

Quantum Algorithms for PDEs via Summation-by-Parts Discretisations

by

Vyom Nilesh Patel

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics (Quantum Information)

Waterloo, Ontario, Canada, 2025

© Vyom Nilesh Patel 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Partial differential equations (PDEs) underpin the mathematical description of physical phenomena across science and engineering. High-order discretisation techniques, such as those based on the Summation-by-Parts (SBP) framework, provide accurate and energy-stable numerical schemes that preserve conservation and stability properties of the continuous equations. These discretisations yield large, linear systems that have significant structure, whose solution constitutes the computational bottleneck of many scientific simulations. Quantum systems can represent exponentially large state spaces using only a polynomial number of qubits. By exploiting this representational capacity, quantum algorithms can achieve substantial, and in some cases exponential, speedups for certain classes of computational problems. However, the practical application of quantum algorithms to PDEs is challenging as a result of the difficulty of efficiently representing the discrete operators as unitary transformations suitable for quantum computation.

This thesis establishes a first step and a significant step toward a unified framework linking high-order SBP discretisations with quantum algorithms based on polynomial spectral transformations. Using Quantum Singular Value Transformation (QSVT), we show how the matrix exponential $\exp(AT)$, governing time evolution in semi-discrete PDE systems, can be efficiently implemented as a quantum circuit. A key contribution is the development of systematically constructible and ancilla-efficient **block-encodings** for structured matrices, including those arising from high-order SBP discretisations of PDEs. These constructions exploit the tensor-product and sparsity structure of the underlying differential operators to enable automated coherent circuit synthesis with polylogarithmic scaling in system size. The resulting framework bridges classical numerical methods and quantum algorithm design by embedding stable, high-order discretisation operators into the coherent quantum model.

The methods are demonstrated on the linear advection equation by block-encoding the SBP semi-discrete operator and applying QSVT to realise its time evolution. This work serves as a blueprint for applying QSVT-based quantum algorithms to high-order discretisations of general linear PDEs.

Acknowledgements

I would like to acknowledge that all of the work presented in this thesis was conducted on the traditional territory of the Neutral, Anishinaabeg, and Haudenosaunee peoples, on lands that form part of the Haldimand Tract granted to the Six Nations along the Grand River.

I am deeply grateful to my supervisor, Prof. David Del Rey Fernández, for his unwavering guidance, patience, and support throughout this work. From the very beginning, he encouraged me to pursue my curiosity and explore new ideas with independence and confidence. His trust allowed me to follow unconventional directions and develop my own perspective on research while always providing thoughtful feedback and steady mentorship. I am deeply grateful for the openness of our discussions, often extending far beyond the immediate scope of my project, which have continually broadened my understanding and sharpened my thinking. Working under his supervision has been an exceptionally formative experience, and his example will continue to influence how I approach both science and mentorship in the future.

I am also grateful to my co-supervisors, Profs. Ala Shayeghi and Hans De Sterck, for their guidance and support throughout my studies. I am especially grateful for their encouragement at every stage of this project and for the many discussions that helped refine both its direction and presentation.

I owe a special debt of gratitude to Prof. Steven Rayan, whose mentorship during my undergraduate studies fundamentally shaped the path that led me here. He was the first person to recognise and nurture my potential for research, welcoming me into his group and encouraging me to explore ideas well beyond the scope of his own interests at the time. His kindness and generosity, both as a teacher and as a mentor, created an environment where curiosity felt valued and learning was a shared pursuit. His belief in my abilities—long before I had any evidence to justify it—gave me the confidence to pursue mathematics with purpose and conviction. It was his teaching, guidance, and example that first showed me what it means to think deeply, to work independently, and to care about ideas for their own sake. I can say without exaggeration that I would not be where I am today without his influence, and I remain profoundly grateful for his continued mentorship and friendship.

I am fortunate that my closest friends from school—Zeenal, Viraj, Tanuja,

and Divya—have remained an inseparable part of my life despite the distance that now separates us. Though we speak across time zones, our conversations continue with the same warmth, humour, and honesty that defined our years together. Growing up with them shaped much of who I am—the way I think, the way I listen, and the way I face challenges. Their friendship has been a constant source of perspective and balance, reminding me of where I come from and of the people who have stood by me through every stage of this journey.

I am especially thankful to the friends who have been a quiet but constant presence during my time in graduate school—Shivani, Honey, Sahil, Ridhi, Vishtasp, Aaron, and Laurena. Graduate school can often feel isolating, and their messages, conversations, and shared humour brought warmth and perspective, helping me stay grounded. I am grateful for their patience and kindness, and for reminding me that life extends well beyond research.

Finally, my heartfelt thanks go to my loving parents, Nilesh and Riya, and my brother, Aarsh, for their unwavering support and encouragement throughout my academic journey. Their belief in me has been a constant source of motivation. Every opportunity I have had, and every step I have taken, rests on the foundation they built through their patience, sacrifices, and unconditional love.

Dedication

Dedicated to my parents, Nilesh and Riya,
and my brother, Aarsh.

Table of Contents

	Author's Declaration	ii
	Abstract	iii
	Acknowledgements	iv
	Dedication	vi
	List of Figures	x
1	Introduction	1
	1.1 Motivation and Background	1
	1.2 Thesis Objectives and Outline	4
2	Stability and Energy Method	7
	2.1 Well Posedness and Stability	7
	2.2 Energy Method	10
	2.3 Linear Advection Equation	10
3	Classical Summation-by-Parts Operators	13
	3.1 Notation and Definitions	13
	3.2 Summation-by-Parts Operators	14
	3.3 Simultaneous Approximation Terms	16

4	Generalised Summation-by-Parts Operators	19
4.1	Notation and Definitions	19
4.2	Construction of Generalised SBP Operators	21
4.3	Curvilinear Geometric Transformation	22
4.4	Summation-by-Parts Operators on Curved Elements	27
4.5	Semi-Discrete Formulation on Curvilinear Meshes	32
4.6	Numerical Experiment	36
5	Introduction to Quantum Computing	41
5.1	Foundations of Quantum Information	41
5.2	Circuit Model of Quantum Computation	43
5.3	Quantum Fourier Transform	49
6	Building Blocks for Quantum Arithmetic and Operator Synthesis	53
6.1	Multi-Controlled X Gate	53
6.2	Multi-Control-Multi-Target Gate	58
6.3	Quantum Adders	59
6.4	Proxy Phasing	66
6.5	Projector Controlled-Unitary Gate	67
6.6	Linear Combination of Unitaries	69
7	Block-Encoding Framework for Structured Linear Operators	75
7.1	Definition	75
7.2	Extensibility Properties of Block-Encodings	77
7.3	Block-Encoding Structured Matrices	81
7.4	Oracle Construction	88
7.5	Efficient Oracle Design under Matrix Structure	91
7.6	Limitations and Future Extensions	97

8	Quantum Signal Processing	101
8.1	Quantum Signal Processing	101
8.2	Polynomial Function Approximation	107
8.3	Fitting Polynomials with QSP	109
9	Quantum Singular Value Transformation	111
9.1	Definitions	111
9.2	Quantum Singular Value Transformation	114
9.3	Matrix Polynomials for Hermitian Matrices	118
9.4	Hamiltonian Simulation via QSVT	120
10	Quantum Algorithm for the Advection Equation	123
10.1	Linear Advection in One Dimension	123
10.2	Quantum Realisation of the Semi-Discrete System	129
10.3	Quantum Circuit Complexity	132
10.4	Limitations and Future Extensions	139
11	Conclusions and Recommendations	143
11.1	Conclusions	143
11.2	Recommendations	145
	Bibliography	146
	Appendices	153
	Appendix A: Supplementary Material for Chapter 6	153
	Appendix B: Supplementary Material for Chapter 7	161
	Appendix C: Supplementary Material for Chapter 10	165

List of Figures

4.1	Collocated grid points on a triangle. Blue circles indicate the grid points, while red squares indicate the boundary points.	22
4.2	Conforming interface.	22
4.3	Non-conforming interface.	22
4.4	Mesh Ω generated as the image of $\hat{\Omega}$ under a curvilinear coordinate transformation.	23
4.5	Warped tetrahedral mesh grid.	37
4.6	Numerical solution for the 3D advection equation obtained using a degree- $p = 3$ SBP-SAT discretisation on a warped mesh with $\varepsilon = 1/16$ and 1296 tetrahedral elements.	38
4.7	Time evolution of the conservation and energy residuals.	39
4.8	Solution error convergence rates computed by fitting a line through the last three values on the refined meshes.	39
5.1	An example of a circuit with gates G_1 , G_2 , and G_3 .	44
5.2	Toffoli gate	44
5.3	Implementing NAND using Toffoli.	44
5.4	Implementing FANOUT using Toffoli.	44
5.5	Notation for a three-qubit diagonal gate.	46
5.6	Decomposition of $C_k^u(U)$ gate. The symbol $\odot$ denotes a generalised control pattern.	46
5.7	Decomposition of the Toffoli gate.	46
5.8	Decomposition of the Toffoli gate up to a diagonal gate.	47
5.9	Commutation of a diagonal gate Δ with a multiplexed unitary U .	47
5.10	The SWAP gate.	47
5.11	Decomposition of the SWAP gate.	47
5.12	Fredkin gate decomposition.	47
5.13	The Hadamard transform on n qubits.	49
5.14	The roots of unity on the unit circle.	50
5.15	Controlled phase rotations for QFT for $m = 5$ qubits.	51
5.16	QFT on 5 qubits.	51
6.1	An MCX_n gate with n control qubits and 1 target qubit.	53
6.2	MCX_n gate with one borrowed qubit.	54
6.3	Example of using conditionally clean ancillae. The unitary U acts as the desired operation only when $a = 1$, and in this subspace it may safely treat (c_1, c_2) as clean workspace in the known state $ 11\rangle$.	56
6.4	Decomposition of MCX_9 using one clean ancilla. Figure adapted from [KG25].	57
6.5	Example of a pair of CCX-gates in a compute-uncompute pattern that can be replaced by a RCCX-gate.	57

6.6	Decomposition of MCX_9 using one dirty ancilla. Figure adapted from [KG25].	58
6.7	Decomposing $\text{MCMT}_n^m(X)$ into one MCX_n gate and $m - 1$ CX gates.	58
6.8	Decomposition of $\text{MCMT}_4^3(X)$ gate.	59
6.9	Adding stored value K in top register to the bottom register.	59
6.10	Draper adder circuit DADD_3 for $n = 3$.	60
6.11	The Van Rentergem adder for n qubits.	61
6.12	Draper constant adder circuit CADD_3 for $n = 3$ and $a = 5 = 101_2$.	61
6.13	Recursive constant addition circuit using a single clean ancilla qubit $ g\rangle = 0\rangle$. The CARRY gate computes and stores the overflow carry of $x_L + a_L$ in g , which then controls an increment on x_H . The register x is split into lower and upper halves x_L and x_H .	62
6.14	Same as Figure 6.13, but using a dirty ancilla qubit g instead of a clean one. The pink control gates are MCMT gates that act on all higher-order bits of x_H .	64
6.15	Subcircuit of the offset gate showing cancellation behaviour when the control condition is not satisfied.	64
6.16	Controlled version of Figure 6.14. Only the CARRY_n^a gates are controlled; all other gates remain unchanged.	65
6.17	Comparison of CX-gate counts for Draper vs. HRS constant adders.	66
6.18	Proxy phasing.	67
6.19	Projector-controlled phase shift gate.	68
6.20	LCU circuit for $k = 4$, using two control qubits (since $\lceil \log_2 4 \rceil = 2$)	71
7.1	Quantum circuit for block-encoding. Post-selecting the ancilla outcome 0^m prepares the data register in a state proportional to $A \psi\rangle$.	77
7.2	Circuit construction of U_A , the block-encoding of a matrix A as established in Theorem 7.6.	82
7.3	Quantum circuit with data oracle split into O_{data}^p and O_{data}^{1-p} components.	85
7.4	Circuit construction of U_A , based on the PREP/UNPREP method as established in Theorem 7.7.	86
7.5	O_{rg} with gates colour-coded and numbered according to the corresponding pair (l, j) . E.g., gate 4 activates when $l = 2$ and $j = 7$ (i.e., $ 111\rangle$).	89
7.6	Example circuit for O_r , with gates colour-coded according to the corresponding pair $(l, k^{(l)})$. E.g., gate 3 activates when $l = 1$ and applies a shift of $+1$.	90
7.7	O_{data} circuit.	91
7.8	Decomposition of multiplexed rotations on four values.	91
7.9	(Left) C_l^c ; (Right) C_l followed by an X on the flag qubit.	92
7.10	Compact implementation of O_{rg} of Figure 7.9.	92
7.11	Merging controls when two control patterns differ by a single bit.	92
7.12	Iterative cancellations of adjacent control patterns applied to Figure 7.9.	93
7.13	Optimisation of O_r when multiple data items share the same offset $k_l = c$. Control patterns are assigned according to a Gray code sequence (here 3, 2, 6, 7).	95
7.14	Block-encoding circuit for an 8×8 Toeplitz matrix.	98
7.15	Block-encoding circuit for an 8×8 circulant matrix.	99
7.16	Two-dimensional Laplacian matrix with 4×4 grid discretisation.	99
7.17	Block-encoding circuit for a 16×16 2D Laplacian matrix.	100
8.1	Geometric interpretation of the unitary matrix $A(x)$ as a rotation in the Bloch sphere.	103
8.2	Geometric interpretation of the unitary matrix $A(x)$ as a reflection across the $ 0\rangle$ state, followed by a reflection across the rotated state $ \psi\rangle$.	104

8.3	The Bernstein ellipse E_ρ with foci at ± 1 and semi-axes $\frac{\rho+\rho^{-1}}{2}$ and $\frac{\rho-\rho^{-1}}{2}$.	108
8.4	QSP polynomial approximations to $\cos(10x)$ and $\sin(10x)$ on $[-1, 1]$ using <code>PYQSP</code> . The real parts match the targets with near-minimax error, while the imaginary parts remain nonzero.	110
9.1	Geometric interpretation of steps 1–4 above. The state $ 0\rangle v_i\rangle$ is mapped via a sequence of unitaries and reflections through a two-dimensional subspace spanned by singular vectors and their orthogonal complements.	116
9.2	Controlled projector for block-encoding.	117
9.3	Quantum circuit implementing the phased alternating sequence for odd d .	118
9.4	Controlled version of Figure 9.3 for odd d .	118
9.5	Block-encoding of $p(A)$ using LCBU to separate the real and imaginary parts.	119
9.6	Optimised circuit for block-encoding $p(A)$ using only one additional ancilla qubit compared to the LCBU construction in Figure 9.5.	119
9.7	Block-encoding of $ip(A)$ using LCBU.	120
9.8	Block-encoding of $\frac{1}{2}e^{itA}$.	120
10.1	$\Omega = [a, b]$ partitioned into n_{elem} non-overlapping elements $\{\Omega_\kappa\}_{\kappa=1}^{n_{\text{elem}}}$ of equal size $\Delta x = \frac{(b-a)}{n_{\text{elem}}}$.	123
10.2	First five orthonormal Legendre polynomials on $[-1, 1]$.	127
10.3	L^2 -error convergence of the classical SBP scheme for 1D linear advection problem.	129
10.4	L^2 -error convergence of the quantum algorithm for polynomial degrees $p = 1$ and 3 with fixed QSVT tolerance $\varepsilon_q = 10^{-13}$.	131
10.5	Explicit $p = 3$ block column for unique-entry analysis	132
10.6	Measured CX-gate counts versus global matrix size $N = n_{\text{elem}}(p + 1)$.	136
10.7	Isocontours of the required QSVT polynomial degree $d(\tau, \varepsilon)$.	138
1	Naive incrementer using a series of MCX gates for $n = 6$.	153
2	Surrounding an increment gate with X gates turns it into a decrement gate.	153
3	Incrementer circuit for n -qubit registers using n dirty ancilla qubits in register $ g\rangle$.	154
4	Incrementer circuit for an n -qubit register with odd n , using a single dirty ancilla qubit $ g\rangle$.	156
5	Incrementer circuit for an n -qubit register with even n , using a single dirty ancilla qubit $ g\rangle$.	157
6	Controlled version of Figure 4. The pink gates indicate the modification where the final subtractor is replaced by an adder conjugated by X-gates.	157
7	Controlled Draper constant adder circuit CADD ₃ for $n = 3$ and $a = 5 = 101_2$.	158
8	Example circuit for computing the final carry bit of $x + 11$ using dirty ancilla qubits g . The pink gates acting on qubit i are present only if the i -th bit of the constant a is 1. For the constant $a = 11 = 1011_2$, the gates on qubit index 2 are omitted, since $a_2 = 0$.	159
9	Controlled CARRY _{n} ^{a} gate.	160
10	Block-encoding Q_g matrix for $p = 1$.	166

Introduction

1.1 Motivation and Background

Partial differential equations (PDEs) form the mathematical foundation of most physical models in science and engineering. Often, they express conservation laws and constitutive relations governing the evolution of quantities such as mass, momentum, and energy in space and time. Since analytical solutions are rarely available for practical problems, numerical approximation of PDEs has become one of the primary means of predictive simulation. The accuracy and efficiency of such simulations depend critically on the choice of spatial and temporal discretisation schemes used to approximate the continuous operators.

High-order discretisation schemes offer superior accuracy per degree of freedom and thus become more computationally efficient than low-order methods when stringent error tolerances are required. For a method of formal order p , the numerical error e satisfies

$$e = \mathcal{O}(h^p), \quad (1.1)$$

where h denotes the mesh spacing and $p > 2$ characterises high-order [Wan07]. Despite their advantages, such methods often exhibit reduced robustness on complex geometries or in the presence of discontinuities. Ongoing research therefore seeks formulations that retain high-order accuracy while improving stability and efficiency.

There exist numerous approaches for constructing high-order discretisations of PDEs, including finite-difference, finite-volume, continuous and discontinuous-Galerkin, and many more. In this work, we adopt the **Summation-by-Parts (SBP)** framework. Originally introduced by Kreiss and Scherer [KS74], SBP operators mimic integration by parts at the discrete level. When coupled with **Simultaneous Approximation Terms (SATs)** for weak enforcement of boundary and interface conditions, SBP schemes lead to provably stable, conservative, and high-order accurate discretisations suitable for complex multi-element and curvilinear geometries [DHZ14, CHDRF⁺18].

Quantum computing offers a fundamentally different computational paradigm based on the principles of superposition, entanglement, and unitary evolution. Unlike classical digital computers, which store and manipulate information using bits that take values in $\{0, 1\}$, a quantum computer operates on **qubits** whose states reside in complex Hilbert spaces. A system of n qubits can represent a superposition of 2^n classical configurations, enabling compact encoding and coherent manipulation of exponentially large state spaces. Computation proceeds through unitary transformations acting on these states, followed by measurement, which probabilistically extracts classical information [NC00].

Theoretical studies have shown that quantum algorithms can outperform their classical counterparts for specific computational tasks. Notable examples include integer factorisation [Sho97], database search [Gro96], and Hamiltonian simulation [LC17]. Of particular relevance to scientific computing are quantum algorithms for linear algebraic operations such as matrix inversion, eigenvalue estimation, and linear systems solvers, which form the foundation of many PDE discretisations [HHL09, CLO21]. These algorithms exploit the ability of quantum computers to efficiently represent and manipulate high-dimensional statevectors, potentially enabling exponential speedups over classical methods for certain problem classes. We are thus motivated to explore how quantum computing techniques can be applied to the numerical solution of PDEs, particularly leveraging high-order stable discretisations.

At an abstract level, a large class of quantum algorithms can be formulated as transformations of matrix spectra or operator functions. Such formulations align naturally with the semi-discrete systems arising from numerical PDE discretisations, where the evolution of the solution is governed by a linear operator A , as in

$$\frac{du_h}{dt} = Au_h. \quad (1.2)$$

Efficiently realising the action of $\exp(tA)$ corresponds to simulating the time evolution under the semi-discrete operator, while matrix inversion appears in the context of implicit time-marching schemes, where one must solve systems of the form, for example, $(\mathbb{1} - \Delta t A)u_h^{n+1} = u_h^n$. In both cases, the ability to implement matrix functions such as $\exp(tA)$ or $(\mathbb{1} - \Delta t A)^{-1}$ efficiently on a quantum computer becomes central to the development of quantum algorithms for PDEs.

A key theoretical breakthrough enabling such operator manipulations is the framework of **Quantum Singular Value Transformation (QSVT)** [GSLW19]. QSVT provides a unified algebraic formalism for implementing polynomial transformations of the singular values of a given operator using coherent quantum circuits. This formalism generalises and subsumes many of the most powerful quantum algorithmic primitives, including phase estimation, Hamiltonian simulation, and quantum linear system solvers, by expressing them as special cases of polynomial spectral transformations [MRTC21]. The essential

insight is that quantum circuits naturally implement orthogonal polynomial recursions under alternating sequences of reflections and phase modulations [LYC16], enabling controlled approximation of analytic functions of operators without explicitly forming their spectral decompositions.

QSVT operates by embedding the singular values of a target operator into the eigenphases of a unitary transformation and then applying a sequence of controlled reflections and phase rotations whose composition implements a desired polynomial transformation of those singular values. Specifically, given access to a unitary operator U whose subspace action corresponds to the operator of interest, QSVT constructs a family of circuits parameterised by a sequence of phase angles $\Phi = \{\varphi_0, \varphi_1, \dots, \varphi_{d-1}\}$ such that the resulting transformation

$$U_{\text{QSVT}}(U, \Phi) = e^{i\varphi_0 Z} U e^{i\varphi_1 Z} U \dots e^{i\varphi_{d-1} Z} U,$$

acts as a polynomial function of the singular values of the embedded operator. The polynomial implemented depends solely on the choice of phase sequence Φ , while the depth d determines the degree of the polynomial approximation and hence the achievable accuracy [GSLW19, LYC16]. For QSVT to act on a general, possibly non-unitary operator A , it is necessary to represent A as a sub-block of a larger unitary matrix. This representation, known as a **block-encoding** [GSLW19], is defined by a unitary $U \in \mathbb{U}(2^{n+a})$ satisfying

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}, \quad \text{for some normalisation constant } \alpha > 0,$$

where the top-left block encodes the scaled operator A/α and the remaining entries complete unitarity. This embedding enables non-unitary linear operations to be manipulated within the coherent framework of quantum computation using only unitary primitives. Block-encodings thus form the fundamental interface between classical linear algebra and quantum circuits: they translate matrices, such as those arising from PDE discretisations, into physical quantum operations that can be composed, inverted, or combined via QSVT.

Constructing block-encodings for arbitrary matrices, however, is a highly nontrivial task. In general, the process requires explicit access to all matrix elements and the ability to coherently load them into quantum amplitudes, which is infeasible for dense operators due to exponential resource requirements. The practical applicability of QSVT-based algorithms therefore hinges on identifying classes of matrices that admit efficient block-encodings, those that can be realised using a number of quantum gates and ancillae scaling polylogarithmically in the matrix dimension. Matrices arising from PDE discretisations, such as those produced by finite-difference or Galerkin methods, are highly structured: they are sparse, banded, and often exhibit tensor-product or block-Toeplitz structure induced by the underlying mesh topology.

Despite this favourable structure, only a limited number of systematic constructions for block-encoding such operators currently exist in the literature. As Seth Lloyd, co-author of the Harrow-Hassidim-Lloyd (HHL) algorithm and a foundational figure in quantum algorithms, remarked, “finding efficient block-encodings of operators arising from differential equations is the sixty-four-thousand-dollar question” [Llo22].

1.2 Thesis Objectives and Outline

The overarching objective of this thesis is to establish a first, significant step toward developing a unified framework connecting high-order numerical discretisations of PDEs with quantum algorithms capable of efficiently realising their semi-discrete dynamics. At the classical level, we seek discretisations that are provably stable, conservative, and systematically extendable to complex geometries. At the quantum level, we aim to construct explicit circuit implementations that reproduce these discrete operators within the coherent unitary model, enabling quantum realisation of time evolution through polynomial spectral transformations. We delineate the specific goals as follows:

- Develop efficient and systematically constructible block-encodings for the structured matrices arising from SBP discretisations of PDEs, and design modular quantum arithmetic and control primitives that enable their coherent synthesis using minimal ancillary qubits. This includes exploring algorithmic strategies for automating block-encoding construction in terms of oracle generation and circuit composition.
- Demonstrate the discretisation and solution workflow on the linear advection equation, linking the semi-discrete stability-preserving SBP formulation to its quantum realisation, and analyse the resulting complexity and accuracy trade-offs.
- Develop a generalised and extensible implementation framework for SBP-based discretisations on unstructured and curvilinear grids, and perform a systematic analysis of their fundamental properties, including consistency, conservation, and stability. This involves a comprehensive, self-contained treatment of the generalised SBP formulation and the development of a high-performance Julia codebase, `SBPLite.jl`, enabling automatic generation of SBP operators on complex geometries.

The rest of this thesis is organised as follows. [Chapter 2](#) introduces the concepts of well-posedness and stability for linear PDEs. [Chapter 3](#) presents classical SBP operators and SATs for constructing consistent, conservative, and energy-stable finite-difference discretisations. [Chapter 4](#) extends the SBP framework to multidimensional and curvilinear elements, analyses their stability and conservation properties, and provides the mathematical foundation on which `SBPLite.jl` is implemented. [Chapter 5](#) introduces the essential

quantum computing concepts and notation used in later chapters. [Chapter 6](#) develops the quantum arithmetic and control primitives required for synthesising block-encoding circuits, including efficient constructions of multi-controlled gates and conditional arithmetic modules. [Chapter 7](#) formalises the block-encoding framework and develops systematic procedures for constructing block-encodings of structured matrices. [Chapter 8](#) introduces Quantum Signal Processing (QSP) and its role in implementing polynomial transformations using phase-modulated single-qubit rotations. [Chapter 9](#) extends this formalism to QSVT, which applies polynomial transformations to the singular values of block-encoded operators. Finally, [Chapter 10](#) demonstrates the complete quantum realisation of the SBP-based semi-discrete advection equation via block-encoding and QSVT and presents the associated error and complexity analyses. Finally, [Chapter 11](#) summarises the main contributions of the thesis and outlines directions for future research.

2

Stability and Energy Method

In this chapter, we review the concept of well-posedness and stability for PDEs. We introduce the energy method, a technique used to derive stability estimates at the PDE level and which later guides the design of stable discretisations. Our goal will be to develop discretisations of well-posed problems that are also discretely well-posed, achieved by constructing numerical methods that closely mimic the stability analysis carried out for the continuous PDE.

2.1 Well Posedness and Stability

Definition 2.1 (Well Posedness [Hado3]). A problem is said to be **well posed** if it satisfies the following three conditions:

- Existence: A solution exists for the problem.
- Uniqueness: The solution is unique.
- The solution depends continuously on the data of the problem.

These conditions ensure that small changes in the initial data lead to small changes in the solution. If any of these conditions are violated, the problem is considered **ill-posed**. The first two conditions are self-explanatory,¹ however we will elaborate on the third condition a bit more rigorously.

Note 1. *Throughout this thesis we restrict attention to linear PDEs. Accordingly, all stability and well-posedness results presented in this chapter apply only to linear problems.*

¹ For many linear PDEs, existence follows from standard semigroup methods [Eva10]. In contrast, existence for nonlinear PDEs can be highly nontrivial; a famous open problem is global existence and smoothness of solutions to the three-dimensional Navier-Stokes equations [Fef06].

Stability can be defined in several ways; in this work, we adopt the formulation presented in [GKO13]. We consider the following initial boundary

value problem (IBVP):

$$\begin{aligned} \frac{\partial \mathcal{U}}{\partial t} &= \mathcal{P}(\vec{x}, t) \mathcal{U} + \mathcal{S}(\vec{x}, t), \quad \forall \vec{x} \in \Omega, \quad \forall t > t_0, \\ \mathcal{B}(\vec{x}, t, \mathcal{U}) &= \mathcal{G}(t), \quad \forall \vec{x} \in \partial\Omega, \quad \forall t > t_0, \\ \mathcal{U}(\vec{x}, t_0) &= \mathcal{F}(\vec{x}), \quad \forall \vec{x} \in \Omega, \quad t = t_0, \end{aligned} \quad (2.1)$$

where $\mathcal{U} := [\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_m]^T$ is an m -component vector-valued function, $\vec{x} := [x_1, x_2, \dots, x_d]^T$ denotes the spatial variables, $\Omega \subset \mathbb{R}^d$ is the spatial domain, and $\partial\Omega$ its boundary. The source term is denoted by $\mathcal{S}$, the initial condition by $\mathcal{F}$, and t_0 is the initial time. The operator $\mathcal{P}$ is a differential operator of order n , given by

$$\mathcal{P}(\vec{x}, t) = \sum_{|i| \leq n} A_i(\vec{x}, t) \left(\frac{\partial}{\partial x_1} \right)^{i_1} \cdots \left(\frac{\partial}{\partial x_d} \right)^{i_d},$$

where each $A_i(\vec{x}, t)$ is a smooth matrix-valued function and $i = (i_1, \dots, i_d)$ is a multi-index. The boundary operator $\mathcal{B}$ is a differential operator of order $k \leq n - 1$.

Definition 2.2 (Stability and Well Posedness [GKO13]). Consider the initial boundary value problem (2.1) with homogeneous boundary conditions (i.e., $\mathcal{S} = 0, \mathcal{G} = 0$). The problem is said to be **stable** if there exists an estimate of the form:

$$\|\mathcal{U}(\vec{x}, t)\|_{\Omega}^2 \leq C e^{\alpha(t-t_0)} \|\mathcal{U}(\cdot, t_0)\|_{\Omega}, \quad (2.2)$$

where C and α are constants independent of the initial data $\mathcal{F}$ and t_0 . In addition, if a unique smooth solution exists, then the problem is said to be **well posed**.

For non-homogeneous data, the stability condition is slightly different:

Definition 2.3 (Stability and Strong Well Posedness [Nor15]). The IBVP (2.1) is said to be **stable** if there exists an estimate of the form:

$$\|\mathcal{U}(\vec{x}, t)\|_{\Omega}^2 \leq C(t, t_0) \left[\|\mathcal{F}(\vec{x})\|_{\Omega}^2 + \int_{t_0}^t \left(\|\mathcal{S}(\cdot, \tau)\|_{\Omega}^2 + \|\mathcal{G}(\tau)\|_{\partial\Omega}^2 \right) d\tau \right], \quad (2.3)$$

where $C(t, t_0)$ is independent of the data and is bounded for all finite time intervals. If a unique smooth solution exists, then the problem is said to be **strongly well posed**.

In both definitions, the norms of the functions involved may be defined differently. In this work, we primarily use the L_2 norm over a finite domain Ω with boundary $\partial\Omega$ for all functions, defined as:

$$\langle \mathcal{U}, \mathcal{V} \rangle_{L_2(\Omega)} := \int_{\Omega} \mathcal{U} \mathcal{V} d\Omega, \quad \|\mathcal{U}\|_{L_2(\Omega)} := \left(\int_{\Omega} \mathcal{U}^2 d\Omega \right)^{1/2},$$

$$\langle \mathcal{U}, \mathcal{V} \rangle_{L_2(\partial\Omega)} := \int_{\partial\Omega} \mathcal{U} \mathcal{V} d\Gamma, \quad \|\mathcal{U}\|_{L_2(\partial\Omega)} := \left(\int_{\partial\Omega} \mathcal{U}^2 d\Gamma \right)^{1/2}.$$

In general, stability is a **necessary but not sufficient** condition for ensuring continuous dependence on the problem data. However, in the case of **linear problems**, stability guarantees both **continuous dependence** and **uniqueness** of the solution [Guso8]. It is important to note that both definitions of stability considered here permit solutions to grow exponentially over finite time intervals. This interpretation of stability differs from the classical notion used in the general theory of differential equations, where stability often implies asymptotic behaviour such as decay to equilibrium. In the present context, stability simply ensures that the solution does not become unbounded or **blow up** in finite time.

Theorem 2.1. *For linear problems, the stability condition (2.3) implies continuous dependence on the data and uniqueness of the solution.*

Proof.

We consider a perturbation of the IBVP (2.1):

$$\begin{aligned} \frac{\partial \mathcal{V}}{\partial t} &= \mathcal{P} \left(\vec{x}, t, \frac{\partial}{\partial x} \right) \mathcal{V} + \mathcal{S}(\vec{x}, t) + \delta \mathcal{S}(\vec{x}, t), \quad \forall \vec{x} \in \Omega, \quad \forall t > t_0, \\ \mathcal{B}(\vec{x}, t, \mathcal{V}) &= \mathcal{G}(t) + \delta \mathcal{G}(t), \quad \forall \vec{x} \in \partial\Omega, \quad \forall t > t_0, \\ \mathcal{V}(\vec{x}, t_0) &= \mathcal{F}(\vec{x}) + \delta \mathcal{F}(\vec{x}), \quad \forall \vec{x} \in \Omega, \quad t = t_0. \end{aligned} \tag{2.4}$$

Let $\mathcal{W} = \mathcal{V} - \mathcal{U}$ and assume $\mathcal{B}$ and $\mathcal{P}$ are linear operators. Subtracting (2.1) from (2.4) gives:

$$\begin{aligned} \frac{\partial \mathcal{W}}{\partial t} &= \mathcal{P} \left(\vec{x}, t, \frac{\partial}{\partial x} \right) \mathcal{W} + \delta \mathcal{S}(\vec{x}, t), \\ \mathcal{B}(\vec{x}, t, \mathcal{W}) &= \delta \mathcal{G}(t), \quad \forall \vec{x} \in \partial\Omega, \quad \forall t > t_0, \\ \mathcal{W}(\vec{x}, t_0) &= \delta \mathcal{F}(\vec{x}), \quad \forall \vec{x} \in \Omega, \quad t = t_0. \end{aligned} \tag{2.5}$$

If (2.1) is stable, then the estimate (2.3) holds for (2.5), and we have:

$$\|\mathcal{W}(\vec{x}, t)\|_{\Omega}^2 \leq C(t, t_0) \left[\|\delta \mathcal{F}(\vec{x})\|_{\Omega}^2 + \int_{t_0}^t \left(\|\delta \mathcal{S}(\cdot, \tau)\|_{\Omega}^2 + \|\delta \mathcal{G}(\tau)\|_{\partial\Omega}^2 \right) d\tau \right].$$

Small perturbations in the input data namely, $\delta \mathcal{F}$, $\delta \mathcal{S}$, and $\delta \mathcal{G}$ lead to correspondingly small perturbations in the solution $\mathcal{W}$. This expresses the notion that the solution to the problem (2.1) depends continuously on the data. Moreover, if the perturbations vanish (i.e., $\delta = 0$), then $\mathcal{V} = \mathcal{U}$, implying that the solution $\mathcal{U}$ is unique. ■

Throughout this thesis, for each problem studied, we formulate a continuous-level stability estimate of the form (2.3) to ensure that this necessary condition for well-posedness is satisfied. We then construct numerical schemes that replicate this stability property at the discrete level. In doing so, we ensure that the resulting schemes are stable which is an essential requirement for the reliable simulation of partial differential equations, as it guarantees that the numerical solution remains bounded as it evolves in time.

2.2 Energy Method

A common approach for deriving estimates of the L_2 norm of the solution in terms of the problem data is the **energy method**, which involves the following steps:

- Multiply the PDE by the transpose of the solution and integrate over the spatial domain Ω .
- Apply integration by parts to transform volume integrals into surface integrals to allow insertion of the boundary conditions.
- Integrate in time to obtain an estimate of the L_2 norm of the solution.

One of the key concepts used in the energy method is integration by parts (IBP) which motivates the design of derivative approximations that discretely replicate this property. Such methods, known as **Summation-by-Parts** (SBP) schemes, result in discrete stability proofs that mimic the PDE-level analysis in a one-to-one fashion. The energy method offers systematic approach for analysing stability in both continuous and discrete settings. It can also, in certain cases, be used to demonstrate uniqueness of the solution.

2.3 Linear Advection Equation

Consider the linear advection equation:

$$\begin{aligned} \frac{\partial \mathcal{U}}{\partial t} + \sum_{i=1}^d \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} &= 0, \quad \forall \vec{x} \in \Omega, \quad \forall t \geq 0, \\ \mathcal{U}(\vec{x}, 0) &= \mathcal{F}(\vec{x}), \quad \forall \vec{x} \in \Omega, \quad t = 0, \end{aligned} \quad (2.6)$$

where we assume the advection velocity field $[a_{x_1}, a_{x_2}, \dots, a_{x_d}]^T$ is constant. Let $\Gamma := \partial\Omega$ denote the boundary of the spatial domain Ω , and let n_{x_i} be the i -th component of the outward unit normal on Γ . We do not impose boundary conditions a priori; instead, we derive them by requiring an energy estimate that ensures stability.

We start by multiplying (2.6) by $\mathcal{U}$ and integrating in space:

$$\int_{\Omega} \mathcal{U} \frac{\partial \mathcal{U}}{\partial t} d\Omega + \frac{1}{2} \sum_{i=1}^d \left(\int_{\Omega} \mathcal{U} \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} d\Omega + \int_{\Omega} \mathcal{U} \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} d\Omega \right), \quad (2.7)$$

where we split the second term into two identical terms for easy cancellation later. We now apply integration by parts to the last term to obtain:

$$\int_{\Omega} \mathcal{U} \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} d\Omega = \oint_{\Gamma} \mathcal{U}^2 a_{x_i} n_{x_i} d\Gamma - \int_{\Omega} \frac{\partial \mathcal{U}}{\partial x_i} a_{x_i} \mathcal{U} d\Omega. \quad (2.8)$$

Substituting (2.8) into (2.7)

$$\int_{\Omega} \mathcal{U} \frac{\partial \mathcal{U}}{\partial t} d\Omega + \frac{1}{2} \sum_{i=1}^d \left(\oint_{\Gamma} \mathcal{U}^2 a_{x_i} n_{x_i} d\Gamma + \cancel{\int_{\Omega} \mathcal{U} \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} d\Omega} - \cancel{\int_{\Omega} \frac{\partial \mathcal{U}}{\partial x_i} a_{x_i} \mathcal{U} d\Omega} \right), \quad (2.9)$$

In one dimension, the integration by parts formula is given by:

$$\int_a^b v \frac{\partial u}{\partial x} dx = vu \Big|_a^b - \int_a^b u \frac{\partial v}{\partial x} dx.$$

In higher dimensions, the formula generalises to:

$$\int_{\Omega} v \frac{\partial u}{\partial x_i} d\Omega = \oint_{\Gamma} uv n_{x_i} d\Gamma - \int_{\Omega} u \frac{\partial v}{\partial x_i} d\Omega,$$

where u and v are scalar functions, Ω is the spatial domain, Γ is its boundary, and n_{x_i} is the i -th component of the outward unit surface normal to Γ .

where the last two terms cancel because

$$\mathcal{U} \frac{\partial(a_{x_i} \mathcal{U})}{\partial x_i} = \mathcal{U} a_{x_i} \frac{\partial \mathcal{U}}{\partial x_i} = \frac{\partial \mathcal{U}}{\partial x_i} a_{x_i} \mathcal{U}.$$

Now, using Leibniz's rule on the left-hand side of (2.7), we have:

$$\int_{\Omega} \mathcal{U} \frac{\partial \mathcal{U}}{\partial t} d\Omega = \frac{1}{2} \frac{d}{dt} \int_{\Omega} \mathcal{U}^2 d\Omega = \frac{1}{2} \frac{d \|\mathcal{U}\|^2}{dt}. \quad (2.10)$$

Substituting (2.10) into (2.9), we arrive at the energy identity:

$$\frac{d \|\mathcal{U}\|^2}{dt} = - \sum_{i=1}^d \oint_{\Gamma} \mathcal{U}^2 a_{x_i} n_{x_i} d\Gamma. \quad (2.11)$$

Integrating (2.11) from 0 to t , rearranging, multiplying through by 2, and using the initial condition in (2.6) gives,

$$\|\mathcal{U}(\cdot, t)\|^2 = \|\mathcal{F}(\cdot)\|^2 - \int_0^t \oint_{\Gamma} \mathcal{U}^2 \sum_{i=1}^d (a_{x_i} n_{x_i}) d\Gamma d\tau. \quad (2.12)$$

At this point, we cannot yet conclude that the problem is stable, since the sign of the boundary integral is undetermined. To ensure non-increasing energy, we impose boundary conditions only on the **inflow** portion of the boundary:

$$\mathcal{U}(\vec{x}, t) = \mathcal{G}(\vec{x}, t), \quad \forall \vec{x} \in \Gamma_- := \left\{ x \in \Gamma : \sum_{i=1}^d (a_{x_i} n_{x_i}) < 0 \right\}.$$

We can now split the boundary integral in (2.12) into inflow and **outflow** components and substitute the prescribed data on Γ_- to complete the energy estimate.

$$\begin{aligned} \|\mathcal{U}(\cdot, t)\|^2 &= \|\mathcal{F}(\cdot)\|^2 - \int_0^t \oint_{\Gamma_+} \mathcal{U}^2 \sum_{i=1}^d (a_{x_i} n_{x_i}) d\Gamma d\tau \\ &\quad - \int_0^t \oint_{\Gamma_-} \mathcal{G}^2 \sum_{i=1}^d (a_{x_i} n_{x_i}) d\Gamma d\tau, \end{aligned} \quad (2.13)$$

$$\therefore \|\mathcal{U}(\cdot, t)\|^2 \leq \|\mathcal{F}(\cdot)\|^2 + \int_0^t \oint_{\Gamma_-} \mathcal{G}^2 \left| \sum_{i=1}^d (a_{x_i} n_{x_i}) \right| d\Gamma d\tau.$$

Alternatively, this can be expressed as a balanced energy identity:

$$\begin{aligned} \|\mathcal{U}(\cdot, t)\|^2 + \int_0^t \oint_{\Gamma_+} \mathcal{U}^2 \sum_{i=1}^d (a_{x_i} n_{x_i}) d\Gamma d\tau &= \|\mathcal{F}(\cdot)\|^2 \\ &\quad + \int_0^t \oint_{\Gamma_-} \mathcal{G}^2 \left| \sum_{i=1}^d (a_{x_i} n_{x_i}) \right| d\Gamma d\tau. \end{aligned}$$

This estimate provides control not only over the solution at time t , but also over its behaviour at the outflow boundary Γ_+ , making it a stronger result than the inequality in (2.13). Importantly, specifying a boundary condition at the outflow boundary Γ_+ would not lead to an energy estimate, making the problem ill-posed.

² The ordering of scalar terms $\mathcal{U}, a_{x_i}$, and $\frac{\partial \mathcal{U}}{\partial x_i}$ can be interchanged due to the commutative property of multiplication.

Periodic Boundary Conditions

For periodic boundary conditions we have that

$$\int_{\Gamma} \mathcal{U}^2 a_{x_i} n_{x_i} d\Gamma = 0.$$

Substituting this condition into the energy estimate (2.13) eliminates the boundary contributions, yielding

$$\|\mathcal{U}(\cdot, t)\|^2 = \|\mathcal{F}\|^2,$$

which implies that the solution norm remains constant in time. Therefore, the problem is stable for all $t \geq 0$.

3

Classical Summation-by-Parts Operators

In this chapter, we introduce classical SBP finite difference operators [DHZ14] and their use in constructing consistent and stable discretisations of PDEs. Building on the energy method developed in Chapter 2, our goal is to design discrete derivative operators and impose data in such a way that stability can be proven through a discrete analogue of integration by parts and the energy method. SBP operators achieve this by mimicking integration by parts at the semi-discrete level, while Simultaneous Approximation Terms (SATs) enable weak imposition of boundary conditions and interface coupling.

3.1 Notation and Definitions

We adopt the following conventions for notation based on the notation in [DHZ14, DBZ14], throughout this thesis:

- **Vectors** in continuous space (e.g., vector-valued functions or fields) are written with over-arrows (e.g., $\vec{u}$), while discretised vectors are denoted in boldface (e.g., $\mathbf{u} \in \mathbb{R}^n$).
- **Matrices** are represented using sans-serif capital letters, for example, D ;
- **Smooth functions** defined on a domain $\Omega \subset \mathbb{R}^d$ are written using calligraphic letters, for example, $\mathcal{U}(\vec{x}) \in C^\infty(\Omega)$, indicating that the function is infinitely differentiable;

When a continuous function $\mathcal{U}$ is sampled at discrete grid points $\mathbf{x} \in \mathbb{R}^n$, we denote the resulting vector as:

$$\mathbf{u} = [\mathcal{U}(x_1), \mathcal{U}(x_2), \dots, \mathcal{U}(x_n)]^T.$$

For brevity, we often write $u_i = \mathcal{U}(x_i)$. Similarly, the restriction of the monomial x^k to the grid $\mathbf{x}$ is written as $\mathbf{x}^k = [x_1^k, x_2^k, \dots, x_n^k]^T$. We set $\mathbf{x}^k = \mathbf{0}$ for $k < 0$. When referring to SBP operators, we distinguish between their **degree** and **order**:

- The **degree** of an SBP operator is the highest degree monomial it can differentiate exactly.
- The **order** of an SBP operator is the smallest exponent of the grid spacing Δx in the leading truncation error across all nodes.

For an operator approximating the m^{th} derivative, the two are related by:¹

$$\text{order} = \text{degree} - m + 1.$$

¹ In particular, for first-derivative operators, the degree and order are equal.

3.2 Summation-by-Parts Operators

Definition 3.1 (SBP Operator). Let $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ denote a grid of n nodes. A matrix operator $D \in \mathbb{R}^{n \times n}$ is said to approximate the derivative $\frac{\partial}{\partial x}$ with order and degree p and satisfy the **Summation-by-Parts (SBP)** property if:

1. $D\mathbf{x}^k = H^{-1}Q\mathbf{x}^k = k\mathbf{x}^{k-1}$ for all integers $k \in [0, p]$;
2. $H = H^T$ and $\mathbf{x}^T H \mathbf{x} > 0 \quad \forall \mathbf{x} \neq \mathbf{0}$; and
3. $Q + Q^T = E_c := \text{diag}(-1, 0, \dots, 0, 1)$.

The operator D satisfies the following **degree equations**:

$$\begin{aligned} D\mathbf{x}^k &= k\mathbf{x}^{k-1}, \quad k \in [0, p], \\ Q\mathbf{x}^k &= kH\mathbf{x}^{k-1}, \quad k \in [0, p]. \end{aligned} \tag{3.1}$$

These properties imply that D approximates the spatial derivative with a truncation error of order p :

$$D\mathbf{u} = \frac{\partial \mathcal{U}}{\partial x}(\mathbf{x}) + \mathbf{e}_p, \tag{3.2}$$

where $D\mathbf{u}$ represents the discrete derivative evaluated at the grid nodes, and the error satisfies $\max(|\mathbf{e}_p|) = \mathcal{O}(\Delta x^p)$, with Δx denoting the grid spacing. The matrix H also known as the **norm** or **mass** matrix serves as a discrete inner product matrix, approximating the continuous L^2 inner product and the induced norm:

$$\begin{aligned} (\mathbf{u}, \mathbf{v})_H &:= \mathbf{u}^T H \mathbf{v} \approx \int_{\Omega} \mathcal{U} \mathcal{V} \, d\Omega, \\ \|\mathbf{u}\|_H^2 &:= (\mathbf{u}, \mathbf{u})_H. \end{aligned} \tag{3.3}$$

In this thesis, we restrict attention to SBP operators with diagonal norm matrices H , where the diagonal entries are positive and define a quadrature rule of degree $q \geq 2p - 1$ [DHZ14]. The matrix Q approximates the integration of a function against a derivative:

$$\mathbf{v}^T H D \mathbf{u} = \mathbf{v}^T Q \mathbf{u} \approx \int_{\Omega} \mathcal{V} \frac{\partial \mathcal{U}}{\partial x} \, d\Omega. \tag{3.4}$$

The SBP property ensures that the operators defined in [Definition 3.1](#) mimic the integration by parts at the discrete level. This correspondence enables the construction of schemes that mirror the continuous stability analysis of [Chapter 2](#). Recall the one-dimensional integration by parts identity:

$$\int_{x_L}^{x_R} \mathcal{V} \frac{\partial \mathcal{U}}{\partial x} dx + \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{V}}{\partial x} dx = \mathcal{V}(x_R) \mathcal{U}(x_R) - \mathcal{V}(x_L) \mathcal{U}(x_L). \quad (3.5)$$

At the semi-discrete level, the SBP operators approximate the left-hand side as²

$$\begin{aligned} \mathbf{v}^T \mathbf{H} \mathbf{D} \mathbf{u} + \mathbf{u}^T \mathbf{H} \mathbf{D} \mathbf{v} &= \mathbf{v}^T \mathbf{Q} \mathbf{u} + \mathbf{u}^T \mathbf{Q} \mathbf{v} \\ &= \mathbf{v}^T \mathbf{Q} \mathbf{u} + (\mathbf{v}^T \mathbf{Q}^T \mathbf{u})^T \\ &= \mathbf{v}^T \mathbf{Q} \mathbf{u} + \mathbf{v}^T \mathbf{Q}^T \mathbf{u} \\ &= \mathbf{v}^T (\mathbf{Q} + \mathbf{Q}^T) \mathbf{u} \\ &= \mathbf{v}^T \mathbf{E}_c \mathbf{u} \\ &= \mathcal{V}(x_n) \mathcal{U}(x_n) - \mathcal{V}(x_1) \mathcal{U}(x_1) \end{aligned} \quad (3.6)$$

where we have used the SBP property $\mathbf{Q} + \mathbf{Q}^T = \mathbf{E}_c$. Thus, (3.6) reproduces integration by parts **exactly**, term by term.

Note 2. The operators defined in [Definition 3.1](#) satisfy the integration by parts formula *exactly*, not up to truncation error. This is the origin of the name *Summation-by-Parts*. Crucially, the property holds for any mesh spacing, not merely in the limit $\Delta x \rightarrow 0$.

For $p = 1$ on a uniform grid with spacing Δx , the classical finite-difference SBP operator takes the form

$$\mathbf{D} = \frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix},$$

with

$$\mathbf{H} = \Delta x \operatorname{diag}\left(\frac{1}{2}, 1, \dots, 1, \frac{1}{2}\right), \quad \mathbf{Q} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}. \quad (3.7)$$

The differentiation matrix $\mathbf{D}$ uses a second-order centred stencil in the interior and first-order forward/backward stencils at the boundaries. More generally, diagonal-norm classical finite-difference SBP operators employ order- $2p$ centred differences in the interior and order- p one-sided stencils at the boundaries [[DHZ14](#)].

²

$$\begin{aligned} \mathbf{v}^T \mathbf{H} \mathbf{D} \mathbf{u} &\approx \int_{x_L}^{x_R} \mathcal{V} \frac{\partial \mathcal{U}}{\partial x} dx, \\ \mathbf{u}^T \mathbf{H} \mathbf{D} \mathbf{v} &\approx \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{V}}{\partial x} dx, \end{aligned}$$

$\mathbf{v}^T \mathbf{Q}^T \mathbf{u}$ is a scalar, so it equals its transpose.

3.3 Simultaneous Approximation Terms

Consider the one-dimensional linear advection equation

$$\begin{aligned} \frac{\partial \mathcal{U}}{\partial t} + a \frac{\partial \mathcal{U}}{\partial x} &= 0, \quad \forall x \in [x_L, x_R], \quad \forall t \geq 0, \quad a > 0, \\ \mathcal{U}(x, 0) &= \mathcal{F}(x), \quad \forall x \in [x_L, x_R], \quad t = 0, \end{aligned} \quad (3.8)$$

with inflow boundary condition

$$\mathcal{U}(x_L, t) = \mathcal{G}(t), \quad \forall t \geq 0.$$

From (2.13) we know that this boundary condition renders the continuous problem strongly stable. We now construct a numerical scheme that replicates this stability analysis at the semi-discrete level.

Semi-Discrete Formulation

Discretise the spatial domain on a uniform grid

$$\mathbf{x} := [x_1, x_2, \dots, x_n]^T, \quad x_1 := x_L, \quad x_n := x_R,$$

and let $\mathbf{u}(t) \in \mathbb{R}^n$ denote the semi-discrete approximation to the solution at time t . Using an SBP operator $\mathbf{D} = \mathbf{H}^{-1}\mathbf{Q}$ to approximate ∂_x , the PDE reduces to

$$\frac{d\mathbf{u}}{dt} = -a\mathbf{D}\mathbf{u}, \quad \mathbf{u}(0) = \mathbf{f}, \quad (3.9)$$

where $\mathbf{f}$ is the restriction of $\mathcal{F}$ to the grid.

To mimic the continuous energy method, we multiply (3.9) by $\mathbf{u}^T \mathbf{H}$, the discrete equivalent of multiplying by $\mathcal{U}$ and integrating over the spatial domain, to obtain

$$\mathbf{u}^T \mathbf{H} \frac{d\mathbf{u}}{dt} = -a\mathbf{u}^T \mathbf{H} \mathbf{D} \mathbf{u} = -a\mathbf{u}^T \mathbf{Q} \mathbf{u}. \quad (3.10)$$

Taking the transpose of (3.10) and adding to itself, we obtain:

$$\begin{aligned} \mathbf{u}^T \mathbf{H} \frac{d\mathbf{u}}{dt} + \frac{d\mathbf{u}^T}{dt} \mathbf{H} \mathbf{u} &= -a\mathbf{u}^T (\mathbf{Q} + \mathbf{Q}^T) \mathbf{u} \\ &= -a\mathbf{u}^T \mathbf{E}_c \mathbf{u} \\ &= -a(u_n^2 - u_1^2). \end{aligned}$$

On the left-hand side we recognise

$$\mathbf{u}^T \mathbf{H} \frac{d\mathbf{u}}{dt} + \frac{d\mathbf{u}^T}{dt} \mathbf{H} \mathbf{u} = 2\mathbf{u}^T \mathbf{H} \frac{d\mathbf{u}}{dt} = \frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2,$$

yielding the semi-discrete energy equation

$$\frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 = -a(u_n^2 - u_1^2). \quad (3.11)$$

This is the discrete analogue of (2.11).

Enforcing Boundary Conditions via SATs

To impose the inflow boundary condition, we add a penalty term **SAT**:³

$$\mathbf{SAT}_L = \beta \mathbf{H}^{-1} \mathbf{e}_1 (u_1 - \mathcal{G}), \quad (3.12)$$

where $\mathbf{e}_1 = [1, 0, \dots, 0]^T$ is the first standard basis vector in $\mathbb{R}^n$, and β is a parameter to be determined. The modified semi-discrete system becomes⁴

$$\frac{d\mathbf{u}}{dt} = -aD\mathbf{u} + \mathbf{SAT}_L. \quad (3.13)$$

Carrying out the same energy analysis yields

$$\begin{aligned} \frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 &= -a(u_n^2 - u_1^2) + 2\beta(u_1^2 - u_1\mathcal{G}) \\ &= -au_n^2 + (a + 2\beta)u_1^2 - 2\beta u_1\mathcal{G} \\ &= -au_n^2 + (a + 2\beta) \left(u_1 - \frac{\beta}{a + 2\beta} \mathcal{G} \right)^2 - \frac{\beta^2 \mathcal{G}^2}{a + 2\beta}. \end{aligned}$$

If $a + 2\beta < 0$, we obtain the stability estimate

$$\frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 \leq + \left| \frac{\beta^2}{a + 2\beta} \right| \mathcal{G}^2. \quad (3.14)$$

Integrating over $[0, t]$ gives

$$\|\mathbf{u}(t)\|_{\mathbf{H}}^2 \leq \|\mathbf{f}\|_{\mathbf{H}}^2 + \int_0^t \left| \frac{\beta^2}{a + 2\beta} \right| \mathcal{G}^2(\tau) d\tau, \quad t \geq 0.$$

which mirrors the continuous bound (2.13).⁵ The semi-discrete system (3.13) can be recast as

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u} + \mathbf{f}(t),$$

with system matrix $\mathbf{A} = -a\mathbf{H}^{-1} (\mathbf{Q} + \mathbf{e}_1 \mathbf{e}_1^T)$. The inequality (3.14) implies that all the eigenvalues of $\mathbf{A}$ have non-positive real parts.

In contrast to strong imposition of boundary conditions, SATs enforce boundary conditions **weakly**. The boundary condition is penalised through an added term that drives the numerical solution toward the correct boundary value. As a result, the boundary condition is satisfied to within the truncation error of the scheme. Upon mesh refinement, the numerical solution at the boundary converges to the prescribed inflow condition $\mathcal{G}(t)$ at the same rate as the interior solution converges to the exact PDE solution [HDRFZ25]. In practice, it is rarely necessary for boundary values to be more accurate than the solution in the interior [HDRFZ25]. What is essential is that the scheme remain stable and convergent under refinement. The derivation of SATs mirrors the continuous setting: the energy method is first applied to the PDE to identify data (boundary and initial conditions, forcing terms) that yield a well-posed problem. The same argument is then carried over to the semi-discrete equations, prescribing the form of the SATs and fixing their parameters (such as β in (3.12)) so that the discrete energy estimate matches the continuous one, ensuring stability.

³ The name “simultaneous approximation term” reflects that, at boundary nodes, both the PDE and the boundary condition are approximated simultaneously.

⁴ In vector form, (3.12) is

$$\mathbf{SAT}_L = \left[\frac{\beta}{\mathbf{H}_{11}} (u_1 - \mathcal{G}), 0, \dots, 0 \right]^T.$$

⁵ The choice $\beta = -a$ recovers the estimate (2.13) exactly.

4

Generalised Summation-by-Parts Operators

In the previous chapter, we introduced classical SBP operators, originally proposed by Kreiss and Scherer [KS74], and reviewed their role in mimicking integration by parts at the discrete level. Since their introduction, many extensions have been developed. In this chapter, we generalise the SBP framework to multidimensional elements, including simplices and curvilinear domains. Multidimensional SBP operators on general elements were first introduced in [HDRFZ16], providing a foundation for discretisations beyond grids with tensor-product elements. We begin by defining generalised SBP operators and outlining their construction on arbitrary elements. We then extend the framework to curvilinear elements, where geometric mappings and metric identities play a crucial role, and finally derive energy-stable discretisations of the linear advection equation using the SBP-SAT methodology. Much of the material in this chapter is drawn from [CHDRF⁺18, HDRFZ25].

4.1 Notation and Definitions

We adopt the following conventions for notation based on the notation in [HDRFZ16, GKNÖ23].

Definition 4.1. The operator $\hat{D}_{\xi_i} \in \mathbb{R}^{n \times n}$ of degree p is said to be a **SBP operator** for the first-derivative $\frac{\partial}{\partial \xi_i}$ on the nodal set $S_{\hat{\Omega}} = \{\vec{\xi}_i\}_{j=1}^n$ of a reference element with domain $\hat{\Omega} \subset \mathbb{R}^d$ if $\hat{D}_{\xi_i}, \hat{Q}_{\xi_i}, \hat{E}_{\xi_i}$, and $\hat{H} \in \mathbb{R}^{n \times n}$ satisfy the following conditions:¹

1. $[\hat{D}_{\xi_i} \mathbf{p}]_j = \frac{\partial \mathcal{P}}{\partial \xi_i}(\vec{\xi}_j)$ for all $\mathcal{P} \in \mathbb{P}_p(\hat{\Omega}), j = \{1, \dots, n\}$;
2. $\hat{D}_{\xi_i} = \hat{H}^{-1} \hat{Q}_{\xi_i}$, where $\hat{H} = \hat{H}^T$ and $\mathbf{x}^T \hat{H} \mathbf{x} > 0 \ \forall \mathbf{x} \neq \mathbf{0}$; and

¹ Here, $\mathbb{P}_r(\hat{\Omega})$ denotes the space of polynomials of maximum total degree r on $\hat{\Omega}$ defined as

$$\mathbb{P}_r(\hat{\Omega}) := \text{span}\{\xi_1^{i_1} \xi_2^{i_2} \dots \xi_d^{i_d} : \mathbf{i} \in \mathcal{N}(r)\},$$

where $\mathcal{N}(r) := \{\mathbf{i} \in \mathbb{N}_0^d : i_1 + \dots + i_d \leq r\}$.

3. $\hat{Q}_{\xi_i} = \hat{S}_{\xi_i} + \frac{1}{2}\hat{E}_{\xi_i}$, where $\hat{S}_{\xi_i}^T = -\hat{S}_{\xi_i}$, $\hat{E}_{\xi_i}^T = \hat{E}_{\xi_i}$, and $\hat{E}_{\xi_i}$ satisfies:

$$\mathbf{q}^T \hat{E}_{\xi_i} \mathbf{p} = \int_{\partial\hat{\Omega}} \mathcal{P} \mathcal{Q} n_{\xi_i} d\Gamma, \quad \forall \mathcal{P}, \mathcal{Q} \in \mathbb{P}_{r \geq p}(\hat{\Omega}), \quad (4.1)$$

where n_{ξ_i} is the component of the outward-pointing normal unit vector in the ξ_i -direction.

As before, the matrix $\hat{H}$ is the **norm** (or **mass**) matrix, which also defines a discrete inner product.² The entries of $\hat{H}$, defined over the nodal set $S_{\hat{\Omega}}$, constitute a degree- $q \geq 2p - 1$ **cubature** rule, which satisfies:³

$$\mathbf{1}^T \hat{H} \mathbf{p} = \int_{\hat{\Omega}} \mathcal{P} d\hat{\Omega} \quad \forall \mathcal{P} \in \mathbb{P}_{q \geq 2p-1}(\hat{\Omega}).$$

Equation (4.1) requires that the symmetric matrix $\hat{E}_{\xi_i} \in \mathbb{R}^{n \times n}$ approximates surface integrals in the ξ_i -direction. All three conditions in Definition 4.1 together imply that the SBP operators discretely mimic integration by parts:

$$\underbrace{\int_{\hat{\Omega}} \mathcal{V} \frac{\partial \mathcal{U}}{\partial \xi_i} d\hat{\Omega}}_{\mathbf{v}^T \hat{Q}_{\xi_i} \mathbf{u}} + \underbrace{\int_{\hat{\Omega}} \mathcal{U} \frac{\partial \mathcal{V}}{\partial \xi_i} d\hat{\Omega}}_{\mathbf{v}^T \hat{Q}_{\xi_i}^T \mathbf{u}} = \underbrace{\int_{\partial\hat{\Omega}} \mathcal{V} \mathcal{U} n_{\xi_i} d\Gamma}_{\mathbf{v}^T \hat{E}_{\xi_i} \mathbf{u}} \quad \forall i \in \{1, \dots, d\}. \quad (4.2)$$

This discrete relationship holds exactly for vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ corresponding to any functions $\mathcal{U}, \mathcal{V}$ that are square-integrable and have square-integrable weak first derivatives.⁴

² We restrict attention to diagonal-norm SBP operators, for which $\hat{H}$ is diagonal with strictly positive diagonal entries.

³ Throughout this thesis, we use the notation $\mathbf{1} \in \mathbb{R}^n$ to denote the vector of all ones.

⁴ i.e. $\mathcal{U}, \mathcal{V} \in H^1(\hat{\Omega})$.

Matrices for Face-based Operations

To weakly impose boundary conditions and couple adjacent elements, we use SATs [CGA94, DRFHZ18]. In the classical SBP setting, the nodal distribution includes points on the boundary $\partial\hat{\Omega}$, allowing SATs to be applied directly and pointwise between coincident nodes of neighbouring elements. In contrast, generalised SBP operators may not include boundary nodes. To address this, we adopt the approach of [DRFHZ18], which decomposes the boundary operator $\hat{E}_{\xi_i}$ as a sum over contributions associated with each non-overlapping facet $\gamma \in \partial\hat{\Omega}$ of the reference element:

$$\hat{E}_{\xi_i} = \sum_{\gamma \in \partial\hat{\Omega}} \hat{E}_{\xi_i}^\gamma = \sum_{\gamma \in \partial\hat{\Omega}} \mathbf{R}_\gamma^T \hat{\mathbf{N}}_\gamma \hat{\mathbf{B}}_\gamma \mathbf{R}_\gamma, \quad (4.3)$$

where:

- $\mathbf{R}_\gamma \in \mathbb{R}^{n_\gamma \times n}$ is an interpolation/extrapolation operator that maps functions from the volume nodal set $S_{\hat{\Omega}}$ to the facet quadrature nodes $S_\gamma = \{\vec{\xi}_{\gamma,j}\}_{j=1}^{n_\gamma}$,
- $\hat{\mathbf{B}}_\gamma \in \mathbb{R}^{n_\gamma \times n_\gamma}$ is a diagonal facet norm matrix that holds the quadrature weights on facet γ ,

- $\hat{\mathbf{N}}_\gamma = \text{diag}[(n_{\xi_i})_1, (n_{\xi_i})_2, \dots, (n_{\xi_i})_{n_\gamma}]$ is a diagonal matrix whose entries are the ξ_i -component of the outward-pointing unit normal vector to facet γ .

To satisfy the SBP property in this setting, we require that:

$$[\mathbf{R}_\gamma \mathbf{p}]_j = [\mathbf{p}_\gamma]_j = \mathcal{P}(\vec{\xi}_{\gamma,j}), \quad \forall \mathcal{P} \in \mathbb{P}_{r \geq p}(\hat{\Omega}), \quad j = \{1, \dots, n_\gamma\}, \quad (4.4)$$

$$\mathbf{1}^T \hat{\mathbf{B}}_\gamma \mathbf{p}_\gamma = \int_{\partial \hat{\Omega}_\gamma} \mathcal{P} \, d\Gamma, \quad \forall \mathcal{P} \in \mathbb{P}_{s \geq 2r}(\partial \hat{\Omega}_\gamma). \quad (4.5)$$

4.2 Construction of Generalised SBP Operators

Let n_r denote the number of monomials in the degree r polynomial space.⁵ We assume that the nodal set $S_{\hat{\Omega}}$ yields a full-column-rank degree- r Vandermonde matrix $\hat{\mathbf{V}}_{\hat{\Omega}} \in \mathbb{R}^{n \times n_r}$ to ensure we maintain unisolvency [MZ22]. Following this setup, the boundary operator can be constructed as follows.⁶

⁵ In d spatial dimensions, this is given by

$$n_r = \binom{r+d}{d}.$$

⁶ Here, $\dagger$ denotes the Moore–Penrose pseudoinverse: $\mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$.

Construction of Generalised SBP Operators

1. Choose cubature rules:

- Select a volume cubature of degree $q \geq 2p - 1$ to define $\hat{\mathbf{H}}$ and nodal set $S_{\hat{\Omega}}$.
- Select a facet quadrature of degree $s \geq 2r$ to define $\hat{\mathbf{B}}_\gamma$ and S_γ for each facet γ .

2. Construct Vandermonde matrices:

- Let $\hat{\mathbf{V}}_{\hat{\Omega}} \in \mathbb{R}^{n \times n_r}$ be the degree- r Vandermonde matrix on $S_{\hat{\Omega}}$.
- Let $\hat{\mathbf{V}}_\gamma \in \mathbb{R}^{n_\gamma \times n_r}$ be the degree- r Vandermonde matrix on S_γ .

3. Define interpolation/extrapolation operator:

$$\mathbf{R}_\gamma := \hat{\mathbf{V}}_\gamma \hat{\mathbf{V}}_{\hat{\Omega}}^\dagger \in \mathbb{R}^{n_\gamma \times n}. \quad (4.6)$$

By construction, the operator $\mathbf{R}_\gamma$ satisfies the interpolation condition in (4.4).

4. Assemble the boundary operator:

$$\hat{\mathbf{E}}_{\xi_i} := \sum_{\gamma \in \partial \hat{\Omega}} \mathbf{R}_\gamma^T \hat{\mathbf{N}}_\gamma \hat{\mathbf{B}}_\gamma \mathbf{R}_\gamma.$$

5. Use Theorem 2 from [DRFHZ18] to compute:

- The discrete derivative operator $\hat{\mathbf{D}}_{\xi_i}$,
- The skew-symmetric matrix $\hat{\mathbf{Q}}_{\xi_i}$.

Diagonal-E SBP Operators

In Figure 4.1, we illustrate an SBP operator where the volume cubature nodes and facet quadrature nodes are collocated. For this special class of operators, the extrapolation operator R_γ becomes a **restriction matrix** that simply selects the volume cubature nodes lying on the facet. Specifically,

$$[R_\gamma]_{ij} = \delta_{ij}, \quad \forall i \in \{1, \dots, n\}, \quad j \in \{1, \dots, n_\gamma\}.$$

Substituting this into (4.3), we obtain a diagonal-E matrix whose nonzero entries are confined to the degrees of freedom on facet γ . Since the facet contributions are non-overlapping, the sum remains diagonal. Thus, the resulting boundary operator $\hat{E}_{\xi_i}$ is diagonal. SBP operators constructed in this manner are referred to as **diagonal-E SBP operators** [CS17]. Moreover, both $\hat{E}_{\xi_i}$ and $\hat{B}_\gamma$ are sparse matrices, enabling efficient evaluation of surface integrals.

Adjacent elements may be coupled either through **conforming** interfaces, where facet nodes coincide (Figure 4.2), or through **non-conforming** interfaces, where nodal sets do not align. In the latter case, mortar techniques can be employed to mediate the coupling (Figure 4.3). In this thesis we restrict attention to conforming discretisations, and refer the reader to [SZ20] for a detailed treatment of non-conforming SBP operators.

For conforming interfaces, the facet nodal sets $S_{K\gamma}$ and $S_{V\gamma}$ of neighbouring elements κ and ν must satisfy the following conditions:

1. Each interface node $\vec{\xi}_{K\gamma,i} \in S_{K\gamma}$ is paired with exactly one node $\vec{\xi}_{V\gamma,j} \in S_{V\gamma}$ such that $\vec{\xi}_{K\gamma,i} = \vec{\xi}_{V\gamma,j}$.
2. If $\vec{\xi}_i$ is a node shared by both $S_{K\gamma}$ and $S_{V\gamma}$, then the corresponding entries of the facet norm matrices agree:

$$[\hat{B}_{K,\gamma}]_{ii} = [\hat{B}_{V,\gamma}]_{jj}.$$

4.3 Curvilinear Geometric Transformation

Definition 4.2 (Curvilinear Coordinates). Let $P \in \mathbb{R}^n$ be a point represented in two coordinate systems:

1. In the original coordinate system $(\xi_1, \xi_2, \dots, \xi_n)$, the point is given by $P = (\xi_1, \xi_2, \dots, \xi_n)$.
2. In a new coordinate system $(x_1, x_2, \dots, x_n)$, the same point is expressed as $P = (x_1, x_2, \dots, x_n)$, where each x_i is a function of the original coordinates:

$$x_i = x_i(\xi_1, \xi_2, \dots, \xi_n), \quad \text{for } i = 1, \dots, n.$$

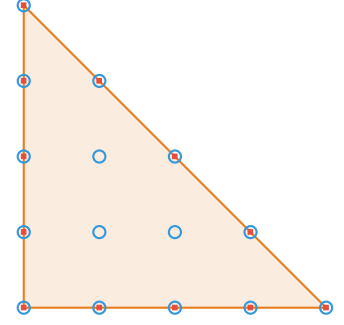


Figure 4.1: Collocated grid points on a triangle. Blue circles indicate the grid points, while red squares indicate the boundary points.

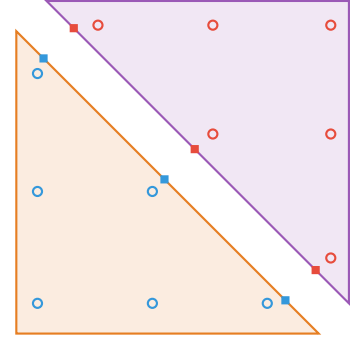


Figure 4.2: Conforming interface.

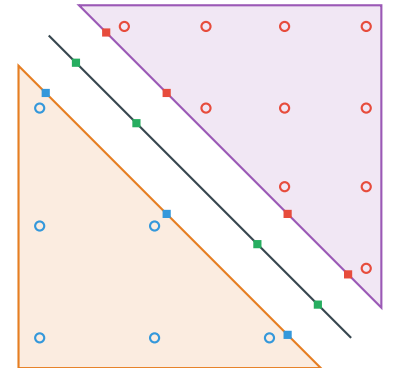


Figure 4.3: Non-conforming interface.

If each function $x_i(\xi_1, \xi_2, \dots, \xi_n)$ is real-valued, twice continuously differentiable, and invertible, then the map

$$\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad \mathbf{F}(\vec{\xi}) = (x_1, \dots, x_n),$$

is called a **coordinate transformation**. When the original coordinates $(\xi_1, \dots, \xi_n)$ are Cartesian:

- the transformation is **affine** if $\mathbf{F}$ is linear,
- and **curvilinear** if $\mathbf{F}$ is nonlinear.

The Jacobian matrix of the map $\mathbf{F}$ plays a crucial role in the transformation of coordinates: its determinant governs whether the mapping is locally invertible and ensures that derivatives can be consistently transferred between coordinate systems. This fact is made precise by the Inverse Function Theorem [HH15, Theorem 2.10.7].

Theorem 4.1 (Inverse Function Theorem). *Let $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be continuously differentiable on an open set $U \subset \mathbb{R}^n$ containing a point $\vec{a}$, and suppose that $\det \mathbf{J}_{\mathbf{F}}(\vec{a}) \neq 0$. Then, there exists an open neighbourhood $V \subset \mathbb{R}^n$ of $\vec{a}$ and an open neighbourhood $W \subset \mathbb{R}^n$ of $\mathbf{F}(\vec{a})$ such that $\mathbf{F} : V \rightarrow W$ is a bijection with a continuously differentiable inverse $\mathbf{F}^{-1} : W \rightarrow V$. Moreover, for all $\vec{y} \in W$, the Jacobian matrix of the inverse satisfies*

$$\mathbf{J}_{\mathbf{F}^{-1}}(\vec{y}) = \left[\mathbf{J}_{\mathbf{F}}(\mathbf{F}^{-1}(\vec{y})) \right]^{-1},$$

where $\mathbf{J}_{\mathbf{F}}(\vec{x})$ denotes the Jacobian matrix of $\mathbf{F}$ evaluated at $\vec{x}$.

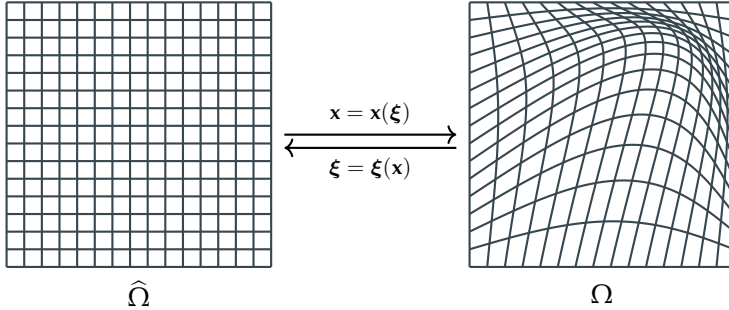


Figure 4.4: Mesh Ω generated as the image of $\hat{\Omega}$ under a curvilinear coordinate transformation.

Consider a smooth, invertible mapping from the computational domain $\hat{\Omega}$ to the physical domain Ω in three dimensions, given by⁷

$$\begin{aligned} x &= x(\xi, \eta, \zeta), & y &= y(\xi, \eta, \zeta), & z &= z(\xi, \eta, \zeta), \\ \xi &= \xi(x, y, z), & \eta &= \eta(x, y, z), & \zeta &= \zeta(x, y, z), \end{aligned}$$

where $\vec{\xi} = [\xi, \eta, \zeta]^T$ are the computational coordinates and $\vec{x} = [x, y, z]^T$ are the physical coordinates. An example of such a mapping is shown in Figure 4.4.⁸ The Jacobian matrix and its determinant (often referred to as the

⁷ For simplicity, we assume $\hat{\Omega} \subseteq \mathbb{R}^3$ with coordinates (ξ, η, ζ) .

⁸ The coordinate transformation is given by:

$$\begin{aligned} x(\xi, \eta) &= \xi + \frac{1}{5} \sin(\pi\xi) \sin(\pi\eta), \\ y(\xi, \eta) &= \eta + \frac{1}{5} \exp(1 - \eta) \sin(\pi\xi) \sin(\pi\eta). \end{aligned}$$

Jacobian) are denoted by:

$$\mathbf{J} = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}, \quad \mathcal{J} = \det(\mathbf{J}).$$

Similarly, the Jacobian of the inverse mapping is given by:⁹

$$\mathbf{J}^{-1} = \frac{\partial \vec{\xi}}{\partial \vec{x}} = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{bmatrix}.$$

The columns of $\mathbf{J}$ are known as the **covariant** basis vectors, denoted by subscripts:¹⁰

$$\vec{a}_\xi = \begin{bmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \\ \frac{\partial z}{\partial \xi} \end{bmatrix}, \quad \vec{a}_\eta = \begin{bmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \\ \frac{\partial z}{\partial \eta} \end{bmatrix}, \quad \vec{a}_\zeta = \begin{bmatrix} \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \zeta} \end{bmatrix}.$$

Similarly, the columns of $\mathbf{J}^{-1}$ form the **contravariant** basis vectors, denoted with superscripts:

$$\vec{a}^\xi = \begin{bmatrix} \frac{\partial \xi}{\partial x} \\ \frac{\partial \xi}{\partial y} \\ \frac{\partial \xi}{\partial z} \end{bmatrix}, \quad \vec{a}^\eta = \begin{bmatrix} \frac{\partial \eta}{\partial x} \\ \frac{\partial \eta}{\partial y} \\ \frac{\partial \eta}{\partial z} \end{bmatrix}, \quad \vec{a}^\zeta = \begin{bmatrix} \frac{\partial \zeta}{\partial x} \\ \frac{\partial \zeta}{\partial y} \\ \frac{\partial \zeta}{\partial z} \end{bmatrix}.$$

Derivative and Metric Invariance

We temporarily adopt index notation and define the following mappings between coordinate indices and their respective symbols:

$$(x_1, x_2, x_3) := (x, y, z), \\ (\xi_1, \xi_2, \xi_3) := (\xi, \eta, \zeta).$$

The covariant and contravariant basis vectors are denoted as $\vec{a}_i$ and $\vec{a}^i$, respectively. These satisfy the following **cyclic identities** for any cyclic permutation $(i, j, k) \in \{(1, 2, 3), (2, 3, 1), (3, 1, 2)\}$:¹¹

$$\vec{a}^i = \frac{1}{\mathcal{J}} \vec{a}_j \times \vec{a}_k, \quad \vec{a}_i = \mathcal{J} \vec{a}^j \times \vec{a}^k, \quad \vec{a}_i \cdot \vec{a}^j = \delta_i^j.$$

To express derivatives with respect to physical coordinates in terms of computational coordinates, we begin with the chain rule:

$$\frac{\partial \mathcal{U}}{\partial x_m} = \sum_{i=1}^d \frac{\partial \mathcal{U}}{\partial \xi_i} \cdot \frac{\partial \xi_i}{\partial x_m} \implies \mathcal{J} \frac{\partial \mathcal{U}}{\partial x_m} = \sum_{i=1}^d \mathcal{J} \frac{\partial \xi_i}{\partial x_m} \frac{\partial \mathcal{U}}{\partial \xi_i}. \quad (4.7)$$

We can rewrite the right-hand side of (4.7) using the product rule to obtain:¹²

⁹ By the inverse function theorem and the chain rule:

$$\frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} \cdot \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \mathbf{1}.$$

¹⁰ This notation allows us to compactly express the Jacobian and its inverse as:

$$\mathbf{J} = [\vec{a}_\xi, \vec{a}_\eta, \vec{a}_\zeta], \quad \mathbf{J}^{-1} = \begin{bmatrix} (\vec{a}^\xi)^T \\ (\vec{a}^\eta)^T \\ (\vec{a}^\zeta)^T \end{bmatrix}.$$

¹¹ These identities follow from the adjugate formula for matrix inversion:

$$\mathbf{J}^{-1} = \frac{\text{adj}(\mathbf{J})}{\mathcal{J}} = \frac{1}{\mathcal{J}} [\vec{a}_2 \times \vec{a}_3, \vec{a}_3 \times \vec{a}_1, \vec{a}_1 \times \vec{a}_2]^T,$$

and from the identity $\mathcal{J} = \vec{a}_1 \cdot (\vec{a}_2 \times \vec{a}_3)$.

¹²

$$\frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} \right) = \mathcal{J} \frac{\partial \mathcal{U}}{\partial \xi_i} \frac{\partial \xi_i}{\partial x_m} + \mathcal{U} \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \right)$$

$$\mathcal{J} \frac{\partial \mathcal{U}}{\partial x_m} = \sum_{i=1}^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} \right) - \mathcal{U} \sum_{i=1}^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \right).$$

The remaining term in the second summation vanishes due to a geometric identity. Specifically, we can express it in vector form as:

$$\begin{aligned} \sum_{i=1}^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \begin{bmatrix} \frac{\partial \xi_i}{\partial x_1} \\ \frac{\partial \xi_i}{\partial x_2} \\ \frac{\partial \xi_i}{\partial x_3} \end{bmatrix} \right) &= \sum_{i=1}^3 \frac{\partial}{\partial \xi_i} (\mathcal{J} \vec{a}^i) \\ &= \sum_{i=1}^3 \frac{\partial}{\partial \xi_i} (\vec{a}_j \times \vec{a}_k) \\ &= \sum_{i=1}^3 \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} \times \frac{\partial \vec{x}}{\partial \xi_k} + \sum_{i=1}^3 \frac{\partial \vec{x}}{\partial \xi_j} \times \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_k} \\ &= \sum_{i=1}^3 \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} \times \frac{\partial \vec{x}}{\partial \xi_k} - \sum_{i=1}^3 \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_k} \times \frac{\partial \vec{x}}{\partial \xi_j} \\ &= \sum_{i=1}^3 \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} \times \frac{\partial \vec{x}}{\partial \xi_k} - \sum_{i=1}^3 \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} \times \frac{\partial \vec{x}}{\partial \xi_k} \\ &= 0. \end{aligned}$$

The derivative of a cross product obeys the product rule:

$$\frac{\partial}{\partial \xi} (\vec{u} \times \vec{v}) = \frac{\partial \vec{u}}{\partial \xi} \times \vec{v} + \vec{u} \times \frac{\partial \vec{v}}{\partial \xi}.$$

In the last step we simply relabel dummy indices. Since (i, j, k) cycles over $(1, 2, 3)$, $(2, 3, 1)$, and $(3, 1, 2)$, we can replace $(i, j, k) \mapsto (j, k, i)$ in the second term without changing the sum.

This is known as the **metric invariant identity**, and it ensures that the second term vanishes identically. We are then left with the expression:

$$\frac{\partial \mathcal{U}}{\partial x_m} = \frac{1}{\mathcal{J}} \sum_{i=1}^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} \right). \quad (4.8)$$

Equation (4.8) is referred to as the **strong-conservative form** of the Cartesian derivative. This formulation is particularly useful when deriving conservative discretisations of PDEs.

Volume and Surface Integrals

Just as we transformed derivatives from the physical domain to the computational domain, we now do the same for volume and surface integrals. We begin by integrating the metric invariant identity (4.3). This yields:

$$\begin{aligned} 0 &= \sum_i^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \right) \\ &= \int_{\hat{\Omega}} \sum_i^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \right) d\hat{\Omega} \\ &= \sum_i^d \oint_{\hat{\Gamma}} \mathcal{J} \frac{\partial \xi_i}{\partial x_m} n_{\xi_i} d\hat{\Gamma}, \end{aligned}$$

which is known as the **surface metric invariant identity**.

We can now relate integrals over the physical domain Ω to integrals over the reference domain $\widehat{\Omega}$ using the volume Jacobian $\mathcal{J}$. Specifically,

$$\int_{\Omega} \mathcal{U} d\Omega = \int_{\widehat{\Omega}} \mathcal{U} \mathcal{J} d\widehat{\Omega}, \quad \int_{\Omega} \mathcal{U} \mathcal{V} d\Omega = \int_{\widehat{\Omega}} \mathcal{U} \mathcal{V} \mathcal{J} d\widehat{\Omega}.$$

Integrating the left-hand side of the conservative form (4.8) over the physical domain yields:

$$\begin{aligned} \int_{\Omega} \frac{\partial \mathcal{U}}{\partial x_m} d\Omega &= \oint_{\Gamma} \mathcal{U} n_{x_m} d\Gamma \\ &= \oint_{\widehat{\Gamma}} \mathcal{U} n_{x_m} \mathcal{J}_s d\widehat{\Gamma} \end{aligned} \quad (4.9)$$

where n_{x_m} denotes the m -th component of the outward-pointing normal vector to the boundary Γ in physical coordinates, and $\mathcal{J}_s$ is the **surface metric Jacobian** associated with the transformation. Similarly, integrating the right-hand side of (4.8) gives:

$$\begin{aligned} \int_{\Omega} \frac{1}{\mathcal{J}} \sum_i^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} \right) d\Omega &= \int_{\widehat{\Omega}} \frac{1}{\mathcal{J}} \sum_i^d \frac{\partial}{\partial \xi_i} \left(\mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} \right) \mathcal{J} d\widehat{\Omega} \\ &= \oint_{\widehat{\Gamma}} \sum_i^d \mathcal{J} \frac{\partial \xi_i}{\partial x_m} \mathcal{U} n_{\xi_i} d\widehat{\Gamma}. \end{aligned} \quad (4.10)$$

Combining (4.9) and (4.10), we obtain:

$$\oint_{\widehat{\Gamma}} \mathcal{U} n_{x_m} \mathcal{J}_s d\widehat{\Gamma} = \oint_{\widehat{\Gamma}} \sum_i^d \frac{\partial \xi_i}{\partial x_m} \mathcal{U} n_{\xi_i} \mathcal{J} d\widehat{\Gamma}.$$

Let $\widehat{\Gamma}$ be a $(d-1)$ -dimensional face of the reference element and define

$$E_m(\widehat{\xi}) := n_{x_m}(\widehat{\xi}) \mathcal{J}_s(\widehat{\xi}) - \sum_{i=1}^d \frac{\partial \xi_i}{\partial x_m}(\widehat{\xi}) n_{\xi_i}(\widehat{\xi}) \mathcal{J}(\widehat{\xi}).$$

E_m is continuous on $\widehat{\Gamma}$ since we assume that the mapping $\mathbf{F}$ is smooth. From the fundamental lemma of the calculus of variations, we have that if

$$\int_{\widehat{\Gamma}} \mathcal{U}(\widehat{\xi}) E_m(\widehat{\xi}) d\widehat{\Gamma} = 0 \quad \text{for all } \mathcal{U} \in C_c^\infty(\widehat{\Gamma}),$$

then E_m is identically zero on $\widehat{\Gamma}$. Thus, we have that

$$n_{x_m} \mathcal{J}_s = \sum_i^d \frac{\partial \xi_i}{\partial x_m} n_{\xi_i} \mathcal{J} \quad \text{for } m = 1, \dots, d. \quad (4.11)$$

Equation (4.11) relates the components of the physical normal vector $\vec{n}$ to those of the reference normal vector $\vec{\tilde{n}}$. It is often referred to as **Nanson's formula** and it plays a crucial role in the transformation of surface integrals and fluxes between computational and physical domains [ZN16, Ode13]. We

can express (4.11) in vector form as:

$$\begin{bmatrix} n_{x_1} \\ n_{x_2} \\ n_{x_3} \end{bmatrix} \mathcal{J}_s = \mathcal{J} \begin{bmatrix} \frac{\partial \xi_1}{\partial x_1} & \frac{\partial \xi_2}{\partial x_1} & \frac{\partial \xi_3}{\partial x_1} \\ \frac{\partial \xi_1}{\partial x_2} & \frac{\partial \xi_2}{\partial x_2} & \frac{\partial \xi_3}{\partial x_2} \\ \frac{\partial \xi_1}{\partial x_3} & \frac{\partial \xi_2}{\partial x_3} & \frac{\partial \xi_3}{\partial x_3} \end{bmatrix} \begin{bmatrix} n_{\xi_1} \\ n_{\xi_2} \\ n_{\xi_3} \end{bmatrix} \quad (4.12)$$

$$\therefore \vec{n} \mathcal{J}_s = \mathcal{J} \mathbf{J}^{-T} \vec{n}.$$

The physical normal $\vec{n}$ is usually taken to have unit length, which implies

$$\|\vec{n}\| = \left\| \frac{\mathcal{J}}{\mathcal{J}_s} \mathbf{J}^{-T} \vec{n} \right\| = 1 \implies \mathcal{J}_s = \left\| \mathcal{J} \mathbf{J}^{-T} \vec{n} \right\|.$$

Substituting back into (4.12), we obtain:¹³

$$\vec{n} = \frac{\mathbf{J}^{-T} \vec{n}}{\left\| \mathbf{J}^{-T} \vec{n} \right\|} \quad (4.13)$$

¹³ Both $\mathcal{J}$ and $\mathcal{J}_s$ are assumed to be positive by construction.

which is the most common form of Nanson's formula found in the literature.

4.4 Summation-by-Parts Operators on Curved Elements

When curved geometries are approximated using linear meshes, the global solution accuracy may degrade to second order, irrespective of the formal order of the discretisation scheme. To fully leverage the accuracy of high-order methods in complex domains, it is essential to use geometrically curved elements. This is particularly important when capturing intricate boundary features or flow structures that are not well-resolved by linear elements [OBB24]. In this section, we outline the construction of SBP volume and interface operators on curved elements, starting from their definitions on a reference element. Our exposition is based on the framework developed in [CHDRF⁺18]. To ensure conservation, freestream preservation, stability, and design-order accuracy, the SBP operators must satisfy several essential conditions: functions must be exactly extrapolated and differentiated on the reference element, and the resulting physical operators must retain the **design order of accuracy** i.e. the approximation of physical derivatives, constructed from degree- p SBP operators on the reference element, must have truncation error of order p [CHDRF⁺18, Section 4].

Let $\Omega \subset \mathbb{R}^d$ be the physical domain of interest, where d denotes the spatial dimension. We partition Ω into n_e non-overlapping (possibly curved) elements:

$$\mathcal{T}_h = \left\{ \{ \Omega_\kappa \}_{\kappa=1}^{n_e} \mid \Omega = \bigcup_{\kappa=1}^{n_e} \bar{\Omega}_\kappa, \quad \Omega_\kappa \cap \Omega_\nu = \emptyset \text{ for } \kappa \neq \nu \right\},$$

where $\bar{\Omega}_\kappa = \Omega_\kappa \cup \partial\Omega_\kappa$ denotes the closure of element Ω_κ . We denote the set of internal interfaces (i.e., shared facets between elements) by Γ^I , defined as:

$$\Gamma^I = \{\partial\Omega_\kappa \cap \partial\Omega_\nu \mid \kappa, \nu \in \{1, \dots, n_e\}, \kappa \neq \nu\}.$$

Throughout, we assume that the boundary $\partial\Omega_\kappa$ of each element is piecewise smooth.

Assumption 4.1. *We assume that valid SBP volume and interface operators satisfying the conditions of Definition 4.1 are defined on the reference element $\hat{\Omega}_\kappa$. Furthermore, we assume the existence of a bijective polynomial mapping*

$$\Phi_\kappa : \hat{\Omega}_\kappa \rightarrow \Omega_\kappa,$$

of degree at most $q \leq p_{\min} + 1$, where $p_{\min} = \min_{\Omega_\kappa \in \mathcal{T}_h} \{p_\kappa\}$. This map transforms any point $\vec{\xi} \in \hat{\Omega}_\kappa$ to its physical counterpart $\vec{x} \in \Omega_\kappa$, i.e.,¹⁴

$$\vec{x} = \Phi_\kappa(\vec{\xi}).$$

¹⁴ A bijective polynomial mapping has a Jacobian determinant that is either strictly positive or strictly negative throughout the element. We assume it is **positive** to ensure that the mapping preserves orientation.

Construction of Boundary Operators

The scaled face normal components that appear along the diagonal of $N_{x_i, \gamma}$ are computed as:

$$[N_{x_i, \gamma}]_{jj} = \left(\sum_{\alpha} \mathcal{J} \frac{\partial \alpha}{\partial x_i} n_\alpha \right)_{(\xi_j, \eta_j, \zeta_j)} \quad \text{for } \alpha \in \{\xi, \eta, \zeta\}, x_i \in \{x, y, z\},$$

$$\text{and } j = 1, \dots, n_\gamma. \quad (4.14)$$

The components describe the physical face normal directions, which are constructed by scaling the reference-space normals by the metric terms. The physical boundary operator E_{x_i} is then assembled by summing contributions from each face $\gamma \in \Gamma_\kappa$ of the element Ω_κ . The decomposition is given by [CHDRF⁺18]:¹⁵

$$E_{x_i} = \sum_{\gamma \in \Gamma_\kappa} E_{x_i}^{\kappa\gamma} = \sum_{\gamma \in \Gamma_\kappa} R_{\kappa\gamma}^T N_{x_i, \gamma} \hat{B}_\gamma R_{\kappa\gamma} \quad (4.15)$$

¹⁵ Note that the reference-space extrapolation operators $R_{\kappa\gamma}$ are reused directly on the physical element without modification.

Proposition 4.2 (Discrete Geometric Divergence [CHDRF⁺18]). *If the conditions of Assumption 4.1 are satisfied, then the discrete boundary operator E_{x_i} satisfies the discrete geometric divergence condition:*

$$\mathbf{1}^T E_{x_i} \mathbf{1} = \int_{\partial\Omega_\kappa} n_{x_i} d\Gamma = 0.$$

Two-Dimensional Curvilinear SBP Operators

We now describe the construction of SBP operators $D_{x_i} = H^{-1}Q_{x_i}$ where $Q_{x_i} = S_{x_i} + \frac{1}{2}E_{x_i}$ on a two-dimensional curvilinear element. The norm

matrix on the physical element is defined as a diagonal scaling of the reference norm matrix:

$$\mathbf{H} = \begin{bmatrix} \mathcal{J}(\xi_1, \eta_1) & 0 & \cdots & 0 \\ 0 & \mathcal{J}(\xi_2, \eta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{J}(\xi_{n_k}, \eta_{n_k}) \end{bmatrix} \begin{bmatrix} \hat{\mathbf{H}} \end{bmatrix}. \quad (4.16)$$

We then construct $\mathbf{Q}_{x_i} \forall x_i \in \{x, y\}$ as $\mathbf{Q}_{x_i} = \mathbf{S}_{x_i} + \frac{1}{2}\mathbf{E}_{x_i}$, where $\mathbf{S}_{x_i}$ is skew-symmetric and defined as:

$$\mathbf{S}_{x_i} = \frac{1}{2} (\Lambda_{\xi, x_i} \mathbf{Q}_\xi + \Lambda_{\eta, x_i} \mathbf{Q}_\eta) - \frac{1}{2} (\mathbf{Q}_\xi^T \Lambda_{\xi, x_i} + \mathbf{Q}_\eta^T \Lambda_{\eta, x_i}). \quad (4.17)$$

Here, Λ_{ξ, x_i} and Λ_{η, x_i} are diagonal matrices of size $n_k \times n_k$ that encode the scaled metric terms:

$$\Lambda_{\xi, x_i} = \begin{bmatrix} \mathcal{J}(\xi_1, \eta_1) \frac{\partial \xi}{\partial x_i}(\xi_1, \eta_1) & 0 & \cdots & 0 \\ 0 & \mathcal{J}(\xi_2, \eta_2) \frac{\partial \xi}{\partial x_i}(\xi_2, \eta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{J}(\xi_{n_k}, \eta_{n_k}) \frac{\partial \xi}{\partial x_i}(\xi_{n_k}, \eta_{n_k}) \end{bmatrix} \quad (4.18)$$

and similarly for Λ_{η, x_i} with $\partial \eta / \partial x_i$. Except for polynomial exactness, the operator $\mathbf{D}_{x_i}$ defined by (4.16), (4.17), and the physical boundary operator (4.15) satisfies the requirements of Definition 4.1: the matrix $\mathbf{E}_{x_i}$ is symmetric, $\mathbf{S}_{x_i}$ is skew-symmetric, and $\mathbf{H}$ is diagonal and positive definite. To ensure conservation, it is also required that $\mathbf{D}_{x_i} \mathbf{1} = 0$, which is established by the following theorem.

Theorem 4.3 (Theorem 6, [CHDRF⁺18]). *Let Assumption 4.1 hold, and let $\mathbf{D}_{x_i}$ be the SBP operator defined by (4.16), (4.17), and the associated boundary operator. Then, $\mathbf{D}_{x_i} \mathbf{1} = 0$ provided the metric matrices Λ_{ξ, x_i} and Λ_{η, x_i} satisfy the discrete geometric divergence-free condition:*

$$(\mathbf{D}_\xi \Lambda_{\xi, x_i} + \mathbf{D}_\eta \Lambda_{\eta, x_i}) \mathbf{1} = \hat{\mathbf{H}}^{-1} (\mathbf{E}_\xi \Lambda_{\xi, x_i} + \mathbf{E}_\eta \Lambda_{\eta, x_i} - \mathbf{E}_{x_i}) \mathbf{1}. \quad (4.19)$$

For two-dimensional problems, the diagonal matrices Λ_{ξ, x_i} and Λ_{η, x_i} storing the exact scaled metrics satisfy the discrete geometric divergence-free condition in (4.19) [CHDRF⁺18, Theorem 7].

Three-Dimensional Curvilinear SBP Operators

The construction of SBP operators $\mathbf{D}_{x_i}$ in three dimensions closely mirrors the two-dimensional case. The norm matrix $\mathbf{H}$ and the skew-symmetric

matrix S_{x_i} are defined as:

$$H = \begin{bmatrix} \mathcal{J}(\xi_1, \eta_1, \zeta_1) & 0 & \cdots & 0 \\ 0 & \mathcal{J}(\xi_2, \eta_2, \zeta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{J}(\xi_{n_k}, \eta_{n_k}, \zeta_{n_k}) \end{bmatrix} \begin{bmatrix} \\ \\ \hat{H} \\ \end{bmatrix},$$

$$\begin{aligned} S_{x_i} &= \frac{1}{2} (\Lambda_{\xi, x_i} Q_\xi + \Lambda_{\eta, x_i} Q_\eta + \Lambda_{\zeta, x_i} Q_\zeta) \\ &\quad - \frac{1}{2} (Q_\xi^T \Lambda_{\xi, x_i} + Q_\eta^T \Lambda_{\eta, x_i} + Q_\zeta^T \Lambda_{\zeta, x_i}). \end{aligned} \tag{4.20}$$

The diagonal metric matrices Λ_{ξ, x_i} , Λ_{η, x_i} , and Λ_{ζ, x_i} play the same structural role as in two dimensions. However, a key distinction is that in three dimensions, these matrices are not necessarily the **exact** scaled metric terms. Instead, they must be computed to satisfy a discrete version of the geometric invariance condition, ensuring that the operator D_{x_i} leads to conservative discretisations.

Theorem 4.4 (Theorem 8, [CHDRF⁺18]). *Let Assumption 4.1 hold, and define the SBP operator as*

$$D_{x_i} = H^{-1} Q_{x_i} \quad \text{where} \quad Q_{x_i} = S_{x_i} + \frac{1}{2} E_{x_i}.$$

Then, the condition $D_{x_i} \mathbf{1} = 0$ holds if the following discrete geometric invariance condition is satisfied:

$$(D_\xi \Lambda_{\xi, x_i} + D_\eta \Lambda_{\eta, x_i} + D_\zeta \Lambda_{\zeta, x_i}) \mathbf{1} = \hat{H}^{-1} (E_\xi \Lambda_{\xi, x_i} + E_\eta \Lambda_{\eta, x_i} + E_\zeta \Lambda_{\zeta, x_i} - E_{x_i}) \mathbf{1}. \tag{4.21}$$

Unlike the two-dimensional case, the metric invariant identity (4.21) does not hold automatically under the assumptions of Assumption 4.1. Instead, the metric terms must be solved explicitly to enforce this condition. Multiplying both sides of (4.21) by $\hat{H}$ yields:

$$(Q_\xi \Lambda_{\xi, x_i} + Q_\eta \Lambda_{\eta, x_i} + Q_\zeta \Lambda_{\zeta, x_i}) \mathbf{1} = (E_\xi \Lambda_{\xi, x_i} + E_\eta \Lambda_{\eta, x_i} + E_\zeta \Lambda_{\zeta, x_i} - E_{x_i}) \mathbf{1}.$$

Rearranging terms, we have:

$$((Q_\xi - E_\xi) \Lambda_{\xi, x_i} + (Q_\eta - E_\eta) \Lambda_{\eta, x_i} + (Q_\zeta - E_\zeta) \Lambda_{\zeta, x_i}) \mathbf{1} = -E_{x_i} \mathbf{1}.$$

Using the SBP property $Q + Q^T = E$, we can rewrite the left-hand side as:

$$(Q_\xi^T \Lambda_{\xi, x_i} + Q_\eta^T \Lambda_{\eta, x_i} + Q_\zeta^T \Lambda_{\zeta, x_i}) \mathbf{1} = E_{x_i} \mathbf{1}.$$

We can now express the above condition in matrix form as:

$$\begin{aligned}
 \left[\begin{array}{c|c|c} Q_\xi^T & Q_\eta^T & Q_\zeta^T \end{array} \right]_{n \times 3n} \left[\begin{array}{c} \Lambda_{\xi, x_i} \\ \Lambda_{\eta, x_i} \\ \Lambda_{\zeta, x_i} \end{array} \right]_{3n \times n} \left[\mathbf{1} \right]_{n \times 1} &= \left[\begin{array}{c} E_x \end{array} \right]_{n \times n} \left[\mathbf{1} \right]_{n \times 1} \\
 \underbrace{\left[\begin{array}{c|c|c} Q_\xi^T & Q_\eta^T & Q_\zeta^T \end{array} \right]_{n \times 3n}}_{Q^T} \underbrace{\left[\begin{array}{c} \text{diagvec}(\Lambda_{\xi, x_i}) \\ \text{diagvec}(\Lambda_{\eta, x_i}) \\ \text{diagvec}(\Lambda_{\zeta, x_i}) \end{array} \right]_{3n \times 1}}_{\mathbf{x}} &= \underbrace{\left[\begin{array}{c} E_x \end{array} \right]_{n \times n}}_{\mathbf{c}} \left[\mathbf{1} \right]_{n \times 1}.
 \end{aligned} \tag{4.22}$$

Our goal is now to compute the diagonal metric matrices Λ_{ξ, x_i} , Λ_{η, x_i} , and Λ_{ζ, x_i} such that the discrete geometric invariance condition (4.21) is satisfied. Following the approach in [CHDRF⁺18], we formulate this task as a convex optimisation problem. Specifically, we seek to minimise the deviation from the exact scaled metric terms subject to a linear constraint:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{b}\|_2^2, \quad \text{subject to } Q^T \mathbf{x} = \mathbf{c}, \tag{4.23}$$

where $\mathbf{b}$ contains the reference values of the **exact** scaled metric terms.¹⁶ The optimal solution $\mathbf{x}^*$ to this problem is the closest vector to $\mathbf{b}$ in the Euclidean norm that also satisfies the constraint $Q^T \mathbf{x} = \mathbf{c}$, which enforces the discrete geometric divergence-free condition (4.21). The vector $\mathbf{b}$ may either contain the exact scaled metric terms or high-order approximations accurate to $\mathcal{O}(h^{p+2})$. A closed form solution to the optimisation problem (4.23) is provided in the following proposition.

Proposition 4.5 (Proposition 2, [CHDRF⁺18]). *The solution to the convex optimisation problem (4.23) is given by*

$$\mathbf{x}^* = \mathbf{b} - (Q^T)^\dagger (Q^T \mathbf{b} - \mathbf{c}), \tag{4.24}$$

where $(Q^T)^\dagger$ denotes the Moore-Penrose pseudoinverse of Q^T . Furthermore,

$$\|\mathbf{x}^* - \mathbf{b}\|_\infty \leq M h^{p+2},$$

where $h > 0$ is the element size and $M > 0$ is a constant independent of h .

For a square matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$, we define

$$\text{diagvec}(A) := \begin{bmatrix} a_{11} \\ a_{22} \\ \vdots \\ a_{nn} \end{bmatrix} \in \mathbb{R}^n.$$

¹⁶ These values are computed from the Jacobian of the mapping $\mathcal{F}_k$, denoted by $\hat{\Lambda}_{\xi, x_i}$, $\hat{\Lambda}_{\eta, x_i}$, and $\hat{\Lambda}_{\zeta, x_i}$. We stack their diagonals into a single vector:

$$\mathbf{b} = \begin{bmatrix} \text{diagvec}(\hat{\Lambda}_{\xi, x_i}) \\ \text{diagvec}(\hat{\Lambda}_{\eta, x_i}) \\ \text{diagvec}(\hat{\Lambda}_{\zeta, x_i}) \end{bmatrix}$$

Note 3. *Strictly speaking, the SBP operators constructed in this section do not satisfy the conditions of Definition 4.1 exactly. In particular, polynomial exactness of a prescribed degree in each coordinate direction is not enforced. Nonetheless, the resulting operators achieve the desired order of accuracy on the physical element. They also mimic integration by parts at the physical level, and may therefore be regarded as an alternative definition of SBP operators, where accuracy is characterised in terms of truncation error rather than polynomial exactness.*

4.5 Semi-Discrete Formulation on Curvilinear Meshes

We now present discretisation of the linear advection equation on curvilinear meshes using the SBP operators developed in the previous sections and SATs to impose inter-element coupling and boundary conditions. Consider the d -dimensional linear advection equation

$$\begin{aligned} \frac{\partial \mathcal{U}}{\partial t} + \sum_{i=1}^d a_i \frac{\partial \mathcal{U}}{\partial x_i} &= 0, \quad \forall \vec{x} \in \Omega, \quad \forall t \geq 0, \\ \mathcal{U}(\vec{x}, 0) &= \mathcal{U}_0, \quad \forall \vec{x} \in \Omega, \quad t = 0, \end{aligned} \quad (4.25)$$

with constant advection velocity $\vec{a} = (a_1, \dots, a_d) \in \mathbb{R}^d$ and periodic boundary conditions on $\partial\Omega$. To motivate construction of SATs, it is useful to recall the weak form of the PDE at the element level, as is common in the analysis of discontinuous Galerkin methods [HW07].¹⁷

Let $\Omega_\kappa \subset \Omega$ denote the domain of an interior element κ with boundary Γ_κ , and let $\mathcal{V} \in C_c^\infty(\Omega_\kappa)$ be a smooth, compactly supported test function. Multiplying (4.25) by $\mathcal{V}$ and integrating over Ω_κ gives

$$\int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{U}}{\partial t} d\Omega + \sum_{i=1}^d \int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{F}_{x_i}}{\partial x_i} d\Omega = 0, \quad (4.26)$$

where $\mathcal{F}_{x_i} = a_i \mathcal{U}$ denotes the flux in the x_i -direction. Applying integration by parts to the second term yields

$$\int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{U}}{\partial t} d\Omega - \sum_{i=1}^d \int_{\Omega_\kappa} \frac{\partial \mathcal{V}}{\partial x_i} \mathcal{F}_{x_i} d\Omega + \sum_{i=1}^d \int_{\Gamma_\kappa} \mathcal{V} \mathcal{F}_{x_i} n_{x_i} d\Gamma = 0. \quad (4.27)$$

Within each element κ , the flux $\mathcal{F}_{x_i}$ depends only on the local solution $\mathcal{U}_\kappa$. At a shared interface $\gamma \subset \Gamma_\kappa \cap \Gamma_\nu$, however, there are two competing traces

$$\mathcal{U}|_{\Gamma_\kappa}, \quad \mathcal{U}|_{\Gamma_\nu},$$

which generally disagree unless continuity is enforced (as in continuous Galerkin methods). Thus, the physical flux is not uniquely defined on γ . To resolve this ambiguity, we introduce a **numerical flux function**

$$\mathcal{F}_{x_i}^*(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m,$$

¹⁷ The discussion here is intended to build intuition; see [HW07] for a rigorous treatment.

which combines the traces from both elements to produce a unique flux. Substituting $\mathcal{F}_{x_i}^*$ into (4.27) gives

$$\int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{U}}{\partial t} d\Omega - \sum_{i=1}^d \int_{\Omega_\kappa} \frac{\partial \mathcal{V}}{\partial x_i} \mathcal{F}_{x_i} d\Omega = - \sum_{i=1}^d \int_{\Gamma_\kappa} \mathcal{V} \mathcal{F}_{x_i}^*(\mathcal{U}_\kappa, \mathcal{U}_\nu) n_{x_i} d\Gamma,$$

where Ω_ν is the neighbouring element of Ω_κ sharing the interface γ . Applying IBP once more to the left-hand side yields

$$\begin{aligned} \int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{U}}{\partial t} d\Omega + \sum_{i=1}^d \int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{F}_{x_i}}{\partial x_i} d\Omega \\ = - \sum_{i=1}^d \int_{\Gamma_\kappa} \mathcal{V} \left(\mathcal{F}_{x_i}^*(\mathcal{U}_\kappa, \mathcal{U}_\nu) - \mathcal{F}_{x_i}(\mathcal{U}_\kappa) n_{x_i} \right) d\Gamma. \end{aligned} \quad (4.28)$$

We can incorporate the normal vector $\vec{n} = (n_{x_1}, \dots, n_{x_d})^T$ directly into the definition of the numerical flux, so that directional dependence is absorbed into $\mathcal{F}^*(\cdot, \cdot; \vec{n})$. Equation (4.28) can then be written more compactly as

$$\begin{aligned} \int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{U}}{\partial t} d\Omega + \sum_{i=1}^d \int_{\Omega_\kappa} \mathcal{V} \frac{\partial \mathcal{F}_{x_i}}{\partial x_i} d\Omega = - \int_{\Gamma_\kappa} \mathcal{V} \mathcal{F}^*(\mathcal{U}_\kappa, \mathcal{U}_\nu; \vec{n}_\kappa) d\Gamma \\ + \sum_{i=1}^d \int_{\Gamma_\kappa} \mathcal{V} \mathcal{F}_{x_i}(\mathcal{U}_\kappa) n_{x_i} d\Gamma. \end{aligned} \quad (4.29)$$

The numerical flux function $\mathcal{F}^*$ satisfies:

1. **Consistency:** $\mathcal{F}^*(\mathcal{U}, \mathcal{U}; \vec{n}) = \sum_{i=1}^d \mathcal{F}_{x_i}(\mathcal{U}) n_{x_i}$,
2. **Conservation:** $\mathcal{F}^*(\mathcal{U}_\kappa, \mathcal{U}_\nu; \vec{n}_\kappa) = -\mathcal{F}^*(\mathcal{U}_\nu, \mathcal{U}_\kappa; \vec{n}_\nu)$, ensuring that fluxes exchanged between neighbouring elements are equal and opposite.

Discretising (4.29) with SBP operators yields

$$\begin{aligned} \mathbf{v}_\kappa^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} + \sum_{i=1}^d \mathbf{v}_\kappa^T \mathbf{H}_\kappa \mathbf{D}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} = - \sum_{\gamma \in \Gamma_\kappa} \mathbf{v}_\kappa^T \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^* + \sum_{i=1}^d \mathbf{v}_\kappa^T \mathbf{E}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} \\ \mathbf{v}_\kappa^T \mathbf{H}_\kappa \left[\frac{d\mathbf{u}_\kappa}{dt} + \sum_{i=1}^d \mathbf{D}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} + \sum_{\gamma \in \Gamma_\kappa} \mathbf{H}_\kappa^{-1} \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^* - \sum_{i=1}^d \mathbf{H}_\kappa^{-1} \mathbf{E}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} \right] = 0. \end{aligned} \quad (4.30)$$

Here $\mathbf{f}_\kappa^{(i)} = a_i \mathbf{u}_\kappa$ denotes the physical flux evaluated at volume quadrature nodes, while $\mathbf{f}_\kappa^* = \mathcal{F}^*(\mathbf{u}_{\kappa \gamma}, \mathbf{u}_{\nu \gamma}; \mathbf{n}_\kappa)$ represents the numerical flux at facet quadrature nodes, with $\mathbf{u}_{\kappa \gamma} = \mathbf{R}_{\kappa \gamma} \mathbf{u}_\kappa$ and $\mathbf{u}_{\nu \gamma} = \mathbf{R}_{\nu \gamma} \mathbf{u}_\nu$. Since $\mathbf{v}_\kappa$ is arbitrary and $\mathbf{H}_\kappa$ is symmetric positive definite, the bracketed term in (4.30) must vanish. This gives the semi-discrete scheme

$$\frac{d\mathbf{u}_\kappa}{dt} + \sum_{i=1}^d \mathbf{D}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} = - \sum_{\gamma \in \Gamma_\kappa} \mathbf{H}_\kappa^{-1} \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^* + \sum_{i=1}^d \mathbf{H}_\kappa^{-1} \mathbf{E}_{x_i \kappa} \mathbf{f}_\kappa^{(i)}. \quad (4.31)$$

The right-hand side of (4.31) consists precisely of the SATs that impose inter-element coupling and boundary conditions (periodic in this case). A further simplification using the SBP property gives

$$\begin{aligned}
 \frac{d\mathbf{u}_\kappa}{dt} &= - \sum_{i=1}^d \left(D_{x_i\kappa} - H_\kappa^{-1} E_{x_i\kappa} \right) \mathbf{f}_\kappa^{(i)} - \sum_{\gamma \in \Gamma_\kappa} H_\kappa^{-1} R_{\kappa\gamma}^T B_\gamma \mathbf{f}_\kappa^* \\
 &= - \sum_{i=1}^d H_\kappa^{-1} (Q_{x_i\kappa} - E_{x_i\kappa}) \mathbf{f}_\kappa^{(i)} - \sum_{\gamma \in \Gamma_\kappa} H_\kappa^{-1} R_{\kappa\gamma}^T B_\gamma \mathbf{f}_\kappa^* \quad (4.32) \\
 &= \sum_{i=1}^d H_\kappa^{-1} Q_{x_i\kappa}^T \mathbf{f}_\kappa^{(i)} - \sum_{\gamma \in \Gamma_\kappa} H_\kappa^{-1} R_{\kappa\gamma}^T B_\gamma \mathbf{f}_\kappa^*.
 \end{aligned}$$

Substituting the curvilinear SBP operators (4.20) into (4.32) gives

$$\frac{d\mathbf{u}_\kappa(t)}{dt} = \sum_{i=1}^d \underbrace{\left[\left(\mathcal{J}_\kappa \hat{H} \right)^{-1} Q_{x_i\kappa}^T \right] \mathbf{f}_\kappa^{(i)}(t)}_{\text{VOLUME}} - \sum_{\gamma \in \Gamma_\kappa} \underbrace{\left[\left(\mathcal{J}_\kappa \hat{H} \right)^{-1} R_{\kappa\gamma}^T B_\gamma \right] \mathbf{f}_\kappa^*(t)}_{\text{FACET}}.$$

The operators marked VOLUME and FACET depend only on the element mapping and its associated metric terms. Hence they can be assembled once and stored. Their precomputation considerably simplifies the implementation of the semi-discrete scheme (4.32).

Conservation

Consider rewriting (4.25) in the following integral form where Ω_κ is an arbitrary control volume within Ω :

$$\begin{aligned}
 \int_{\Omega_\kappa} \frac{\partial \mathcal{U}}{\partial t} d\Omega_\kappa &= - \sum_{i=1}^d \int_{\Omega_\kappa} \frac{\partial \mathcal{F}_{x_i}}{\partial x_i} d\Omega_\kappa \\
 \frac{d}{dt} \int_{\Omega_\kappa} \mathcal{U} d\Omega_\kappa &= - \sum_{i=1}^d \int_{\partial\Omega_\kappa} \mathcal{F}_{x_i}(\mathcal{U}) n_{x_i} d\Gamma.
 \end{aligned} \quad (4.33)$$

Equation (4.33) makes the conservation property of the advection equation explicit: the rate of change of the total mass $\int_{\Omega_\kappa} \mathcal{U} d\Omega_\kappa$ within Ω_κ is equal to the net flux across its boundary $\partial\Omega_\kappa$. The conservation property is equally crucial for numerical schemes. The Lax-Wendroff theorem [LW60] states that a convergent and conservative discretisation of (4.25) converges to weak solutions of (4.33).

In the semi-discrete scheme (4.32), conservation follows from the SBP property $D_{x_i\kappa} \mathbf{1} = 0$ together with the use of a unique, consistent numerical flux $\mathcal{F}^*$ on each interface. A common choice for the advection equation is

$$\mathbf{f}_\kappa^*(\mathbf{u}_{\kappa\gamma}, \mathbf{u}_{\nu\gamma}; \mathbf{n}_\kappa) = \frac{1}{2} A_{n\kappa\gamma} (\mathbf{u}_{\kappa\gamma} + \mathbf{u}_{\nu\gamma}) + \frac{\lambda |A_{n\kappa\gamma}|}{2} (\mathbf{u}_{\kappa\gamma} - \mathbf{u}_{\nu\gamma}) \quad (4.34)$$

where

$$A_{n\kappa\gamma} = \sum_{i=1}^d a_i N_{x_i\kappa\gamma}, \quad N_{x_i\kappa\gamma} \text{ as in (4.14)}.$$

The parameter λ interpolates between the **symmetric SAT** ($\lambda = 0$) and the **upwind SAT** ($\lambda = 1$).

Multiplying (4.32) by $\mathbf{1}^T \mathbf{H}_\kappa$ gives

$$\begin{aligned} \mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} &= \sum_{i=1}^d \mathbf{1}^T \mathbf{Q}_{x_i \kappa}^T \mathbf{f}_\kappa^{(i)} - \sum_{\gamma \in \Gamma_\kappa} \mathbf{1}^T \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^* \\ &= - \sum_{\gamma \in \Gamma_\kappa} \mathbf{1}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*, \quad (\because \mathbf{Q}_{x_i \kappa} \mathbf{1} = \mathbf{0}, \mathbf{R}_{\kappa \gamma} \mathbf{1} = \mathbf{1}) \end{aligned}$$

so that the right-hand side approximates the net flux across Γ_κ . This shows that the integrated quantity $\mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} \approx \int_{\Omega_\kappa} \frac{\partial \mathcal{U}}{\partial t} d\Omega_\kappa$ is conserved element-wise. Summing over all elements yields

$$\sum_{\kappa \in \mathcal{T}_h} \mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} = - \sum_{\kappa \in \mathcal{T}_h} \left(\underbrace{\sum_{\gamma \in \Gamma_\kappa \cap \Gamma} \mathbf{1}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*}_{\text{boundary fluxes}} + \underbrace{\sum_{\gamma \in \Gamma_\kappa \setminus \Gamma} \mathbf{1}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*}_{\text{interface fluxes}} \right).$$

Here we have split the flux contributions into those from the physical boundary Γ and those from interior interfaces. Since the numerical flux satisfies $\mathbf{f}_\kappa^* = -\mathbf{f}_\nu^*$ on any $\gamma \subset \Gamma_\kappa \cap \Gamma_\nu$, the interior contributions cancel pairwise. The global conservation statement therefore reduces to

$$\sum_{\kappa \in \mathcal{T}_h} \mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} = - \sum_{\kappa \in \mathcal{T}_h} \sum_{\gamma \in \Gamma_\kappa \cap \Gamma} \mathbf{1}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*. \quad (4.35)$$

For periodic boundary conditions the right-hand side vanishes, and we obtain

$$\sum_{\kappa \in \mathcal{T}_h} \mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} = 0,$$

showing that the semi-discrete scheme is globally conservative.

Energy Stability

To establish energy stability, we again apply the energy method. Multiplying (4.32) by $\mathbf{u}_\kappa^T \mathbf{H}_\kappa$ gives

$$\mathbf{u}_\kappa^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt} = \sum_{i=1}^d \mathbf{u}_\kappa^T \mathbf{Q}_{x_i \kappa}^T \mathbf{f}_\kappa^{(i)} - \sum_{\gamma \in \Gamma_\kappa} \mathbf{u}_\kappa^T \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*.$$

The left-hand side can be written as $\frac{1}{2} \frac{d}{dt} \|\mathbf{u}_\kappa\|_{\mathbf{H}_\kappa}^2$. Using the SBP property $\mathbf{Q}_{x_i \kappa}^T = -\mathbf{S}_{x_i \kappa} + \frac{1}{2} \mathbf{E}_{x_i \kappa}$,

$$\mathbf{u}_\kappa^T \mathbf{Q}_{x_i \kappa}^T \mathbf{f}_\kappa^{(i)} = -\mathbf{u}_\kappa^T \mathbf{S}_{x_i \kappa} \mathbf{f}_\kappa^{(i)} + \frac{1}{2} \mathbf{u}_\kappa^T \mathbf{E}_{x_i \kappa} \mathbf{f}_\kappa^{(i)}.$$

Substituting $\mathbf{f}_\kappa^{(i)} = a_i \mathbf{u}_\kappa$ and using skew-symmetry of $\mathbf{S}_{x_i \kappa}$, the first term vanishes. Hence

$$\frac{d}{dt} \|\mathbf{u}_\kappa\|_{\mathbf{H}_\kappa}^2 = \sum_{i=1}^d a_i \mathbf{u}_\kappa^T \mathbf{E}_{x_i \kappa} \mathbf{u}_\kappa - \sum_{\gamma \in \Gamma_\kappa} 2 \mathbf{u}_\kappa^T \mathbf{R}_{\kappa \gamma}^T \mathbf{B}_\gamma \mathbf{f}_\kappa^*. \quad (4.36)$$

Summing (4.36) over all elements and grouping contributions facetwise yields

$$\begin{aligned} \sum_{\kappa \in \mathcal{T}_h} \frac{d}{dt} \|\mathbf{u}_\kappa\|_{H_\kappa}^2 &= -2 \sum_{\gamma \in \Gamma_I} \left(\mathbf{u}_\kappa^T R_{\kappa\gamma}^T B_\gamma \mathbf{f}_\kappa^* + \mathbf{u}_\nu^T R_{\nu\gamma}^T B_\gamma \mathbf{f}_\nu^* \right) \\ &\quad + \sum_{\gamma \in \Gamma_I} \sum_{i=1}^d a_i \left(\mathbf{u}_\kappa^T E_{x_i\kappa} \mathbf{u}_\kappa + \mathbf{u}_\nu^T E_{x_i\nu} \mathbf{u}_\nu \right). \end{aligned} \quad (4.37)$$

Using the expression $E_{x_i\kappa} = R_{\kappa\gamma}^T N_{x_i,\kappa\gamma} B_\gamma R_{\kappa\gamma}$ and the fact that $N_{x_i,\nu\gamma} = -N_{x_i,\kappa\gamma}$, the second term simplifies to

$$\sum_{\gamma \in \Gamma_I} \sum_{i=1}^d a_i \left(\mathbf{u}_\kappa^T N_{x_i,\kappa\gamma} B_\gamma \mathbf{u}_\kappa - \mathbf{u}_\nu^T N_{x_i,\kappa\gamma} B_\gamma \mathbf{u}_\nu \right). \quad (4.38)$$

Meanwhile, the first term in (4.37), using conservation of the numerical flux and substituting (4.34), gives

$$\begin{aligned} -2 \sum_{\gamma \in \Gamma_I} (\mathbf{u}_\kappa^T - \mathbf{u}_\nu^T) B_\gamma \mathbf{f}_\kappa^* &= - \sum_{\gamma \in \Gamma_I} \sum_{i=1}^d a_i \left(\mathbf{u}_\kappa^T N_{x_i,\kappa\gamma} B_\gamma \mathbf{u}_\kappa - \mathbf{u}_\nu^T N_{x_i,\kappa\gamma} B_\gamma \mathbf{u}_\nu \right) \\ &\quad - \lambda \sum_{\gamma \in \Gamma_I} (\mathbf{u}_\kappa^T - \mathbf{u}_\nu^T) B_\gamma |A_{n\kappa\gamma}| (\mathbf{u}_\kappa - \mathbf{u}_\nu). \end{aligned} \quad (4.39)$$

Adding (4.38) and (4.39) cancels the facet-local terms, leaving

$$\sum_{\kappa \in \mathcal{T}_h} \frac{d}{dt} \|\mathbf{u}_\kappa\|_{H_\kappa}^2 = -\lambda \sum_{\gamma \in \Gamma_I} (\mathbf{u}_\kappa - \mathbf{u}_\nu)^T B_\gamma |A_{n\kappa\gamma}| (\mathbf{u}_\kappa - \mathbf{u}_\nu).$$

Integrating in time gives

$$\sum_{\kappa \in \mathcal{T}_h} \|\mathbf{u}_\kappa(t)\|_{H_\kappa}^2 = \sum_{\kappa \in \mathcal{T}_h} \|\mathbf{u}_\kappa(0)\|_{H_\kappa}^2 - \lambda \int_0^t \sum_{\gamma \in \Gamma_I} \mathbf{u}_\gamma(\tau)^T B_\gamma |A_{n\kappa\gamma}| \mathbf{u}_\gamma(\tau) d\tau, \quad (4.40)$$

where $\mathbf{u}_\gamma = \mathbf{u}_\kappa - \mathbf{u}_\nu$. Since B_γ is positive definite and $|A_{n\kappa\gamma}|$ is positive semi-definite, the integral is non-negative. Thus, for $\lambda \in \{0, 1\}$,

$$\sum_{\kappa \in \mathcal{T}_h} \|\mathbf{u}_\kappa(t)\|_{H_\kappa}^2 \leq \sum_{\kappa \in \mathcal{T}_h} \|\mathbf{u}_\kappa(0)\|_{H_\kappa}^2,$$

showing that the semi-discrete scheme is energy stable for both the symmetric ($\lambda = 0$) and upwind ($\lambda = 1$) SATs.

4.6 Numerical Experiment

We conclude this chapter with a numerical experiment verifying the accuracy, conservation, and stability of the semi-discrete scheme (4.32) on curvilinear meshes. We solve the three-dimensional linear advection equation on the periodic domain

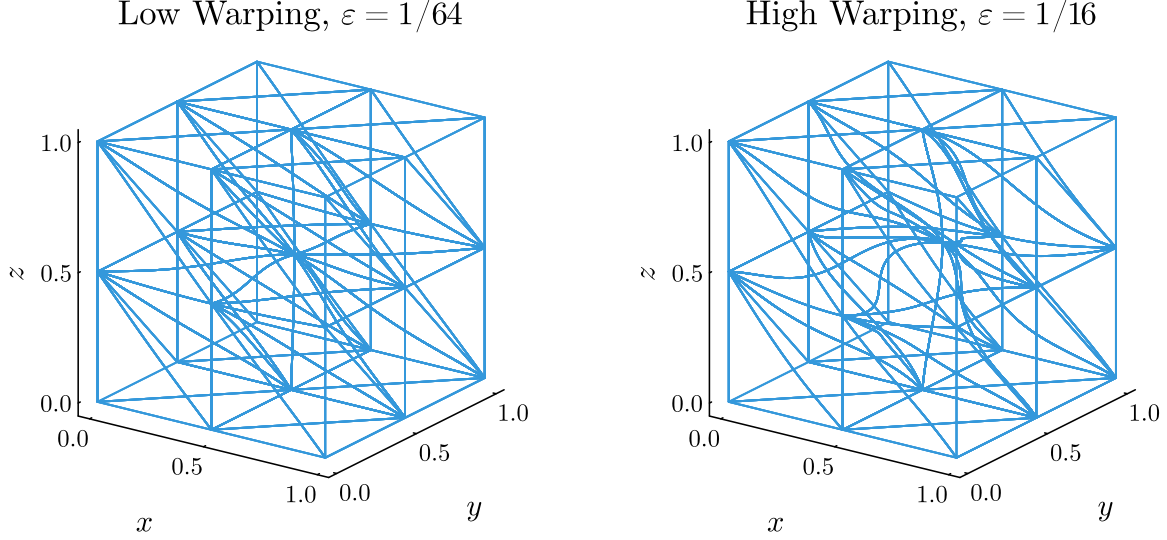
$$\Omega = [0, 1]^3, \quad \mathbf{a} = (1, 1, 1)^T,$$

with initial condition

$$\mathcal{U}_0(\vec{x}) = \sin(2\pi x) \sin(2\pi y) \sin(2\pi z).$$

The exact solution is given by

$$\mathcal{U}(\vec{x}, t) = \sin(2\pi(x - t)) \sin(2\pi(y - t)) \sin(2\pi(z - t)).$$



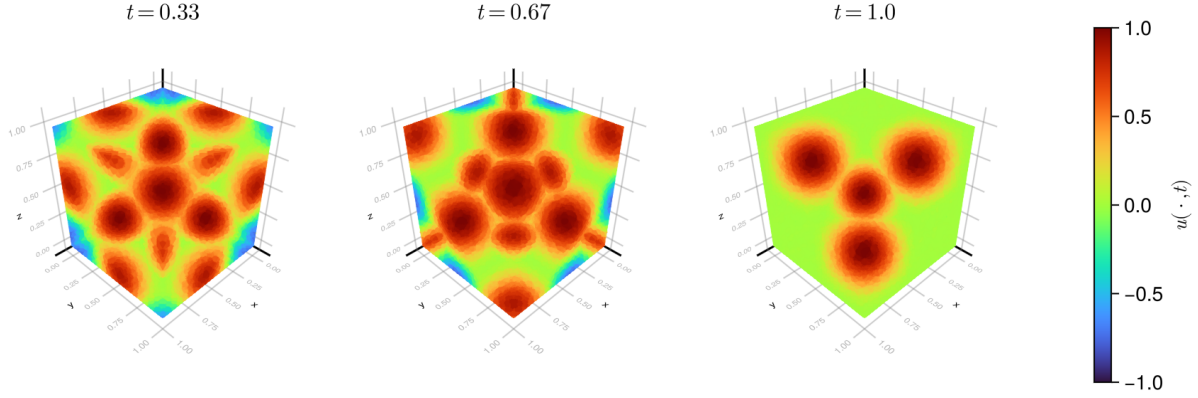
The domain is discretised using tetrahedral meshes generated with GMSH [GR09]. We consider two curvilinear meshes, with low and high warping based on [CDRFC19, Section 5.2]. The warping is applied via the transformation

$$\begin{aligned}\tilde{y} &\leftarrow y + \varepsilon \cos\left(3\pi\left(x - \frac{1}{2}\right)\right) \cos\left(\pi\left(y - \frac{1}{2}\right)\right) \cos\left(\pi\left(z - \frac{1}{2}\right)\right), \\ \tilde{x} &\leftarrow x + \varepsilon \cos\left(\pi\left(x - \frac{1}{2}\right)\right) \sin\left(4\pi\left(\tilde{y} - \frac{1}{2}\right)\right) \cos\left(\pi\left(z - \frac{1}{2}\right)\right), \\ \tilde{z} &\leftarrow z + \varepsilon \cos\left(\pi\left(\tilde{x} - \frac{1}{2}\right)\right) \cos\left(\pi\left(\tilde{y} - \frac{1}{2}\right)\right) \cos\left(\pi\left(z - \frac{1}{2}\right)\right),\end{aligned}$$

with $\varepsilon = 1/64$ for low warping and $\varepsilon = 1/16$ for high warping. Figure 4.5 illustrates the resulting tetrahedral meshes for the two warping levels. The new node positions $(\tilde{x}, \tilde{y}, \tilde{z})$ are then used to construct the curvilinear mapping $\Phi_\kappa : \hat{\Omega} \rightarrow \Omega_\kappa$ for each element κ through Lagrange interpolating polynomials of degree $d = 3$. The resulting systems of ordinary differential equations are integrated in time until $t = 1$ using the classical fourth-order Runge-Kutta method. The time step is chosen as

$$\Delta t = \frac{0.1 h}{\|\mathbf{a}\|_2}$$

Figure 4.5: Warped tetrahedral mesh grid.



where h denotes the minimum distance between volume quadrature nodes in the mesh. For spatial discretisation, the numerical flux (4.34) with the upwind SAT ($\lambda = 1$) is used. Figure 4.6 illustrates the numerical solution at various time instances on the highly warped mesh using degree $p = 3$ SBP operators.

Conservation and Stability

To assess the discrete conservation and stability properties, we track two diagnostic quantities over time:¹⁸

$$\text{Conservation residual: } \mathcal{R}_c(t) = \sum_{\kappa \in \mathcal{T}_h} \mathbf{1}^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt}, \quad \text{and,}$$

$$\text{Energy residual: } \mathcal{E}(t) = \sum_{\kappa \in \mathcal{T}_h} \mathbf{u}_\kappa^T \mathbf{H}_\kappa \frac{d\mathbf{u}_\kappa}{dt}.$$

Figure 4.7 shows the evolution of $\mathcal{R}_c(t)$ and $\mathcal{E}(t)$ for $p = 4$ on the highly warped mesh for both symmetric and upwind SATs. The mesh used for this test consists of 384 elements and is intentionally coarser to demonstrate that conservation and stability are maintained even at finite resolution, consistent with the theoretical analysis of the previous section.

Figure 4.6: Numerical solution for the 3D advection equation obtained using a degree- $p = 3$ SBP-SAT discretisation on a warped mesh with $\varepsilon = 1/16$ and 1296 tetrahedral elements.

¹⁸ $\frac{d\mathbf{u}}{dt}$ is computed by performing the right-hand side evaluations in (4.32).

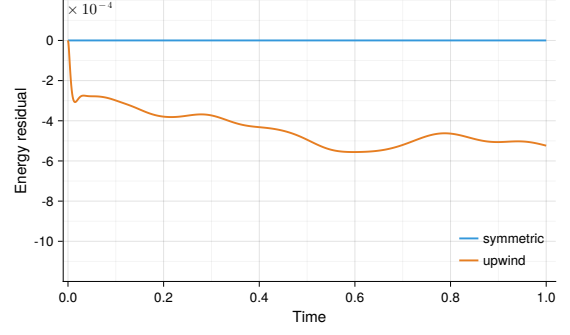
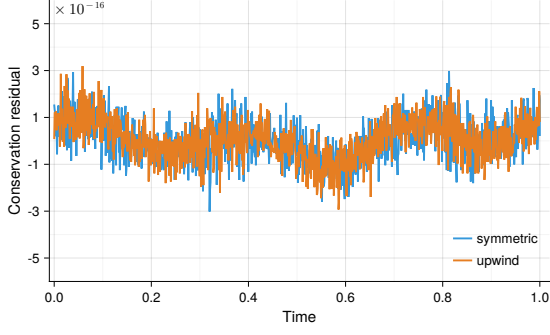


Figure 4.7: Time evolution of the conservation and energy residuals.

Accuracy

The discrete L_2 error is computed as

$$\|\mathbf{u} - \mathbf{u}_h\|_H = \sqrt{\sum_{\kappa \in \mathcal{T}_h} (\mathbf{u}_\kappa - \mathbf{u}_{h,\kappa})^T \mathbf{H}_\kappa (\mathbf{u}_\kappa - \mathbf{u}_{h,\kappa})},$$

where $\mathbf{u}_\kappa$ and $\mathbf{u}_{h,\kappa}$ denote the exact and numerical solutions evaluated at the volume quadrature nodes of element κ . Figure 4.8 shows the resulting solution errors and convergence rates obtained under uniform mesh refinement for SBP operators of degree $p = 1$ through $p = 5$, on both the low- and high-warping meshes. The reference-space SBP operators used in these tests are the Diagonal-E operators described in Section 4.2.

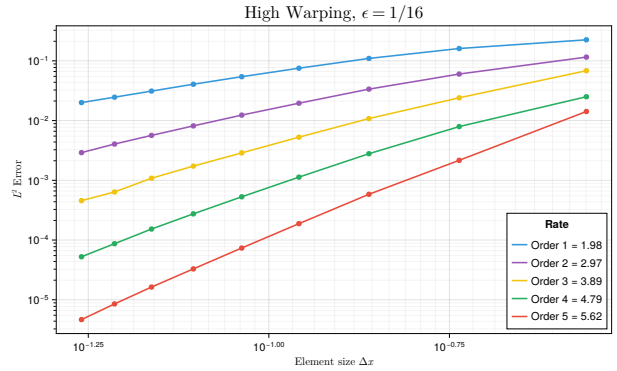
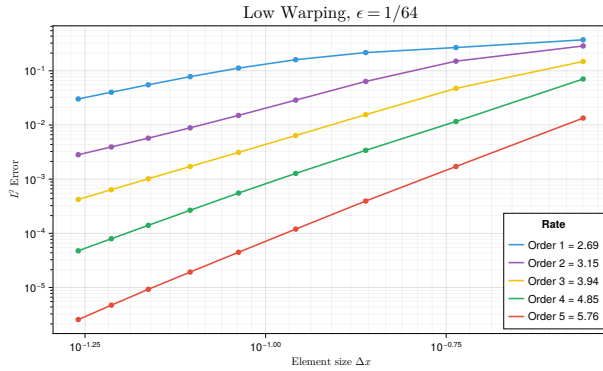


Figure 4.8: Solution error convergence rates computed by fitting a line through the last three values on the refined meshes.

5

Introduction to Quantum Computing

5.1 Foundations of Quantum Information

We will start by introducing the basic principles of quantum mechanics that are necessary to understand quantum computing. We follow the notation and conventions of [NCoo].

- A vector ψ in a complex Hilbert space $\mathcal{H}$ is written as **ket** $|\psi\rangle \in \mathcal{H}$, and its conjugate transpose (adjoint) as a **bra** $\langle\psi| \in \mathcal{H}^*$.¹
- The inner product of two vectors $\psi, \varphi \in \mathcal{H}$ is written $\langle\psi|\varphi\rangle \in \mathbb{C}$.
- The outer product $|\psi\rangle\langle\varphi|$ defines a rank-1 linear operator on $\mathcal{H}$.

¹ $|\psi\rangle$ is represented as a column vector, and $\langle\psi|$ as a row vector.

Basic Postulates of Quantum Mechanics

The core principles of quantum mechanics that govern the behaviour of quantum systems are:

1. **State space:** A d -dimensional quantum system is described by a Hilbert space $\mathcal{H} \cong \mathbb{C}^d$. A quantum state is a vector $|\psi\rangle \in \mathcal{H}$ satisfying $\langle\psi|\psi\rangle = 1$.
2. **Composite systems:** The joint state space of two systems with Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ is the **tensor product** space $\mathcal{H}_1 \otimes \mathcal{H}_2$.
3. **Unitary evolution:** Allowed (coherent) operations on a closed quantum system are **unitary** operators $U : \mathcal{H} \rightarrow \mathcal{H}$.
4. **Measurement (Born rule):** Let $\{|i\rangle\}_{i=1}^d$ be an orthonormal basis for the Hilbert space $\mathcal{H}$, and suppose the system is in the state

$$|\psi\rangle = \sum_{i=1}^d \alpha_i |i\rangle.$$

Measuring $|\psi\rangle$ in this basis yields outcome i with probability $|\alpha_i|^2$, and the post-measurement state collapses to $|i\rangle$.

5. **Measurement on subsystems (Extended Born rule):** Let $\mathcal{H}_1 \otimes \mathcal{H}_2$ be a bipartite system with orthonormal bases $\{|i\rangle\}$ for $\mathcal{H}_1$ and $\{|j\rangle\}$ for $\mathcal{H}_2$. Suppose the joint state is

$$|\psi\rangle = \sum_{i,j} \alpha_{ij} |i\rangle \otimes |j\rangle.$$

Measuring subsystem $\mathcal{H}_1$ in the $\{|i\rangle\}$ basis yields outcome i with probability $p_i = \sum_j |\alpha_{ij}|^2$, and the post-measurement state becomes²

$$|\psi_i\rangle = |i\rangle \otimes \frac{1}{\sqrt{p_i}} \sum_j \alpha_{ij} |j\rangle.$$

² Multiplying any $|\varphi_j\rangle$ by a **phase** $e^{i\theta}$ leaves measurement statistics unchanged. Two states that differ only by a global phase (e.g. $|\psi\rangle$ and $e^{i\theta} |\psi\rangle$) are physically indistinguishable.

Qubits and Composite Systems

- A quantum bit or **qubit** for short, is a vector in a two-dimensional complex Hilbert space $\mathbb{C}^2$ such that $\| |\psi\rangle \| = 1$. A canonical basis for $\mathbb{C}^2$ is given by the vectors:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

- A general qubit state is represented as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1.$$

Here, α and β are called **amplitudes**.

- An n -qubit system is described by the Hilbert space $(\mathbb{C}^2)^{\otimes n}$, with computational basis states labelled $|b\rangle$ for each bit string $b \in \{0, 1\}^n$. For instance, if $|\psi\rangle, |\varphi\rangle \in \mathbb{C}^2$ represent two single-qubit states, the joint system is described by their tensor product $|\psi\rangle \otimes |\varphi\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^4$.³ In the computational basis, this expands as:

$$\begin{aligned} |\psi\rangle \otimes |\varphi\rangle &= (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle) \\ &= \alpha_1 \alpha_2 |00\rangle + \alpha_1 \beta_2 |01\rangle + \beta_1 \alpha_2 |10\rangle + \beta_1 \beta_2 |11\rangle. \end{aligned}$$

³ This is often written as $|\psi\varphi\rangle$ or $|\psi\rangle |\varphi\rangle$ for brevity.

Entanglement and Superposition

- A state $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ is a **product state** if $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$; otherwise, it is **entangled**.
- A state of the form $|\psi\rangle = \alpha |\varphi\rangle + \beta |\chi\rangle$, with $\alpha, \beta \neq 0$, is a **superposition** of $|\varphi\rangle$ and $|\chi\rangle$.
- Interference** arises when amplitudes combine. For example:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Measuring either in the computational basis yields 0 or 1 with equal probability. However, their superposition:

$$\frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) = |0\rangle$$

leads to outcome 0 with certainty, due to **constructive interference** in the $|0\rangle$ amplitude.

5.2 Circuit Model of Quantum Computation

We will describe quantum algorithms using the **quantum circuit model**, a standard framework in which computations are performed on a fixed collection of qubits. In this model, a quantum circuit is understood as a unitary transformation applied to the system, with all qubits conventionally initialised in the state $|0\rangle$. Quantum circuits are constructed from a sequence of elementary gates, each acting non-trivially on a small subset of qubits. These gates are represented by unitary matrices that operate non-trivially only on the corresponding subsystem of the Hilbert space. When embedded into the full system, such gates are extended via tensoring with identities. The **gate complexity** of a circuit refers to the total number of one- and two-qubit operations it uses.

Definition 5.1 (Unitary Matrix). A complex matrix $U \in \mathcal{M}_n(\mathbb{C})$ is **unitary** if⁴

$$UU^\dagger = U^\dagger U = \mathbb{1}_n,$$

where $\dagger$ denotes the conjugate transpose and $\mathbb{1}_n$ is the identity matrix of size n . The set of all such matrices forms the **unitary group**, denoted

$$\mathbb{U}(n) := \left\{ U \in \mathcal{M}_n(\mathbb{C}) \mid UU^\dagger = \mathbb{1}_n \right\}.$$

The coherent evolution of an n -dimensional quantum state $|\psi\rangle \in \mathbb{C}^n$ is governed by a unitary operator $U \in \mathbb{U}(n)$. The transformed state is

$$|\varphi\rangle = U|\psi\rangle.$$

The restriction to unitaries imposes strong structural constraints on admissible transformations.⁵ A particularly important case is $\mathbb{U}(2)$, the group of all single-qubit unitaries. Every element of $\mathbb{U}(2)$ admits a compact factorisation, as stated in the following example.

Example 5.1 (Euler's Decomposition). Let $A \in \mathbb{U}(2)$ be arbitrary. Then there exist real parameters $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ such that

$$A = e^{i\alpha} \begin{bmatrix} e^{-i\beta} & 0 \\ 0 & e^{i\beta} \end{bmatrix} \begin{bmatrix} \cos \gamma & -\sin \gamma \\ \sin \gamma & \cos \gamma \end{bmatrix} \begin{bmatrix} e^{-i\delta} & 0 \\ 0 & e^{i\delta} \end{bmatrix}.$$

The first and last factors are diagonal phase shifts, the middle factor is a real rotation, and $e^{i\alpha}$ is a global phase, physically irrelevant to measurement outcomes.

⁴ $\mathcal{M}_n(\mathbb{C})$ denotes the set of all $n \times n$ complex matrices.

⁵ While a general $n \times n$ complex matrix has $2n^2$ real parameters, the unitary group $\mathbb{U}(n)$ is a smooth manifold of dimension n^2 .

Quantum Gates

A **quantum gate** is a mathematically well-defined operation that can be implemented physically on a quantum device. Throughout this thesis, we restrict attention to gates acting on one or two qubits, as these are the native operations available on most hardware platforms [KSB⁺20]. Table 5.1 summarises commonly used gates, including their circuit symbols, matrix representations, and action on computational basis states. Many gates are **parametric**, defined by a continuous parameter, typically an angle that governs the strength or duration of the operation. Examples include the single-qubit rotation gates $R_x(\theta), R_y(\theta), R_z(\theta)$.

Matrix Representation Quantum gates are represented by unitary matrices acting on the system's state vector. When multiple gates are applied in sequence, the overall transformation is the product of their matrices. However, the order of multiplication is reversed relative to the left-to-right order of circuit diagrams. For example, if a circuit applies gates G_1, G_2, G_3 from left to right (see Figure 5.1), the overall operator is

$$U = G_3 G_2 G_1,$$

since each gate matrix acts on the state vector from the left.

Reversible Circuits and Computation

Quantum mechanics is inherently reversible: every operation on a quantum state $|\psi\rangle$ must be invertible. This requirement is captured mathematically by the fact that all valid operations are represented by unitary matrices, which are invertible by definition. Reversibility, however, is not unique to quantum systems; classical computation can also be reformulated in reversible terms.

Classical logic circuits are typically built from gates such as NAND and FANOUT [MC13]. To make such circuits reversible, the computation must be encoded in a way that preserves all input information. A canonical example is the **Toffoli gate**, a three-bit reversible gate defined by

$$(a_0, a_1, a_2) \mapsto (a_0, a_1, a_2 \oplus a_0 \cdot a_1),$$

where $\oplus$ denotes bitwise XOR. This transformation performs a controlled-controlled-NOT, embedding the classical AND operation in a reversible framework (Figure 5.2). Importantly, the Toffoli gate is both unitary and self-inverse, making it suitable for quantum computations. It can further be used to simulate the NAND and FANOUT gates (Figure 5.3, Figure 5.4). Since any classical Boolean circuit can be expressed solely in terms of NAND and FANOUT [MC13], and both can be realised using Toffoli gates, it follows that any classical logic circuit can be embedded into a quantum circuit of comparable size. This observation establishes that quantum circuits are at least as powerful as their classical counterparts.

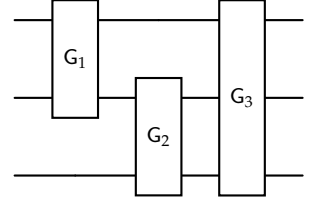


Figure 5.1: An example of a circuit with gates G_1, G_2 , and G_3 .

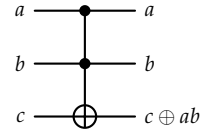


Figure 5.2: Toffoli gate

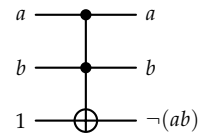


Figure 5.3: Implementing NAND using Toffoli.

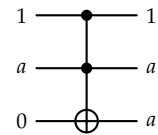


Figure 5.4: Implementing FANOUT using Toffoli.

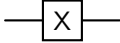
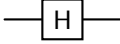
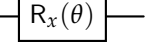
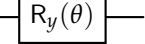
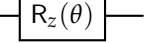
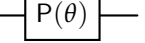
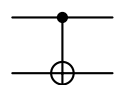
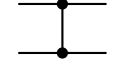
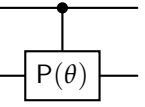
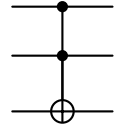
Name	Notation	Circuit Diagram	Matrix Representation	Action on Basis State
Pauli X	X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\beta 0\rangle + \alpha 1\rangle$
Pauli Y	Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$-i\beta 0\rangle + i\alpha 1\rangle$
Pauli Z	Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\alpha 0\rangle - \beta 1\rangle$
Hadamard	H		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$\frac{1}{\sqrt{2}}(\alpha + \beta) 0\rangle + \frac{1}{\sqrt{2}}(\alpha - \beta) 1\rangle$
Phase	S		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	$\alpha 0\rangle + i\beta 1\rangle$
T-gate	T		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	$\alpha 0\rangle + e^{i\pi/4}\beta 1\rangle$
X-rotation	$R_x(\theta)$		$\begin{bmatrix} \cos(\theta/2) & -i \sin(\theta/2) \\ -i \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$	$\cos(\theta/2)\alpha 0\rangle - i \sin(\theta/2)\beta 1\rangle$
Y-rotation	$R_y(\theta)$		$\begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$	$\cos(\theta/2)\alpha 0\rangle - \sin(\theta/2)\beta 1\rangle$
Z-rotation	$R_z(\theta)$		$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$	$e^{-i\theta/2}\alpha 0\rangle + e^{i\theta/2}\beta 1\rangle$
Phase	$P(\theta)$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$	$\alpha 0\rangle + e^{i\theta}\beta 1\rangle$
CNOT	CX		$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$	$\alpha_1\alpha_2 00\rangle + \alpha_1\beta_2 01\rangle + \beta_1\beta_2 10\rangle + \beta_1\alpha_2 11\rangle$
CZ	CZ		$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix}$	$\alpha_1\alpha_2 00\rangle + \alpha_1\beta_2 01\rangle + \alpha_2\beta_1 10\rangle - \beta_1\beta_2 11\rangle$
CP	$CP(\theta)$		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{bmatrix}$	$\alpha_1\alpha_2 00\rangle + \alpha_1\beta_2 01\rangle + \beta_1\alpha_2 10\rangle + \beta_1\beta_2 e^{i\theta} 11\rangle$
Toffoli	CCX		$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \\ & & & & & & 0 & 1 \\ & & & & & & 1 & 0 \end{bmatrix}$	$\alpha_1\alpha_2\alpha_3 000\rangle + \alpha_1\alpha_2\beta_3 001\rangle + \alpha_1\beta_2\alpha_3 010\rangle + \alpha_1\beta_2\beta_3 011\rangle + \beta_1\alpha_2\alpha_3 100\rangle + \beta_1\alpha_2\beta_3 101\rangle + \beta_1\beta_2\beta_3 110\rangle + \beta_1\beta_2\alpha_3 111\rangle$

Table 5.1: Common quantum gates

Diagonal Gates A quantum gate Δ acting on n qubits is called **diagonal** if its matrix representation in the computational basis is diagonal. Equivalently,

$$\Delta = \sum_{x \in \{0,1\}^n} e^{i\theta_x} |x\rangle \langle x|, \quad \theta_x \in \mathbb{R},$$

so that each basis state $|x\rangle$ is left unchanged up to a phase factor $e^{i\theta_x}$. In explicit form,

$$\Delta = \text{diag}(e^{i\theta_0}, e^{i\theta_1}, \dots, e^{i\theta_{2^n-1}}).$$

A common circuit symbol for a diagonal gate is shown in Figure 5.5.

Uniformly Controlled Gates Let $C_k^u(U)$ denote a quantum gate acting on k control qubits and ℓ target qubits. The gate applies one of 2^k distinct ℓ -qubit unitary operations, depending on the computational basis state of the control register:

$$C_k^u(U) |c\rangle |\psi\rangle = |c\rangle \otimes U_c |\psi\rangle, \quad c \in \{0,1\}^k,$$

where $U = \{U_c\}_{c \in \{0,1\}^k}$ is a family of unitary matrices on the target qubits. Such an operator is called a **uniformly controlled gate** (UCG), also known as a **multiplexed gate** [ICK⁺16]. Its matrix representation is block-diagonal:

$$C_k^u(U) = \bigoplus_{c \in \{0,1\}^k} U_c,$$

which makes it especially useful for structured decompositions and conditional logic. Any multi-controlled unitary, such as those in UCG decompositions, can be written using only on-controls by conjugating with X gates, simplifying synthesis.

Toffoli Gate The **Toffoli gate** (or CCX-gate) is a fundamental building block in reversible and quantum computing. Since most hardware supports only one- and two-qubit gates, it must be decomposed into elementary operations. A standard decomposition uses six CX gates together with single-qubit rotations [NCoo] (Figure 5.7).

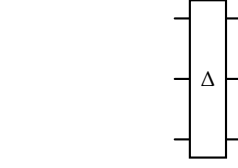
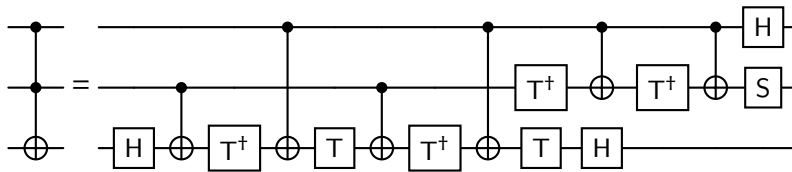


Figure 5.5: Notation for a three-qubit diagonal gate.

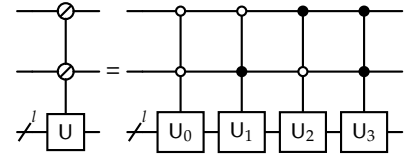


Figure 5.6: Decomposition of $C_k^u(U)$ gate. The symbol $\otimes$ denotes a generalised control pattern.

Figure 5.7: Decomposition of the Toffoli gate.

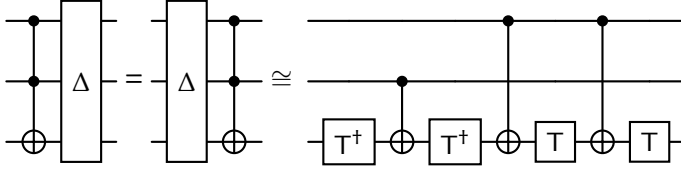
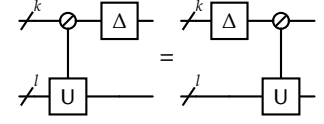


Figure 5.8: Decomposition of the Toffoli gate up to a diagonal gate.

If nontrivial phase shifts on intermediate states are permitted, the gate can instead be implemented with only three CX gates and four single-qubit rotations [ICK⁺16, Lemma 6] (Figure 5.8). In this decomposition, the diagonal gate satisfies $\Delta|010\rangle = -|010\rangle$ while acting trivially on all other basis states. Such relative-phase decompositions are especially useful in circuits that follow compute-uncompute patterns: diagonal gates commute with uniformly controlled (and hence controlled) unitaries, allowing them to pass through conditional logic without affecting control behaviour [ICK⁺16, Lemma 7] (Figure 5.9). We refer to this variant as the **relative-phase Toffoli gate** (RCCX-gate).

Figure 5.9: Commutation of a diagonal gate Δ with a multiplexed unitary U .

Swap Gate The **SWAP** gate is a two-qubit operation that exchanges the states of its inputs:

$$\text{SWAP}(|\psi\rangle \otimes |\varphi\rangle) = |\varphi\rangle \otimes |\psi\rangle.$$

In the computational basis its matrix is

$$\begin{bmatrix} 1 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix}.$$

Alternative circuit symbols are shown in Figure 5.10. The gate can be decomposed into three CX gates (Figure 5.11). To verify this, consider a basis state $|x\rangle \otimes |y\rangle$, with $x, y \in \{0, 1\}$:

$$\begin{aligned} \text{SWAP}(|x\rangle \otimes |y\rangle) &= \text{CX}_{12}\text{CX}_{21}\text{CX}_{12}(|x\rangle \otimes |y\rangle) \\ &= \text{CX}_{12}\text{CX}_{21}(|x\rangle \otimes |x \oplus y\rangle) \\ &= \text{CX}_{12}(|x \oplus (x \oplus y)\rangle \otimes |x \oplus y\rangle) \\ &= \text{CX}_{12}(|y\rangle \otimes |x \oplus y\rangle) \quad (\because x \oplus x = 0) \\ &= |y\rangle \otimes |(x \oplus y) \oplus y\rangle = |y\rangle \otimes |x\rangle. \end{aligned}$$

Fredkin Gate The **Fredkin gate** [FT82], or **controlled-SWAP**, exchanges two target qubits conditioned on a control qubit. Its action on the computational basis is

$$\text{FREDKIN } |x\rangle |y\rangle |z\rangle = \begin{cases} |x\rangle |y\rangle |z\rangle, & x = 0, \\ |x\rangle |z\rangle |y\rangle, & x = 1. \end{cases}$$

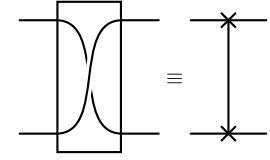


Figure 5.10: The SWAP gate.

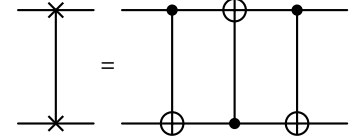


Figure 5.11: Decomposition of the SWAP gate.

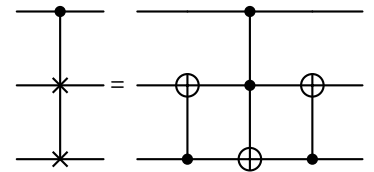


Figure 5.12: Fredkin gate decomposition.

That is, the second and third qubits are swapped if and only if the first qubit is $|1\rangle$. A decomposition into CX and Toffoli gates is shown in [Figure 5.12](#). Like the Toffoli gate, the Fredkin gate is universal for classical reversible computation [\[FT82\]](#).

Universal Gate Sets

Definition 5.2 (Universal Gate Set). A set of quantum gates $\mathcal{G}$ is **universal** if every unitary on n qubits can be approximated to arbitrary precision using only gates from $\mathcal{G}$. Concretely, for any $U \in \mathbb{U}(2^n)$ and any $\varepsilon > 0$, there exists a sequence $U_1, U_2, \dots, U_t \in \mathcal{G}$ such that

$$\|U - U_t \cdots U_2 U_1\| \leq \varepsilon,$$

where $\|\cdot\|$ denotes the spectral norm.⁶

It is known that all single-qubit unitaries together with the CX gate form a universal set, i.e., any unitary on n qubits can be exactly implemented using only these gates [\[BBC⁺95\]](#). Moreover, the **Solovay–Kitaev theorem** states that if $\mathcal{G}$ is a universal gate set closed under inverses and generating a dense subgroup of $\text{SU}(2)$, then any single-qubit unitary can be approximated to precision ε using $\mathcal{O}(\log^c(1/\varepsilon))$ gates from $\mathcal{G}$, for some constant $c > 3$. Hence such sets are not only complete but also efficient for circuit synthesis.

The Clifford+T gate set is a widely used universal gate set consisting of:

- **Clifford gates:** the Hadamard gate H, the phase gate S, and the CX gate.
- **Non-Clifford gate:** the T gate.

This set is universal for quantum computation in the approximate sense, meaning any unitary can be efficiently approximated to arbitrary precision using only Clifford and T gates [\[KMM13\]](#).

The Clifford+T set is central to **fault-tolerant quantum computation**. It suffices to realise fault-tolerant implementations of these gates, together with computational-basis measurements, to achieve universality [\[NC00\]](#). In typical error-correcting codes, Clifford gates admit straightforward fault-tolerant constructions, whereas T-gates require more costly techniques such as **magic state injection** and **distillation**. Nonetheless, provided physical error rates lie below a threshold, fault-tolerant methods suppress errors exponentially with only logarithmic overhead in circuit size [\[ABO99\]](#).

Circuits composed solely of Clifford gates are called **stabilizer circuits**. Despite the presence of quantum entanglement and superposition, they can be simulated efficiently on a classical computer. The **Gottesman–Knill theorem** states that stabilizer circuits on n qubits, with polynomially many gates and computational-basis deterministic measurements, can be simulated in time $\Theta(n^3)$ [\[Got97\]](#), later improved to $\Theta(n^2)$ [\[AG04\]](#), and more recently to $\Theta(n)$ via Gidney’s `Stim` simulator [\[Gid21\]](#). This classical tractability highlights

⁶ The spectral norm is defined as

$$\|A\| := \sup_{|\psi\rangle \neq 0} \frac{\|A|\psi\rangle\|}{\| |\psi\rangle \|}.$$

the necessity of non-Clifford gates for quantum advantage. The T gate is a standard choice, and the addition of even a few such gates already makes simulation intractable for large n in general.

5.3 Quantum Fourier Transform

The Fourier transform is ubiquitous in scientific computing, with the fast Fourier transform forming the backbone of many classical algorithms. It appears in diverse applications, from signal processing and data compression to complexity theory. Analogously, the **quantum Fourier transform** (QFT) is a central primitive in quantum algorithms, playing a key role in phase estimation, Shor's algorithm, amplitude estimation, and related procedures.

Throughout this section we adopt the notation of [Ric24]. Let $n \in \mathbb{N}$, and let $\mathcal{H}_n := (\mathbb{C}^2)^{\otimes n}$ denote the Hilbert space of n qubits. Computational basis states are written as

$$|x\rangle = |x_{n-1}\rangle \otimes \cdots \otimes |x_0\rangle,$$

where $x \in \{0, 1, \dots, 2^n - 1\}$ is the corresponding integer in little-endian binary encoding, with x_0 the least significant bit (LSB).

Hadamard Transform

Before introducing the quantum Fourier transform, it is useful to examine the structure and action of the multi-qubit **Hadamard transform** $H^{\otimes n}$. By definition,

$$H^{\otimes n} := H \otimes H \otimes \cdots \otimes H \in \mathbb{U}(2^n).$$

Applied to the all-zero input, it produces a **uniform superposition**:

$$H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

For a general basis state $|x\rangle$, where $x = x_{n-1} \dots x_1 x_0 \in \{0, 1\}^n$, we compute:

$$\begin{aligned} H^{\otimes n} |x\rangle &= (H |x_{n-1}\rangle) \otimes \cdots \otimes (H |x_1\rangle) \otimes (H |x_0\rangle) \\ &= \bigotimes_{j=0}^{n-1} \left(\frac{1}{\sqrt{2}} \sum_{y_j \in \{0,1\}} (-1)^{x_j y_j} |y_j\rangle \right) \\ &= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x_{n-1} y_{n-1} + \cdots + x_1 y_1 + x_0 y_0} |y\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{\vec{x} \cdot \vec{y}} |y\rangle, \end{aligned}$$

where $\vec{x} \cdot \vec{y}$ denotes the bitwise dot product modulo 2.

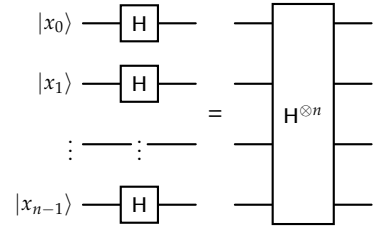


Figure 5.13: The Hadamard transform on n qubits.

A standard circuit representation is shown in Figure 5.13. This final expression highlights a structural property of the Hadamard transform: it maps computational basis states to uniform superpositions with phases determined by mod-2 inner products. This feature underlies its role in algorithms exploiting interference, such as the Deutsch-Jozsa, Simon's, and Grover's algorithms.

Fourier Transform

For a positive integer $n \in \mathbb{N}$, define

$$\omega := e^{2\pi i/n}.$$

The number ω is a **primitive n -th root of unity**, lying on the unit circle in the complex plane (Figure 5.14). By construction, $\omega^n = 1$, and the set $\{1, \omega, \omega^2, \dots, \omega^{n-1}\}$ forms a cyclic group under multiplication. A fundamental identity is

$$1 + \omega + \omega^2 + \dots + \omega^{n-1} = 0,$$

and more generally, for any $k \in \{1, \dots, n-1\}$,

$$1 + \omega^k + \omega^{2k} + \dots + \omega^{(n-1)k} = \sum_{j=0}^{n-1} \omega^{jk} = 0.$$

Definition 5.3 (Discrete Fourier Transform Modulo n). The **Discrete Fourier transform modulo n** is the unitary matrix $F_n \in \mathbb{U}(n)$ defined by

$$(F_n)_{j,k} := \frac{1}{\sqrt{n}} \omega^{jk}, \quad j, k \in \{0, 1, \dots, n-1\}.$$

Equivalently,

$$F_n = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{bmatrix}.$$

For any $a \in \mathbb{Z}_n$, the action of F_n on a computational basis state is⁷

$$F_n |a\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega^{ja} |j\rangle,$$

which is the a -th column of F_n and is called a **Fourier basis state**.⁸ Since F_n is unitary, it defines a quantum gate acting on $\log_2 n$ qubits, mapping an n -dimensional state vector to another such vector. When realised as a circuit, this transformation is the what we refer to as the **quantum Fourier transform (QFT)**.

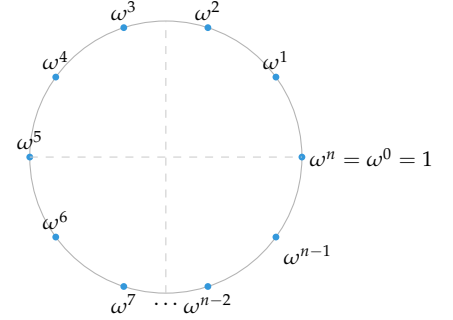


Figure 5.14: The roots of unity on the unit circle.

The inverse Fourier transform is the Hermitian adjoint,

$$(F_n^*)_{j,k} = \frac{1}{\sqrt{n}} \omega^{-jk}.$$

⁷ Basis states are indexed by integers $x \in \{0, \dots, 2^n - 1\}$ instead of bitstrings for notational simplicity.

⁸ Similarly,

$$F_n^* |a\rangle = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega^{-ja} |j\rangle.$$

Implementation of the Quantum Fourier Transform

In this thesis we restrict to the case $m = 2^n$, so that the QFT is a unitary operator $F_{2^n} \in \mathbb{U}(2^n)$ on n qubits. A key ingredient in its implementation is the use of controlled-phase rotations of the form⁹

$$|y\rangle |a\rangle \mapsto \omega_{2^n}^{ya} |y\rangle |a\rangle,$$

which apply phases determined by the binary expansion of y .

The QFT is constructed recursively: the $(n - 1)$ -qubit transform is combined with a Hadamard on the remaining qubit and a layer of controlled-phase gates. Finally, a sequence of SWAP gates is used to reverse the order of the qubits. An example for $n = 5$ is shown in Figure 5.16. The resulting circuit uses n Hadamard gates, $\frac{n(n-1)}{2}$ controlled-phase gates, and up to $\frac{n}{2}$ SWAP gates (each decomposes into three CX gates). The total gate complexity is therefore $\mathcal{O}(n^2)$, compared with $\mathcal{O}(2^n n)$ for the classical FFT.

By omitting rotations with very small angles, we obtain an **approximate QFT** with complexity $\mathcal{O}(n \log n)$ [CW00]. The inverse QFT is obtained simply by reversing the circuit and replacing each gate with its adjoint.

⁹ These gates commute and may be applied in any order.

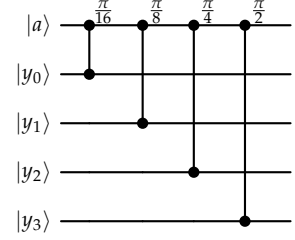


Figure 5.15: Controlled phase rotations for QFT for $m = 5$ qubits.

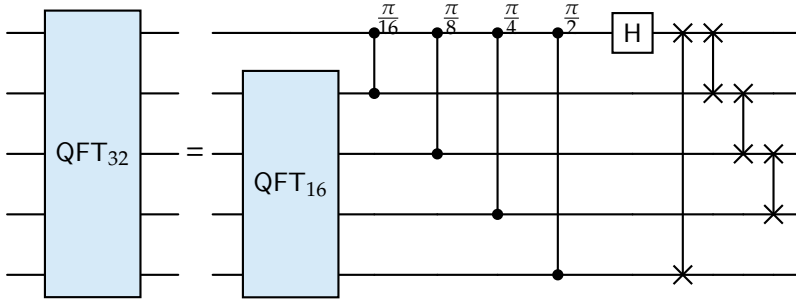


Figure 5.16: QFT on 5 qubits.

6

Building Blocks for Quantum Arithmetic and Operator Synthesis

This chapter develops the quantum arithmetic and control primitives that underpin the block-encoding and QSVT constructions. We begin by analysing ancilla management strategies and implementation of multi-controlled operations such as the MCX and MCMT gates, focussing on efficient decompositions and cost trade-offs. We then construct a hierarchy of arithmetic modules—incrementers, constant adders, and conditional adders culminating in circuits capable of performing controlled arithmetic using a constant number of dirty qubits. Building on these arithmetic primitives, we next introduce the **Projector-Controlled Phase Shift Gate**, which realises selective phase rotations conditioned on arbitrary projectors, the **Linear Combination of Unitaries (LCU)** framework, which expresses structured matrix operations as coherent weighted sums of simpler unitaries, and finally, we describe **Oblivious Amplitude Amplification**, an essential technique for amplifying success probability while preserving unitarity within block-encoded subspaces.

6.1 Multi-Controlled X Gate

Definition 6.1 (Multi-Controlled X Gate (MCX)). The **multi-controlled X gate**, denoted MCX_n , is the unitary operator

$$\text{MCX}_n : \mathcal{H}_{n+1} \rightarrow \mathcal{H}_{n+1},$$

acting on n control qubits and a single target qubit. On computational basis states, its action is defined by

$$\text{MCX}_n |x_0, x_1, \dots, x_{n-1}\rangle |t\rangle = |x_0, x_1, \dots, x_{n-1}\rangle |t \oplus (x_0 \wedge \dots \wedge x_{n-1})\rangle,$$

where $t \in \{0, 1\}$ is the value of the target qubit, and $\oplus$ denotes bitwise XOR. Hence, the target qubit is flipped if and only if all n control qubits are in the $|1\rangle$ state, thereby generalising the standard CCX (Toffoli) gate.

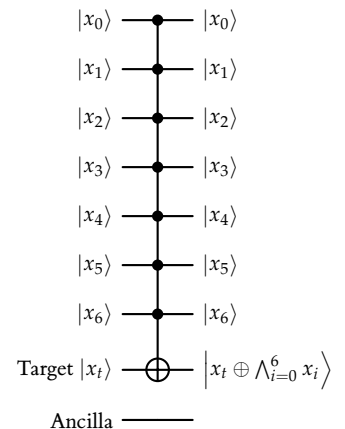


Figure 6.1: An MCX_n gate with n control qubits and 1 target qubit.

The (MCX) gate arises ubiquitously in quantum arithmetic subroutines, oracle constructions, phase-kickback primitives, etc. In the context of block-encoding, circuit complexity depends critically on the ability to implement MCX gates efficiently. Since early fault-tolerant architectures will operate with few logical qubits, our focus is on **ancilla-minimal decompositions** that balance qubit count, circuit depth, and gate complexity. Throughout this section we examine such decompositions for a circuit with n control qubits, one target, and optionally one ancilla, as illustrated in Figure 6.1.

Ancilla Qubits

Before discussing circuit constructions, we introduce the notion of **ancilla qubits** as described in [Cra15]. Ancilla qubits are auxiliary qubits that support a quantum circuit without being part of its logical input or output. They allow intermediate storage, enable reversible constructions, and often reduce depth or gate count in multi-controlled operations. Depending on their initialisation and restoration requirements, ancillae are classified into four types:

- **Burnable:** Start in $|0\rangle$ but may end in any state. They can be freely overwritten and final clean-up is unnecessary.
- **Zeroed:** Start and end in $|0\rangle$. These require explicit uncomputation but permit safe reuse in later stages.
- **Garbage:** Begin in an unknown state and need not be restored. Their use requires reversible techniques such as **toggle detection** [Cra15].
- **Borrowed:** Also start in an arbitrary state but must be returned unchanged. They often correspond to temporarily idle qubits elsewhere in the circuit.

Throughout this thesis we focus on the **borrowable ancilla** case, as it represents the most general setting and requires the fewest dedicated resources. Such constructions are particularly suitable for early fault-tolerant architectures, where minimising qubit overhead is of primary importance.

Uncomputation and Toggle Detection

Figure 6.2 shows the decomposition of an MCX_n gate into two $\text{MCX}_{n/2}$ gates using one ancilla qubit. The first two gates perform a simple reduction: the ancilla acts as an intermediate flag that is toggled by one half of the controls and then serves as a combined control for the other half.

However, this procedure **pollutes** the ancilla since its state is modified mid-computation. Because the operation of the first $\text{MCX}_{n/2}$ gate is reversible, we can restore the ancilla to its original value by applying the same gate again. This process of reversing intermediate side effects is often called the

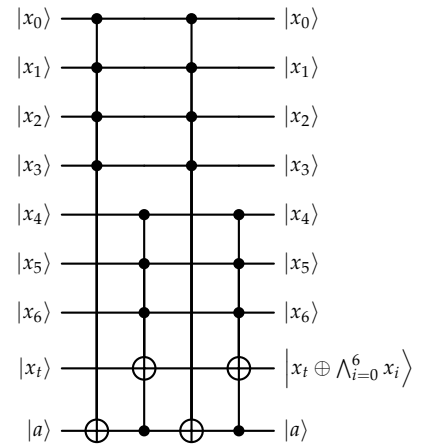


Figure 6.2: MCX_n gate with one borrowed qubit.

uncomputation trick, and it ensures that ancillary qubits return to their initial state once their temporary role is complete.

When the ancilla's initial state is unknown, uncomputation is not enough. We need to additionally perform **toggle detection** [Cra15]. The idea is to conditionally flip the target qubit both before and after the possible toggling of the ancilla. If the ancilla was never toggled, the two conditional flips cancel; if it was toggled, the target experiences a net inversion. This guarantees that the overall transformation remains reversible while the ancilla (potentially in an arbitrary state), is left unaffected at the end of the computation. This decomposition can be applied recursively, however it leads to the recurrence $T(n) = 4T(n/2)$, resulting in a total gate count of $T(n) = \mathcal{O}(n^2)$.

Decomposition using Conditionally Clean Ancillae

A recently introduced technique, **conditionally clean ancillae** [NZS24, KG25], extends the standard notions of clean and dirty ancillae by introducing a new intermediate resource. Unlike dirty ancillae, which may begin in unknown states, conditionally clean ancillae behave as if they were clean within subspaces selected by a **predicate register**. This property allows them to function as temporary workspace **without** the overhead of toggle detection, enabling notable reductions in gate count and depth for building blocks such as MCX gates, incrementers, and comparators.

Let x_1, x_2 be control qubits (possibly entangled with a reference system R), with joint state

$$|\Phi\rangle_{x_1 x_2 R} = \sum_{x \in \{0,1\}^2} \alpha_x |x\rangle_{x_1 x_2} |\psi_x\rangle_R.$$

Append a clean target qubit t in state $|0\rangle$ and apply a CCX controlled by (x_1, x_2) :

$$\text{CCX}(x_1, x_2, t) |\Phi\rangle = \sum_{x \neq 11} \alpha_x |x\rangle_{x_1 x_2} |\psi_x\rangle_R |0\rangle_t + \alpha_{11} |11\rangle_{x_1 x_2} |\psi_{11}\rangle_R |1\rangle_t. \quad (6.1)$$

Define the predicate P as “ $t = 1$ ” with corresponding projector $\Pi_P = \mathbb{1} \otimes |1\rangle\langle 1|_t$. Restricting (6.1) to the $t = |1\rangle$ subspace gives

$$\Pi_P \text{CCX}(x_1, x_2, t) |\Phi\rangle = \alpha_{11} |11\rangle_{x_1 x_2} |\psi_{11}\rangle_R |1\rangle_t. \quad (6.2)$$

Within this subspace, the register (x_1, x_2) collapses to the definite classical value $|11\rangle$, independent of R . In the terminology of conditionally clean ancillae:

- the **predicate register** is t ,
- the **conditionally clean ancilla register** is (x_1, x_2) ,
- the associated **known value** is $f(P) = 11$.

Thus, after marking with a CCX-gate and restricting to $t = 1$, the pair (x_1, x_2) may be treated as clean ancillae **conditioned** on the predicate P . Any operation controlled on t can safely reuse these qubits as workspace in the known state $|11\rangle$, performing temporary computations without requiring toggle detection or explicit cleanup.

To illustrate the use of conditionally clean ancillae, consider Figure 6.3. Our goal is to implement $\text{MCX}(x_0, x_1, x_2, x_3, x_t)$ using a single clean ancilla a . The construction follows a **mark-use-unmark** pattern that isolates a subset of the controls, uses them as temporary workspace, and then restores all registers to their initial states.

1. **Mark.** Compute the predicate P that $(x_2, x_3) = (1, 1)$ by applying $\text{CCX}(x_2, x_3, a)$. In the subspace $a = 1$, the pair (x_2, x_3) is deterministically $|11\rangle$, hence conditionally clean with known value $f(P) = 11$.
2. **Use (all gates controlled on a).**
 - (a) Apply X to x_2 , converting $|1\rangle \mapsto |0\rangle$. Qubit x_2 now serves as a clean zeroed ancilla.
 - (b) Compute $\text{CCX}(x_0, x_1, x_2)$ so that x_2 temporarily stores $x_0 \wedge x_1$.
 - (c) Flip the target via $\text{CCX}(x_2, a, x_t)$, toggling x_t iff $a = 1$ and $x_2 = 1$, i.e., precisely when $x_0 = x_1 = x_2 = x_3 = 1$.
3. **Uncompute.** Apply the inverse of steps 2(a)–2(b) to restore x_2 to its original value within the $a = 1$ subspace.
4. **Unmark.** Reapply $\text{CCX}(x_2, x_3, a)$ to reset the ancilla a to $|0\rangle$.

This circuit realises a unitary U that acts as the identity whenever $a = 0$, since the compute-uncompute pattern leaves all registers unchanged. When $a = 1$, it flips the target qubit t exactly when $x_0 = x_1 = x_2 = x_3 = 1$, thereby reproducing the action of MCX_4 .¹

In Figure 6.3, the pair (x_2, x_3) become conditionally clean once the predicate register a is marked. Within the same subspace, these qubits can then be reused as workspace to generate **further** conditionally clean ancillae: for instance, (x_0, x_1) may be treated as conditionally clean whenever $a = 1$. Iterating this idea yields a recursive construction in which conditionally clean ancillae give rise to additional ones, enabling an efficient synthesis of MCX_n .

Figure 6.4 illustrates the general MCX_n construction using one clean ancilla, proceeding in four stages:

1. **Up ladder:** A sequence of $\lfloor n/2 \rfloor$ CCX gates progressively accumulates the logical AND of control values onto conditionally clean qubits.
2. **Down ladder:** A second ladder of $n - \lfloor n/2 \rfloor - 2$ CCX gates propagates the accumulated values downward until the AND of all n controls is stored on the single qubit x_0 . This qubit now acts as the **effective global control**.

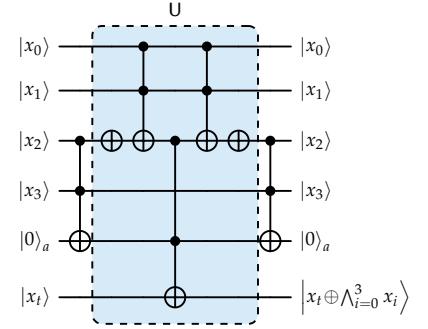


Figure 6.3: Example of using conditionally clean ancillae. The unitary U acts as the desired operation only when $a = 1$, and in this subspace it may safely treat (c_1, c_2) as clean workspace in the known state $|11\rangle$.

¹ In this example, only one of the two conditionally clean ancillae is actually used, yet the full structure generalises naturally to larger control sets.

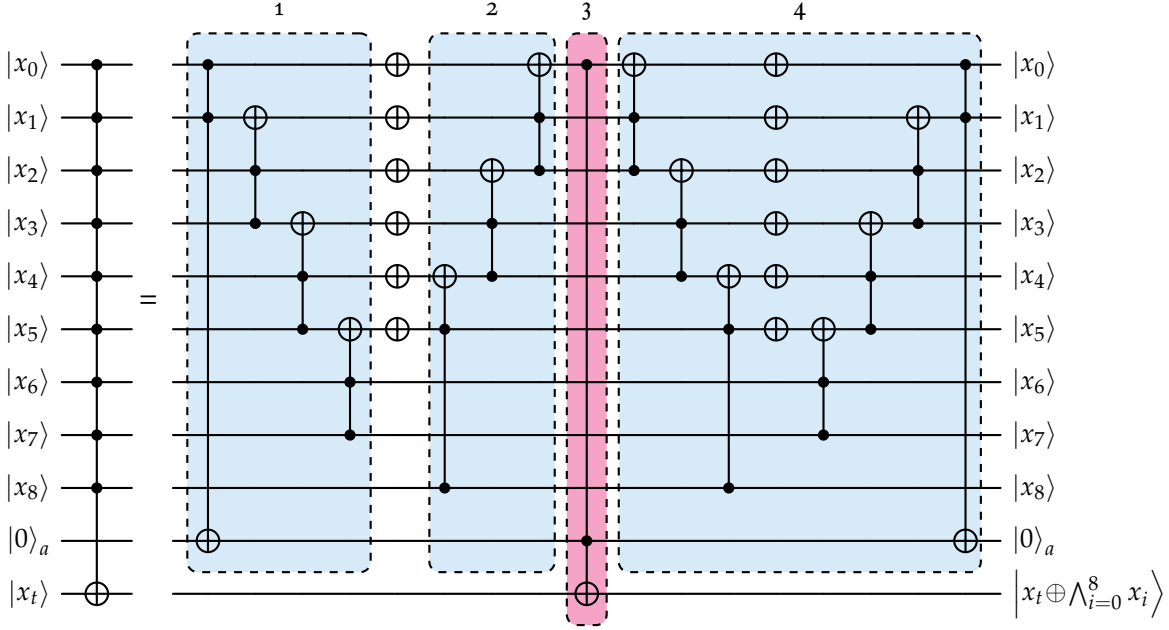


Figure 6.4: Decomposition of MCX_9 using one clean ancilla. Figure adapted from [KG25].

3. **Application:** One additional CCX-gate (not part of a compute-uncompute pair) applies the desired n -controlled NOT to the target.
4. **Uncomputation:** Both ladders are then reversed, restoring all clean and conditionally clean ancillae to their initial states and leaving only the intended flip on the target qubit.

The full construction uses $2n - 3$ CCX gates and has depth $\mathcal{O}(n)$. Decomposing each CCX into one- and two-qubit gates yields $12n - 18$ CX gates. Since all but one CCX occur in compute-uncompute pairs, they can be replaced by relative-phase CCX (RCCX) as shown in Figure 6.5, reducing the total to $6n - 6$ CX gates.

Using the toggle-detection trick, the clean ancilla a can be replaced with a dirty ancilla (Figure 6.6). This raises the CCX count to $4n - 8$, but since all except two appear in compute-uncompute pairs, they can be implemented with RCCX gates, giving a total of $12n - 18$ CX gates.

Finally, employing two ancillae, either both clean or both dirty, reduces the circuit depth from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$, while preserving the same overall gate count as the single-ancilla construction. Detailed implementations are provided in [KG25, Sections 5.3 and 5.5].

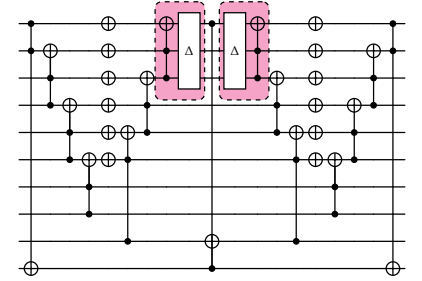


Figure 6.5: Example of a pair of CCX-gates in a compute-uncompute pattern that can be replaced by a RCCX-gate.

$$\text{Total CX count for } \text{MCX}_n = \begin{cases} 6n - 6, & \text{with one/two clean ancilla,} \\ 12n - 18, & \text{with one/two dirty ancilla.} \end{cases}$$

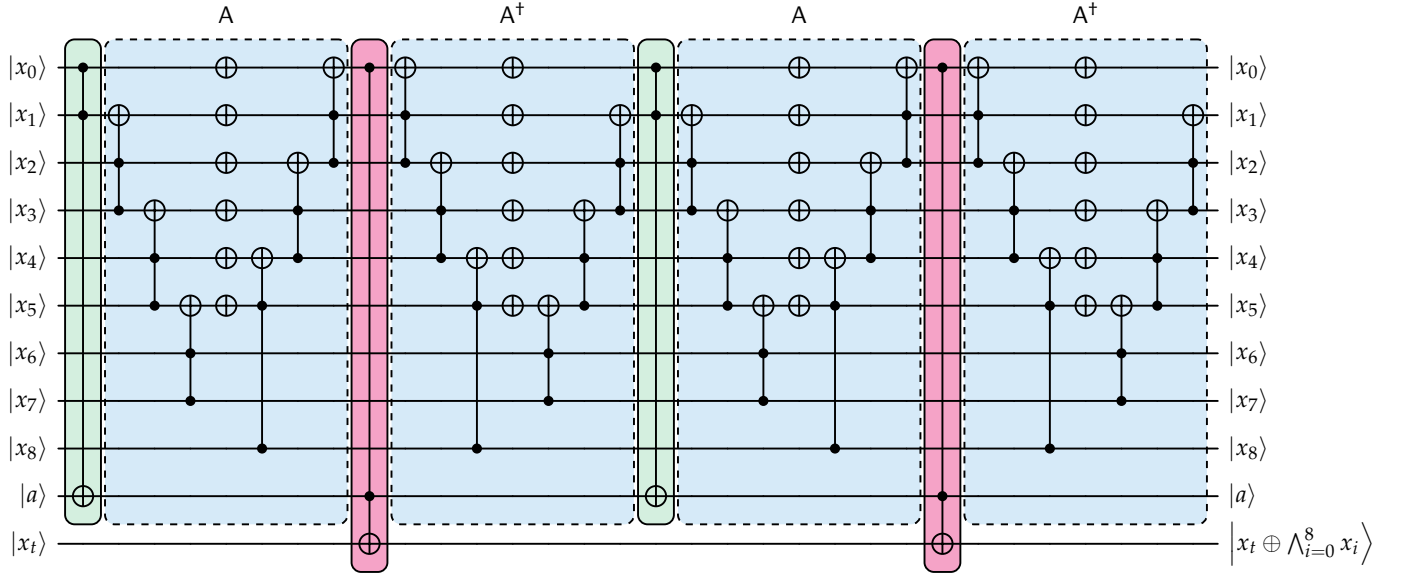


Figure 6.6: Decomposition of MCX_9 using one dirty ancilla. Figure adapted from [KG25].

6.2 Multi-Control-Multi-Target Gate

A natural extension of the MCX gate is the **multi-control multi-target NOT** gate, denoted $\text{MCMT}_c^t(X)$. It acts on c control qubits and t target qubits, flipping **each** target if and only if all controls are $|1\rangle$:

$$\text{MCMT}_c^t(X) |x\rangle |y\rangle = \begin{cases} |x\rangle X^{\otimes t} |y\rangle, & \text{if } x = 1^c, \\ |x\rangle |y\rangle, & \text{otherwise.} \end{cases}$$

Here $x \in \{0, 1\}^c$, $y \in \{0, 1\}^t$, and 1^c denotes the c -bit string of all ones.

Efficient Decomposition

A naive realisation would apply t separate MCX_c gates, one per target qubit, but this approach scales poorly. A more efficient implementation (Figure 6.7) uses a single MCX_c gate together with $2t - 2$ additional CX gates [Gid18, Fig. 24]. Using the MCX decomposition from previous sections, this yields a total cost of $12c + 2t - 20$ CX gates.

When the number of targets satisfies $t \geq c - 1$, the additional targets can act as ancillary workspace, enabling the $c - 2$ -ancilla construction from [ICK⁺16, Lemma 8]. This improves the MCX_c cost to $8c - 6$ CX gates and the overall complexity to $8c + 2t - 8$ CX.

$$\text{Total CX count for } \text{MCMT}_c^t(X) = \begin{cases} 12c + 2t - 20, & t < c - 1, \\ 8c + 2t - 8, & t \geq c - 1. \end{cases}$$

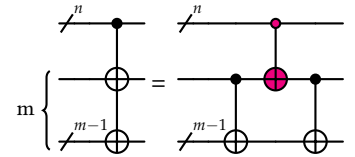


Figure 6.7: Decomposing $\text{MCMT}_n^m(X)$ into one MCX_n gate and $m - 1$ CX gates.

To validate these constructions in practice, the MCX and MCMT circuits described above were implemented by the author in Qiskit [JATK⁺24]. These implementations were contributed via pull requests that have since been merged into the official repository, making the techniques directly available to the broader quantum computing community.

6.3 Quantum Adders

We write computational basis states as

$$|x\rangle = |x_{n-1}\rangle \otimes \cdots \otimes |x_0\rangle,$$

where $x \in \{0, 1, \dots, 2^n - 1\}$ is the integer encoded in binary, with x_0 being the least significant bit (LSB). When considering two such registers, the sum $x + y$ is taken modulo 2^n , consistent with binary arithmetic on fixed-width registers. We adopt the notation from [Gid18], where adding the contents of a qubit register $|x\rangle$, encoding an integer A in binary into another register $|y\rangle$ is denoted as shown in Figure 6.9.

Quantum-Quantum Adders

Definition 6.2 (In-Place Quantum Adder). The **in-place quantum adder** is the unitary operator $\text{ADD}_n : \mathcal{H}_n \otimes \mathcal{H}_n \rightarrow \mathcal{H}_n \otimes \mathcal{H}_n$, defined by its action on computational basis states as

$$\text{ADD}_n |x\rangle |y\rangle = |x\rangle |(x + y) \bmod 2^n\rangle.$$

Definition 6.3 (Controlled Rotation Layer). The **controlled rotation layer**, denoted CROT_n , is the unitary operator $\text{CROT}_n : \mathcal{H}_n \otimes \mathcal{H}_{n+1} \rightarrow \mathcal{H}_n \otimes \mathcal{H}_{n+1}$, acting on an n -qubit control register and an $(n + 1)$ -qubit target register. For computational basis states $|x\rangle |\tilde{y}\rangle$, where $|\tilde{y}\rangle = \text{QFT}_{n+1} |y\rangle$ is the quantum Fourier transform of an $(n + 1)$ -qubit register, its action is²

$$\text{CROT}_n |x\rangle |\tilde{y}\rangle = \left(\prod_{j=0}^{n-1} \prod_{k=j}^{n-1} \text{CP}_{x_j \rightarrow y_k} \left(\frac{2\pi}{2^{k-j+1}} \right) \right) \left(\prod_{j=0}^{n-1} \text{CP}_{x_j \rightarrow y_n} \left(\frac{2\pi}{2^{n-j}} \right) \right) |x\rangle |\tilde{y}\rangle.$$

This layer is visually highlighted in Figure 6.10.

Draper Quantum Adder [Dra00, RPGE17]: The **Draper quantum adder** performs modular addition in the Fourier basis. It is defined as the unitary

$$\text{DADD}_n := (\mathbb{1} \otimes \text{QFT}_{n+1}^\dagger) \text{CROT}_n (\mathbb{1} \otimes \text{QFT}_{n+1}),$$

where QFT_{n+1} denotes the quantum Fourier transform on $n + 1$ qubits, and CROT_n applies the controlled phase rotations conditioned on the value of the first register.³ On computational basis states, the operator acts as

$$\text{DADD}_n |x\rangle |y\rangle |0\rangle = |x\rangle |(x + y) \bmod 2^n\rangle |c_n\rangle,$$

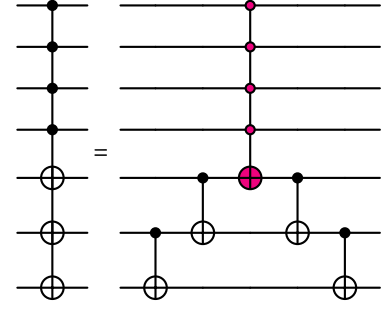


Figure 6.8: Decomposition of $\text{MCMT}_4^3(X)$ gate.

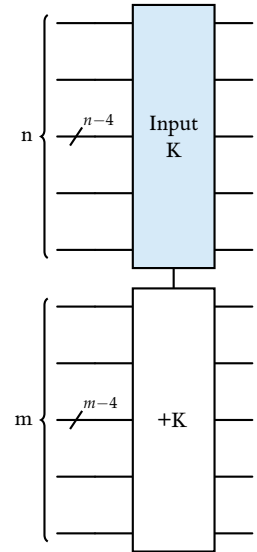


Figure 6.9: Adding stored value K in top register to the bottom register.

² The first product performs phase additions between control qubits x_j and target qubits $y_0, \dots, y_{n-1}$. The second product propagates overflow into the carry qubit y_n with the correct phase resolution.

³ The carry qubit is included in the Fourier transform so that it naturally encodes the carry-out bit. If a carry-out is not required, this qubit may be omitted, yielding a fixed adder.

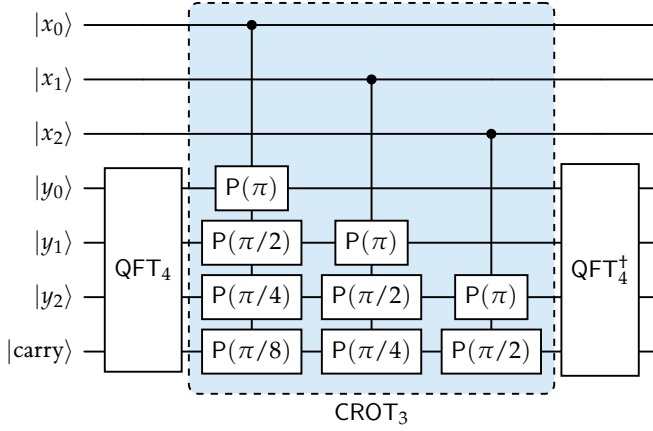


Figure 6.10: Draper adder circuit DADD_3 for $n = 3$.

where $c_n \in \{0, 1\}$ is the final carry-out bit indicating whether $x + y \geq 2^n$. The full circuit is shown in Figure 6.10.

The overall cost of DADD_n arises from two Fourier transforms and one controlled rotation layer. Each QFT_{n+1} requires $\mathcal{O}(n^2)$ one- and two-qubit gates, while CROT_n contains $\mathcal{O}(n^2)$ controlled phase rotations. Hence the total asymptotic complexity of the Draper adder is

Total CX count: $\mathcal{O}(n^2)$.

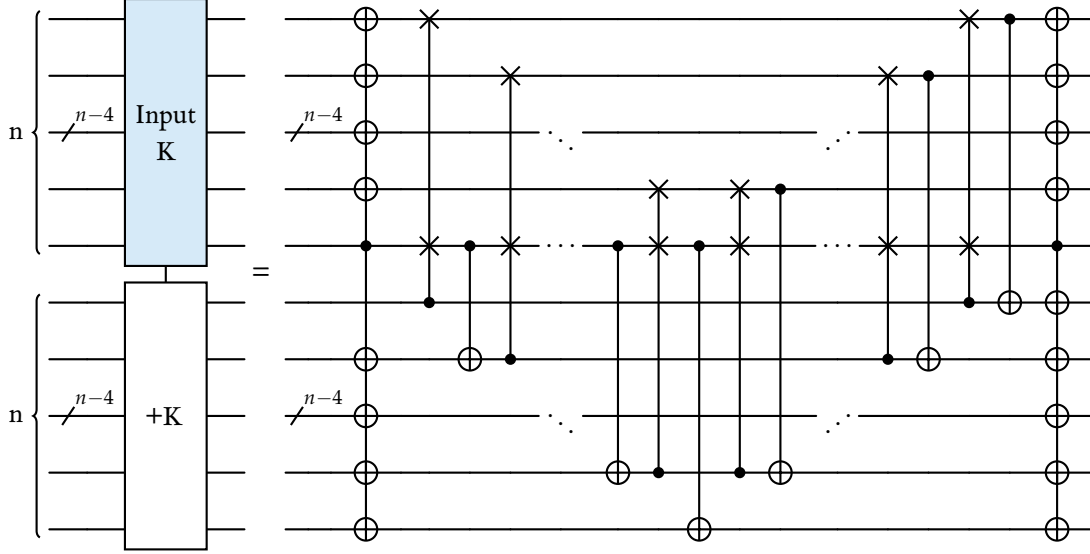
Van Rentergem Fixed-Adder The **Van Rentergem adder** implements the in-place quantum addition operator ADD_n using only CCX and CX gates, and requires no ancillary qubits. The original construction by Van Rentergem [RV05] employed Fredkin gates and included a carry-in bit. Here, we follow the simplified form of [Gid18, Fig. 15], which omits the carry-in and decomposes Fredkin gates into CCX and CX operations. The complete circuit is shown in Figure 6.11. A straightforward gate count shows:

Total CX count: $16n - 15$.

This fixed-length version assumes both registers are of size n . For a generalised case allowing differing register sizes, see [Gid18, Fig. 17]. The author's Qiskit implementation of this adder has been merged into the official repository via [pull request #14816](#).

Classical-Quantum Adders

Up to this point, our discussion has focused on **quantum-quantum adders**, where two quantum registers are summed. We now turn to **classical-quantum addition**, in which a known classical constant is added to a quantum register. Performing this operation reversibly is often more challenging than adding two quantum registers with unknown values. Intuitively, one might expect that specialising a circuit for a fixed constant would simplify its design. In practice, the opposite occurs: since the added value is known at compile time, there is no second input register available to act as workspace. Introducing additional ancillae could resolve this, but would increase the overall qubit


 Figure 6.11: The Van Rentergem adder for n qubits.

count—contrary to our goal of early fault-tolerant compatibility. Instead, we aim to construct adders that exploit **borrowable ancillae**, as introduced earlier. These designs achieve reversibility without permanent workspace overhead, maintaining low depth and qubit requirements. Formally, they are defined as follows:

Definition 6.4. Let $a \in \{0, 1\}^n$ be a classical constant and $|x\rangle \in \mathcal{H}_n$ an n -qubit register. The **in-place quantum constant adder** or **offset gate** is the unitary operator

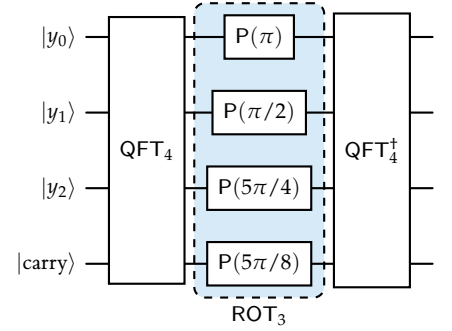
$$\text{ADD}_a : \mathcal{H}_n \rightarrow \mathcal{H}_n, \quad \text{ADD}_a |x\rangle = |(x + a) \bmod 2^n\rangle,$$

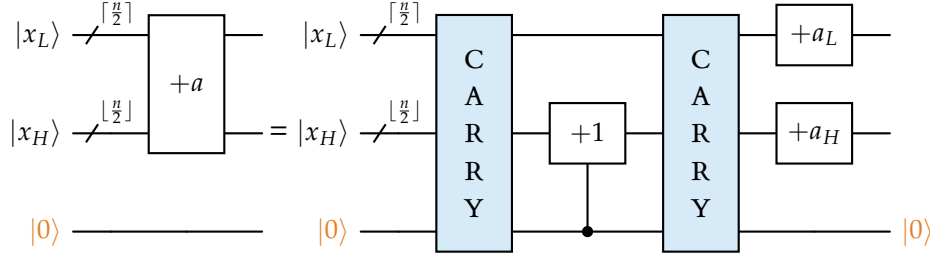
which adds the classical value a to the register x modulo 2^n without requiring additional output or workspace.

We first consider a straightforward adaptation of the Draper adder for constant addition, then present a more efficient construction that uses a single dirty ancilla qubit.

Draper Constant Adder As discussed in previously, the Draper adder performs addition via phase rotations in the Fourier basis. This construction can be specialised to offset gates. Let $c \in \{0, \dots, 2^n - 1\}$ be a known classical constant. Since the value of $|x\rangle$ is fixed, each controlled-phase gate $\text{CP}_{x_j \rightarrow y_k}$ in the CROT-layer simplifies: if $x_j = 1$, it becomes a single-qubit phase rotation on y_k ; if $x_j = 0$, it is omitted. The result is a constant-phase rotation layer that applies fixed phase shifts to the qubits of $|y\rangle$ according to the binary expansion of c . The overall transformation is $|y\rangle \mapsto |(y + c) \bmod 2^n\rangle$, implemented by the circuit

$$\text{CADD}_n(c) := \text{QFT}_n^\dagger \Phi_n(c) \text{QFT}_n,$$


 Figure 6.12: Draper constant adder circuit CADD_3 for $n = 3$ and $a = 5 = 101_2$.



where $\Phi_n(c)$ applies the pre-computed single-qubit rotations corresponding to c , as shown in Figure 6.12.

Because c is known, all controlled-phase gates in the CROT_n layer can be replaced by unconditional phase rotations. When multiple such rotations act on the same qubit, their angles may be summed at compile time, reducing the number of phase gates from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. However, the QFT_n and $\text{QFT}_n^\dagger$ layers each still require $\mathcal{O}(n^2)$ gates, so the overall circuit complexity remains quadratic. Thus, the Draper constant adder is one of the simplest reversible constructions for classical addition, but it is not optimal. Moreover, it requires high-precision phase rotations with exponentially small angles, which are challenging to implement fault-tolerantly.⁴

Häner, Roetteler, and Svore (HRS) Constant Adder To implement ADD_a more efficiently, we adopt the construction of [HRS17], which realises constant addition using only a single dirty ancilla qubit. The circuit for the offset gate is shown in Figure 6.13. The two key primitives used in this construction are the **controlled incrementer** and the **carry gate**. During the course of this project, the author implemented this adder in Qiskit, a construction with many intricate components that are seldom documented in detail. To address this gap, a comprehensive step-by-step exposition of all subroutines is provided in Appendix A, along with explicit gate counts (beyond asymptotic notation) useful for practical resource estimation. Here, we summarise the main ideas.

Definition 6.5 (Quantum Incrementer). The **quantum incrementer** INC_n is a unitary operator

$$\text{INC}_n : \mathcal{H}_n \rightarrow \mathcal{H}_n, \quad \text{INC}_n |x\rangle = |(x+1) \bmod 2^n\rangle,$$

which adds one to the binary value of its input register modulo 2^n ⁵. The controlled version of the incrementer applies the increment operation only when qubits in a control register are set to $|1\rangle$:

$$\text{CINC}_n |x\rangle |y\rangle = \begin{cases} |x\rangle |(y+1) \bmod 2^n\rangle, & \text{if } x = 1^c, \\ |x\rangle |y\rangle, & \text{otherwise.} \end{cases}$$

Figure 6.13: Recursive constant addition circuit using a single clean ancilla qubit $|g\rangle = |0\rangle$. The CARRY gate computes and stores the overflow carry of $x_L + a_L$ in g , which then controls an increment on x_H . The register x is split into lower and upper halves x_L and x_H .

⁴ An approximate QFT with complexity $\mathcal{O}(n \log n)$ [CW00] can reduce the asymptotic scaling, but large constant factors and the need for exponentially precise rotations remain.

⁵ This corresponds to an increment in two's-complement arithmetic.

Definition 6.6 (Overflow Carry Gate). Let $|x\rangle \in \mathcal{H}_n$ be the input register and $a \in \{0, 1\}^n$ a classical constant. Let $|g\rangle \in \mathcal{H}_{n-1}$ denote dirty ancillae in arbitrary states, and $|q\rangle \in \mathcal{H}_1$ a clean target qubit. The **carry gate** CARRY_n^a is a reversible unitary that computes the overflow bit of $x + a$:

$$\text{CARRY}_n^a |x\rangle |g\rangle |0\rangle_q = |x\rangle |g\rangle |\text{carry}(x, a)\rangle,$$

where $\text{carry}(x, a) = 1$ iff $x + a \geq 2^n$, and 0 otherwise.⁶ All dirty ancillae are restored to their initial states. The detailed Boolean logic, reversible realisation, and gate-count analysis of the carry gate is provided in [Appendix A](#).

⁶ The name “carry gate” is somewhat misleading: it outputs only the global overflow bit, not a full carry chain.

Having defined these components, the key idea now is that the modular sum

$$|x\rangle \mapsto |(x + a) \bmod 2^n\rangle$$

can be decomposed into two half-sized additions, coupled only by a single overflow carry bit. Consider $|x\rangle = |x_H\rangle |x_L\rangle$ and $a = a_H || a_L$, where each half has size $n/2$. The addition proceeds as follows:

- **Base case:** For $n = 1$, ADD_a reduces to a single-bit XOR,

$$\text{ADD}_a |x\rangle = |x \oplus a\rangle,$$

implemented by applying X if $a = 1$.

- **Recursive case:** For $n > 1$, the algorithm performs:
 1. Compute the carry-out of $x_L + a_L$ via $\text{CARRY}_{n/2}^{a_L}$ using $|x_H\rangle$ as dirty ancillae and a clean ancilla $|q\rangle$ to store the overflow:

$$|x_L\rangle |x_H\rangle |0\rangle \mapsto |x_L\rangle |x_H\rangle |\text{carry}(x_L, a_L)\rangle.$$

2. Conditionally increment $|x_H\rangle$ using $|q\rangle$ as control:

$$|x_H\rangle |q\rangle \mapsto |(x_H + q) \bmod 2^{n/2}\rangle |q\rangle.$$

3. Recursively apply ADD_{a_L} on $|x_L\rangle$ and ADD_{a_H} on $|x_H\rangle$.

Why it works? This construction works because binary addition can be decoupled at the midpoint once the carry-out from the lower half is known.

- The carry from $x_L + a_L$ is computed first and propagated to x_H through a controlled increment⁷.
- The recursive additions then apply independently to each half, completing the modular sum.

⁷ The lower half $|x_L\rangle$ remains unchanged during this step.

Each recursion level halves the register size and performs $\mathcal{O}(n)$ work, yielding an overall cost of $\mathcal{O}(n \log n)$ gates using only a single clean ancilla.

Using a Dirty Ancilla. The clean-ancilla construction of Figure 6.13 can be adapted to use a borrowed dirty qubit, enabling fully in-place constant addition without additional clean workspace. At a high level, the dirty-ancilla version replaces the single controlled increment with a symmetric pair of increment-decrement operations that conditionally cancel depending on the carry bit, ensuring the borrowed qubit is restored to its initial state. The circuit is shown in Figure 6.14.

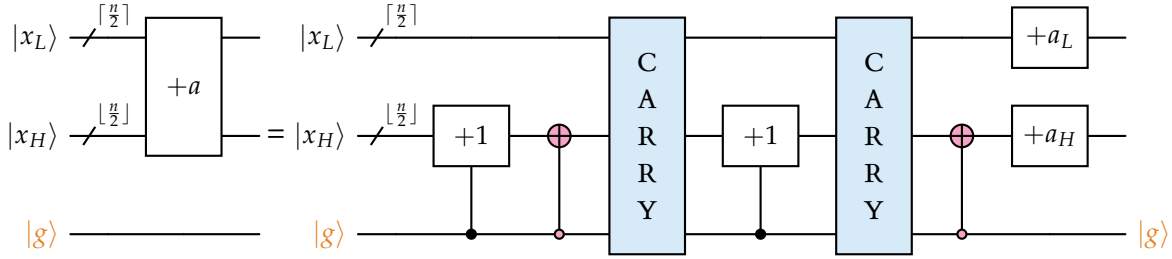


Figure 6.14: Same as Figure 6.13, but using a **dirty** ancilla qubit g instead of a clean one. The pink control gates are MCMT gates that act on all higher-order bits of x_H .

Constant-Specific Optimisations. The efficiency of the constant adder can be further improved when the constant a has trailing zeros in its binary expansion. In such cases, part of the register remains unaffected by the addition, allowing the recursive circuit to shrink proportionally. A detailed analysis and complexity derivation are provided in Appendix A.

Controlled Classical-Quantum Adders

We now extend the constant-adder construction to the **controlled** setting, where addition by a classical constant is conditioned on a quantum control register. Such controlled adders are essential in block-encoding circuits, as discussed later in Chapter 7.⁸

Naively applying the control condition to every one- and two-qubit gate within the constant would lead to severe overhead since each gate becomes multi-controlled on the entire control register. Instead, we employ more efficient constructions that factor the control logic structurally rather than gate-by-gate. We denote the control register by $|\text{ctrl}\rangle = |c_0\rangle \otimes \cdots \otimes |c_{m-1}\rangle$ where $c_i \in \{0, 1\}$, and the n -qubit target register as the system on which the classical constant is added. Although not used directly in our implementation, the controlled Draper adder is reviewed in Appendix A for completeness and comparison with the approaches adopted here.

Controlled Constant Adder Extending the constant adder to its controlled form requires only minimal changes to the underlying circuit—surprisingly, the only modification is replacing each CARRY_n^a gate, with its controlled

⁸ In general, we require **multiplexed** addition.

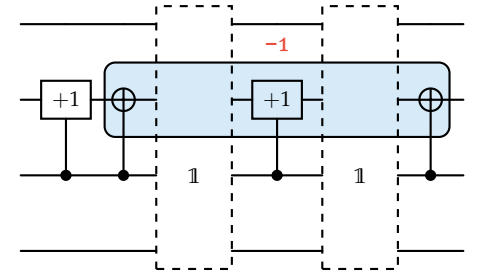
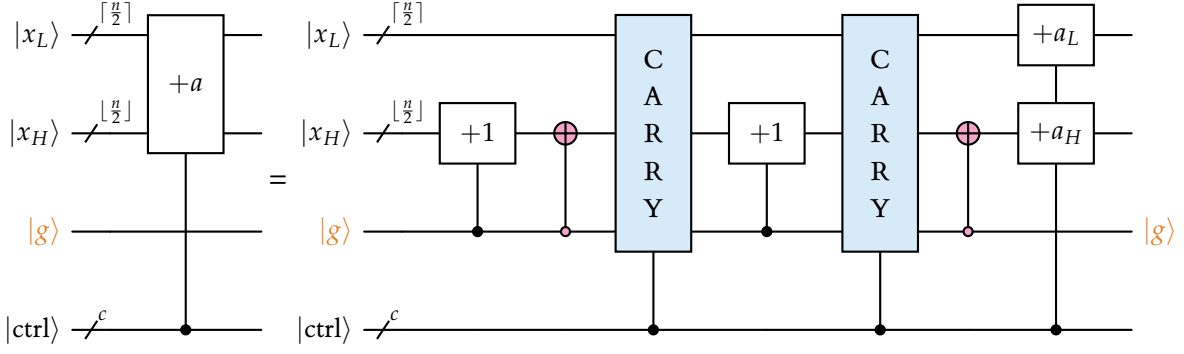


Figure 6.15: Subcircuit of the offset gate showing cancellation behaviour when the control condition is not satisfied.



counterpart; all other gates remain unchanged, yielding a remarkably simple and elegant construction as shown in Figure 6.16.

Why it works? Consider the case where the control condition is inactive. The controlled CARRY_n^a gates act as identities, leaving the carry register $|g\rangle$ untouched. As illustrated in Figure 6.15:

- The first incrementer acts on the high bits.
- The second incrementer, **sandwiched** between two controlled MCMT gates, becomes a decrementer.
- This controlled decrementer cancels the previous incrementer, leaving the high bits unchanged.
- The same cancellation pattern recurses throughout the circuit.

When the control condition is active, the controlled CARRY_n^a gates operate exactly as in the original adder, computing the carries and triggering the correct sequence of increment operations. The resulting circuit thus implements controlled constant addition faithfully.⁹ The CX-gate complexity is

$$\text{Total CX count} = \mathcal{O}((n - k) \log(n - k) + c),$$

where k is the number of trailing zeros in the binary expansion of a .

Controlled Carry Gate. The controlled version of the CARRY_n^a gate extends the clean-ancilla carry logic to the controlled setting, following the same compute-uncompute principle as before. A detailed circuit construction, together with its gate-count analysis, is provided in Appendix A.

Figure 6.17 compares the asymptotic and empirical scaling of the controlled Draper and the HRS constant adders in terms of CX-gate count as a function of register size. As expected, Draper-based adders exhibit quadratic scaling

Figure 6.16: Controlled version of Figure 6.14. Only the CARRY_n^a gates are controlled; all other gates remain unchanged.

⁹ Only the dirty-ancilla case is shown; the clean-ancilla variant follows trivially.

with n , while the HRS construction achieves a near-linear $\mathcal{O}(n \log n)$ scaling, making it substantially more efficient for large registers. The horizontal axis in the figure extends up to $n = 33$ qubits, which, as we will later see, corresponds to operations on a “data” subspace of dimension 2^n . Hence, $n = 30$ already represents arithmetic over a billion-dimensional state space—precisely the scale at which quantum computers are expected to outperform their classical counterparts.

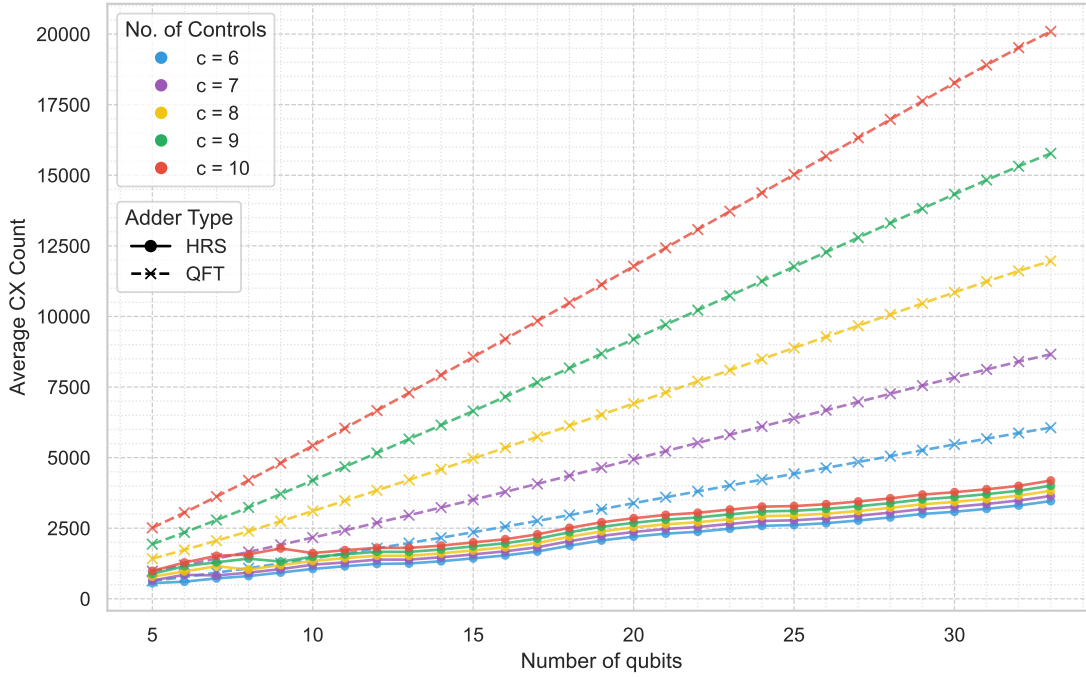


Figure 6.17: Comparison of CX-gate counts for Draper vs. HRS constant adders.

6.4 Proxy Phasing

In quantum algorithms, we often need to apply phase rotations conditioned on a particular quantum state. When the state is entangled or otherwise difficult to isolate, applying such a phase directly can be nontrivial. A useful technique called **proxy phasing** [Cra17] addresses this challenge. The core idea is simple: instead of applying a phase directly to the target state, we temporarily “copy” information about the state into an ancilla qubit, apply the phase there, and then uncompute the ancilla. Although the phase is physically applied to a different qubit, the net effect on the system is the same.

To illustrate the idea, suppose we wish to apply $P(\theta)$ to a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, as if acting directly on $|\psi\rangle$. The circuit in Figure 6.18

achieves this using an ancilla:

1. Start in state $\alpha |0\rangle + \beta |1\rangle$.
2. Add an ancilla in $|0\rangle$, giving $(\alpha |0\rangle + \beta |1\rangle) \otimes |0\rangle = \alpha |00\rangle + \beta |10\rangle$.
3. Apply a CX-gate (control: original qubit; target: ancilla): $\alpha |00\rangle + \beta |11\rangle$.
4. Apply $P(\theta)$ to the ancilla qubit: $\alpha |00\rangle + e^{i\theta} \beta |11\rangle$.
5. Uncompute with another CX-gate: $\alpha |00\rangle + \beta e^{i\theta} |10\rangle = (\alpha |0\rangle + e^{i\theta} \beta |1\rangle) |0\rangle$.

The ancilla returns to its original state $|0\rangle$, while the desired phase $e^{i\theta}$ has been applied to the $|1\rangle$ component of $|\psi\rangle$, exactly as intended.

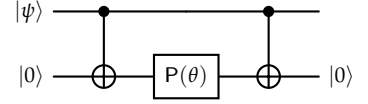


Figure 6.18: Proxy phasing.

6.5 Projector Controlled-Unitary Gate

Let $U \in \mathbb{U}(2^n)$, and Π be an orthogonal projector of size $2^m \times 2^m$. The **projector-controlled U gate** is an operation on $m + n$ qubits defined as:

$$U \otimes \Pi + \mathbb{1}_n \otimes (\mathbb{1}_m - \Pi).$$

This gate applies U to the target register when the control register lies in the subspace onto which Π projects; otherwise, it applies the identity. When $\Pi = |1\rangle\langle 1|$, we often refer to the resulting gate as a controlled- U gate.

Projector-controlled NOT Gate

The **projector-controlled NOT gate** is a special instance of a projector-controlled unitary gate, where the unitary U is the NOT gate (i.e., X) defined by the operator:

$$C_{\Pi}\text{NOT} := X \otimes \Pi + \mathbb{1}_2 \otimes (\mathbb{1}_{2^m} - \Pi).$$

This construction can be used to implement an analogue of a projective measurement, allowing quantum algorithms to apply operations conditioned on subspace membership without collapsing the state. A canonical example is the standard CX gate, which corresponds to a projector-controlled X gate where the control projector is $|1\rangle\langle 1|$:

$$X \otimes |1\rangle\langle 1| + \mathbb{1}_2 \otimes |0\rangle\langle 0|.$$

Note 4. Up to a Hadamard conjugation on the target qubit, the $C_{\Pi}\text{NOT}$ -gate implements a controlled reflection of the form $\mathbb{1} - 2\Pi$. This structure appears in Grover's algorithm and amplitude amplification, where reflections over subspaces are essential.

Projector-controlled Phase Shift Gate

Let $\Pi \in \mathbb{C}^{2^m \times 2^m}$ be an orthogonal projector and let $\varphi \in \mathbb{R}$ be a phase angle. The **projector-controlled phase shift gate** is defined by:

$$\Pi_\varphi := e^{i\varphi(2\Pi - \mathbb{1}_{2^m})}.$$

This operator applies a relative phase to quantum states based on whether they lie in the image or the orthogonal complement of Π . We can expand the operator as:¹⁰

$$\begin{aligned} \Pi_\varphi &= e^{i\varphi(2\Pi - \mathbb{1})} = \cos(\varphi)\mathbb{1} + i\sin(\varphi)(2\Pi - \mathbb{1}) \\ &= \frac{1}{2}(e^{i\varphi} + e^{-i\varphi})\mathbb{1} + \frac{1}{2}(e^{i\varphi} - e^{-i\varphi})(2\Pi - \mathbb{1}) \quad (6.3) \\ &= e^{-i\varphi}\mathbb{1} + (e^{i\varphi} - e^{-i\varphi})\Pi. \end{aligned}$$

To understand the action of Π_φ , let $|\psi_\Pi\rangle$ be a state that lies in the image of Π , and $|\psi_\perp\rangle$ be orthogonal to it. Since $\Pi|\psi_\Pi\rangle = |\psi_\Pi\rangle$ and $\Pi|\psi_\perp\rangle = 0$, it follows that:

$$\Pi_\varphi |\psi_\Pi\rangle = e^{i\varphi} |\psi_\Pi\rangle, \quad \Pi_\varphi |\psi_\perp\rangle = e^{-i\varphi} |\psi_\perp\rangle. \quad (6.4)$$

Up to a global phase of $e^{i\varphi}$, (6.4) is equivalent to:

$$\Pi_\varphi |\psi_\Pi\rangle = e^{2i\varphi} |\psi_\Pi\rangle, \quad \Pi_\varphi |\psi_\perp\rangle = |\psi_\perp\rangle,$$

which is a selective phase shift on the subspace defined by Π .

The unitary Π_φ can be implemented using a quantum circuit that applies a phase rotation to a target qubit conditioned on whether the control register lies in the image of Π . The construction uses a pair of C_Π NOT-gates and a single-qubit phase rotation:

$$\begin{aligned} \Pi_\varphi &= (\mathbb{X} \otimes \Pi + \mathbb{1} \otimes (\mathbb{1} - \Pi)) (e^{-i\varphi Z} \otimes \mathbb{1}) (\mathbb{X} \otimes \Pi + \mathbb{1} \otimes (\mathbb{1} - \Pi)) \\ &= \mathbb{X} e^{-i\varphi Z} \mathbb{X} \otimes \Pi^2 + e^{-i\varphi Z} \mathbb{X} \otimes (\mathbb{1} - \Pi) \overset{0}{\xrightarrow{\Pi}} \quad (\because \Pi^2 = \Pi) \\ &\quad + \mathbb{X} e^{-i\varphi Z} \otimes \Pi(\mathbb{1} - \Pi) \overset{0}{\xrightarrow{\Pi}} + e^{-i\varphi Z} \otimes (\mathbb{1} - \Pi)^2 \\ &= \mathbb{X} e^{-i\varphi Z} \mathbb{X} \otimes \Pi + e^{-i\varphi Z} \otimes (\mathbb{1} - \Pi). \end{aligned} \quad (6.5)$$

We verify that this operator behaves as expected when applied to $|0\rangle \otimes |\psi\rangle$:

- If $|\psi\rangle \in \text{Im}(\Pi)$, then:

$$\begin{aligned} \Pi_\varphi (|0\rangle \otimes |\psi_\Pi\rangle) &= \mathbb{X} e^{-i\varphi Z} \mathbb{X} |0\rangle \otimes |\psi_\Pi\rangle \\ &= |0\rangle \otimes e^{i\varphi} |\psi_\Pi\rangle. \end{aligned} \quad (6.6)$$

- If $|\psi\rangle \in \ker(\Pi)$, then:

$$\begin{aligned} \Pi_\varphi (|0\rangle \otimes |\psi_\perp\rangle) &= e^{-i\varphi Z} |0\rangle \otimes |\psi_\perp\rangle \\ &= |0\rangle \otimes e^{-i\varphi} |\psi_\perp\rangle. \end{aligned} \quad (6.7)$$

¹⁰ $e^{i\varphi A} = \cos(\varphi)\mathbb{1} + i\sin(\varphi)A$ for any operator A such that $A^2 = \mathbb{1}$.

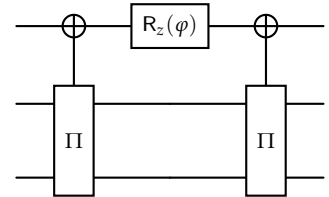


Figure 6.19: Projector-controlled phase shift gate.

Again, up to a global phase $e^{i\varphi}$, this results in a 2φ phase shift on the subspace defined by Π , and no change on its orthogonal complement. The circuit implementing this operator is shown in [Figure 6.19](#).

Note 5 (Connection to Proxy Phasing). *The projector-controlled phase shift gate $\Pi_\varphi = e^{i\varphi(2\Pi-\mathbb{1})}$ can be viewed as a generalisation of the **proxy phasing** technique [Cra17] introduced earlier. Proxy phasing handles the special case where the projector Π corresponds to a single computational basis state (e.g., $|1^n\rangle\langle 1^n|$), and the phase is applied using an ancilla qubit and a pair of MCX-gates. The projector-controlled gate extends this idea to arbitrary projectors Π , allowing phases to be applied coherently based on subspace membership rather than exact classical patterns. Both constructions reflect the same core principle: introducing a conditional phase without directly measuring or collapsing the system.*

6.6 Linear Combination of Unitaries

Suppose we wish to implement a unitary operator V that is expressed as a linear combination of unitary operators U_i

$$V = \sum_{i=1}^k \alpha_i U_i,$$

where $\alpha_i \in \mathbb{C}$ are complex coefficients. For simplicity, we assume that all α_i are real and non-negative. This is without loss of generality, since any complex phase in α_i can be absorbed into U_i by redefining $U_i \leftarrow e^{i\theta_i} U_i$ for appropriate phases θ_i .¹¹ The **linear combination of unitaries** (LCU) method provides a way to implement such an operator V efficiently using a quantum circuit.

Assume that we can implement each U_i in a controlled fashion. Specifically, we assume access to a two-register unitary known as the **SELECT** operator, defined by:

$$\text{SELECT}(U) := \sum_{i=1}^k |i\rangle\langle i| \otimes U_i.$$

This operator applies U_i to the target register, conditioned on the control register being in the state $|i\rangle$.¹²

$$\text{SELECT}(U) (|i\rangle \otimes |\psi\rangle) = |i\rangle \otimes U_i |\psi\rangle.$$

We also require a unitary operator A , called the **PREP** operator, which prepares a weighted superposition over the control states:

$$A |0^m\rangle = \frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle, \quad \text{where } \alpha := \sum_{i=1}^k \alpha_i. \quad (6.8)$$

We now describe how to combine the PREP and SELECT operators to implement the LCU operator $V = \sum_i \alpha_i U_i$.

¹¹ This is equivalent to applying a global phase to the circuit implementing U_i .

¹² The control register selects which unitary U_i is applied to the target.

1. Prepare the initial state $|0^m\rangle \otimes |\psi\rangle$, where $|\psi\rangle$ is the input state and $m = \lceil \log_2 k \rceil$ is the number of control qubits.
2. Apply the PREP operator A to the control register:

$$A |0^m\rangle \otimes |\psi\rangle = \frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes |\psi\rangle.$$

3. Apply the SELECT operator $U := \sum_i |i\rangle \langle i| \otimes U_i$ to obtain:

$$U \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes |\psi\rangle \right) = \frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle.$$

4. Applying $A^\dagger$ to the control register yields:

$$A^\dagger \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle \right).$$

Define the projector onto the control state $|0^m\rangle$ as:

$$\Pi := |0^m\rangle \langle 0^m| \otimes \mathbb{1}.$$

To isolate the part of the state where the control register returns to $|0^m\rangle$, insert the identity $\mathbb{1} = \Pi + (\mathbb{1} - \Pi)$ and decompose the state as:

$$\begin{aligned} & \Pi A^\dagger \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle \right) + (\mathbb{1} - \Pi) A^\dagger \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle \right) \\ &= (|0^m\rangle \otimes \mathbb{1}) \left(\frac{1}{\sqrt{\alpha}} \sum_{j=1}^k \sqrt{\alpha_j} \langle j| \otimes \mathbb{1} \right) \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle \right) + |\Phi^\perp\rangle \\ &= \frac{1}{\alpha} \sum_{i=1}^k |0^m\rangle \otimes \alpha_i U_i |\psi\rangle + |\Phi^\perp\rangle \\ &= \frac{1}{\alpha} |0^m\rangle \otimes V |\psi\rangle + |\Phi^\perp\rangle, \end{aligned}$$

Since $A |0^m\rangle = \frac{1}{\sqrt{\alpha}} \sum_j \sqrt{\alpha_j} |j\rangle$, it follows that $\langle 0^m| A^\dagger = \frac{1}{\sqrt{\alpha}} \sum_j \sqrt{\alpha_j} \langle j|$.

where the orthogonal component is defined by:

$$|\Phi^\perp\rangle := (\mathbb{1} - \Pi) A^\dagger \left(\frac{1}{\sqrt{\alpha}} \sum_{i=1}^k \sqrt{\alpha_i} |i\rangle \otimes U_i |\psi\rangle \right).$$

By construction, $\Pi |\Phi^\perp\rangle = 0$, since $\Pi(\mathbb{1} - \Pi) = 0$.

Thus, with probability $p = 1/\alpha^2$, a projective measurement on the control register yields $|0^m\rangle$, and the target register contains $V |\psi\rangle$. The failure component $|\Phi^\perp\rangle$ lies entirely in the orthogonal subspace. An example quantum circuit implementing the LCU method is shown in [Figure 6.20](#).

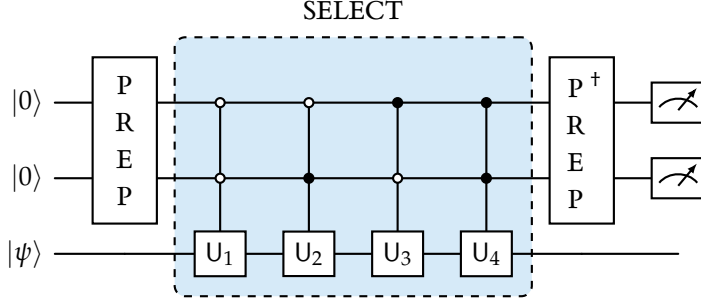


Figure 6.20: LCU circuit for $k = 4$, using two control qubits (since $\lceil \log_2 4 \rceil = 2$)

Oblivious Amplitude Amplification

We have seen that the LCU construction implements the operator V with success amplitude $1/\alpha$, so the overall success probability is $p = 1/\alpha^2$. We now describe how to boost this probability to 1 using a variant of amplitude amplification [BHMT02] known as **oblivious amplitude amplification** (OAA) [BCC⁺14]. From the LCU construction, we obtain a unitary $W := A^\dagger U A$ that acts as:

$$W |0^m\rangle |\psi\rangle = \sin(\theta) |0^m\rangle V |\psi\rangle + \cos(\theta) |\Phi^\perp\rangle,$$

where $\theta = \arcsin(1/\alpha)$, and $|\Phi^\perp\rangle$ is orthogonal to the subspace in which the control register is $|0^m\rangle$. To isolate the desired component $|0^m\rangle V |\psi\rangle$, we define the projector: $\Pi := |0^m\rangle \langle 0^m| \otimes \mathbb{1}$ so that $\Pi |\Phi^\perp\rangle = 0$ by construction. We now introduce a reflection operator about this subspace:

$$R := 2\Pi - \mathbb{1},$$

and define the composite unitary:

$$S := -WRW^\dagger R.$$

The operator S acts as an **amplitude amplifier**: repeated applications of S increase the amplitude on the desired subspace, without requiring any knowledge of $|\psi\rangle$. This motivates the name **oblivious amplitude amplification** (OAA).¹³ The following lemma formalises this process:

Lemma 6.1 (Oblivious Amplitude Amplification [BCC⁺14, Lemma 3.6]). *Let W and V be unitary matrices on $n + m$ qubits and n qubits respectively, and let $\theta \in (0, \pi/2)$. Suppose that for any n -qubit state $|\psi\rangle$,*

$$W |0^m\rangle |\psi\rangle = \sin(\theta) |0^m\rangle V |\psi\rangle + \cos(\theta) |\Phi^\perp\rangle,$$

where $|\Phi^\perp\rangle$ is an $(n + m)$ -qubit state that depends on $|\psi\rangle$ and satisfies $\Pi |\Phi^\perp\rangle = 0$, where $\Pi = |0^m\rangle \langle 0^m| \otimes \mathbb{1}$. Let $R := 2\Pi - \mathbb{1}$ and $S := -WRW^\dagger R$. Then for any $t \in \mathbb{Z}$,

$$S^t W |0^m\rangle |\psi\rangle = \sin((2t + 1)\theta) |0^m\rangle V |\psi\rangle + \cos((2t + 1)\theta) |\Phi^\perp\rangle.$$

¹³ This is a key distinction from standard amplitude amplification, which typically requires knowledge of the input state $|\psi\rangle$ to construct the relevant reflections. In contrast, OAA amplifies the desired component without needing any access to or assumptions about $|\psi\rangle$.

Hence, if we can choose an integer t such that $(2t + 1)\theta = \frac{\pi}{2}$, then

$$S^t W |0^m\rangle |\psi\rangle = |0^m\rangle V |\psi\rangle,$$

yielding a perfect implementation of V using $\mathcal{O}(t) = \mathcal{O}(\alpha)$ applications of W and $W^\dagger$. We now consider two cases, depending on whether such a t exists.

Case 1: $\frac{\pi}{2\theta}$ is an odd integer. In this case, there exists $t \in \mathbb{Z}$ such that $2t + 1 = \frac{\pi}{2\theta}$. The amplification is exact:¹⁴

¹⁴ since $\sin((2t + 1)\theta) = \sin(\pi/2) = 1$

$$S^t W |0^m\rangle |\psi\rangle = |0^m\rangle V |\psi\rangle.$$

This gives a perfect implementation of V using $\mathcal{O}(\alpha)$ applications of W and $W^\dagger$, and corresponds to the ideal setting.

Case 2: $\frac{\pi}{2\theta}$ is not an odd integer. Let $2t + 1$ be the smallest odd integer greater than $\frac{\pi}{2\theta}$, and define θ' such that¹⁵

¹⁵ Then $\sin(\theta') < \sin(\theta) = \frac{1}{\alpha}$.

$$(2t + 1)\theta' = \frac{\pi}{2}, \quad \text{so that } \theta' < \theta.$$

Introduce a single ancilla qubit in the state $|0\rangle$ and apply a $R_y(2\varphi)$ rotation such that

$$R_y(2\varphi) |0\rangle = |0\rangle + \sqrt{1 - r^2} |1\rangle, \quad \text{where } r = \frac{\sin(\theta')}{\sin(\theta)}, \quad \varphi = \arccos(r).$$

Now define a modified unitary:

$$W' := (R_y(2\varphi) \otimes \mathbb{1}) W.$$

This creates the state:

$$\begin{aligned} W'(|0\rangle |0^m\rangle |\psi\rangle) &= (R_y(2\varphi) \otimes \mathbb{1}) \left(\sin(\theta) |0\rangle |0^m\rangle V |\psi\rangle + \cos(\theta) |0\rangle |\Phi^\perp\rangle \right) \\ &= r \sin(\theta) |0\rangle |0^m\rangle V |\psi\rangle + r \cos(\theta) |0\rangle |\Phi^\perp\rangle \\ &\quad + \sqrt{1 - r^2} \sin(\theta) |1\rangle |0^m\rangle V |\psi\rangle \\ &\quad + \sqrt{1 - r^2} \cos(\theta) |1\rangle |\Phi^\perp\rangle. \end{aligned}$$

The remaining terms lie in a subspace orthogonal to $|0\rangle |0^m\rangle V |\psi\rangle$. Let us verify that the remainder has norm $\cos(\theta')$. The total squared amplitude of the orthogonal component is:

Since $r \sin(\theta) = \sin(\theta')$, the amplitude of $|0\rangle |0^m\rangle V |\psi\rangle$ is now $\sin(\theta')$.

$$\begin{aligned} &(r \cos(\theta))^2 + (\sqrt{1 - r^2} \sin(\theta))^2 + (\sqrt{1 - r^2} \cos(\theta))^2 \\ &= r^2 \cos^2(\theta) + (1 - r^2)(\sin^2(\theta) + \cos^2(\theta)) \\ &= r^2 \cos^2(\theta) + (1 - r^2)(1) = 1 - r^2 \sin^2(\theta) \\ &= 1 - \sin^2(\theta') = \cos^2(\theta'). \end{aligned}$$

Taking the square root confirms that the remaining state has norm $\cos(\theta')$, so we obtain:

$$W'(|0\rangle|0^m\rangle|\psi\rangle) = \sin(\theta')|0\rangle|0^m\rangle V|\psi\rangle + \cos(\theta')|\Phi'^\perp\rangle.$$

We are now back in Case 1, with effective success angle θ' such that $(2t + 1)\theta' = \pi/2$. Therefore, we can apply S'^t with:

$$S' := -W'RW'^\dagger R$$

to obtain a perfect implementation of V using $\mathcal{O}(t) = \mathcal{O}(\alpha)$ gates. This completes the construction of oblivious amplitude amplification and its application to the LCU method.

Note 6. A key assumption in the LCU construction is that the operator V is exactly unitary. In practice, we are often interested in situations where V is only δ -close (in spectral norm) to some unitary $\tilde{V}$, where $\tilde{V}$ itself can be expressed as a linear combination of unitaries U_i . In such cases, the standard OAA procedure must be adapted to handle approximation error, leading to what is known as approximate oblivious amplitude amplification [Kot14, Theorem 2.5] or robust oblivious amplitude amplification [BCK15, Lemma 6].

7

Block-Encoding Framework for Structured Linear Operators

Block-encoding provides a unifying framework for representing general, non-unitary linear operators within the unitary dynamics of a quantum computer. In essence, a matrix $A \in \mathbb{C}^{N \times N}$ is embedded as a scaled sub-block of a larger unitary $U \in \mathbb{U}(2^n)$. This construction enables non-unitary operations to be manipulated within coherent quantum circuit model using only unitary primitives. The present chapter develops this concept in full generality: we first formalise the mathematical definition of block-encodings and establish their closure under linear combinations and products, thereby showing how complex operators can be composed from simpler constituents. We then extend these foundations to structured matrices, introducing systematic procedures for generating the required data oracles and synthesising their corresponding circuits. Together, these developments provide the algorithmic backbone for the quantum singular value transformation and PDE solvers constructed in subsequent chapters.

7.1 Definition

Definition 7.1 (Block-encoding, general form [GSLW19, Definition 11]). Let $A \in \mathbb{C}^{r \times c}$ and let $d \geq \max\{r, c\}$. We say that $U \in \mathbb{C}^{d \times d}$ is a **block-encoding** of A if there exist isometries $B_{L,1} \in \mathbb{C}^{d \times r}$, $B_{R,1} \in \mathbb{C}^{d \times c}$ (with orthonormal columns) satisfying

$$B_{L,1}^\dagger U B_{R,1} = A. \quad (7.1)$$

Equivalently, choose any **unitary completions** $B_L = \begin{bmatrix} B_{L,1} & B_{L,2} \end{bmatrix}$ and $B_R = \begin{bmatrix} B_{R,1} & B_{R,2} \end{bmatrix}$ so that $B_L, B_R \in \mathbb{U}(d)$.¹ Then

$$\begin{bmatrix} B_{L,1}^\dagger U B_{R,1} & B_{L,1}^\dagger U B_{R,2} \\ B_{L,2}^\dagger U B_{R,1} & B_{L,2}^\dagger U B_{R,2} \end{bmatrix} = \begin{bmatrix} A & * \\ * & * \end{bmatrix},$$

¹ If $V \in \mathbb{C}^{d \times r}$ has orthonormal columns, a **unitary completion** is any $U = \begin{bmatrix} V & V_\perp \end{bmatrix} \in \mathbb{C}^{d \times d}$ with $U^\dagger U = \mathbb{1}_d$, where $V_\perp$ spans $\ker(V^\dagger)$.

so A appears as the top-left $r \times c$ block of U in the (B_L, B_R) basis. Let $\Pi_L := B_{L,1} B_{L,1}^\dagger$ and $\Pi_R := B_{R,1} B_{R,1}^\dagger$ denote the projectors onto the encoding subspaces. Then

$$B_L^\dagger (\Pi_L U \Pi_R) B_R = \begin{bmatrix} \mathbb{1}_r \\ \mathbf{0} \end{bmatrix} (B_{L,1}^\dagger U B_{R,1}) \begin{bmatrix} \mathbb{1}_c & \mathbf{0} \end{bmatrix} = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

We typically use the special case where the encoding subspaces align with the computational basis. Taking $B_{L,1} = \begin{bmatrix} \mathbb{1}_r \\ \mathbf{0} \end{bmatrix}$ and $B_{R,1} = \begin{bmatrix} \mathbb{1}_c \\ \mathbf{0} \end{bmatrix}$ (the first r and c columns of $\mathbb{1}_d$) gives

$$\Pi_L = \begin{bmatrix} \mathbb{1}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \Pi_R = \begin{bmatrix} \mathbb{1}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \implies \Pi_L U \Pi_R = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

When d, r, c are powers of two, all operators can be expressed as qubit unitaries. In this case, (7.1) becomes

$$(\langle 0^{a_L} | \otimes \mathbb{1}_r) U (| 0^{a_R} \rangle \otimes \mathbb{1}_c) = A,$$

where $a_L = \log_2(d/r)$ and $a_R = \log_2(d/c)$. Unless stated otherwise, we adopt this computational-basis formulation in what follows.

Definition 7.2 (Approximate block-encoding). Let A be an n -qubit operator ($N = 2^n$). A unitary U_A acting on $m + n$ qubits is an (α, m, ε) -**block-encoding** of A if there exist parameters $\alpha, \varepsilon \in \mathbb{R}_+$ such that²

$$\|A - \alpha (\langle 0^m | \otimes \mathbb{1}_N) U_A (| 0^m \rangle \otimes \mathbb{1}_N)\| \leq \varepsilon,$$

where m denotes the number of ancilla qubits. Equivalently, U_A admits the block structure³

$$U_A = \begin{bmatrix} \frac{A}{\alpha} + E & * \\ * & * \end{bmatrix}, \quad \|E\| \leq \varepsilon/\alpha,$$

where the top-left block corresponds to the subspace with ancilla qubits in state $|0^m\rangle$.

We now record a simple bound on the operator norm of an encoded matrix.

Proposition 7.1. *If U_A is an (α, m, ε) -block-encoding of A , then*

$$\|A\| \leq \alpha + \varepsilon.$$

Proof.

By definition,

$$A = (A - \alpha (\langle 0^m | \otimes \mathbb{1}) U_A (| 0^m \rangle \otimes \mathbb{1})) + \alpha (\langle 0^m | \otimes \mathbb{1}) U_A (| 0^m \rangle \otimes \mathbb{1}).$$

² The norm here is the spectral norm.

³ The exact form (7.1) is sometimes called a **projected encoding**, while the approximate version above is often referred to simply as a **block-encoding** [GSLW19]. We adopt this latter convention.

Applying the triangle inequality,

$$\|A\| \leq \|A - \alpha(\langle 0^m | \otimes \mathbb{1}) U_A (|0^m\rangle \otimes \mathbb{1})\| + \alpha \|(\langle 0^m | \otimes \mathbb{1}) U_A (|0^m\rangle \otimes \mathbb{1})\|.$$

The first term is at most ε by assumption. For the second, note that $(\langle 0^m | \otimes \mathbb{1}) U_A (|0^m\rangle \otimes \mathbb{1}) = Q U_A Q^\dagger$ for some isometry Q with $\|Q\| = 1$, hence the product has norm at most 1. Thus, $\|A\| \leq \alpha + \varepsilon$. ■

Lemma 7.2. *Let U_A be an $(\alpha, m, 0)$ -block-encoding of an operator A , and let $|\psi\rangle$ be an arbitrary state. Then there exists a quantum circuit that prepares the normalised state*

$$\frac{A|\psi\rangle}{\|A|\psi\rangle\|}$$

with success probability $\|A|\psi\rangle\|^2/\alpha^2$.

Proof.

Any block-encoding U_A can be written as

$$U_A = |0^m\rangle\langle 0^m| \otimes \frac{A}{\alpha} + \sum_{\neg(x=0^m \wedge y=0^m)} |x\rangle\langle y| \otimes C_{xy},$$

where each C_{xy} is an $N \times N$ operator acting on the data register. Let $|\psi\rangle$ be an arbitrary state. The action of U_A on the joint state $|0^m\rangle |\psi\rangle$ of ancilla and data registers is

$$U_A(|0^m\rangle |\psi\rangle) = |0^m\rangle \frac{A|\psi\rangle}{\alpha} + \sum_{x \neq 0^m} |x\rangle C_{x0^m} |\psi\rangle = |0^m\rangle \frac{A|\psi\rangle}{\alpha} + \beta |\perp\rangle,$$

where $|\perp\rangle$ is normalised and orthogonal to $|0^m\rangle A|\psi\rangle$.⁴ Projecting onto the ancilla outcome $|0^m\rangle$ thus succeeds with probability $\|A|\psi\rangle\|^2/\alpha^2$, and in this case the post-measurement state is proportional to $A|\psi\rangle$, as claimed. ■

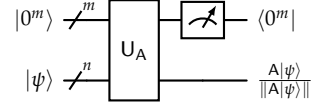


Figure 7.1: Quantum circuit for block-encoding. Post-selecting the ancilla outcome 0^m prepares the data register in a state proportional to $A|\psi\rangle$.

7.2 Extensibility Properties of Block-Encodings

Having established how a single block-encoding can probabilistically apply its encoded operator to a quantum state, we now turn to how such block-encodings can be manipulated and combined.

Linear Combinations of Block-encodings

In Section 6.6, we described how LCU can be implemented efficiently. The same principle extends to block-encodings: by combining multiple block-encodings, we can construct a new block-encoding that represents their linear combination.

⁴ The first term has ancilla $|0^m\rangle$; the remaining terms involve orthogonal ancilla states $|x \neq 0^m\rangle$.

Definition 7.3 (State Preparation Pair [GSLW19, Definition 51]). Let $y \in \mathbb{C}^m$ with $\|y\|_1 \leq \beta$ for some $\beta > 0$. A pair of unitaries (V_L, V_R) is called a (β, b, ε) -state-preparation pair if

$$V_L|0^b\rangle = \sum_{j=0}^{2^b-1} c_j |j\rangle, \quad V_R|0^b\rangle = \sum_{j=0}^{2^b-1} d_j |j\rangle,$$

such that:

$$\sum_{j=0}^{m-1} |\beta c_j^* d_j - y_j| \leq \varepsilon, \quad \text{and} \quad c_j = d_j = 0 \quad \text{for all } j \geq m.$$

A common choice is $V_L = V^\dagger, V_R = V$ with $c_j = d_j = \sqrt{y_j}/\beta$, as in the PREP operator from Equation (6.8).

Theorem 7.3 (Linear combination of block-encoding (LCBU) [GSLW19, Lemma 52]). Let $A = \sum_{j=0}^{m-1} y_j A_j$ with $A_j \in \mathbb{C}^{N \times N}$ an n -qubit operator and $\vec{y} \in \mathbb{C}^m$. Suppose each A_j admits an $(\alpha, a, \varepsilon_2)$ -block-encoding U_j on a ancillae, i.e.,

$$\tilde{A}_j := (\langle 0^a | \otimes \mathbb{1}_N) U_j (|0^a\rangle \otimes \mathbb{1}_N) = A_j/\alpha + E_j, \quad \|E_j\| \leq \varepsilon_2/\alpha.$$

Let (V_L, V_R) be a $(\beta, b, \varepsilon_1)$ -state-preparation pair for $\vec{y}$. Define the multiplexed unitary on $a+b$ ancillae and N -dimensional system:

$$W := \sum_{j=0}^{m-1} |j\rangle\langle j| \otimes U_j + \left(\mathbb{1}_N - \sum_{j=0}^{m-1} |j\rangle\langle j| \right) \otimes \mathbb{1}_{2^a} \otimes \mathbb{1}_{2^b}.$$

Then,

$$U := (V_L^\dagger \otimes \mathbb{1}) W (V_R \otimes \mathbb{1})$$

is an $(\alpha\beta, a+b, \alpha\varepsilon_1 + \beta\varepsilon_2)$ -block-encoding of A , i.e.,⁵

$$\left\| A - \alpha\beta (\langle 0^{(a+b)} | \otimes \mathbb{1}_N) U (|0^{(a+b)}\rangle \otimes \mathbb{1}_N) \right\| \leq \alpha\varepsilon_1 + \beta\varepsilon_2.$$

Proof.

Let $|\psi_R\rangle = V_R|0^b\rangle = \sum_j d_j |j\rangle$ and $\langle\psi_L| = \langle 0^b|V_L^\dagger = \sum_j c_j^* \langle j|$, so that $c_j = d_j = 0$ for $j \geq m$. Then,⁶

$$\begin{aligned} & (\langle 0^b | \otimes \langle 0^a | \otimes \mathbb{1}_N) U (|0^b\rangle \otimes |0^a\rangle \otimes \mathbb{1}_N) \\ &= (\langle\psi_L| \otimes \langle 0^a | \otimes \mathbb{1}_N) W (|\psi_R\rangle \otimes |0^a\rangle \otimes \mathbb{1}_N) \\ &= \sum_{j=0}^{m-1} c_j^* d_j (\langle 0^a | \otimes \mathbb{1}_N) U_j (|0^a\rangle \otimes \mathbb{1}_N) \\ &= \sum_{j=0}^{m-1} c_j^* d_j (A_j/\alpha + E_j). \end{aligned}$$

Multiplying by $\alpha\beta$ and subtracting $A = \sum_j y_j A_j$ results in,

$$\begin{aligned} \alpha\beta (\langle 0^{(a+b)} | \otimes \mathbb{1}_N) U (|0^{(a+b)}\rangle \otimes \mathbb{1}_N) - A &= \sum_{j=0}^{m-1} (\beta c_j^* d_j - y_j) A_j \\ &\quad + \alpha\beta \sum_{j=0}^{m-1} c_j^* d_j E_j. \end{aligned}$$

⁵ [GSLW19, Lemma 52] contains a small typo in the final error bound; the corrected version is stated here.

⁶ By construction $c_j = d_j = 0$ for $j \geq m$

We now bound the two terms separately. First,

$$\left\| \sum_j (\beta c_j^* d_j - y_j) A_j \right\| \leq \sum_j |\beta c_j^* d_j - y_j| \|A_j\| \leq \alpha \sum_j |\beta c_j^* d_j - y_j| \leq \alpha \varepsilon_1.$$

Second, since V_L, V_R are unitaries, $\sum_j |c_j|^2 = \sum_j |d_j|^2 = 1$, and by Cauchy-Schwarz, $\sum_j |c_j^* d_j| \leq 1$. Therefore,

$$\begin{aligned} \left\| \alpha \beta \sum_j c_j^* d_j E_j \right\| &\leq \alpha \beta \sum_j |c_j^* d_j| \|E_j\| \\ &\leq \alpha \beta \max_j \|E_j\| \left(\sum_j |c_j^* d_j| \right) \\ &\leq \alpha \beta (\varepsilon_2 / \alpha) = \beta \varepsilon_2. \end{aligned}$$

Combining these bounds yields the error $\alpha \varepsilon_1 + \beta \varepsilon_2$, as claimed. $\blacksquare$

Power-of-two padding. If $m < 2^b$, pad to $k = 2^b$ terms by setting $U_j = \mathbb{1}$ and $y_j = 0$ for $j \geq m$. Then $W = \sum_{j=0}^{k-1} |j\rangle \langle j| \otimes U_j$ and the same proof applies unchanged.

Product of Block-encoded matrices

Theorem 7.4 (Product of block-encodings [GSLW19, Lemma 4]). *Let $A, B \in \mathbb{C}^{N \times N}$ be n -qubit operators. Suppose U_A is an (α, a, ε) -block-encoding of A , and V_B is a (β, b, δ) -block-encoding of B . Assume further that the ancilla registers of U_A and V_B are disjoint. Then, the composed unitary*

$$W := (\mathbb{1}_{2^b} \otimes U_A)(\mathbb{1}_{2^a} \otimes V_B),$$

defined on $a + b$ ancillae, and the $N = 2^n$ -dimensional system, is an $(\alpha\beta, a + b, \alpha\delta + \beta\varepsilon)$ -block-encoding of the product AB , i.e.,

$$\left\| AB - \alpha\beta \left(|0^{(a+b)}\rangle \otimes \mathbb{1}_N \right) W \left(|0^{(a+b)}\rangle \otimes \mathbb{1}_N \right) \right\| \leq \alpha\delta + \beta\varepsilon.$$

Proof.

By the definition of block-encoding,

$$\tilde{A} := (|0^a\rangle \otimes \mathbb{1}_N) U_A (|0^a\rangle \otimes \mathbb{1}_N) = A/\alpha + E_A, \quad \|E_A\| \leq \varepsilon/\alpha,$$

$$\tilde{B} := (|0^b\rangle \otimes \mathbb{1}_N) V_B (|0^b\rangle \otimes \mathbb{1}_N) = B/\beta + E_B, \quad \|E_B\| \leq \delta/\beta.$$

The block encoded by $(\mathbb{1}_{2^b} \otimes U_A)(\mathbb{1}_{2^a} \otimes V_B)$ on $a + b$ ancillae is

$$(|0^{(a+b)}\rangle \otimes \mathbb{1}_N) (\mathbb{1}_{2^b} \otimes U_A)(\mathbb{1}_{2^a} \otimes V_B) (|0^{(a+b)}\rangle \otimes \mathbb{1}_N) = \tilde{A} \tilde{B}.$$

We wish to bound

$$\begin{aligned} \left\| AB - \alpha\beta \tilde{A} \tilde{B} \right\| &= \left\| AB - \alpha \tilde{A} B + \alpha \tilde{A} B - \alpha\beta \tilde{A} \tilde{B} \right\| \\ &= \left\| (A - \alpha \tilde{A}) B + \alpha \tilde{A} (B - \beta \tilde{B}) \right\|. \end{aligned}$$

Using the triangle inequality,

$$\|AB - \alpha\beta\tilde{A}\tilde{B}\| \leq \|(A - \alpha\tilde{A})B\| + \|\alpha\tilde{A}(B - \beta\tilde{B})\|.$$

Now,

$$\begin{aligned} \|(A - \alpha\tilde{A})B\| &\leq \|A - \alpha\tilde{A}\| \|B\| \\ &\leq \alpha\|E_A\| \|B\| \\ &\leq \kappa(\varepsilon/\kappa) \beta = \varepsilon\beta. \end{aligned}$$

Similarly for the second term,

$$\|\alpha\tilde{A}(B - \beta\tilde{B})\| \leq \alpha\|\tilde{A}\| \|B - \beta\tilde{B}\| \leq \alpha\delta.$$

Combining these bounds gives

$$\|AB - \alpha\beta\tilde{A}\tilde{B}\| \leq \varepsilon\beta + \alpha\delta.$$

Thus, $(\mathbb{1}_{2^b} \otimes U_A)(\mathbb{1}_{2^a} \otimes V_B)$ is an $(\alpha\beta, a + b, \alpha\delta + \beta\varepsilon)$ -block-encoding of AB . $\blacksquare$

Conjugate Transpose of a Block-encoding

Theorem 7.5 (Conjugate transpose of a block-encoding). *If U_A is an (α, a, ε) -block-encoding of an n -qubit operator A , then $U_A^\dagger$ is an (α, a, ε) -block-encoding of $A^\dagger$.*

Proof.

In computational-basis form, U_A being an (α, a, ε) -block-encoding of A means

$$\|A - \alpha(\langle 0^a | \otimes \mathbb{1}_N) U_A (|0^a\rangle \otimes \mathbb{1}_N)\| \leq \varepsilon.$$

Let

$$\tilde{A} := (\langle 0^a | \otimes \mathbb{1}_N) U_A (|0^a\rangle \otimes \mathbb{1}_N), \quad \text{so that } \|A - \alpha\tilde{A}\| \leq \varepsilon.$$

Taking adjoints and using $\|Q^\dagger\| = \|Q\|$ for the spectral norm, results in,

$$\|A^\dagger - \alpha\tilde{A}^\dagger\| = \|(A - \alpha\tilde{A})^\dagger\| \leq \varepsilon.$$

Since $(\langle 0^a | \otimes \mathbb{1}_N)^\dagger = (|0^a\rangle \otimes \mathbb{1}_N)$, then,

$$\tilde{A}^\dagger = ((\langle 0^a | \otimes \mathbb{1}_N) U_A (|0^a\rangle \otimes \mathbb{1}_N))^\dagger = (\langle 0^a | \otimes \mathbb{1}_N) U_A^\dagger (|0^a\rangle \otimes \mathbb{1}_N).$$

Hence, $U_A^\dagger$ is an (α, a, ε) -block-encoding of $A^\dagger$. The gate complexity is unchanged, as $U_A^\dagger$ is implemented by reversing the circuit for U_A and inverting its gates. the circuit for U_A and inverting its gates. $\blacksquare$

General-basis version. If $B_{L,1}^\dagger U B_{R,1} = A/\alpha + E$, then $B_{R,1}^\dagger U^\dagger B_{L,1} = (B_{L,1}^\dagger U B_{R,1})^\dagger = A^\dagger/\alpha + E^\dagger$, so $U^\dagger$ block-encodes $A^\dagger$ with left/right isometries swapped.

7.3 Block-Encoding Structured Matrices

We have seen how block-encodings can be combined to produce new operators. Here we discuss how to construct block-encodings for **structured matrices** commonly encountered in numerical linear algebra. Our exposition follows [YYL⁺24], which studies sparse matrices arising in simulations of ocean-acoustics. The challenge in quantum settings is to represent sparse matrices efficiently, storing only the nonzero entries while keeping circuit depth and ancilla costs low.

Assumption 7.1 (Matrix Data Items). *Let $A \in \mathbb{C}^{N \times N}$ with $N = 2^n$. Assume that the nonzero entries of A can be partitioned into s_0 disjoint subsets $S_0, S_1, \dots, S_{s_0-1}$ such that:*

- *For each $l \in \{0, \dots, s_0 - 1\}$, there exists a complex value A_l and index sets $\mathcal{R}_l$ (rows) and $\mathcal{C}_l$ (columns), together with an injective map $r_l : \mathcal{C}_l \rightarrow \mathcal{R}_l$ satisfying⁷*

$$A_{r_l(j), j} = A_l, \quad \forall j \in \mathcal{C}_l.$$

⁷ The mapping r_l is fixed for each data item l ; that is, all entries corresponding to A_l share the same row-column relation $i = r_l(j)$.

- *Equivalently, for all (i, j) ,*

$$A_{ij} = \sum_{l=0}^{s_0-1} A_l \mathbf{1}_{j \in \mathcal{C}_l} \mathbf{1}_{i=r_l(j)}. \quad (7.2)$$

In particular, $S_l = \{(r_l(j), j) : j \in \mathcal{C}_l\}$, and the sets S_l are pairwise disjoint.

Hence, every nonzero entry of A is uniquely identified by a pair (l, j) , where l indexes the data item and j its column, with $r_l(j)$ determining the corresponding row.

Note 7. Condition (2) in [Assumption 7.1](#) restricts how nonzero entries are grouped into data items. For each value A_l , the row indices where it appears must be recoverable from the column indices through a fixed injective function r_l . E.g., if the same numeric value occurs in multiple rows for a given column, two different injective functions would be required, and therefore, the occurrences must be treated as distinct data items.

Base Scheme for Block-Encoding Sparse Matrices

We use two registers: a data-item register $|a\rangle \in \mathcal{H}_a$ with $2^{a-1} < s_0 \leq s = 2^a$ to store the index l , and a system (or column) register $|j\rangle \in \mathcal{H}_n$. Based on [Assumption 7.1](#), we define the following quantum oracles that provide access to the matrix entries and their index structure.

- **Data Loading Oracle O_{data} :** Loads the value A_l corresponding to the data

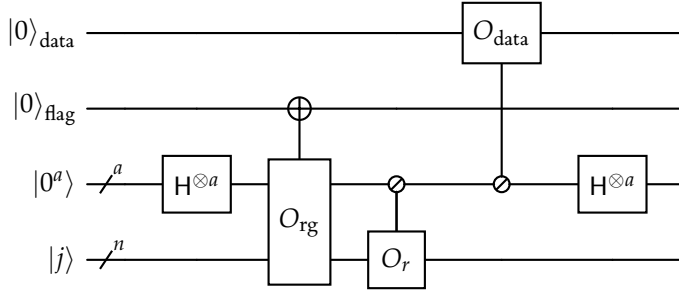


Figure 7.2: Circuit construction of U_A , the block-encoding of a matrix A as established in [Theorem 7.6](#).

item indexed by l :

$$O_{\text{data}} |0\rangle_{\text{data}} |l\rangle = \begin{cases} \left(\frac{A_l}{\|A\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |l\rangle, & l \in [0, s_0 - 1], \\ |0\rangle_{\text{data}} |l\rangle, & l \in [s_0, s - 1], \end{cases} \quad (7.3)$$

where $\|A\|_{\max} = \max_l |A_l|$, and $|0\rangle_{\text{data}}$ is the ancilla qubit used to load the data value.

- **Out-of-Range Oracle O_{rg} :** Flags whether the indices (l, j) are valid:

$$O_{\text{rg}} |0\rangle_{\text{flag}} |l\rangle |j\rangle = \begin{cases} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], j \in \mathcal{C}_l, \\ |1\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1] \text{ or } j \notin \mathcal{C}_l, \end{cases} \quad (7.4)$$

where the ancilla $|0\rangle_{\text{flag}}$ indicates whether the input indices are in range.

- **Row Oracle O_r :** Computes the row index $i = r_l(j)$ corresponding to (l, j) :⁸

$$O_r |l\rangle |j\rangle = \begin{cases} |l\rangle |r_l(j)\rangle, & l \in [0, s_0 - 1], \\ |l\rangle |j\rangle, & l \in [s_0, s - 1]. \end{cases} \quad (7.5)$$

⁸ We deviate from the notation of [\[YYL⁺24\]](#): here we write $r_l(j)$ instead of $c_l(j)$ for the function, and O_r instead of O_c for the oracle. This choice better reflects their functionality.

Theorem 7.6 (Block-Encoding Sparse Matrices [\[YYL⁺24, Theorem 3.3\]](#)).

Let $A \in \mathbb{C}^{N \times N}$ with $N = 2^n$ be a matrix with s_0 data items as in [Assumption 7.1](#).

Let O_{data} , O_{rg} , and O_r be the oracles defined in (7.3), (7.4), and (7.5), respectively.

Set $a = \lceil \log_2(s_0) \rceil$ and $s = 2^a$. Then

$$U_A := (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes H^{\otimes a} \otimes \mathbb{1}_N) \cdot \left[(O_{\text{data}} \otimes \mathbb{1}_{N+2}) (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r) (\mathbb{1}_2 \otimes O_{\text{rg}}) \right] \cdot (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes H^{\otimes a} \otimes \mathbb{1}_N)$$

is an $(s\|A\|_{\max}, 2 + \log_2 s, 0)$ -block-encoding of A . The corresponding circuit is shown in [Figure 7.2](#).

Proof.

Consider the initial state $|0\rangle_{\text{data}} |0\rangle_{\text{flag}} |0^a\rangle |j\rangle$, where $|j\rangle$ encodes an arbitrary column index $j \in [0, N-1]$. Postselecting all ancilla registers onto $|0\rangle$ yields the amplitude $\langle 0|_{\text{data}} \langle 0|_{\text{flag}} \langle 0^a| \langle i| U_A |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |0^a\rangle |j\rangle$, which equals the (i, j) -entry of the encoded matrix up to normalisation. Thus, we compute

$$\begin{aligned}
 & \langle 0|_{\text{data}} \langle 0|_{\text{flag}} \langle 0^a| \langle i| U_A |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |0^a\rangle |j\rangle \\
 &= \frac{1}{s} \sum_{k,l=0}^{s-1} \langle 0|_{\text{data}} \langle 0|_{\text{flag}} \langle k| \langle i| (O_{\text{data}} \otimes \mathbb{1}_{N+2}) (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r) (\mathbb{1}_2 \otimes O_{\text{rg}}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \frac{1}{s} \sum_{k,l=0}^{s-1} \mathbf{1}_{l < s_0} \mathbf{1}_{j \in \mathcal{C}_l} \langle 0|_{\text{data}} \langle k| \langle i| (O_{\text{data}} \otimes \mathbb{1}_N) (\mathbb{1}_2 \otimes O_r) |0\rangle_{\text{data}} |l\rangle |j\rangle \\
 &= \frac{1}{s} \sum_{k=0}^{s-1} \sum_{l=0}^{s_0-1} \mathbf{1}_{j \in \mathcal{C}_l} \langle 0|_{\text{data}} \langle k| \langle i| (O_{\text{data}} \otimes \mathbb{1}_N) (\mathbb{1}_2 \otimes O_r) |0\rangle_{\text{data}} |l\rangle |j\rangle \\
 &= \frac{1}{s} \sum_{k=0}^{s-1} \sum_{l=0}^{s_0-1} \mathbf{1}_{j \in \mathcal{C}_l} \langle 0|_{\text{data}} \langle k| (O_{\text{data}} \otimes \mathbb{1}_N) |0\rangle_{\text{data}} |l\rangle \underbrace{\langle i| r_l(j) \rangle}_{=\delta_{i,r_l(j)}} \\
 &= \frac{1}{s \|A\|_{\max}} \sum_{l=0}^{s-1} \sum_{l=0}^{s_0-1} \mathbf{1}_{j \in \mathcal{C}_l} A_l \underbrace{\langle k| l \rangle}_{=\delta_{k,l}} \delta_{i,r_l(j)} \\
 &= \frac{1}{s \|A\|_{\max}} \sum_{l=0}^{s_0-1} A_l \mathbf{1}_{j \in \mathcal{C}_l} \mathbf{1}_{i=r_l(j)} \\
 &= \frac{A_{ij}}{s \|A\|_{\max}}, \tag{7.6}
 \end{aligned}$$

where the last equality follows from the dictionary identity (7.2) in [Assumption 7.1](#). Hence, U_A block-encodes A with scaling factor $s \|A\|_{\max}$: the factor $1/s$ comes from preparing/unpreparing the uniform superposition via $H^{\otimes a}$, and the factor $1/\|A\|_{\max}$ from the normalisation in O_{data} . Thus, U_A is an $(\alpha, 2 + \log_2 s, 0)$ block-encoding with

$$\alpha = s \|A\|_{\max}.$$

We take the **data-loading cost** to be s_0 , i.e., the number of distinct values A_l that must be loaded by O_{data} .⁹ ■

The order of applying O_{data} , O_{rg} , and O_r in U_A may be interchanged freely, provided that O_{rg} is always applied before O_r . The proof is provided in [Appendix B](#).

PREP/UNPREP Scheme for Block-Encoding Sparse Matrices

We now present an alternative scheme for block-encoding sparse matrices with data items, which in some cases achieves a better subnormalisation

⁹ The data loading cost refers to the number of distinct values A_l that need to be loaded, each requiring at least one 2-qubit gate.

factor. The scheme follows the PREP/UNPREP framework of [YYL⁺24] and [SCC24], in which state-preparation unitaries are used to load the data items.

The construction begins by splitting the data-loading oracle O_{data} into two parts, parameterised by a choice of $0 \leq p \leq 1$. Using the commutation property of O_{data} , the oracles can be rearranged as in Figure 7.3. Each half acts on a separate ancilla qubit, denoted $|0\rangle_{\text{data}_0}$ and $|0\rangle_{\text{data}_1}$. The first part is defined by

$$O_{\text{data}}^p |0\rangle_{\text{data}} |l\rangle = \begin{cases} \left(\frac{A_l^p}{\|A\|_{\max}^p} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^{2p}} |1\rangle_{\text{data}} \right) |l\rangle, & l \in [0, s_0 - 1], \\ |0\rangle_{\text{data}} |l\rangle, & l \in [s_0, s - 1]. \end{cases} \quad (7.7)$$

The second oracle O_{data}^{1-p} is defined analogously, replacing p by $1 - p$.

A direct calculation shows that the circuit in Figure 7.3 has the same action as U_A from Theorem 7.6 when both data ancillae are postselected to $|0\rangle$. To see this, define the projection,

$$\Pi = |00\rangle \langle 00|_{\text{data}_0, \text{data}_1} \otimes \mathbb{1}_{2^a},$$

onto the joint ancilla state $|00\rangle$. It suffices to evaluate the action on

$|0\rangle_{\text{data}_0} |0\rangle_{\text{data}_1} |l\rangle_a$ for an arbitrary $l \in [0, s - 1]$.¹⁰

¹⁰ We omit subscripts for brevity.

$$\begin{aligned} & \Pi O_{\text{data}}^{1-p} O_{\text{data}}^p |0\rangle |0\rangle |l\rangle \\ &= \Pi O_{\text{data}}^{1-p} \begin{cases} \left(\frac{A_l^p}{\|A\|_{\max}^p} |0\rangle + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^{2p}} |1\rangle \right) |0\rangle |l\rangle, & l \in [0, s_0 - 1], \\ |00\rangle |l\rangle, & l \in [s_0, s - 1], \end{cases} \\ &= \Pi \begin{cases} \left(\frac{A_l^p}{\|A\|_{\max}^p} |0\rangle + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^{2p}} |1\rangle \right) \left(\frac{A_l^{1-p}}{\|A\|_{\max}^{1-p}} |0\rangle + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^{2(1-p)}} |1\rangle \right) |l\rangle, & l \in [0, s_0 - 1], \\ |00\rangle |l\rangle, & l \in [s_0, s - 1], \end{cases} \\ &= \Pi \begin{cases} \frac{A_l}{\|A\|_{\max}} |00\rangle |l\rangle + |\perp\rangle, & l \in [0, s_0 - 1], \\ |00\rangle |l\rangle, & l \in [s_0, s - 1], \end{cases} \\ &= \begin{cases} \frac{A_l}{\|A\|_{\max}} |l\rangle, & l \in [0, s_0 - 1], \\ |l\rangle, & l \in [s_0, s - 1], \end{cases} \end{aligned} \quad (7.8)$$

where $|\perp\rangle$ is orthogonal to $|00\rangle$, namely $|01\rangle$, $|10\rangle$, or $|11\rangle$. Thus, the circuit in Figure 7.3 is also an $(s\|A\|_{\max}, 3 + \log_2 s, 0)$ -block-encoding of A . In particular, the subnormalisation factor remains **unchanged**.

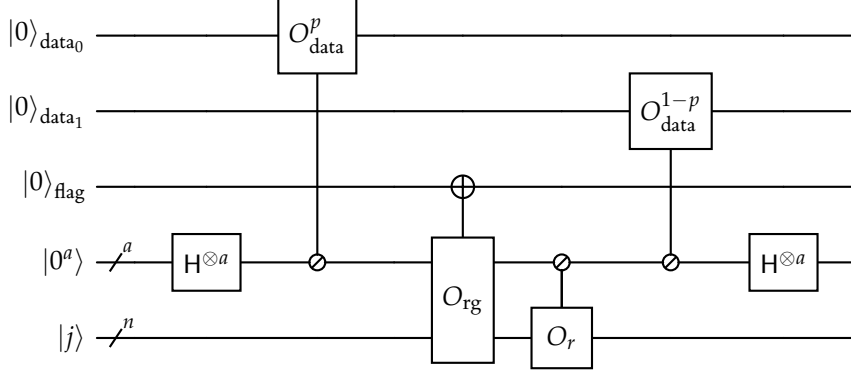


Figure 7.3: Quantum circuit with data oracle split into O_{data}^p and O_{data}^{1-p} components.

The key observation is that the Hadamard transforms, together with the data-loading oracles, prepare a quantum state whose amplitudes encode the data items; the subnormalisation factor arises solely from these two components. From Equation (7.7), each of O_{data}^p and O_{data}^{1-p} is therefore naturally combined with the surrounding Hadamard transforms.¹¹ For the first part, we compute

$$\begin{aligned}
 & \langle 0 |_{\text{data}_0} O_{\text{data}}^p (\mathbb{1}_2 \otimes H^{\otimes a}) | 0 \rangle_{\text{data}_0} | 0^a \rangle \\
 &= \frac{1}{\sqrt{s}} \left(\sum_{l=0}^{s_0-1} \frac{A_l^p}{\|A\|_{\max}^p} |l\rangle + \sum_{l=s_0}^{s-1} |l\rangle \right) \\
 &= \frac{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}}{\sqrt{s} \|A\|_{\max}^p} \cdot \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}} \sum_{l=0}^{s_0-1} A_l^p |l\rangle + \frac{1}{\sqrt{s}} \sum_{l=s_0}^{s-1} |l\rangle \\
 &= \frac{x_1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}} \sum_{l=0}^{s_0-1} A_l^p |l\rangle + \frac{1}{\sqrt{s}} \sum_{l=s_0}^{s-1} |l\rangle, \tag{7.9}
 \end{aligned}$$

where

$$x_1 = \frac{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}}{\sqrt{s} \|A\|_{\max}^p}.$$

Similarly,

$$\begin{aligned}
 \langle 0 |_{\text{data}_1} \langle 0^a | (\mathbb{1}_2 \otimes H^{\otimes a}) O_{\text{data}}^{1-p} | 0 \rangle_{\text{data}_1} &= \frac{x_2}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}} \sum_{l=0}^{s_0-1} A_l^{1-p} \langle l | \\
 &\quad + \frac{1}{\sqrt{s}} \sum_{l=s_0}^{s-1} \langle l |, \tag{7.10}
 \end{aligned}$$

where

$$x_2 = \frac{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2-2p}}}{\sqrt{s} \|A\|_{\max}^{1-p}}.$$

¹¹ A related approach for improving the subnormalisation factor is **uniform singular value amplification** [GSLW19], which employs QSVT as part of the block-encoding process. See [YYL⁺24, SCC24] for further details.

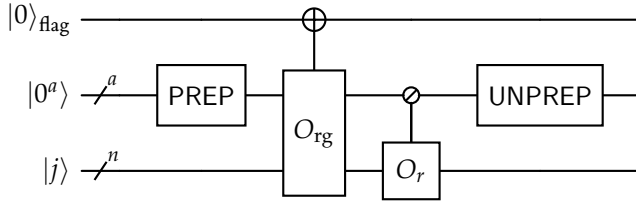


Figure 7.4: Circuit construction of U_A , based on the PREP/UNPREP method as established in [Theorem 7.7](#).

Both x_1 and x_2 are bounded by 1 and represent precisely the contributions to the subnormalisation factor, which we now seek to reduce. In (7.9) and (7.10), the terms $\frac{1}{\sqrt{s}} \sum_{l=s_0}^{s-1} |l\rangle$ and $\frac{1}{\sqrt{s}} \sum_{l=s_0}^{s-1} \langle l|$ correspond to **out-of-range** indices. The relevant (normalised) states are therefore

$$\frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}} \sum_{l=0}^{s_0-1} A_l^p |l\rangle, \quad \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2-2p}}} \sum_{l=0}^{s_0-1} A_l^{1-p} \langle l|,$$

which have zero amplitude on out-of-range positions. Thus, we replace the Hadamard layers and data-loading oracles by state-preparation unitaries PREP and UNPREP, which directly prepare these normalised amplitude states.

$$\begin{aligned} \text{PREP } |0^a\rangle &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p}}} \sum_{l=0}^{s_0-1} A_l^p |l\rangle + \sum_{l=s_0}^{s-1} \mathbf{0} \cdot |l\rangle, \\ \langle 0^a| \text{UNPREP} &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2-2p}}} \sum_{l=0}^{s_0-1} A_l^{1-p} \langle l| + \sum_{l=s_0}^{s-1} \mathbf{0} \cdot \langle l|. \end{aligned} \tag{7.11}$$

The resulting subnormalisation factor is the product of these two normalisation constants:

$$\alpha_p = \sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \cdot \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}.$$

We summarise this formally in [Theorem 7.7](#).

Theorem 7.7 (PREP/UNPREP Block-Encoding Sparse Matrices [[YYL⁺24](#), Theorem 3.7]). *Let $A \in \mathbb{C}^{N \times N}$ with $N = 2^n$ be a matrix with s_0 data items as in [Assumption 7.1](#). Let O_{rg} and O_r be the oracles defined in (7.4) and (7.5), and let PREP and UNPREP be the state-preparation unitaries defined in (7.11) for some $0 \leq p \leq 1$. Set $a = \lceil \log_2(s_0) \rceil$ and $s = 2^a$. Then,*

$$U_A := (\mathbb{1}_2 \otimes \text{UNPREP} \otimes \mathbb{1}_N) (\mathbb{1}_2 \otimes O_r) O_{rg} (\mathbb{1}_2 \otimes \text{PREP} \otimes \mathbb{1}_N)$$

is an $(\alpha_p, 1 + \log_2(s), 0)$ -block-encoding of A , where

$$\alpha_p = \sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}.$$

The corresponding circuit for U_A is shown in [Figure 7.4](#).

Proof. We begin by applying U_A to the initial state $|0\rangle_{\text{flag}}|0^a\rangle|j\rangle$, where $|j\rangle$ is the system register for an arbitrary column index $j \in [0, N-1]$ and then postselect on $|0\rangle_{\text{flag}}|0^a\rangle|i\rangle$ for some row index $i \in [0, N-1]$. Thus, we compute

$$\begin{aligned}
 & \langle 0|_{\text{flag}} \langle 0^a| \langle i| U_A |0\rangle_{\text{flag}} |0^a\rangle |j\rangle \\
 &= \langle 0|_{\text{flag}} \langle 0^a| \langle i| (\mathbb{1}_2 \otimes \text{UNPREP} \otimes \mathbb{1}_N) (\mathbb{1}_2 \otimes O_r) O_{\text{rg}} (\mathbb{1}_2 \otimes \text{PREP} \otimes \mathbb{1}_N) |0\rangle_{\text{flag}} |0^a\rangle |j\rangle \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} \sum_{k,l=0}^{s_0-1} A_l^p A_k^{1-p} \langle 0|_{\text{flag}} \langle k| \langle i| (\mathbb{1}_2 \otimes O_r) O_{\text{rg}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} \sum_{k,l=0}^{s_0-1} A_l^p A_k^{1-p} \mathbf{1}_{l < s_0} \mathbf{1}_{j \in \mathcal{C}_l} \langle k| \langle i| (\mathbb{1}_2 \otimes O_r) |l\rangle |j\rangle \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} \sum_{k,l=0}^{s_0-1} A_l^p A_k^{1-p} \mathbf{1}_{j \in \mathcal{C}_l} \langle k|l\rangle \langle i|r_l(j)\rangle \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} \sum_{k,l=0}^{s_0-1} A_l^p A_k^{1-p} \mathbf{1}_{j \in \mathcal{C}_l} \delta_{kl} \delta_{i,r_l(j)} \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} \sum_{l=0}^{s_0-1} A_l^p \mathbf{1}_{j \in \mathcal{C}_l} \mathbf{1}_{i=r_l(j)} \\
 &= \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}} A_{ij}. \tag{7.12}
 \end{aligned}$$

Hence, U_A is an $(\alpha_p, 1 + \log_2 s, 0)$ -block-encoding of A with

$$\alpha_p(A) = \sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}}.$$

The data-loading cost is $2s$, since both PREP and UNPREP must load s values each. $\blacksquare$

Compared to the base scheme of [Theorem 7.6](#), the subnormalisation factor α_p is always smaller:

$$\begin{aligned}
 \alpha_p &= \sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}} \\
 &\leq \sqrt{s_0 \|A\|_{\max}^{2p} \cdot s_0 \|A\|_{\max}^{2-2p}} \\
 &= s_0 \|A\|_{\max} \\
 &\leq s \|A\|_{\max}. \tag{7.13}
 \end{aligned}$$

Since α_p is symmetric about $p = \frac{1}{2}$, i.e. $\alpha_p = \alpha_{1-p}$, we may restrict to $p \in [\frac{1}{2}, 1]$ when minimising α_p . To determine the optimum value, we invoke the following result.

Lemma 7.8 (Callebaut’s Inequality [Cal65]). *Let $\{a_i\}_{i=0}^{m-1}$ and $\{b_i\}_{i=0}^{m-1}$ be sequences of non-negative real numbers. Then for $0 \leq y \leq x \leq 1$,*

$$\sum_{i=0}^{m-1} a_i^{1+y} b_i^{1-y} \sum_{j=0}^{m-1} a_j^{1-y} b_j^{1+y} \leq \sum_{i=0}^{m-1} a_i^{1+x} b_i^{1-x} \sum_{j=0}^{m-1} a_j^{1-x} b_j^{1+x}. \quad (7.14)$$

Hence, the function

$$f(r) = \left(\sum_{i=0}^{m-1} a_i^{1+r} b_i^{1-r} \right) \left(\sum_{j=0}^{m-1} a_j^{1-r} b_j^{1+r} \right),$$

is monotone increasing for $r \in [0, 1]$, and thus attains its minimum at $r = 0$. Setting $a_i = |A_i|$ and $b_i = 1$, and performing the change of variables $2p = 1 + r$ shows that $r = 0$ corresponds to $p = \frac{1}{2}$.¹² Therefore,

$$\alpha_p = \sqrt{f(2p-1)} = \sqrt{\sum_{l=0}^{s_0-1} |A_l|^{2p} \sum_{k=0}^{s_0-1} |A_k|^{2-2p}},$$

is minimised at $p = \frac{1}{2}$. The optimal subnormalisation factor is¹³

$$\alpha_{\text{opt}} = \sum_{l=0}^{s_0-1} |A_l|. \quad (7.15)$$

¹² Note that A_i are nonzero by definition for $i \in [0, s_0 - 1]$.

¹³ Moreover, $\alpha_{1/2} \leq \alpha_p \leq \alpha_q$ for all $\frac{1}{2} \leq p \leq q \leq 1$.

7.4 Oracle Construction

We now describe the construction of the oracles O_{rg} and O_r used in the block-encoding schemes. Recall from [Assumption 7.1](#) that the nonzero entries of $A \in \mathbb{C}^{N \times N}$ are partitioned into s_0 data items indexed by $l \in [0, s_0 - 1]$. Each item is specified by a value A_l , a column set $\mathcal{C}_l \subseteq [0, N - 1]$, and an injective map

$$r_l : \mathcal{C}_l \rightarrow [0, N - 1],$$

which determines the corresponding row index. Hence every nonzero entry of A is uniquely labelled by a pair (l, j) , with $j \in \mathcal{C}_l$ and $r_l(j)$ giving the row.

The mappings r_l form the core of the oracle design: O_r computes $r_l(j)$, O_{rg} verifies whether (l, j) lies within the valid domain $\mathcal{C}_l$, and O_{data} loads the corresponding value A_l . To make their arithmetic structure explicit, it is convenient to write

$$r_l(j) = j + k_l, \quad j \in \mathcal{C}_l,$$

where each integer offset $k_l \in [-N + 1, N - 1]$ encodes the row as a shifted column index. Such **integer shift/offset maps** underlie most sparse matrix

formats and directly determine the circuit complexity of the oracle implementations. Constructing the functions $r_l(j)$ is therefore a central step in the overall block-encoding process.

For illustration, consider the matrix proportional to the fourth-order centred finite-difference stencil for the one-dimensional derivative operator:

$$A = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 \\ -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 & 0 \\ \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 \\ 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 \\ 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 \\ 0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ 0 & 0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 \end{bmatrix} \end{matrix} \cdot (7.16)$$

We use this stencil as a running example for the oracle constructions. Its nonzero entries can be grouped into $s_0 = 4$ data items with

$$\begin{aligned} \mathcal{C}_0 &= \{2, 3, 4, 5, 6, 7\}, & \mathcal{C}_1 &= \{1, 2, 3, 4, 5, 6, 7\}, \\ \mathcal{C}_2 &= \{0, 1, 2, 3, 4, 5, 6\}, & \mathcal{C}_3 &= \{0, 1, 2, 3, 4, 5\}. \end{aligned}$$

Each colour class corresponds to one offset $k_l \in \{-2, -1, +1, +2\}$, illustrating the direct mapping between finite-difference stencils and oracle offset patterns.

Out-of-Range Oracle

The out-of-range oracle O_{rg} identifies whether a pair (l, j) lies outside the valid domain of r_l . It flips a flag qubit if either

$$l \in [s_0, s-1] \quad \text{or} \quad j \notin \mathcal{C}_l.$$

The first condition ($l \geq s_0$) can be checked using comparator circuits. With conditionally clean ancillae, this requires $\mathcal{O}(n)$ CX-gates and $\log^*(n)$ clean ancillae [KG25]. In the PREP/UNPREP scheme of Theorem 7.7, these indices are automatically assigned zero amplitude by the state-preparation unitaries, so only the membership check $j \in \mathcal{C}_l$ for $l < s_0$ remains.

Membership testing is implemented using MCX-gates, whose control pattern depends on both the binary encoding of elements $j \in \mathcal{C}_l^c$ and the register storing l . For the stencil matrix in Equation (7.16), the out-of-range column sets are

$$\mathcal{C}_0^c = \{0, 1\}, \quad \mathcal{C}_1^c = \{0\}, \quad \mathcal{C}_2^c = \{7\}, \quad \mathcal{C}_3^c = \{6, 7\}.$$

The corresponding circuit for O_{rg} is shown in Figure 7.5. Each MCX-gate is controlled on both the register l and the binary pattern of an element $j \in \mathcal{C}_l^c$. Thus, each gate flips the flag qubit precisely when (l, j) is an out-of-range pair.

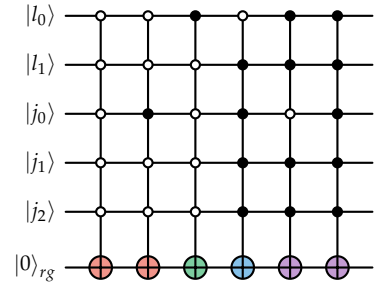


Figure 7.5: O_{rg} with gates colour-coded and numbered according to the corresponding pair (l, j) . E.g., gate 4 activates when $l = 2$ and $j = 7$ (i.e., $|111\rangle$).

Row-Index Oracle

The row-index oracle O_r computes the functions $r_l(j)$ for all $l \in [0, s_0 - 1]$, each taking the arithmetic form

$$r_l(j) = j + k_l, \quad k_l \in [-N + 1, N - 1],$$

implemented via the offset gate described in [Controlled Constant Adder](#). These offsets k_l are obtained by partitioning the nonzero entries of A according to their cyclic shifts, as detailed in [Algorithm 1](#). As an illustration, for $N = 8$ and $s_0 = 2$ with

$$r_0(j) = j - 2, \quad r_1(j) = j - 1, \quad r_2(j) = j + 1, \quad r_3(j) = j + 2,$$

the corresponding circuit is shown in [Figure 7.6](#).

To minimise circuit cost, each offset k is chosen to maximise the number of trailing zero bits in $|k|$, exposing powers of two and enabling the **shift-then-add** optimisation introduced in [Appendix A](#). This factorisation reduces the total controlled-adder count by decomposing each operation into smaller, cheaper sub-adders. The procedure `CanonicalOffset`, given in [Appendix B](#), selects these offsets automatically.

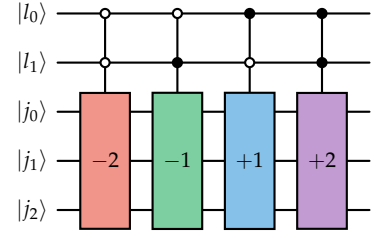


Figure 7.6: Example circuit for O_r , with gates colour-coded according to the corresponding pair $(l, k^{(l)})$. E.g., gate 3 activates when $l = 1$ and applies a shift of $+1$.

Algorithm 1: Offset Partitioning

Input: Matrix $A \in \mathbb{C}^{N \times N}$ with $N = 2^n$.

Output: Collection $\mathcal{I} = \{(l, v, k_l, \mathcal{C}_l, r_l)\}$, where $r_l(j) = (j + k_l) \bmod N$.

```

1: (Optional) Normalise entries: Round  $A$  to suppress tiny floating discrepancies.
2:  $\mathcal{V} \leftarrow \text{Unique}(\{A_{ij}\}_{A_{ij} \neq 0})$ 
3:  $\mathcal{I} \leftarrow \emptyset$  ▷ list of  $(l, v, k_l, \mathcal{C}_l, r_l)$ 
4: for  $v \in \mathcal{V}$  do
5:    $\mathcal{P}_v \leftarrow \{(i, j) : A_{ij} = v\}$  ▷ positions where  $v$  occurs
6:    $\mathcal{G} \leftarrow \emptyset$  ▷ offset  $\mapsto$  list of  $(i, j)$ 
7:   for  $(i, j) \in \mathcal{P}_v$  do
8:      $k \leftarrow \text{CanonicalOffset}(i, j, N)$  ▷  $k \equiv (i - j) \bmod N$ ; choose canonical representative for efficiency
9:      $\mathcal{G}[k] \leftarrow \mathcal{G}[k] \cup \{(i, j)\}$ 
10:  end for
11:  for  $k_l \in \text{Keys}(\mathcal{G})$  do
12:     $\mathcal{C}_l \leftarrow \text{Sort}\{j \mid (i, j) \in \mathcal{G}[k_l]\}$  ▷ unique columns
13:     $r_l(j) \leftarrow (j + k_l) \bmod N$  ▷ row map for all  $j \in [0, N - 1]$ 
14:     $\mathcal{I} \leftarrow \mathcal{I} \cup \{(l, v, k_l, \mathcal{C}_l, r_l)\}$ 
15:  end for
16: end for
17: return  $\mathcal{I}$ 

```

Data-Loading Oracle

The data-loading oracle O_{data} encodes the values A_l into amplitudes of a quantum state. Any complex number $z = a + ib$ can be represented via angles $\theta, \varphi \in [0, 2\pi)$ as

$$z = e^{-i\theta/2} \cos\left(\frac{\varphi}{2}\right),$$

where $\varphi = 2 \arccos(|z|)$ and $\theta = -2 \arg(z)$. Consider the action of $R_x(\varphi)R_z(\theta)$ on $|0\rangle$:¹⁴

$$R_x(\varphi)R_z(\theta)|0\rangle = e^{-i\theta/2} \cos\left(\frac{\varphi}{2}\right)|0\rangle - ie^{-i\theta/2} \sin\left(\frac{\varphi}{2}\right)|1\rangle.$$

The coefficient of $|0\rangle$ is exactly z ; since the block-encoding postselects on $|0\rangle$, the amplitude of $|1\rangle$ is irrelevant. Hence O_{data} can be implemented using controlled R_x and R_z gates with angles

$$\varphi_l = 2 \arccos\left(\frac{|A_l|}{\|A\|_{\max}}\right), \quad \theta_l = -2 \arg(A_l),$$

for $l \in [0, s_0 - 1]$, as illustrated in Figure 7.7.

The resulting multi-controlled rotations are examples of uniformly controlled gates. Such gates, acting on s values with angles $\{\theta_i\}$, can be decomposed into s CX gates and s single-qubit rotations with modified angles $\{\tilde{\theta}_i\}$, obtained from $\{\theta_i\}$ via the Walsh-Hadamard transform [SBMo6]. An example decomposition is shown in Figure 7.8.

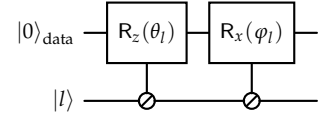
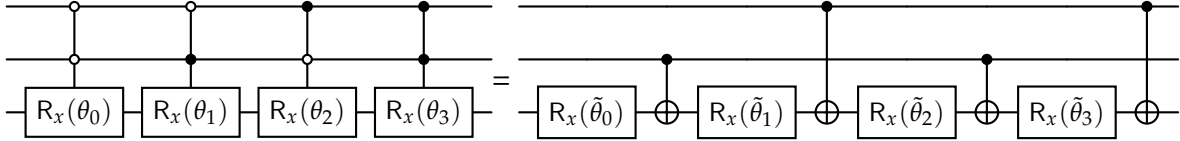


Figure 7.7: O_{data} circuit.



While the above construction uses a single ancilla register, alternative data-loading strategies are possible when additional ancillae or approximations are allowed. A common option is quantum read-only memory (QROM) [BGB⁺18], which lowers T- or CX-gate counts at the cost of extra ancillae. Likewise, preparing the states required for PREP and UNPREP in Theorem 7.7 generally requires $\Omega(N)$ gates, where N is the Hilbert-space dimension, if no auxiliary qubits are used [ICK⁺16]. In our setting, however, $s \ll N$, therefore, the cost of O_{data} remains subdominant relative to the other oracles.

Figure 7.8: Decomposition of multiplexed rotations on four values.

7.5 Efficient Oracle Design under Matrix Structure

The oracle constructions described above are generic and make no assumptions about the structure of the data items. In many applications, however, algebraic or arithmetic regularities can be exploited to reduce oracle cost. The out-of-range oracle O_{rg} benefits from structured column sets, while

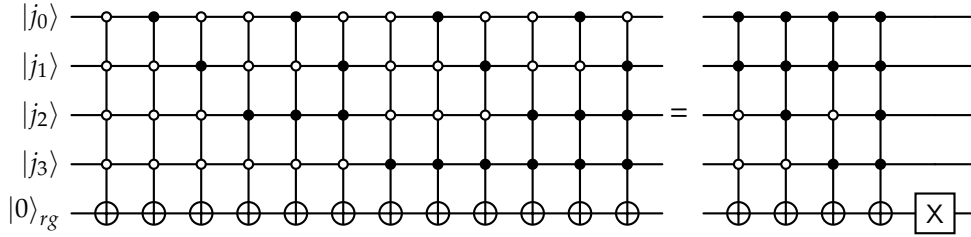
the row-index oracle O_r can be simplified when the shift maps $r_l(j)$ admit arithmetic patterns. Optimising O_{data} reduces to the problem of optimal state preparation, which lies beyond the scope of this thesis.

Optimising O_{rg} : Out-of-Range Detection

Consider the following column set:

$$\mathcal{C}_l = \{3, 7, 11, 15\}, \quad \mathcal{C}_l^c = \{0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 13, 14\}.$$

A membership test for $\mathcal{C}_l^c$ can be implemented as in Figure 7.9 (Left). Equivalently, we can test for $\mathcal{C}_l$ and apply a single X-gate to the flag qubit, as in Figure 7.9 (Right), which reduces the total number of MCX-gates whenever $|\mathcal{C}_l| \leq |\mathcal{C}_l^c|$.



A further improvement arises when $\mathcal{C}_l$ has an explicit arithmetic form. In the present example,

$$\mathcal{C}_l = \{j \in [0, 15] \mid j \bmod 4 = 3\},$$

which defines a periodic residue class modulo 4. We refer to such sets as **structured column sets**. Their membership test can be realised using a single MCX-gate, as in Figure 7.10, rather than a generic cascade of multi-controlled gates, yielding a substantial reduction in total CX count.

More generally, when two control patterns differ in only one bit, their corresponding gates can be merged into a single gate with one fewer control, provided the base operation G is identical. This reduction rule is illustrated in Figure 7.11 and can be applied iteratively to simplify the circuit in Figure 7.9 into the compact form of Figure 7.10.

Boolean Expressions for Membership Tests Each MCX-gate in the oracle circuits corresponds to a conjunction of Boolean literals. For example, the first gate in Figure 7.9 (Right) acts only when

$$x_0 \wedge x_1 \wedge \bar{x}_3 \wedge \bar{x}_4,$$

where x_i denotes the state of qubit i and $\bar{x}_i$ its negation. The flag qubit flips precisely when this **conjunction** is satisfied. Thus, every controlled operation in the circuit implements one Boolean clause.

Figure 7.9: (Left) $\mathcal{C}_l^c$; (Right) $\mathcal{C}_l$ followed by an X on the flag qubit.

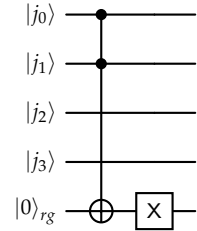


Figure 7.10: Compact implementation of O_{rg} of Figure 7.9.

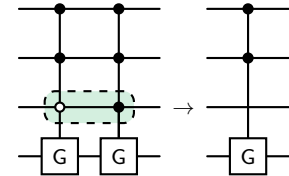


Figure 7.11: Merging controls when two control patterns differ by a single bit.

Sum of Products (SOP) Form Membership in a set of bitstrings can be represented as a Boolean function in **sum-of-products** (SOP) form namely, a disjunction of conjunctions:

$$f(x_0, \dots, x_{n-1}) = \bigvee_k \bigwedge_{i \in P_k} \ell_i,$$

where each literal $\ell_i \in \{x_i, \bar{x}_i\}$. Each **product term** $\bigwedge_{i \in P_k} \ell_i$ ¹⁵ is implemented by a single MCX-gate. To realise the full disjunction, ancilla qubits may be used to store the outputs of each clause, which are then combined into the flag qubit through an OR network. For example, the membership test for $\{0011, 1001\}$ is written in SOP form as

$$(\bar{x}_0 \bar{x}_1 x_2 x_3) \vee (x_0 \bar{x}_1 \bar{x}_2 x_3).$$

corresponding to two product clauses, each represented by an MCX-gate, whose outcomes can then be combined into the flag qubit via an OR.

In contrast to the SOP construction, the out-of-range oracle O_{rg} exhibits a slightly different structure. Because all product clauses act on a single target qubit, multiple activations interfere modulo two. Consequently, the oracle realises an **exclusive sum of products** (ESOP) form:

$$f(x_0, \dots, x_{n-1}) = \bigoplus_k \bigwedge_{i \in P_k} \ell_i.$$

The ESOP representation captures this cancellation behaviour, enabling algebraic simplifications. When two clauses differ in only one literal, their XOR reduces to a single clause with that literal removed.¹⁶ This rule corresponds precisely to the reduction shown in [Figure 7.11](#): adjacent control patterns differing in a single bit merge into a simpler gate. Repeated application of this rule, as illustrated in [Figure 7.12](#), produces algebraically reduced ESOP expressions and circuits with fewer or less-controlled MCX-gates. The ESOP viewpoint thus provides the Boolean-algebraic foundation for the structural simplifications exploited in out-of-range oracles.

¹⁵ A product term $\bigwedge_{i \in P_k} \ell_i$ is often called a **cube**, as it defines a subcube of the Boolean hypercube $\{0, 1\}^n$.

¹⁶ For example,

$$(\bar{x}_0 \bar{x}_1 x_2 x_3) \oplus (x_0 \bar{x}_1 x_2 x_3) = \bar{x}_1 x_2 x_3.$$

Here the dependence on x_0 cancels, leaving a simpler product term. In circuit terms, the two MCX-gates collapse to a single gate with one fewer control.

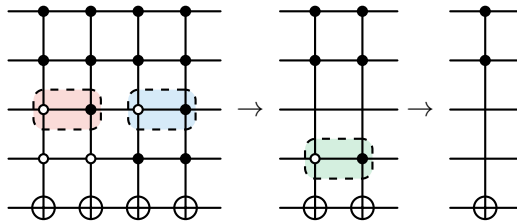


Figure 7.12: Iterative cancellations of adjacent control patterns applied to [Figure 7.9](#).

Note 8. The key difference between SOP and ESOP lies in how clauses are combined: SOP uses ordinary disjunction (OR), so overlapping clauses never cancel, while ESOP uses exclusive disjunction (XOR), so overlapping clauses cancel modulo two. In certain cases such as disjoint clauses differing in one literal, both forms reduce to the same simplified formula. Their semantics, however, remain distinct: SOP represents set union, whereas ESOP represents parity. Moreover, for many Boolean functions minimal ESOPs use fewer cubes than minimal SOPs [MP01], making ESOP a more compact representation in practice.

Algorithm 2: ESOP-Based Synthesis of O_{rg}

Input: Group encodings specified as a mapping $l \mapsto \mathcal{C}_l^c \subseteq [0, N - 1]$.

Output: Quantum circuit for O_{rg} obtained via ESOP reduction.

```

1:  $\mathcal{U} \leftarrow \{ \mathcal{C}_l^c \mid l \}$  ▷ set of distinct out-of-range column sets
2:  $\mathcal{H} \leftarrow \emptyset$  ▷ map  $S \mapsto (E_S, \nu)$  where  $E_S$  is ESOP,  $\nu \in \{\text{false}, \text{true}\}$ 
3: for  $S \in \mathcal{U}$  do
4:    $(E_S, \nu) \leftarrow \text{ESOPMinimise}(S; \text{allowNegation} = \text{true})$ 
5:    $\mathcal{H}[S] \leftarrow (E_S, \nu)$ 
6: end for

7:  $\mathcal{T} \leftarrow \emptyset$  ▷ conjunction terms for the full oracle
8: for  $l$  over data items do
9:   if  $\mathcal{C}_l^c = \emptyset$  then
10:    continue ▷ no out-of-range constraint for this  $l$ 
11:  end if
12:   $(E_{\text{col}}, \nu) \leftarrow \mathcal{H}[\mathcal{C}_l^c]$ 
13:   $E_{\text{data}} \leftarrow \text{Encode}(l)$  ▷ Boolean pattern for data index  $l$ 
14:   $\mathcal{T} \leftarrow \mathcal{T} \cup \{ E_{\text{col}} \wedge E_{\text{data}} \}$ 
15:  if  $\nu = \text{true}$  then
16:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{ E_{\text{data}} \}$  ▷ extra clause if column expression is negated
17:  end if
18: end for

19:  $(E_{\text{final}}, \nu_{\text{final}}) \leftarrow \text{ESOPMinimise}(\mathcal{T}; \text{allowNegation} = \text{true})$  ▷ global ESOP reduction over all terms
20:  $Q \leftarrow \text{BuildCircuitFromESOP}(E_{\text{final}})$ 
21: if  $\nu_{\text{final}} = \text{true}$  then
22:   apply X to the flag qubit
23: end if
24: return  $Q$ 
    
```

Extending ESOP Reductions to Data Indices Thus far, ESOP reductions have been applied only to the column-index register $|j\rangle$, assuming all MCX-gates

share identical controls on the data-index register $|l\rangle$. In practice, many data items exhibit identical column-set patterns, creating redundancy that allows ESOP reductions to be extended jointly to the data qubits. This yields significant savings in structured settings such as high-order finite-difference stencils or block-structured discretisations where repeated column sets frequently arise. The complete procedure, combining column- and data-qubit reductions and including optimisation over negated ESOP forms, is summarised in [Algorithm 2](#). The subroutines `Encode`, `ESOPMinimise`, and `BuildCircuitFromESOP` implement, respectively, the Boolean pattern for the data index l , ESOP minimisation with optional negation, and circuit synthesis from ESOP expressions.

Optimising O_r : Row-Index Oracle

When multiple data items share the same offset k_l , the corresponding circuit contains several controlled-adders with identical base operations that differ only in their control patterns ([Figure 7.13](#), Left). As in the out-of-range oracle, these can be merged via the reduction rule of [Figure 7.11](#), yielding a single controlled-adder with fewer controls ([Figure 7.13](#), Right). This optimisation is particularly impactful since offset gates are among the most costly components in our block-encoding schemes.

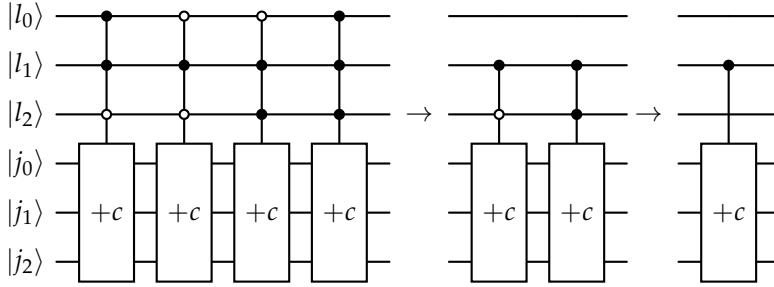


Figure 7.13: Optimisation of O_r when multiple data items share the same offset $k_l = c$. Control patterns are assigned according to a Gray code sequence (here 3, 2, 6, 7).

Unlike O_{rg} , however, the control patterns of O_r are not fixed by the matrix structure: **they are chosen by us**. Specifically, they depend on how data items are assigned to offsets k_l . By ordering the assignments of data items that share the same base offset gate so that successive control patterns differ in only one bit, we expose additional opportunities for gate merging. Such an ordering is provided by a **Gray code** sequence, ensuring that adjacent indices admit maximal reductions.

 Algorithm 3: Gray Code Assignment for O_r Optimisation

Input: Table $\mathcal{D}$ of data items i , each with location location_i and offset k_i .

Output: Updated table $\mathcal{D}$ with field location_gray_i defining new data locations.

```

1:  $\mathcal{G} \leftarrow \text{GroupBy}(\mathcal{D}, k)$  ▷ partition indices by offset  $k$ 
2:  $\mathcal{R} \leftarrow \mathcal{G} \triangleright \text{Filter}(|L| > 1) \triangleright \text{Sort}(\text{key} = |L|, \text{reverse} = \text{true})$ 
3:  $\mathcal{S} \leftarrow \mathcal{G} \triangleright \text{Filter}(|L| = 1) \triangleright \text{Sort}(\text{key} = k)$ 
(Here  $\triangleright$  denotes left-to-right composition of list operations.)

4:  $M \leftarrow 2^{\lceil \log_2(|\{i\}|) \rceil}$  ▷ maximum index range
5:  $\mathcal{U} \leftarrow \emptyset$  ▷ set of used indices
6:  $\mathcal{A} \leftarrow \emptyset$  ▷ map old index  $\mapsto$  new index
7:  $\mathcal{N} \leftarrow \emptyset$  ▷ list of non-contiguous groups
▷ — Allocate contiguous Gray-code blocks for repeated groups —

8: for  $(k, \text{indices}) \in \mathcal{R}$  do
9:    $f \leftarrow |\text{indices}|$  ;  $b \leftarrow 2^{\lceil \log_2(f) \rceil}$ 
10:   $\text{blockStart} \leftarrow \text{FindFreeBlock}(b)$  ▷ first free block of size  $b$  within  $[0, M)$ 
11:  if  $\text{blockStart} \geq 0$  then
12:     $\text{GraySeq} \leftarrow \text{GenerateGrayCodes}(\log_2 b)$ 
13:    for  $i = 0, \dots, f - 1$  do
14:       $\mathcal{A}[\text{indices}[i]] \leftarrow \text{blockStart} + \text{GraySeq}[i]$ 
15:       $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathcal{A}[\text{indices}[i]]\}$ 
16:    end for
17:  else
18:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{(k, \text{indices})\}$ 
19:  end if
20: end for
▷ — Handle leftover repeated groups (non-contiguous) —
▷ available indices

21:  $\mathcal{V} \leftarrow \text{Sort}([0, \dots, M - 1] \setminus \mathcal{U})$ 
22: for  $(k, \text{indices}) \in \mathcal{N}$  do
23:    $f \leftarrow |\text{indices}|$ 
24:    $\text{GraySeq} \leftarrow \text{GenerateGrayCodes}(\lceil \log_2(f) \rceil)$ 
25:    $\text{GrayToIndex} \leftarrow \text{Assoc}(\text{GraySeq}[0:f], \mathcal{V}[0:f])$  ▷ associate first  $f$  codes with first  $f$  free indices
26:   for  $i = 0, \dots, f - 1$  do
27:      $\mathcal{A}[\text{indices}[i]] \leftarrow \text{GrayToIndex}[\text{GraySeq}[i]]$ 
28:   end for
29:    $\mathcal{V} \leftarrow \mathcal{V}[f:]$ 
30: end for

31: for  $(k, \text{indices}) \in \mathcal{S}$  do ▷ — Allocate singleton groups —
32:    $\mathcal{A}[\text{indices}[0]] \leftarrow \text{PopFirst}(\mathcal{V})$ 
33: end for
34: return updated table  $\mathcal{D}$  with field  $\text{location\_gray}$  defined by  $\mathcal{A}$ 

```

7.6 Limitations and Future Extensions

Limitations

The framework developed thus far provides an efficient and fully automated procedure for constructing block-encodings of structured linear operators whose sparsity patterns are regular and whose indices admit simple binary arithmetic. Nevertheless, several intrinsic limitations constrain the class of matrices to which our approach applies.

Dimensional and structural constraints. The present construction assumes that the target operator is **square** and that its dimension N is a power of two, i.e., $N = 2^n$ for some integer n . This ensures a direct mapping between matrix indices and the computational basis states of n qubits, permitting all addressing and arithmetic to be performed using bitwise operations. Extending to rectangular or non-power-of-two matrices requires either zero-padding to the next power of two or an explicit subspace embedding.

Arithmetic complexity in modular checks. A further limitation arises in the O_{rg} oracle, where Boolean simplification relies on expressing structured index sets in **exclusive-sum-of-products** (ESOP) form. In simple cases—such as column sets repeating every 2^k entries—the modular condition reduces to inspecting only the first k bits of the column index j . E.g., verifying $j \bmod 4 = 3$ merely checks that the lowest two bits of j are 1. Such patterns permit extensive cancellation among adjacent ESOP terms, often collapsing the entire condition to a single MCX-gate in the optimised oracle.

However, such simplification does not generalise to arbitrary moduli. When the repetition pattern follows a period of $p \neq 2^k$, the condition $j \bmod p = \ell$ may no longer be expressed as a low-depth Boolean pattern over the binary digits of j . Instead, we might need to implement a reversible long-division circuit to compute the remainder of j/p , compare it with ℓ , and uncompute the quotient register. In the worst case (under the $\mathcal{O}(1)$ assumption), this requires $\mathcal{O}(2^n)$ MCX gates for an n -qubit index register, implying a gate complexity that scales linearly with matrix dimension N , rather than logarithmically. The resulting block-encoding remains **correct** but it ceases to be **efficient** in the quantum-algorithmic sense.

A natural question arising from the above discussion is: **what classes of structured matrices can be block encoded efficiently under these restrictions?** In other words, when does the ESOP reduction and modular arithmetic required by O_{r} and O_{rg} remain polylogarithmic in the system size? We illustrate several representative examples below, each exhibiting a pattern that permits low-depth Boolean conditions and minimal ancilla overhead.

Toeplitz Matrices. Consider an 8×8 Toeplitz matrix with 5 unique diagonals:

$$\begin{bmatrix} \alpha_2 & \alpha_1 & \alpha_0 & & & & & \\ \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & & & & \\ \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & & & \\ & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & & \\ & & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & \\ & & & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \\ & & & & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 \\ & & & & & \alpha_4 & \alpha_3 & \alpha_2 \end{bmatrix}.$$

The optimised block-encoding circuit for this matrix is shown in Figure 7.14, with normalisation factor $\alpha = \sum_{i=0}^4 |\alpha_i|$.

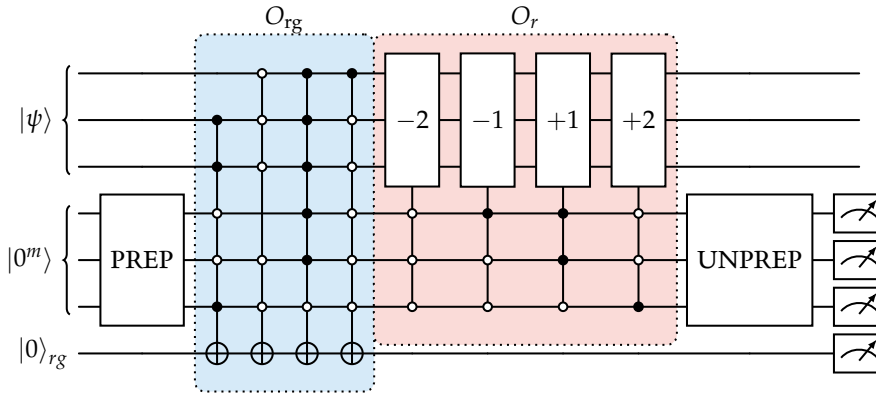


Figure 7.14: Block-encoding circuit for an 8×8 Toeplitz matrix.

Circulant Matrices. Circulant matrices are closely related to Toeplitz matrices, with the additional property that each row is a **cyclic shift** of the previous one. Consider the following 8×8 circulant matrix:

$$\begin{bmatrix} \alpha_2 & \alpha_3 & \alpha_4 & & & \alpha_0 & \alpha_1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & & & \alpha_0 \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & & \\ & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \\ & & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ & & & \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_4 & & & & \alpha_0 & \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 & & & & \alpha_0 & \alpha_1 \end{bmatrix}.$$

Since every data element appears in all column positions, the $_{rg}$ oracle is unnecessary. The resulting block-encoding circuit is shown in Figure 7.15, with normalisation factor $\alpha = \sum_{i=0}^4 |\alpha_i|$.

Two-dimensional Laplacian. We now consider a two-dimensional Laplacian operator discretised on a 4×4 grid. The resulting 16×16 matrix exhibits a characteristic block-Toeplitz structure as shown in Figure 7.16. The corresponding block-encoding circuit is shown in Figure 7.17, with normalisation

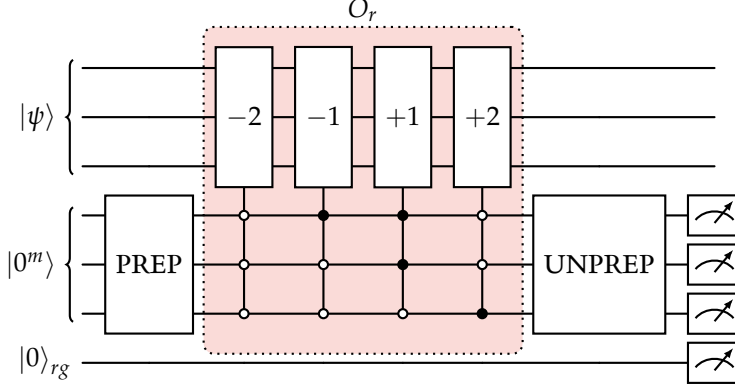


Figure 7.15: Block-encoding circuit for an 8×8 circulant matrix.

factor $\alpha = |\alpha_0| + |\alpha_1| + |\alpha_1| + |\alpha_2| + |\alpha_2|$. Note that α_1 and α_2 appear twice, as they are treated as distinct data items corresponding to the row shifts.

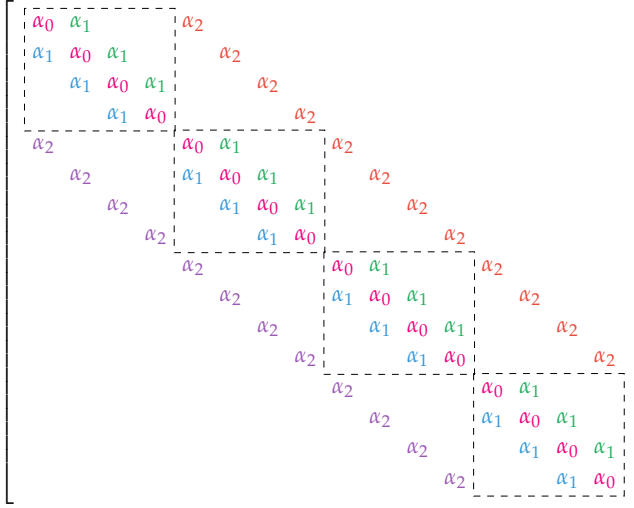


Figure 7.16: Two-dimensional Laplacian matrix with 4×4 grid discretisation.

These examples highlight a broad class of matrices that can be efficiently block encoded due to their regular structure. Since we can efficiently evaluate the column index modulo a power of two, any operator exhibiting a block-repeating pattern with block size 2^k can be encoded with polylogarithmic overhead. Such matrices naturally arise in element-based discretisations, where local operators are repeatedly assembled into a global system with uniform coupling structure. In particular, the global discretisation operator introduced in [Chapter 10](#) adheres precisely to this form, enabling an efficient block-encoding of the entire semi-discrete system.

Note 9. The examples shown in this section have been explored in prior works on block-encoding of structured matrices, including [YYL⁺24, SCC24]. However, the prior works investigated these examples in isolation, requiring manual design of each block-encoding circuit. In contrast, our framework provides a fully automated and generalised procedure for efficiently block-encoding structured matrices.

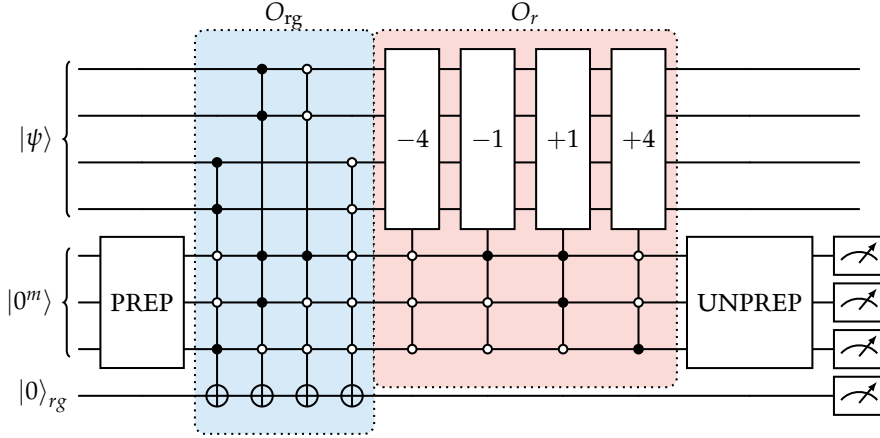


Figure 7.17: Block-encoding circuit for a 16×16 2D Laplacian matrix.

Future Extensions

Relaxing ancillary qubit constraints. Throughout this work, our focus has been on achieving block-encodings with minimal ancillary qubit requirements. If this restriction is relaxed to allow up to $\mathcal{O}(n)$ ancillae, the reversible long-division required for general modulus checks can be implemented efficiently [WLC24]. This trade-off between ancilla count and circuit depth offers a practical path to extending the class of efficiently encodable operators.

Handling non-power-of-two dimensions. To address the power-of-two limitation within the current construction, we can pad each block matrix with zeros to the next power of two and subsequently apply a permutation that restores the original ordering. Such permutations admit efficient circuit realisations [DLX25], and the overhead is often modest since adding a single ancillary qubit doubles the representable matrix dimension.

Incorporating dual-oracle frameworks. Our approach builds upon the single-oracle formulation of [YYL⁺24], where only a row or column oracle must be provided. By contrast, the framework of [SCC24] permits explicit specification of both row and column oracles, significantly broadening the class of matrices that can be efficiently block encoded. Automated synthesis within such a dual-oracle setting constitutes a promising next step.

Generalising the projector construction. Finally, the present block-encoding design assumes projectors implemented as controlled-phase projectors with zero-valued controls. An open direction is to generalise this to arbitrary control patterns or non-diagonal projectors. Whether such general projectors admit efficient circuit realisations remains an open question and, to our knowledge, has not yet been systematically explored in the literature.

8

Quantum Signal Processing

Quantum Signal Processing (QSP) provides a systematic method for encoding polynomial transformations into unitary circuits. At its core, QSP exploits the fact that certain unitaries naturally generate polynomials under repeated application. To build intuition, we begin by studying a simple rotation matrix whose powers reveal this polynomial structure explicitly. We then switch to a geometric viewpoint, framing QSP in terms of reflections on the Bloch sphere. Finally, we turn to the practical question of polynomial function approximation, showing how Chebyshev techniques enable the approximation of analytic functions by admissible polynomials suitable for implementation via QSP.

8.1 Quantum Signal Processing

Polynomial Encodings in Single-Qubit Rotations

Define the following family of 2×2 matrices, indexed by a real parameter $x \in [-1, 1]$:¹

$$A(x) := \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix}. \quad (8.1)$$

¹ Note that this operator is a unitary operator as long as $x \in [-1, 1]$.

Let us compute a few successive powers of $A(x)$ to observe how its structure evolves:

$$\begin{aligned} A^2(x) &= \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix} \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix} \\ &= \begin{bmatrix} 2x^2 - 1 & -2x\sqrt{1-x^2} \\ 2x\sqrt{1-x^2} & 2x^2 - 1 \end{bmatrix} \\ A^3(x) &= \begin{bmatrix} 2x^2 - 1 & -2x\sqrt{1-x^2} \\ 2x\sqrt{1-x^2} & 2x^2 - 1 \end{bmatrix} \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix} \\ &= \begin{bmatrix} 4x^3 - 3x & (1-4x^2)\sqrt{1-x^2} \\ -(1-4x^2)\sqrt{1-x^2} & 4x^3 - 3x \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A^4(x) &= \begin{bmatrix} 4x^3 - 3x & (1 - 4x^2)\sqrt{1 - x^2} \\ -(1 - 4x^2)\sqrt{1 - x^2} & 4x^3 - 3x \end{bmatrix} \begin{bmatrix} x & -\sqrt{1 - x^2} \\ \sqrt{1 - x^2} & x \end{bmatrix} \\ &= \begin{bmatrix} 8x^4 - 8x^2 + 1 & -4x(2x^2 - 1)\sqrt{1 - x^2} \\ 4x(2x^2 - 1)\sqrt{1 - x^2} & 8x^4 - 8x^2 + 1 \end{bmatrix}. \end{aligned}$$

In each case, the matrix entries are polynomials in x , and notably, these correspond to well-known orthogonal polynomials:

- The top-left entry $A_{11}^k(x)$ equals the **Chebyshev polynomial of the first kind** $T_k(x)$,
- The off-diagonal entries are proportional to the **Chebyshev polynomial of the second kind** $U_{k-1}(x)$, scaled by $\sqrt{1 - x^2}$.

We now formalise the pattern observed earlier.

Lemma 8.1. *For any integer $k \geq 0$, the k -th power of $A(x)$ takes the form*

$$A^k(x) := \begin{bmatrix} T_k(x) & -U_{k-1}(x)\sqrt{1 - x^2} \\ U_{k-1}(x)\sqrt{1 - x^2} & T_k(x) \end{bmatrix}$$

where $T_k(x)$ and $U_{k-1}(x)$ are the Chebyshev polynomials of the first and second kind, respectively.

Proof.

We prove the lemma by identifying a recursive pattern. Let us write:

$$A^k(x) = \begin{bmatrix} P_k(x) & -Q_k(x)\sqrt{1 - x^2} \\ Q_k(x)\sqrt{1 - x^2} & P_k(x) \end{bmatrix},$$

for some polynomials $P_k(x), Q_k(x)$. Substituting the expressions

$$\begin{aligned} A(x) &= \begin{bmatrix} x & -\sqrt{1 - x^2} \\ \sqrt{1 - x^2} & x \end{bmatrix}, \\ A^{k-1}(x) &= \begin{bmatrix} P_{k-1}(x) & -Q_{k-1}(x)\sqrt{1 - x^2} \\ Q_{k-1}(x)\sqrt{1 - x^2} & P_{k-1}(x) \end{bmatrix}, \end{aligned}$$

into $A^k(x) = A(x) \cdot A^{k-1}(x)$, we find:

$$A^k(x) = \begin{bmatrix} xP_{k-1} + (x^2 - 1)Q_{k-1} & -(xQ_{k-1} + P_{k-1})\sqrt{1 - x^2} \\ (xQ_{k-1} + P_{k-1})\sqrt{1 - x^2} & xP_{k-1} + (x^2 - 1)Q_{k-1} \end{bmatrix}$$

From this, we extract the recurrence:

$$\begin{aligned} P_k(x) &= xP_{k-1}(x) + (x^2 - 1)Q_{k-1}(x), \\ Q_k(x) &= xQ_{k-1}(x) + P_{k-1}(x). \end{aligned} \tag{8.2}$$

Substituting $Q_k(x)$ into the expression for $P_k(x)$ yields:

$$(x^2 - 1)Q_k(x) = xP_k(x) - P_{k-1}(x).$$

Finally, substituting back into (8.2), we obtain the second-order recursions:

$$\begin{aligned} P_k(x) &= 2xP_{k-1}(x) - P_{k-2}(x), \\ Q_k(x) &= 2xQ_{k-1}(x) - Q_{k-2}(x). \end{aligned} \quad (8.3)$$

These are exactly the Chebyshev recurrences for T_k and U_{k-1} . Hence $P_k = T_k$ and $Q_k = U_{k-1}$, establishing the claim. ■

Geometric Interpretation

To gain further insight into the structure of the matrix $A(x)$, we reinterpret it geometrically as a rotation operator on the Bloch sphere. When $x \in [-1, 1]$, the matrix $A(x)$ acts as a rotation in the two-dimensional subspace spanned by the standard basis vectors $|0\rangle$ and $|1\rangle$. In particular, this matrix performs a rotation by an angle 2θ , where $\theta = \arccos(x)$.² We can visualise (as shown in Figure 8.1) this by applying $A(x)$ to the initial state $|0\rangle$, yielding:

$$A(x) |0\rangle = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = |\psi\rangle,$$

a point on the Bloch sphere rotated by an angle θ from the $|0\rangle$ state.

Let us now explore what happens when we alternate reflections across two different axes: first about $|0\rangle$, and then about the rotated state $|\psi\rangle$. Let

- $A_{|0\rangle} = 2|0\rangle\langle 0| - \mathbb{1}$ be the reflection across the $|0\rangle$ axis,
- $A_{|\psi\rangle} = 2|\psi\rangle\langle \psi| - \mathbb{1}$ be the reflection across the rotated state $|\psi\rangle = A(x)|0\rangle$.

Now apply this pair of reflections to the initial state $|0\rangle$. We obtain:

$$A_{|\psi\rangle} A_{|0\rangle} A(x) |0\rangle.$$

Geometrically, each such double reflection rotates the vector by an additional angle of 2θ . Since we began at angle θ , the result lies at an angle of 3θ as shown in Figure 8.2. The final state is:

$$A_{|\psi\rangle} A_{|0\rangle} A(x) |0\rangle = \begin{bmatrix} \cos(3\theta) \\ \sin(3\theta) \end{bmatrix}.$$

Repeating the process with $|1\rangle$ instead of $|0\rangle$, we can similarly show that:³

$$A_{|\psi\rangle} A_{|1\rangle} A(x) |1\rangle = \begin{bmatrix} \cos(3\theta) \\ -\sin(3\theta) \end{bmatrix}.$$

Thus, we can write the composite operation as:

$$A_{|\psi\rangle} A_{|0\rangle} A(x) = \begin{bmatrix} \cos(3\theta) & -\sin(3\theta) \\ \sin(3\theta) & \cos(3\theta) \end{bmatrix}.$$

² In the block sphere representation, this corresponds to an $R_y(2\theta)$ rotation.

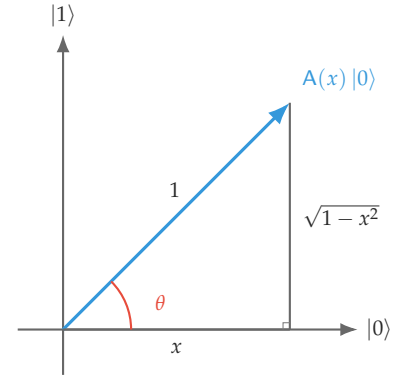


Figure 8.1: Geometric interpretation of the unitary matrix $A(x)$ as a rotation in the Bloch sphere.

³ We do not redefine the reflection axes: we will use the same $A_{|0\rangle}$ and $A_{|\psi\rangle}$ with $|\psi\rangle = A(x)|0\rangle$.

This confirms that the sequence of two reflections implements a rotation by 2θ , which, when composed with the initial application of $A(x)$, results in a total rotation of 3θ . Such rotation-by-reflection constructions are fundamental and also appear in Grover's algorithm [Gro96]. We can make this intuition concrete using algebra. Recall that the reflection operator about the rotated state $|\psi\rangle = A(x)|0\rangle$ is given by:

$$\begin{aligned} A_{|\psi\rangle} &= 2|\psi\rangle\langle\psi| - \mathbb{1} \\ &= 2(A(x)|0\rangle\langle 0|A(x)^\dagger) - \mathbb{1} \\ &= 2(A(x)|0\rangle\langle 0|A(x)^\dagger) - A(x)A(x)^\dagger \\ &= A(x)(2|0\rangle\langle 0| - \mathbb{1})A(x)^\dagger \\ &= A(x)A_{|0\rangle}A(x)^\dagger. \end{aligned}$$

This tells us that reflecting about $|\psi\rangle$ is equivalent to conjugating the $|0\rangle$ -reflection by $A(x)$. Similarly, we can show that $A_{|0\rangle}A(x)A_{|0\rangle} = A(x)$. Now substitute into our composite operation:

$$\begin{aligned} A_{|\psi\rangle}A_{|0\rangle}A(x) &= \begin{bmatrix} \cos(3\theta) & -\sin(3\theta) \\ \sin(3\theta) & \cos(3\theta) \end{bmatrix} \\ &= (A(x)A_{|0\rangle}A(x)^\dagger)A_{|0\rangle}A(x) \\ &= A(x)(A_{|0\rangle}A(x)^\dagger A_{|0\rangle})A(x) \\ &= A(x)A(x)A(x) \\ &= A^3(x). \end{aligned}$$

This reasoning extends naturally to higher powers, leading to the following general result:

Lemma 8.2. *Let $A(x)$ be the unitary matrix defined in (8.1) and $\theta = \arccos(x)$. Define the following reflections:*

- $A_{|0\rangle} = 2|0\rangle\langle 0| - \mathbb{1}$,
- $A_{|\psi\rangle} = 2A(x)|0\rangle\langle 0|A(x)^\dagger - \mathbb{1}$.

Then for any odd integer $k = 2m + 1$, the operator:

$$U_k := (A_{|\psi\rangle}A_{|0\rangle})^m A(x)$$

*satisfies:*⁴

$$U_k = \begin{bmatrix} \cos(k\theta) & -\sin(k\theta) \\ \sin(k\theta) & \cos(k\theta) \end{bmatrix} = \begin{bmatrix} T_k(x) & -U_{k-1}(x)\sqrt{1-x^2} \\ U_{k-1}(x)\sqrt{1-x^2} & T_k(x) \end{bmatrix}.$$

Thus by interpreting $A(x)$ as a rotation and composing it with reflections, we have uncovered a second route to the same result as before. Repeated applications of $A(x)$ let us generate Chebyshev polynomials, but this does not

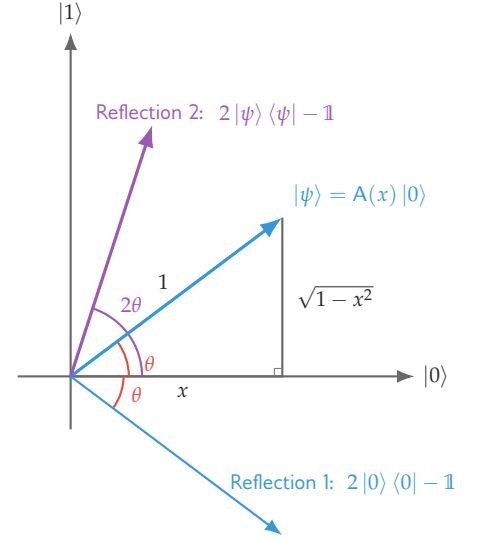


Figure 8.2: Geometric interpretation of the unitary matrix $A(x)$ as a reflection across the $|0\rangle$ state, followed by a reflection across the rotated state $|\psi\rangle$.

⁴ This lemma shows that the repeated application of $A(x)$, interspersed with reflections, yield matrices whose entries are Chebyshev polynomials: $T_k(x)$ on the diagonal and $U_{k-1}(x)\sqrt{1-x^2}$ on the off-diagonals.

give us the ability to produce arbitrary polynomials. For instance, it is unclear how to directly obtain a target polynomial like $f(x) = 3x^3$. To gain more control over the output, we need a mechanism for tuning the coefficients. A natural way to introduce such flexibility is by inserting phase rotations between applications of $A(x)$. Specifically, consider modifying the basic construction by inserting a Z -axis rotation:

$$A(x) e^{i\varphi Z} A(x),$$

where $e^{i\varphi Z} = \begin{bmatrix} e^{i\varphi} & 0 \\ 0 & e^{-i\varphi} \end{bmatrix}$ is a phase rotation. Let us compute the full product $A(x) e^{i\varphi Z} A(x)$:

$$\begin{aligned} &= \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix} \begin{bmatrix} e^{i\varphi} & 0 \\ 0 & e^{-i\varphi} \end{bmatrix} \begin{bmatrix} x & -\sqrt{1-x^2} \\ \sqrt{1-x^2} & x \end{bmatrix} \\ &= \begin{bmatrix} e^{i\varphi}x^2 - e^{-i\varphi}(1-x^2) & -e^{i\varphi}x\sqrt{1-x^2} - e^{-i\varphi}x\sqrt{1-x^2} \\ e^{i\varphi}x\sqrt{1-x^2} + e^{-i\varphi}x\sqrt{1-x^2} & e^{-i\varphi}x^2 - e^{i\varphi}(1-x^2) \end{bmatrix}. \end{aligned}$$

By varying the phase φ , we can steer the resulting matrix entries—essentially tuning the coefficients of the polynomial. For example, setting $\varphi = 0$ recovers the second Chebyshev polynomial:

$$A(x) A(x) = \begin{bmatrix} 2x^2 - 1 & -2x\sqrt{1-x^2} \\ 2x\sqrt{1-x^2} & 2x^2 - 1 \end{bmatrix}$$

corresponding to $T_2(x)$ on the diagonal. This motivates a more general construction: insert a sequence of phase rotations between each application of $A(x)$, yielding:

$$U_\Phi = e^{i\varphi_0 Z} A(x) e^{i\varphi_1 Z} A(x) \dots e^{i\varphi_k Z} A(x). \quad (8.4)$$

By choosing the sequence of phases $\Phi = \{\varphi_0, \varphi_1, \dots, \varphi_k\}$ appropriately, this allows us to construct polynomials of degree k in x , encoded in matrix entries of U_Φ . What we ultimately want, however, is the **inverse problem**: given a polynomial $p(x)$, can we find a sequence of phase rotations Φ such that the resulting matrix U_Φ encodes $p(x)$? This is the essence of Quantum Signal Processing (QSP) [LYC16, LC19, LC17].

Theorem 8.3 (Quantum Signal Processing [GSLW19, Theorem 3]). *Let $x \in [-1, 1]$, and define the **signal rotation operator**:*

$$W(x) := \begin{bmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{bmatrix} = e^{i\arccos(x)X} \quad (8.5)$$

where X is the Pauli X -gate. Then, for any integer $d \geq 1$, there exists a sequence of real phases $\Phi = \{\varphi_0, \dots, \varphi_d\} \in \mathbb{R}^{d+1}$ such that for all $x \in [-1, 1]$:

$$e^{i\varphi_0 Z} \cdot \prod_{j=1}^d \left(W(x) \cdot e^{i\varphi_j Z} \right) = \begin{bmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{bmatrix}$$

if and only if $P, Q \in \mathbb{C}[x]$ such that

1. $\deg(P) \leq d$ and $\deg(Q) \leq d - 1$,
2. P has parity $\equiv d \pmod{2}$ and Q has parity $\equiv d - 1 \pmod{2}$,
3. $\forall x \in [-1, 1]: |P(x)|^2 + (1 - x^2)|Q(x)|^2 = 1$.

An efficient algorithm exists to compute such a phase sequence Φ using $\mathcal{O}(d^2)$ arithmetic operations [GSLW19, Theorem 3]. This theorem illustrates the core mechanism behind QSP: By combining controlled signal rotations with tunable Z-axis phase gates, we can engineer a unitary whose matrix entries encode polynomial functions of x .

Conventions and Alternative Forms The definition of $W(x)$ in (8.5) differs from the signal rotation operator $A(x)$ used in earlier sections. However, the two are related via conjugation by fixed Z-rotations:⁵

$$A(x) = e^{i\frac{\pi}{4}Z} W(x) e^{-i\frac{\pi}{4}Z}. \quad (8.6)$$

We now define a Hermitian variant of the signal rotation, which is more convenient for block-encoding applications:

$$R(x) = \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}. \quad (8.7)$$

Then, the following corollary from [GSLW19, Corollary 8] holds:

Corollary 8.4. *Let $P \in \mathbb{C}[x]$ be a degree- d polynomial satisfying:*

1. P has parity $\equiv d \pmod{2}$,
2. $\forall x \in [-1, 1]: |P(x)| \leq 1$.
3. $\forall x \in (-\infty, -1] \cup [1, \infty): |P(x)| \geq 1$,
4. if d is even, then $\forall x \in \mathbb{R}: P(x)P^*(x) \geq 1$,

then there exists a sequence of phases $\Phi = \{\varphi_1, \dots, \varphi_d\} \in \mathbb{R}^d$ such that⁶

$$\prod_{j=1}^d \left(e^{i\varphi_j Z} \cdot R(x) \right) = \begin{bmatrix} P(x) & * \\ * & * \end{bmatrix}. \quad (8.8)$$

Moreover, for $x \in \{-1, 1\}$, we have that $P(x) = x^d \prod_{j=1}^d e^{i\varphi_j}$, and for d even $P(0) = e^{-i \sum_{j=1}^d (-1)^j \varphi_j}$.

This version of QSP departs from the original in several important ways. First, the signal operator $R(x)$, often called the **reflection operator** [MRTC21], is Hermitian and integrates naturally with the block-encoding framework discussed in Chapter 7. Second, we now use d phase parameters instead of $d + 1$. This change ensures that the parity of the polynomial matches its degree and reduces the circuit depth by one, saving an application of the block-encoded operator in QSVT constructions. In (8.8), only the top-left matrix entry is shown, as the others are irrelevant for our purposes⁷.

⁵ This relation allows the effect of switching from $W(x)$ to $A(x)$ to be absorbed into the choice of phases. Specifically, subtracting $\frac{\pi}{4}$ from the first phase φ_0 and adding $\frac{\pi}{4}$ to the last phase φ_k recovers an equivalent construction.

⁶ Both the signal rotation $R(x)$ and the phase rotation $e^{i\varphi_j Z}$ are unitary operators on a two-dimensional Hilbert space, and thus can be viewed as single-qubit operations.

⁷ This is because QSVT works with a projected subspace, allowing us to ignore off-diagonal terms and absorb certain phases.

Lemma 8.5. Let $R(x)$ be the signal rotation operator defined in (8.7), and let $P(x) \in \mathbb{C}[x]$ be a degree- d polynomial satisfying the conditions of [Corollary 8.4](#) (so that the $(1, 1)$ entry equals $P(x)$). Let $S_\Phi(x)$ be the product in [Equation \(8.8\)](#). Then replacing Φ with $-\Phi = (-\varphi_1, \dots, -\varphi_d)$ yields:

$$S_{-\Phi}(x) = \prod_{j=1}^d \left(e^{-i\varphi_j Z} \cdot R(x) \right) = \begin{bmatrix} P^*(x) & * \\ * & * \end{bmatrix},$$

i.e., the $(1, 1)$ entry becomes the complex conjugate $P^*(x)$.

Proof.

Since $R(x)$ is real, we have $R(x)^\dagger = R(x)$. Moreover, $(e^{i\varphi_j Z})^* = e^{-i\varphi_j Z}$ because $Z = \text{diag}(1, -1)$. Applying entrywise complex conjugation to $S_\Phi(x)$ gives

$$\begin{aligned} S_\Phi^*(x) &= \begin{bmatrix} P^*(x) & * \\ * & * \end{bmatrix} = \prod_{j=1}^d \left((e^{i\varphi_j Z})^* \cdot R(x)^* \right) \\ &= \prod_{j=1}^d \left(e^{-i\varphi_j Z} \cdot R(x) \right) \\ &= S_{-\Phi}(x). \end{aligned}$$

Therefore the $(1, 1)$ entry satisfies

$$P^*(x) = S_\Phi^*(x)_{11} = S_{-\Phi}(x)_{11}.$$

■

We will primarily focus on the case where $P(x)$ is a real polynomial, for which the following result holds:

Corollary 8.6 (Real Quantum Signal Processing [[GSLW19](#), Corollary 10]).

Let $P_{\Re}(x) \in \mathbb{R}[x]$ be a degree- d polynomial for some $d \geq 1$, such that

- $P_{\Re}$ has parity- $(d \bmod 2)$, and
- for all $x \in [-1, 1] : |P_{\Re}(x)| \leq 1$.

Then there exists $P \in \mathbb{C}[x]$ that satisfies the requirements of [Corollary 8.4](#) and $\Re[P(x)] = P_{\Re}(x)$.

8.2 Polynomial Function Approximation

Our objective is to use QSP to implement a target function $f(x)$ via a quantum circuit. Since QSP realises only **admissible polynomials** $p(x)$ ⁸, the standard approach is to approximate f by such a polynomial on a prescribed interval. Specifically, we seek a polynomial p of **minimal degree** such that

$$\begin{aligned} |p(x) - f(x)| &\leq \epsilon, \quad \forall x \in [a, b], \\ |p(x)| &\leq 1, \quad \forall x \in [-1, 1]. \end{aligned}$$

⁸ As characterised in [Corollary 8.4](#).

The degree d directly determines the complexity of the resulting QSP circuit: a degree- d polynomial corresponds to a sequence of d phase rotations and d applications of the signal rotation operator $R(x)$, so the overall query and gate complexity scales as $\mathcal{O}(d)$.

Chebyshev Polynomials

Uniform polynomial approximation on $[-1, 1]$ is most effectively carried out in the Chebyshev basis. These polynomials admit several equivalent descriptions. For a comprehensive overview, see [Tre13, Chapter 3].

Definition 8.1 (Chebyshev Polynomials of the First Kind). For $n \in \mathbb{N}$, the degree- n Chebyshev polynomial is defined by the identities

$$T_n(x) = T_n\left(\frac{1}{2}(z + z^{-1})\right) = \frac{1}{2}(z^n + z^{-n}) = \cos(n\theta) = \cos(n \arccos(x)),$$

where $z \in \mathbb{C}$ with $z = e^{i\theta}$, $\theta \in [-\pi, \pi]$, and $x = \cos \theta = \frac{1}{2}(z + z^{-1})$. It follows immediately that $|T_n|_{[-1,1]} \leq 1$ and $T_n(-x) = (-1)^n T_n(x)$.

Definition 8.2 (Chebyshev Coefficients and Truncation). Let $f : [-1, 1] \rightarrow \mathbb{C}$ be Lipschitz continuous (i.e. $|f(x) - f(y)| \leq L|x - y|$ for finite L). Then f admits a unique expansion in Chebyshev polynomials:⁹

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x), \quad a_k = \frac{2}{\pi} \int_{-1}^1 \frac{f(x) T_k(x)}{\sqrt{1-x^2}} dx.$$

This decomposition converges uniformly and absolutely. For a function f with Chebyshev coefficients $\{a_k\}$, the **degree- d Chebyshev truncation** is defined as

$$f_d(x) := \sum_{k=0}^d a_k T_k(x).$$

Approximating f then reduces to truncating its Chebyshev series. The error is governed by the decay of the Chebyshev coefficients:

$$\|f - f_d\|_{[-1,1]} = \left\| \sum_{k=d+1}^{\infty} a_k T_k(x) \right\|_{[-1,1]} \leq \sum_{k=d+1}^{\infty} |a_k|.$$

If f is analytic in a sufficiently large region around $[-1, 1]$, the coefficients $\{a_k\}$ decay exponentially. In particular, if f is analytic with $|f(x)| \leq M$ inside the Bernstein ellipse (Figure 8.3)

$$E_\rho = \left\{ z \in \mathbb{C} : z = \frac{1}{2}(z + z^{-1}), |z| = \rho \geq 1 \right\},$$

then $|a_k| \leq 2M\rho^{-k}$, and consequently

$$\|f - f_d\|_{[-1,1]} \leq \frac{2M\rho^{-d}}{\rho - 1}.$$

⁹ For $k = 0$, the factor $2/\pi$ is replaced by $1/\pi$.

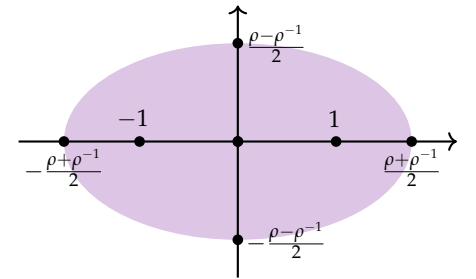


Figure 8.3: The Bernstein ellipse E_ρ with foci at ± 1 and semi-axes $\frac{\rho + \rho^{-1}}{2}$ and $\frac{\rho - \rho^{-1}}{2}$.

Thus, the degree d required for ε -accuracy scales as

$$d = \mathcal{O}\left(\log\left(\frac{M}{\varepsilon}\right)\right).$$

We refer the reader to [Tre13, Theorem 8.2] for a proof and further details on bounding polynomial approximation errors via Chebyshev truncation. For our purposes, the most relevant case is the exponential function $f(x) = e^{itx}$ with $t \in \mathbb{R}$ on $[-1, 1]$. For such standard functions, one can derive **tight** error bounds directly from the truncated Chebyshev series, without appealing only to general Bernstein ellipse arguments.

Recall from Corollary 8.4 that QSP requires the target polynomial $P(x)$ to have definite parity. However, e^{itx} does not satisfy this condition. To resolve this, we decompose it into even and odd parts,

$$e^{itx} = \cos(tx) + i \sin(tx),$$

so that each component can be approximated separately and later recombined, as described in Section 9.3. The following theorem provides explicit polynomial approximations for $\cos(tx)$ and $\sin(tx)$ with controlled error on $[-1, 1]$.

Theorem 8.7 ([GSLW19, Lemma 57, 59]). *Let $0 < \varepsilon < \frac{1}{e}$, $t \in \mathbb{R} \setminus \{0\}$, and set*

$$R = \left\lfloor \frac{1}{2} d\left(\frac{e|t|}{2}, \frac{5}{4}\varepsilon\right) \right\rfloor.$$

Then the following degree- $2R$ and degree- $(2R + 1)$ polynomials satisfy

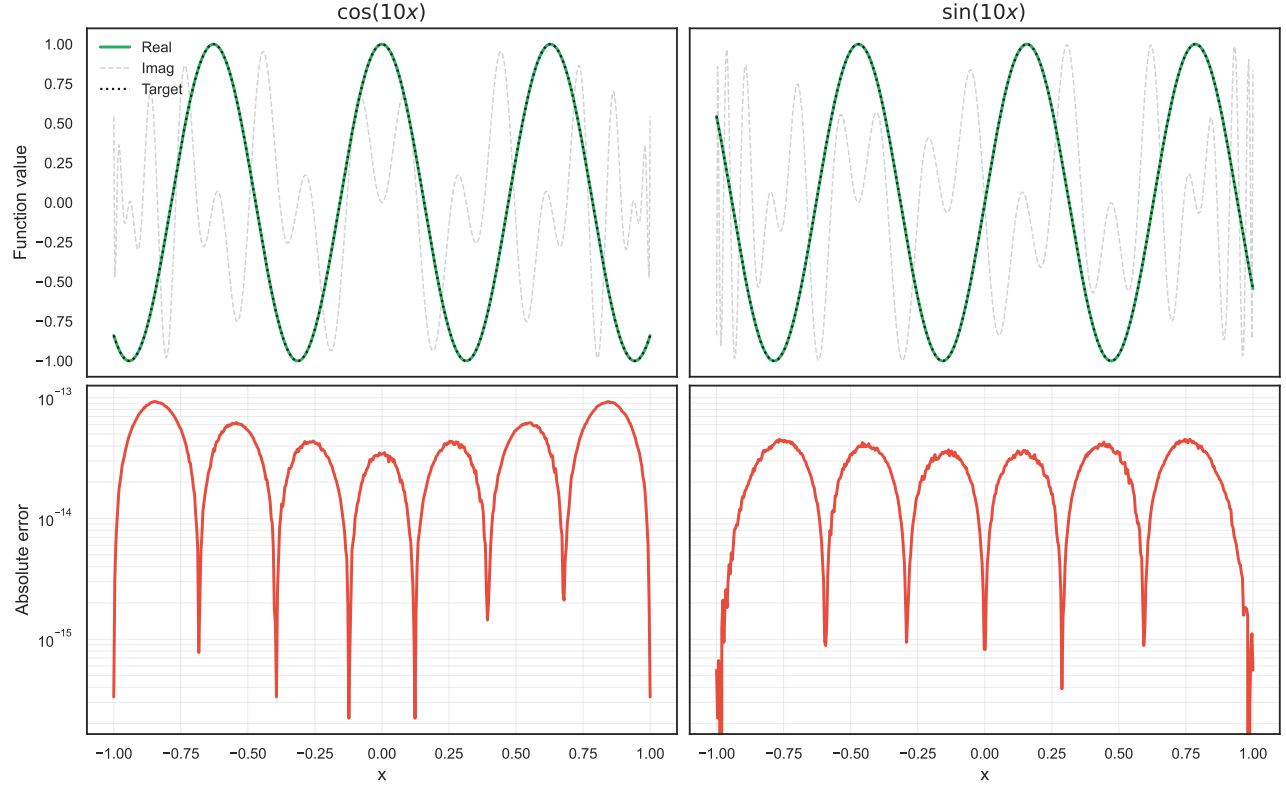
$$\begin{aligned} \left\| \cos(tx) - J_0(t) + 2 \sum_{k=1}^R (-1)^k J_{2k}(t) T_{2k}(x) \right\|_{[-1,1]} &\leq \varepsilon, \\ \left\| \sin(tx) - 2 \sum_{k=0}^R (-1)^k J_{2k+1}(t) T_{2k+1}(x) \right\|_{[-1,1]} &\leq \varepsilon, \end{aligned}$$

where $J_m(t)$ denotes the Bessel function of the first kind. Moreover, the required degree satisfies

$$d(t, \varepsilon) = \Theta\left(|t| + \frac{\ln(1/\varepsilon)}{\ln\left(e + \frac{\ln(1/\varepsilon)}{|t|}\right)}\right).$$

8.3 Fitting Polynomials with QSP

So far we have studied the theoretical admissibility of polynomials in QSP. In practice, we want to fit a target function to a QSP-realizable polynomial. We compute the phase factors Φ numerically using `pyQSP`, a Python package implementing the L-BFGS based optimisation algorithm from [DMWL21]. Figure 8.4 illustrates typical results for approximating $\cos(10x)$ and $\sin(10x)$. The optimisation procedure finds a complex polynomial $P(x)$ whose real part



reproduces the target function with near-minimax error. Consequently, the imaginary part is generally nonzero, as shown in the plots. This phenomenon reflects the fact that QSP naturally realises complex-valued polynomials. In the following chapter we will show how to handle such cases by combining even and odd components into admissible QSVT constructions.

Figure 8.4: QSP polynomial approximations to $\cos(10x)$ and $\sin(10x)$ on $[-1, 1]$ using `PYQSP`. The real parts match the targets with near-minimax error, while the imaginary parts remain nonzero.

9

Quantum Singular Value Transformation

In [Chapter 8](#), we studied how single-qubit signal processing enables polynomial transformations of matrix entries, with the phase sequence determining the implemented polynomial. The central idea of **Quantum Singular Value Transformation** (QSVT) is to lift this idea to matrices: by acting on qubitised subspaces of a block-encoding, it applies a chosen polynomial to all singular values in parallel. This chapter introduces the geometric picture underlying QSVT, formalises it in the alternating phase modulation sequence and the QSVT theorem, and presents circuit constructions for various polynomial classes. We conclude with an application to Hamiltonian simulation.

9.1 Definitions

Definition 9.1 (Singular Value Decomposition [\[BK22\]](#)). Let $A \in \mathbb{C}^{r \times c}$ be any matrix. Then there exist unitary matrices $W \in \mathbb{C}^{r \times r}$ and $V \in \mathbb{C}^{c \times c}$, and a diagonal matrix $\Sigma \in \mathbb{R}^{r \times c}$, such that

$$A = W\Sigma V^\dagger.$$

The entries of Σ are the non-negative real numbers $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > 0$ called the **singular values** of A , where $k = \text{rank}(A)$. The columns of W and V are called the **left** and **right singular vectors** of A , respectively.¹ Equivalently, the SVD can be expressed as a sum of rank-one outer products:

$$A = \sum_{i=1}^k \sigma_i |w_i\rangle \langle v_i|,$$

where $\{|w_i\rangle\} \subset \mathbb{C}^r$ and $\{|v_i\rangle\} \subset \mathbb{C}^c$ are orthonormal sets.

Definition 9.2 (Spectral Decomposition). Let $A \in \mathbb{C}^{r \times r}$ be a **normal** matrix, i.e., $AA^\dagger = A^\dagger A$. Then there exists a unitary matrix $V \in \mathbb{C}^{r \times r}$ and a diagonal matrix $D \in \mathbb{C}^{r \times r}$ such that

$$A = VDV^\dagger.$$

¹ Here $V^\dagger$ denotes the conjugate transpose of V , and all diagonal entries of Σ beyond the first k are zero.

The entries of D are the (not necessarily distinct) eigenvalues of A , and the columns of V are the corresponding orthonormal eigenvectors. This decomposition is called the **spectral decomposition** of A .² In particular, if A is Hermitian ($A = A^\dagger$), then the eigenvalues of A are real.

² If A is real and symmetric, then $A = Q\Lambda Q^\top$ with $Q \in \mathbb{R}^{r \times r}$ orthonormal and $\Lambda \in \mathbb{R}^{r \times r}$ diagonal.

Definition 9.3 ([Gan60]). Let $A \in \mathbb{C}^{r \times r}$ be a matrix with eigenvalues $\lambda_1, \dots, \lambda_s$, where each λ_i has algebraic multiplicity n_i . A function $f : \mathbb{C} \rightarrow \mathbb{C}$ is said to be **defined on the spectrum of A** if all derivatives

$$f^{(j)}(\lambda_i), \quad \text{for } 0 \leq j \leq n_i - 1 \text{ and } 1 \leq i \leq s,$$

exist. These values are collectively referred to as the values of the function f on the spectrum of A .

Definition 9.4 (Jordan Canonical Form [HJ13]). Let $A \in \mathbb{C}^{r \times r}$ be a square matrix. Then there exists an invertible matrix $S \in \mathbb{C}^{r \times r}$ such that

$$A = S J_A S^{-1},$$

where $J_A \in \mathbb{C}^{r \times r}$ is a block-diagonal matrix called the **Jordan canonical form** of A . Each diagonal block has the form

$$J_{m_k}(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k \times m_k},$$

where $\lambda_k \in \mathbb{C}$ is an eigenvalue of A and $m_k \in \mathbb{N}$ is the size of the block. The full Jordan form is the direct sum³

$$J_A = J_{m_1}(\lambda_1) \oplus \dots \oplus J_{m_s}(\lambda_s).$$

³ The Jordan form encodes both the eigenvalues of A and the structure of its generalised eigenspaces.

Definition 9.5 (Matrix Function [Higo8]). Let $A \in \mathbb{C}^{r \times r}$ be a square matrix, and let f be defined on the spectrum of A . Suppose A admits a Jordan decomposition

$$A = S J S^{-1}.$$

Then the **matrix function** $f(A)$ is defined by

$$f(A) := S f(J) S^{-1} = S \operatorname{diag}(f(J_1), \dots, f(J_s)) S^{-1},$$

where each block $f(J_k)$ is given by⁴

$$f(J_k) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix} \in \mathbb{C}^{m_k \times m_k}.$$

⁴ This construction requires that f has sufficiently many derivatives at each eigenvalue λ_k .

Definition 9.6 (Singular Value Transformation [GSLW19, Definition 16]). Let $A \in \mathbb{C}^{r \times c}$ have singular value decomposition $A = \sum_{i=1}^k \sigma_i |w_i\rangle \langle v_i|$, where $k = \min(r, c)$. Let $f : \mathbb{R} \rightarrow \mathbb{C}$ be a function that is either even or odd. Then the **singular value transformation** of A by f is defined as

$$f^{\text{SV}}(A) := \begin{cases} \sum_{i=1}^k f(\sigma_i) |w_i\rangle \langle v_i|, & \text{if } f \text{ is odd,} \\ \sum_{i=1}^c f(\sigma_i) |v_i\rangle \langle v_i|, & \text{if } f \text{ is even,} \end{cases} \quad (9.1)$$

where σ_i is defined to be zero for $i > k$,⁵

When $f(x) = p(x)$ is a polynomial that is either even or odd, the singular value transformation $p^{\text{SV}}(A)$ reduces to an ordinary matrix polynomial in A and $A^\dagger$. For instance,

$$\begin{aligned} p(x) = x^2 + 5 &\Rightarrow p^{\text{SV}}(A) = A^\dagger A + 5\mathbb{1}, \\ p(x) = x^3 + 5x &\Rightarrow p^{\text{SV}}(A) = AA^\dagger A + 5A. \end{aligned}$$

⁵ Odd functions preserve the left and right singular vector structure, while even functions apply only to the singular values, expressed in the right singular basis.

Eigen- and Singular- Value Transformation of Hermitian Matrices

Let $A \in \mathbb{C}^{r \times r}$ be Hermitian. By the spectral theorem, there exists a unitary V such that

$$A = VDV^\dagger, \quad D = \text{diag}(\lambda_1, \dots, \lambda_r) \in \mathbb{R}^{r \times r}.$$

Since $A^\dagger A = A^2 = VD^2V^\dagger$, the singular values of A are

$$\sigma_i = |\lambda_i|, \quad \sqrt{A^\dagger A} = V |D| V^\dagger,$$

where $|D| = \text{diag}(|\lambda_1|, \dots, |\lambda_r|)$. Writing $D = \text{sign}(D) |D|$ and defining $W := V \text{sign}(D)$, we obtain

$$A = W\Sigma V^\dagger, \quad \Sigma = |D|,$$

which gives the singular value decomposition of A .⁶ Thus, for Hermitian matrices the singular values are simply the absolute values of the eigenvalues, and the left and right singular vectors are eigenvectors of A up to a sign. Now let $f : \mathbb{R} \rightarrow \mathbb{C}$ be a function of definite parity. Then

$$f^{\text{SV}}(A) = \begin{cases} Wf(\Sigma)V^\dagger = Vf(\text{sign}(D)\Sigma)V^\dagger = Vf(D)V^\dagger = f(A), & f \text{ odd,} \\ Vf(\Sigma)V^\dagger = Vf(D)V^\dagger = f(A), & f \text{ even.} \end{cases} \quad (9.2)$$

Therefore, if A is Hermitian and f has definite parity, **the eigenvalue and singular value transformations of A are the same**. This structural equivalence breaks down for non-Hermitian matrices and there is no clear connection between the eigenvalue and singular value transformations.

⁶ Here, $\text{sign}(D)$ is diagonal with entries $\text{sign}(\lambda_i) \in \{-1, 0, 1\}$. If $A \succeq 0$, then $D = \Sigma$ and $W = V$, so the spectral and singular value decompositions coincide.

9.2 Quantum Singular Value Transformation

Geometric Intuition

Following the constructive approach of [Chapter 8](#), we now develop a geometric approach toward understanding **Quantum Singular Value Transformation** (QSVT). The objective is to extend the single-qubit polynomial transformations to the multi-qubit case, operating on a larger Hilbert space that contains a block-encoding of the matrix of interest.

Let $U_A : \mathbb{C}^{2^{a+n}} \rightarrow \mathbb{C}^{2^{a+n}}$ be an $(1, a, 0)$ -block-encoding of A in accordance with [Definition 7.2](#). For any $|\psi\rangle \in \mathbb{C}^{2^n}$, the action of U_A on the state $|0^a\rangle$ is given by⁷

$$U_A |0^a\rangle |\psi\rangle = |0^a\rangle A |\psi\rangle + \delta |\perp_\psi\rangle,$$

where $|\perp_\psi\rangle$ is orthogonal to the subspace spanned by $\{|0^a\rangle A |\psi\rangle : \psi \in \mathbb{C}^{2^n}\}$ and δ is a normalisation factor. Now let $A = \sum_{i=1}^k \sigma_i |w_i\rangle \langle v_i|$ be the singular value decomposition of A . Consider how U_A acts on the **right** singular vector $|v_i\rangle$:⁸

$$\begin{aligned} U_A |0\rangle |v_i\rangle &= |0\rangle A |v_i\rangle + \delta |\tilde{\perp}_i\rangle \\ &= \sigma_i |0\rangle |w_i\rangle + \sqrt{1 - \sigma_i^2} |\tilde{\perp}_i\rangle, \end{aligned} \quad (9.3)$$

where $|\tilde{\perp}_i\rangle$ is orthogonal to $|0\rangle |w_i\rangle$. Similarly, applying $U_A^\dagger$ to the **left** singular vector gives:

$$\begin{aligned} U_A^\dagger |0\rangle |w_i\rangle &= |0\rangle A^\dagger |w_i\rangle + \beta |\perp_i\rangle \\ &= \sigma_i |0\rangle |v_i\rangle + \sqrt{1 - \sigma_i^2} |\perp_i\rangle, \end{aligned} \quad (9.4)$$

where $|\perp_i\rangle$ is orthogonal to $|0\rangle |v_i\rangle$. Now, applying $U_A^\dagger$ to both sides of (9.3) yields:

$$\begin{aligned} U_A^\dagger U_A |0\rangle |v_i\rangle &= U_A^\dagger \left(\sigma_i |0\rangle |w_i\rangle + \sqrt{1 - \sigma_i^2} |\tilde{\perp}_i\rangle \right) \\ \mathbb{1} |0\rangle |v_i\rangle &= \sigma_i^2 |0\rangle |v_i\rangle + \sigma_i \sqrt{1 - \sigma_i^2} |\perp_i\rangle + \sqrt{1 - \sigma_i^2} U_A^\dagger |\tilde{\perp}_i\rangle \quad (9.5) \\ \therefore U_A^\dagger |\tilde{\perp}_i\rangle &= \sqrt{1 - \sigma_i^2} |0\rangle |v_i\rangle - \sigma_i |\perp_i\rangle. \end{aligned}$$

Similarly, applying U_A to both sides of (9.4) gives:

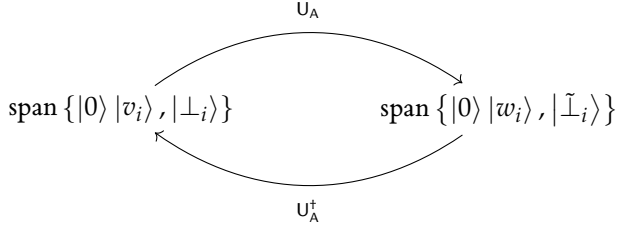
$$\begin{aligned} U_A U_A^\dagger |0\rangle |w_i\rangle &= U_A \left(\sigma_i |0\rangle |v_i\rangle + \sqrt{1 - \sigma_i^2} |\perp_i\rangle \right) \\ \mathbb{1} |0\rangle |w_i\rangle &= \sigma_i^2 |0\rangle |w_i\rangle + \sigma_i \sqrt{1 - \sigma_i^2} |\tilde{\perp}_i\rangle + \sqrt{1 - \sigma_i^2} U_A |\perp_i\rangle \quad (9.6) \\ \therefore U_A |\perp_i\rangle &= \sqrt{1 - \sigma_i^2} |0\rangle |w_i\rangle - \sigma_i |\tilde{\perp}_i\rangle. \end{aligned}$$

These calculations reveal an important structure: the two subspaces are

⁷ For clarity we temporarily assume an exact block-encoding ($\epsilon = 0$) and $\alpha = 1$.

⁸ For notational clarity, we will assume $a = 1$:

preserved under the actions of U_A and $U_A^\dagger$:



This procedure forms the algebraic foundation of singular value transformation. To build further intuition, we trace a sequence of operations that reveal the underlying two-dimensional structure:

1. Begin with the right singular vector state $|0\rangle |v_i\rangle$, and apply the block-encoding unitary U_A :

$$U_A |0\rangle |v_i\rangle = \sigma_i |0\rangle |w_i\rangle + \sqrt{1 - \sigma_i^2} |\tilde{\perp}_i\rangle.$$

2. Next, reflect across the state $|0\rangle |w_i\rangle$ using $R_{|0\rangle |w_i\rangle} := 2\Pi_L - \mathbb{1}$ where $\Pi_L = |0\rangle |w_i\rangle \langle 0| \langle w_i|$:⁹

$$(2\Pi_L - \mathbb{1}) U_A |0\rangle |v_i\rangle = \sigma_i |0\rangle |w_i\rangle - \sqrt{1 - \sigma_i^2} |\tilde{\perp}_i\rangle.$$

⁹ This reflection flips the sign of the orthogonal component while preserving the $|0\rangle |w_i\rangle$ term.

3. Apply $U_A^\dagger$ to return to the original subspace:

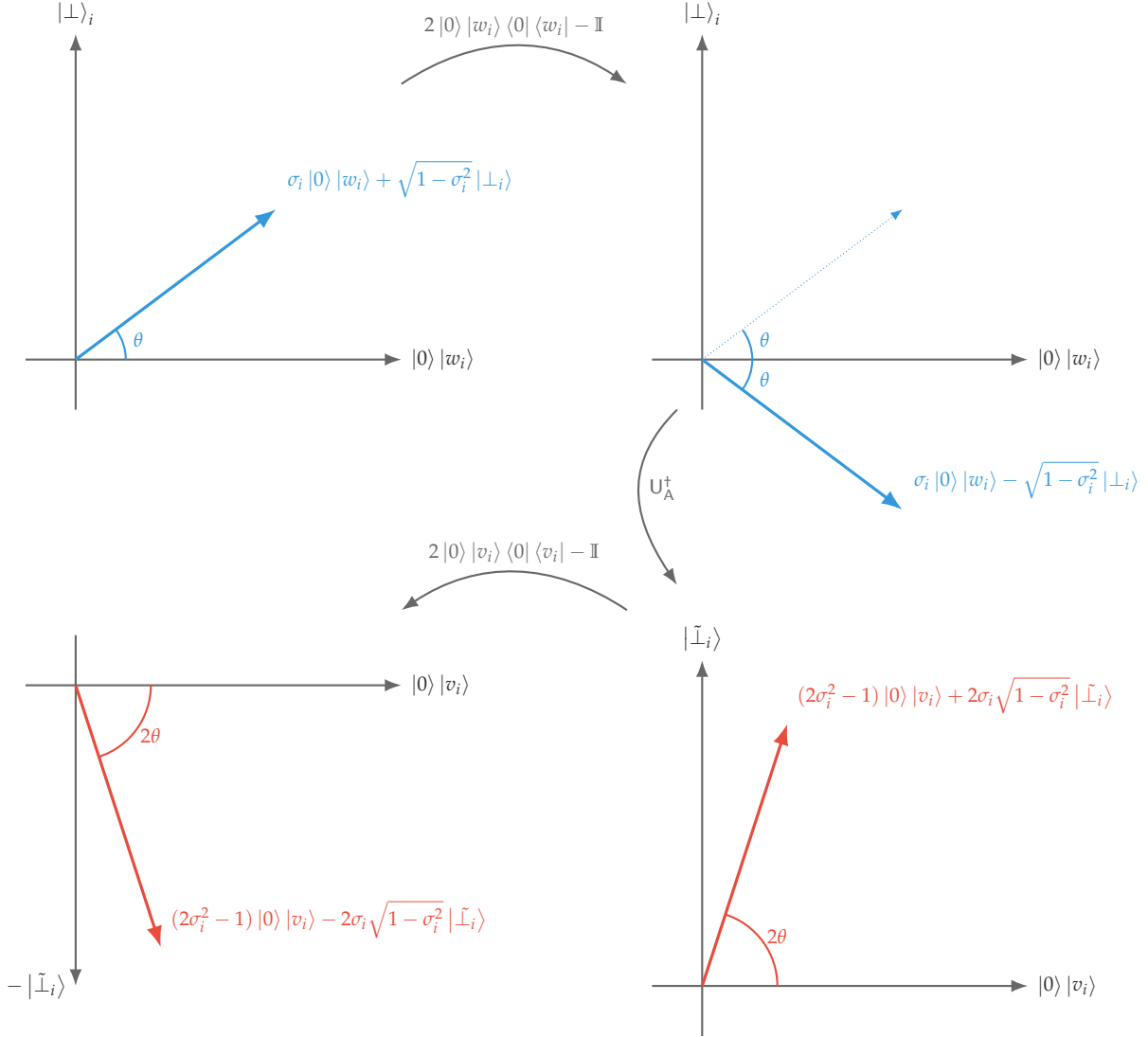
$$\begin{aligned} U_A^\dagger (2\Pi_L - \mathbb{1}) U_A |0\rangle |v_i\rangle &= \sigma_i^2 |0\rangle |v_i\rangle + \sigma_i \sqrt{1 - \sigma_i^2} |\perp_i\rangle \\ &\quad - (1 - \sigma_i^2) |0\rangle |v_i\rangle + \sigma_i \sqrt{1 - \sigma_i^2} |\perp_i\rangle \\ &= (2\sigma_i^2 - 1) |0\rangle |v_i\rangle + 2\sigma_i \sqrt{1 - \sigma_i^2} |\perp_i\rangle. \end{aligned}$$

4. Reflect once more, this time across the state $|0\rangle |v_i\rangle$:¹⁰

$$\begin{aligned} (2\Pi_R - \mathbb{1}) U_A^\dagger (2\Pi_L - \mathbb{1}) U_A |0\rangle |v_i\rangle &= (2\sigma_i^2 - 1) |0\rangle |v_i\rangle \\ &\quad - 2\sigma_i \sqrt{1 - \sigma_i^2} |\perp_i\rangle. \end{aligned}$$

¹⁰ Here, $\Pi_R = |0\rangle |v_i\rangle \langle 0| \langle v_i|$.

Examining the final expression, we see that the coefficient of $|0\rangle |v_i\rangle$ is $T_2(\sigma_i)$, the second Chebyshev polynomial, while the coefficient of $|\perp_i\rangle$ is proportional to $U_1(\sigma_i)$, the first Chebyshev polynomial of the second kind. This emergence of Chebyshev structure mirrors the pattern observed earlier in QSP, once again revealing the polynomial geometry underlying the transformation. Because this derivation holds for every singular value σ_i , the same sequence of operations applies simultaneously to all singular values. In effect, QSVT performs QSP-like rotations on each singular value individually, yet in parallel across all singular directions—an instance of the so-called **qubitised** subspaces.



The previous analysis considered the special initial state $|0\rangle |v_i\rangle$. More generally, we may begin with an arbitrary superposition supported on the right singular subspace of A . In this case, the reflection projector becomes

$$\Pi_R = |0^a\rangle \langle 0^a| \otimes \sum_{i=1}^k |v_i\rangle \langle v_i| = |0^a\rangle \langle 0^a| \otimes \mathbb{I}_{2^n},$$

where $|0^a\rangle$ denotes the ancilla register. The left projector Π_L is defined analogously in terms of the left singular vectors. The combined unitary

$$(2\Pi_R - \mathbb{I}) U_A^\dagger (2\Pi_L - \mathbb{I}) U_A$$

then acts invariantly on each two-dimensional subspace spanned by $\{|0^a\rangle |v_i\rangle, |\tilde{\perp}_i\rangle\}$.

Figure 9.1: Geometric interpretation of steps 1–4 above. The state $|0\rangle |v_i\rangle$ is mapped via a sequence of unitaries and reflections through a two-dimensional subspace spanned by singular vectors and their orthogonal complements.

To adjust the action of these reflections, we introduce phase parameters exactly as in QSP (Equation (8.4)). In place of single-qubit Z-rotations, we now use **projector-controlled rotations** (Section 6.5). These gates extend the QSP rotation operators by applying phase shifts conditioned on the projectors Π_L and Π_R . Unlike the single-qubit case, however, the relevant subspaces are embedded within a larger block-encoded matrix, and the projector formalism is essential to isolate them.

Definition 9.7 (Alternating Phase Modulation Sequence [GSLW19, Definition 15]). Let U_A be a block-encoding of a matrix A and $\Phi = \{\varphi_i\}_{i=1}^d \in \mathbb{R}^d$ be a phase vector. The **phased alternating sequence** associated with (U_A, Φ) is defined as¹¹

$$U_A(\Phi) := \begin{cases} e^{i\varphi_1(2\Pi_L - \mathbb{1})} U_A \prod_{i=1}^{\frac{d-1}{2}} \left(e^{i\varphi_{2i}(2\Pi_R - \mathbb{1})} U_A^\dagger e^{i\varphi_{2i+1}(2\Pi_L - \mathbb{1})} U_A \right), & d \text{ odd}, \\ \prod_{i=1}^{\frac{d}{2}} \left(e^{i\varphi_{2i-1}(2\Pi_R - \mathbb{1})} U_A^\dagger e^{i\varphi_{2i}(2\Pi_L - \mathbb{1})} U_A \right), & d \text{ even}. \end{cases} \quad (9.7)$$

The circuit implementing the phased alternating sequence is shown in Figure 9.3. In the block-encoding setting, projector-controlled reflections simplify to zero-controlled MCX_n gates, as illustrated in Figure 9.2. We are now ready to state the **Quantum Singular Value Transformation Theorem**, which formalises the connection between block-encodings and singular value transformations.

Theorem 9.1 (Quantum Singular Value Transformation Theorem [GSLW19, Theorem 10, Lemma 22]). Let U_A be an (α, a, ε) block-encoding of a matrix A , and let $\Phi = \{\varphi_i\}_{i=1}^d \subset \mathbb{R}$ be a phase sequence that implements the degree- d polynomial $p(x) \in \mathbb{C}[x]$ as in Corollary 8.4. Then the phased alternating sequence $U_A(\Phi)$ is an $(\alpha, a + 1, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encoding of $p^{SV}(A)$:

$$\begin{cases} \Pi_L U_A(\Phi) \Pi_R = \begin{bmatrix} p^{SV}(A) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = p^{SV}(A) (\Pi_L U_A \Pi_R), & p \text{ odd}, \\ \Pi_R U_A(\Phi) \Pi_R = \begin{bmatrix} p^{SV}(A) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \Pi_R p^{SV}(A) (\Pi_L U_A \Pi_R) \Pi_R, & p \text{ even}. \end{cases} \quad (9.8)$$

Controlled Phased Alternating Sequence

In many applications we require a **controlled** version of the phased alternating sequence. At first glance, this appears costly: one might expect to control all d occurrences of U_A and $U_A^\dagger$. However, the structure of the sequence contains many compute-uncompute pairs, which lead to the following simplification:

¹¹ Products are ordered left-to-right, so φ_d is applied first.

When working specifically with block-encodings (i.e., $\Pi_L = |0^a\rangle\langle 0^a| \otimes \mathbb{1}$), the projector-controlled rotation reduces to

$$e^{i\varphi(2\Pi_L - \mathbb{1})} = \begin{bmatrix} e^{i\varphi} \mathbb{1} & \mathbf{0} \\ \mathbf{0} & e^{-i\varphi} \mathbb{1} \end{bmatrix},$$

where the block structure corresponds to the image of Π_L and its orthogonal complement.

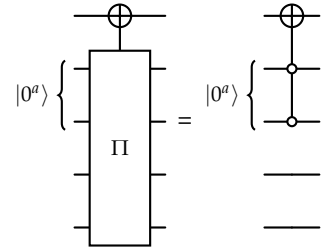


Figure 9.2: Controlled projector for block-encoding.

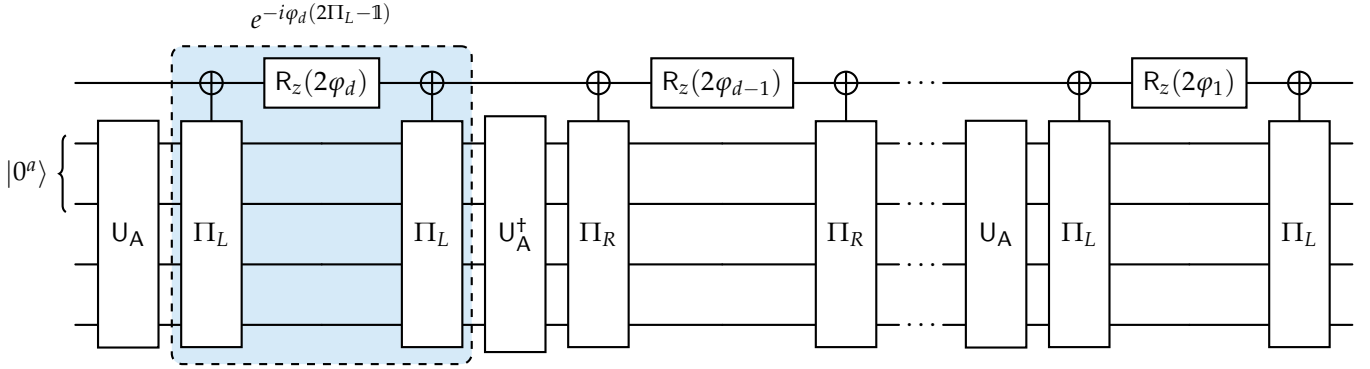


Figure 9.3: Quantum circuit implementing the phased alternating sequence for odd d .

- If d is even, the controlled sequence is obtained by controlling only the single-qubit $R_z(\varphi)$ rotations.
- If d is odd, it suffices to control the $R_z(\varphi)$ rotations **and** a single instance of the block-encoding U_A .

The reasoning is straightforward: if the control qubit is in the $|0\rangle$ state, none of the controlled rotations are applied. The overall circuit therefore collapses to the identity when d is even, or to a single U_A when d is odd. Conversely, when the control qubit is in the $|1\rangle$ state, the full phased alternating sequence is applied as usual. The resulting controlled construction is shown in Figure 9.4.

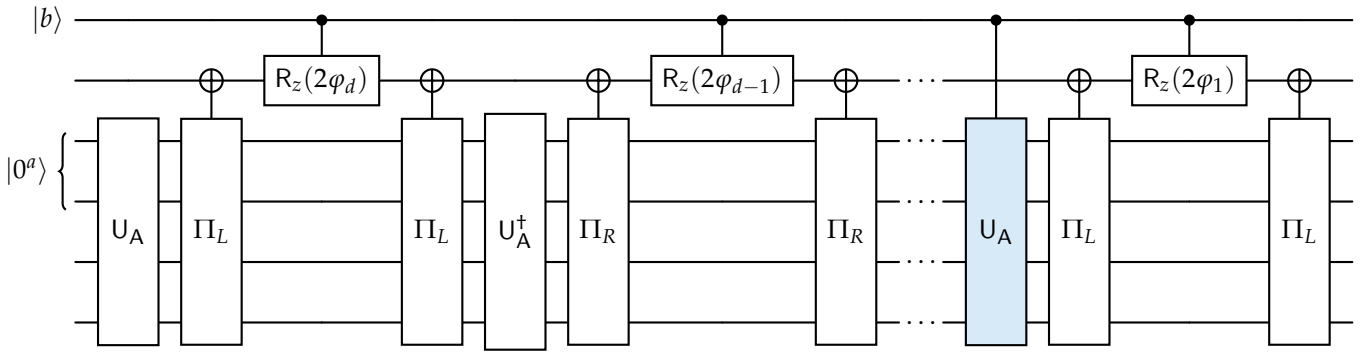


Figure 9.4: Controlled version of Figure 9.3 for odd d .

9.3 Matrix Polynomials for Hermitian Matrices

We now focus on the case where the target matrix A is Hermitian. In this setting, singular value transformation and eigenvalue transformation coincide. Suppose U_A is an (α, a, ε) -block-encoding of A . Our goal is to implement block-encodings of matrix polynomials $p(A)$ for various classes of polynomials. The complexity bounds we obtain are consistent with those in [GSLW19, Theorem 56], but our presentation breaks the cases down explicitly and

provides concrete circuit constructions, making the implementation details transparent.

Real-Valued Polynomials with Definite Parity

Let $p(x) \in \mathbb{R}[x]$ be a polynomial satisfying the conditions of [Corollary 8.6](#). Then there exists a phase sequence Φ such that QSP implements a polynomial $q(x) \in \mathbb{C}[x]$ with $\text{Re}(q) = p$. By [Theorem 9.1](#), $U_A(\Phi)$ is an $(\alpha, a + 1, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encoding of $q(A)$, and from [Theorem 7.5](#) we know that $U_A(-\Phi)$ encodes $q^*(A)$. Since

$$p(A) = \frac{1}{2} (q(A) + q^*(A)),$$

we may use [Theorem 7.3](#) with state-preparation pair (H, H) to obtain an $(\alpha, a + 2, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encoding of $p(A)$. The corresponding circuit is shown in [Figure 9.5](#).

Ancilla optimisation. Closer inspection of the QSVT circuit in [Figure 9.3](#) reveals that one ancilla qubit can be saved. The control ancilla naturally switches the sign of all phase rotations: when in $|0\rangle$ the circuit applies $R_z(\varphi)$, and when in $|1\rangle$ it applies $R_z(-\varphi)$. Thus the entire unitary may be expressed as

$$\tilde{U} = |0\rangle\langle 0| \otimes U_A(\Phi) + |1\rangle\langle 1| \otimes U_A(-\Phi).$$

Conjugating by Hadamards gives

$$(|0\rangle\langle 0| \otimes \mathbb{1})(H \otimes \mathbb{1}) \tilde{U} (H \otimes \mathbb{1})(|0\rangle\langle 0| \otimes \mathbb{1}) = \frac{1}{2} (U_A(\Phi) + U_A(-\Phi)),$$

which is precisely the block-encoding of $p(A)$ we require. This optimised construction (shown in [Figure 9.6](#)) achieves the same functionality with a single ancilla qubit and two additional Hadamards, yielding an $(\alpha, a + 1, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encoding of $p(A)$. From this point onward we adopt this improved design for real polynomials of definite parity.

Imaginary multiples. Finally, if we wish to implement a block-encoding of $ip(A)$ for such a polynomial $p(x)$, this may be achieved by applying a global phase of i to the ancilla qubit in [Figure 9.5](#), realised by inserting a Z gate. The resulting circuit is shown in [Figure 9.7](#).

Real-Valued Polynomials with Indefinite Parity

Consider a real polynomial $p(x) \in \mathbb{R}[x]$ with $|p(x)| \leq 1$ on $[-1, 1]$, but without any parity constraint. We may always decompose p into even and odd components:

$$p_{\text{even}}(x) = \frac{1}{2}(p(x) + p(-x)), \quad p_{\text{odd}}(x) = \frac{1}{2}(p(x) - p(-x)).$$

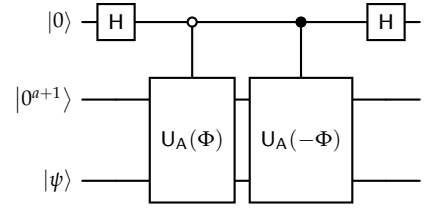


Figure 9.5: Block-encoding of $p(A)$ using LCBU to separate the real and imaginary parts.

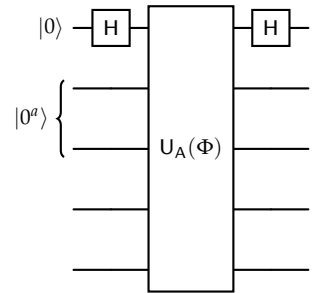


Figure 9.6: Optimised circuit for block-encoding $p(A)$ using only one additional ancilla qubit compared to the LCBU construction in [Figure 9.5](#).

Since p_{even} and p_{odd} satisfy the conditions of the previous section, they can be implemented using the same constructions. Finally, by applying [Theorem 7.3](#) with the state-preparation pair (H, H) , we obtain an $(2\alpha, a + 2, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block encoding of $p(A)$.

General Polynomials

Let $p(x) \in \mathbb{C}[x]$ be an arbitrary polynomial without parity constraints. We first decompose it into real and imaginary parts:

$$p(x) = g(x) + ih(x), \quad g, h \in \mathbb{R}[x].$$

We then construct block-encodings of $g(A)$ and $h(A)$ separately. Two cases arise:

1. **Both g and h have definite parity.** Using the optimised construction of [Figure 9.6](#), we obtain $(\alpha, a + 1, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encodings of $g(A)$ and $h(A)$. Combining them via LCU with the state-preparation pair (H, H) yields a $(2\alpha, a + 2, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block block-encoding of $p(A)$.
2. **At least one of g or h lacks definite parity.** In this case, we decompose the non-parity component into its even and odd parts as before. This produces $(2\alpha, a + 2, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block-encodings of $g(A)$ and $h(A)$. Applying LCU once more with (H, H) yields a final $(4\alpha, a + 3, 4d\sqrt{\frac{\varepsilon}{\alpha}})$ -block encoding of $p(A)$.

9.4 Hamiltonian Simulation via QSVT

In the previous section, we described how to implement a block-encoding of a polynomial transformation $p(A)$ of a Hermitian matrix A . We now apply this framework to Hamiltonian simulation, where the goal is to implement the time-evolution operator $\exp(iAT)$ for a Hermitian matrix A up to precision ε . Suppose we are given an $(\alpha, a, 0)$ -block-encoding of A . We first rescale time as

$$\exp(iAT) = \exp\left(i \frac{A}{\alpha} \cdot \alpha T\right) = \exp(iA'\tau),$$

where $A' = A/\alpha$ and $\tau = \alpha T$. Note that A' is Hermitian with spectrum contained in $[-1, 1]$. Thus, it suffices to construct a block-encoding of $\exp(iA'\tau)$ up to precision ε given a $(1, a, 0)$ -block encoding of A' . For notational simplicity we relabel $A' \rightarrow A$.

As before, we decompose $e^{i\tau x}$ into its even and odd components

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

so that $\cos(\tau x)$ and $\sin(\tau x)$ can be approximated to precision $\varepsilon/6$ by polynomials $p_{\text{even}}(x)$ and $p_{\text{odd}}(x)$, respectively, via [Theorem 8.7](#). Using [Figure 9.6](#),

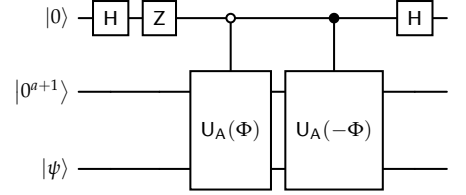


Figure 9.7: Block-encoding of $ip(A)$ using LCBU.

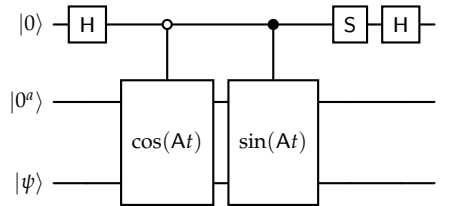


Figure 9.8: Block-encoding of $\frac{1}{2}e^{itA}$.

this yields a $(2, a + 2, \frac{\varepsilon}{6})$ -block-encoding of

$$U_{\exp(A)} = \cos(A\tau) + i \sin(A\tau).$$

The corresponding circuit is shown in [Figure 9.8](#). To remove the factor of 2 in the normalisation, we must apply amplitude amplification. Since $U_{\exp(A)}$ is only $\delta = \frac{1}{2}$ -close to unitary, we require the **robust** version of oblivious amplitude amplification (OAA) as described in [Section 6.6](#). Alternatively, QSVT itself can be used to perform the amplification.

Amplification via QSVT

Observe that $\exp(iA\tau)$ is unitary, hence all its singular values are equal to 1, and we may write

$$\exp(iA\tau) = \sum_{i=1}^r 1 |w_i\rangle \langle v_i|, \quad r = \text{rank}(A).$$

In contrast, the block-encoding constructed above gives

$$\frac{1}{2} \exp(iA\tau) = \sum_{i=1}^r \frac{1}{2} |w_i\rangle \langle v_i|.$$

Applying QSVT with the polynomial $p(x) = -T_3(x) = 3x - 4x^3$ amplifies the singular values:

$$p^{\text{SV}}\left(\frac{1}{2} \exp(iA\tau)\right) = \sum_{i=1}^r p\left(\frac{1}{2}\right) |w_i\rangle \langle v_i| = \sum_{i=1}^r 1 |w_i\rangle \langle v_i| = \exp(iA\tau).$$

Thus, by applying QSVT with $p(x) = -T_3(x)$ to the block-encoding of $\frac{1}{2} \exp(iA\tau)$, we obtain a block encoding of $\exp(iA\tau)$ itself. The phase sequence implementing $-T_3(x)$ under [Corollary 8.4](#) is simply

$$\Phi = \left\{ 2\pi, -\frac{\pi}{2}, -\frac{\pi}{2} \right\}.$$

This demonstrates another equivalence between QSVT and established procedures such as robust OAA: both can be used to amplify success probability and restore exact unitarity.

Complexity Analysis

A complete complexity analysis that incorporates block-encoding error is more delicate than simple error propagation. While we could in principle track errors directly through successive block-encodings, this approach leads to suboptimal estimates in the Hamiltonian simulation setting. A careful analysis based on additional inequalities provides sharper control of error accumulation, but this lies beyond the scope of this thesis. We therefore state the final result without proof and refer the reader to [\[GSLW19, Corollary 62\]](#) for a complete treatment.¹²

¹² There is a typo in [\[GSLW19, Corollary 62\]](#): the number of ancillae is stated as $a + 2$ rather than $a + 3$.

Theorem 9.2 (Robust Block-Hamiltonian Simulation [GSLW19, Corollary 62]).

Let $T \in \mathbb{R}$, $\varepsilon \in (0, 1)$, and let U_A be an $(\alpha, a, \frac{\varepsilon}{2|T|})$ -block-encoding of a Hermitian matrix A . Then we can implement an $(1, a + 3, \varepsilon)$ -block-encoding of $\exp(iAT)$ with

$$6\alpha|T| + 9 \log\left(\frac{12}{\varepsilon}\right)$$

queries to U_A and $U_A^\dagger$, three queries to controlled- U_A and controlled- $U_A^\dagger$, and

$$\mathcal{O}(a\alpha|T| + \log(1/\varepsilon))$$

two-qubit gates.

To build intuition, we sketch the resource requirements step by step:

- **Cosine block-encoding.** Constructing $\cos(A\tau)$ requires d queries to $U_A/U_A^\dagger$, together with d controlled rotations and $2d$ MCX _{n} gates.
- **Sine block-encoding.** Constructing $\sin(A\tau)$ similarly requires d queries to $U_A/U_A^\dagger$, plus one query to controlled- $U_A/U_A^\dagger$, and again d controlled rotations and $2d$ MCX _{n} gates.
- **Half-exponential.** Combining the two yields a block-encoding of $\frac{1}{2} \exp(iA\tau)$ at a cost of $2d$ queries to $U_A/U_A^\dagger$, one controlled query, $2d$ controlled rotations, and $4d$ MCX _{n} gates.
- **Amplification.** The QSVT amplification step requires three applications of the block-encoding of $\frac{1}{2} \exp(iA\tau)$, together with three applications of controlled projectors.

Collecting terms, the full cost of Hamiltonian simulation via QSVT is

$$6d = 6 \Theta(\alpha|T| + \log(1/\varepsilon))$$

queries to $U_A/U_A^\dagger$, three queries to controlled- $U_A/U_A^\dagger$, together with $6d$ controlled rotations and $12d$ MCX _{n} gates.

In the applications considered in this thesis, we assume a **perfect** block-encoding (i.e., $\varepsilon_{BE} = 0$) and therefore account only for the approximation error arising from the polynomial representations of $\cos(AT)$ and $\sin(AT)$. Consequently, our Hamiltonian simulation inherits the asymptotic complexity

$$\Theta\left(\alpha|T| + \frac{\ln(1/\varepsilon)}{\ln\left(e + \frac{\ln(1/\varepsilon)}{\alpha|T|}\right)}\right).$$

in accordance with [GSLW19, Theorem 58].

Quantum Algorithm for the Advection Equation

This chapter unifies the classical and quantum components of our framework, demonstrating how the SBP-based semi-discrete formulation of the linear advection equation can be realised as a quantum algorithm through block-encoding and QSVT. Building upon an assumed state-preparation procedure, we describe the quantum realisation of the semi-discrete system, detailing its time evolution and solution reconstruction, and derive an explicit bound connecting the classical discretisation and quantum approximation errors. The chapter concludes with an analysis of quantum circuit complexity, highlighting the scaling of the block-encoding normalisation factor, QSVT query depth, and the optimal selection of polynomial degree and element count.

10.1 Linear Advection in One Dimension

Consider the linear advection equation in one spatial dimension with a positive unit wave speed and periodic boundary conditions:

$$\begin{aligned}\frac{\partial \mathcal{U}}{\partial t} + \frac{\partial \mathcal{U}}{\partial x} &= 0, \quad \forall x \in \Omega := [a, b], \quad \forall t \geq 0, \\ \mathcal{U}(a, t) &= \mathcal{U}(b, t), \quad \forall t > 0, \\ \mathcal{U}(x, 0) &= \mathcal{F}(x), \quad \forall x \in \Omega, \quad t = 0.\end{aligned}$$

Domain Decomposition and Mapping

The domain is partitioned into n_{elem} uniform elements of size $\Delta x = \frac{b-a}{n_{\text{elem}}}$, with nodes $x_0 = a < \dots < x_{n_{\text{elem}}} = b$ as shown in Figure 10.1. We label each element by $\kappa = 1, \dots, n_{\text{elem}}$, so that

$$\Omega_\kappa = [x_{\kappa-1}, x_\kappa] = [a + (\kappa - 1)\Delta x, a + \kappa\Delta x].$$

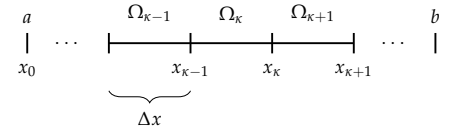


Figure 10.1: $\Omega = [a, b]$ partitioned into n_{elem} non-overlapping elements $\{\Omega_\kappa\}_{\kappa=1}^{n_{\text{elem}}}$ of equal size $\Delta x = \frac{(b-a)}{n_{\text{elem}}}$.

Each element is mapped to the reference interval $\hat{\Omega} = [-1, 1]$ using the affine transformation

$$\Phi_\kappa(\xi) = \frac{\Delta x}{2} \xi + \frac{x_{\kappa-1} + x_\kappa}{2}, \quad \xi \in [-1, 1].$$

The Jacobian of this mapping is constant on each element:

$$J = \frac{dx}{d\xi} = \frac{\Delta x}{2}, \quad \mathcal{J} = \det(J) = \frac{\Delta x}{2}.$$

Reference Element Operators

On the reference element $\hat{\Omega} = [-1, 1]$, there are two boundary facets,

$$\hat{\Gamma}_L = \{-1\}, \quad \hat{\Gamma}_R = \{+1\},$$

corresponding to the left and right boundaries. For these facets we define the following operators:

$$\begin{aligned} B_L &= [1], B_R = [1], \\ N_L &= [-1], N_R = [+1]. \end{aligned}$$

Here B encodes the boundary quadrature weights, and N stores the outward normal directions. The reference SBP identity takes the form

$$\hat{E} = \hat{Q} + \hat{Q}^T = R_R^T R_R - R_L^T R_L,$$

where $\hat{Q}$ is the differentiation operator on $\hat{\Omega}$, and R_L, R_R are the extrapolation operators to the left and right facets, respectively.

Semi-Discrete Form

Starting from the general semi-discrete SBP-SAT formulation in (4.32), the case $d = 1$ with unit positive wave speed reduces to

$$\frac{d\mathbf{u}_\kappa}{dt} = \left(\mathcal{J}_\kappa \hat{H} \right)^{-1} Q_\kappa^T \mathbf{u}_\kappa - \sum_{\gamma \in \Gamma_\kappa} \left(\mathcal{J}_\kappa \hat{H} \right)^{-1} R_{\kappa\gamma}^T B_\gamma \mathbf{f}_\kappa^*, \quad (10.1)$$

where $\mathbf{u}_\kappa$ are the degrees of freedom on element κ , $\Gamma_\kappa = \{\gamma_L, \gamma_R\}$ denotes the left and right facets of the element, and $\mathbf{f}_\kappa^*$ is the numerical flux.¹ Using the constant Jacobian $\mathcal{J}_\kappa = \frac{\Delta x}{2} \mathbb{1}$, (10.1) simplifies to

$$\frac{d\mathbf{u}_\kappa}{dt} = \frac{2}{\Delta x} \hat{H}^{-1} \left(Q_\kappa^T \mathbf{u}_\kappa - \sum_{\gamma \in \Gamma_\kappa} R_\gamma^T \mathbf{f}_\kappa^* \right). \quad (10.2)$$

We employ symmetric SATs with parameter $\lambda = 0$ in (4.34), so that

$$\mathbf{f}^*(\mathbf{u}^+, \mathbf{u}^-; \mathbf{n}^+) = \frac{1}{2} N_\gamma^+ (\mathbf{u}^+ + \mathbf{u}^-),$$

¹ We suppress explicit time dependence in $\mathbf{u}$ and $\mathbf{f}^*$ for readability.

on each facet $\gamma \in \Gamma_\kappa$, where $\mathbf{u}^-$ and $\mathbf{u}^+$ denote interior and exterior traces, respectively. Substituting the flux into (10.2) yields

$$\begin{aligned} \sum_{\gamma \in \Gamma_\kappa} \mathbf{R}_\gamma^T \mathbf{f}_\kappa^* &= \frac{1}{2} \left(\mathbf{R}_L^T \mathbf{N}_L^+ (\mathbf{u}_L^+ + \mathbf{u}_L^-) + \mathbf{R}_R^T \mathbf{N}_R^+ (\mathbf{u}_R^+ + \mathbf{u}_R^-) \right) \\ &= \frac{1}{2} \left(\mathbf{R}_L^T \mathbf{N}_L \mathbf{R}_L \mathbf{u}_\kappa + \mathbf{R}_R^T \mathbf{N}_R \mathbf{R}_R \mathbf{u}_\kappa + \mathbf{R}_L^T \mathbf{N}_L \mathbf{R}_R \mathbf{u}_{\kappa-1} + \mathbf{R}_R^T \mathbf{N}_R \mathbf{R}_L \mathbf{u}_{\kappa+1} \right) \\ &= \frac{1}{2} \left((\mathbf{R}_R^T \mathbf{R}_R - \mathbf{R}_L^T \mathbf{R}_L) \mathbf{u}_\kappa + \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L \mathbf{u}_{\kappa+1} - \frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R \mathbf{u}_{\kappa-1} \right). \end{aligned} \quad (10.3)$$

Here, we used that the exterior state at the left facet satisfies $\mathbf{u}_L^- = \mathbf{R}_R \mathbf{u}_{\kappa-1}$, while the exterior state at the right facet satisfies $\mathbf{u}_R^- = \mathbf{R}_L \mathbf{u}_{\kappa+1}$. Inserting (10.3) into (10.2) gives the final semi-discrete evolution equation:

$$\frac{d\mathbf{u}_\kappa}{dt} = -\frac{2}{\Delta x} \hat{\mathbf{H}}^{-1} \left(\mathbf{S}_\kappa \mathbf{u}_\kappa + \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L \mathbf{u}_{\kappa+1} - \frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R \mathbf{u}_{\kappa-1} \right), \quad (10.4)$$

where the local stiffness operator is defined as

$$\mathbf{S}_\kappa = \mathbf{Q}_\kappa - \frac{1}{2} (\mathbf{R}_R^T \mathbf{R}_R - \mathbf{R}_L^T \mathbf{R}_L).$$

Equation (10.4) shows explicitly the contributions from the **element interior**, the coupling to the **right neighbour**, and the coupling to the **left neighbour**. The system of ODEs in (10.4) can be combined into a global system of the form

$$\frac{d}{dt} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{N-1} \\ \mathbf{u}_N \end{bmatrix} = -\frac{2}{\Delta x} \hat{\mathbf{H}}^{-1} \begin{bmatrix} \mathbf{S}_1 & \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L & \mathbf{0} & \dots & -\frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R \\ -\frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R & \mathbf{S}_2 & \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L & \dots & \mathbf{0} \\ \vdots & -\frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R & \mathbf{S}_3 & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L \\ \frac{1}{2} \mathbf{R}_R^T \mathbf{R}_L & \dots & \mathbf{0} & -\frac{1}{2} \mathbf{R}_L^T \mathbf{R}_R & \mathbf{S}_N \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{N-1} \\ \mathbf{u}_N \end{bmatrix}. \quad (10.5)$$

Here the periodic boundary conditions are enforced by coupling the **first and last elements**. The global semi-discrete system can be written compactly as

$$\frac{d\mathbf{u}}{dt} = \mathbf{A} \mathbf{u}, \quad \mathbf{A} = -\frac{2}{\Delta x} \hat{\mathbf{H}}^{-1} \mathbf{Q}_g, \quad (10.6)$$

where $\mathbf{u}$ collects all element degrees of freedom and $\mathbf{Q}_g$ denotes the global system matrix.

The exact time evolution is given by

$$\mathbf{u}(t) = \exp(\mathbf{A}t) \mathbf{u}(0). \quad (10.7)$$

If we use the classical SBP operators (cf. 3.7), the matrix Q_g becomes skew-symmetric. However, the associated norm matrix $\hat{H}$ is not a scaled identity, so the discrete operator $A = -\hat{H}^{-1}Q_g$ is not skew-symmetric in general. We are particularly interested in the case where A is skew-symmetric, since then

$$\exp(At) = \exp\left(i \frac{A}{i} t\right),$$

and $\frac{A}{i}$ is Hermitian. This allows the time evolution operator $\exp(At)$ to be implemented on a quantum computer via QSVT as a polynomial transformation of the Hermitian operator $\frac{A}{i}$.

Modal Discretisation

To obtain a skew-symmetric global operator A , we employ a modal discretisation with **orthonormal** Legendre basis functions. On each element κ , the solution takes the form

$$u_h^{(\kappa)}(x, t) = \sum_{i=0}^p \hat{u}_i^{(\kappa)}(t) \varphi_i(\Phi_\kappa^{-1}(x)) = \sum_{i=0}^p \hat{u}_i^{(\kappa)}(t) \varphi_i(\xi), \quad x \in \Omega_\kappa, \quad (10.8)$$

where $\{\hat{u}_i^{(\kappa)}\}_{i=0}^p$ are the **modal coefficients** and φ_i is the i -th orthonormalised Legendre polynomial. The Legendre polynomials $P_i(\xi)$ satisfy

$$\int_{-1}^1 P_i(\xi) P_j(\xi) d\xi = \frac{2}{2i+1} \delta_{ij}.$$

Consequently, defining

$$\varphi_i(\xi) = \sqrt{\frac{2i+1}{2}} P_i(\xi),$$

we obtain an orthonormal basis on $[-1, 1]$:

$$\int_{-1}^1 \varphi_i(\xi) \varphi_j(\xi) d\xi = \delta_{ij}.$$

Figure 10.2 shows the first few orthonormal Legendre polynomials on $[-1, 1]$.

Given a continuous function $\mathcal{G}(x)$, the modal coefficients are obtained by projection onto the basis,

$$\hat{u}_i^{(\kappa)} = \int_{-1}^1 \mathcal{G}(\Phi_\kappa(\xi)) \varphi_i(\xi) d\xi. \quad (10.9)$$

In practice, these integrals are approximated using high-order quadrature rules to ensure accuracy of the modal representation. We employ adaptive Gauss–Kronrod quadrature based on the `QuadGK.jl` package [Jul25] with a tight absolute tolerance of $\varepsilon_q = 10^{-15}$ to suppress numerical errors in the projection.

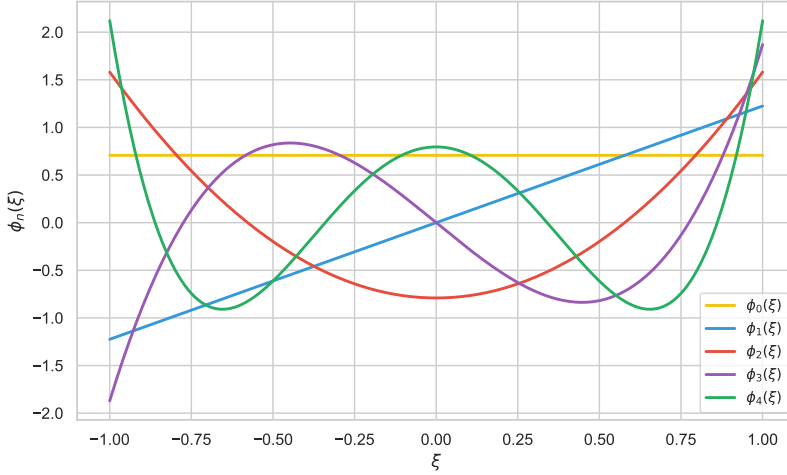


Figure 10.2: First five orthonormal Legendre polynomials on $[-1, 1]$.

Recall that the diagonal mass matrix defines a discrete inner product on the reference element. For the orthonormal Legendre basis $\{\varphi_i\}_{i=0}^p$, the modal mass matrix is

$$\hat{H}_{ij} = \int_{-1}^1 \varphi_i(\xi) \varphi_j(\xi) d\xi = \delta_{ij}. \quad (10.10)$$

Thus $\hat{H} = \mathbb{1}_{p+1}$, so $\hat{H}^{-1} = \hat{H} = \mathbb{1}_{p+1}$. Similarly, the entries $Q_{\kappa ij}$ are given by

$$\begin{aligned} Q_{\kappa, ij} &= \int_{x_{\kappa-1}}^{x_{\kappa}} \varphi_i(\Phi_{\kappa}^{-1}(x)) \frac{\partial}{\partial x} \varphi_j(\Phi_{\kappa}^{-1}(x)) dx \\ &= \int_{-1}^1 \varphi_i(\xi) \varphi_j'(\xi) \frac{2}{\Delta x} \frac{\Delta x}{2} d\xi \\ &= \int_{-1}^1 \varphi_i(\xi) \varphi_j'(\xi) d\xi, \end{aligned} \quad (10.11)$$

where we performed the change of variables $\xi = \Phi_{\kappa}^{-1}(x)$. Hence Q_{κ} is **independent** of the element index κ , depending only on the local basis functions. The final expression in (10.11) has a closed-form solution (see [Appendix C](#)):

$$Q_{ij} = \int_{-1}^1 \varphi_i(\xi) \varphi_j'(\xi) d\xi = \begin{cases} \sqrt{(2i+1)(2j+1)}, & i+j \text{ odd}, j > i, \\ 0, & \text{otherwise.} \end{cases} \quad (10.12)$$

Finally, the extrapolation operators to the left and right boundaries of the reference element are given by evaluation of the basis functions:

$$\begin{aligned} R_L &= [\varphi_0(-1) \quad \varphi_1(-1) \quad \cdots \quad \varphi_p(-1)], \\ R_R &= [\varphi_0(+1) \quad \varphi_1(+1) \quad \cdots \quad \varphi_p(+1)]. \end{aligned} \quad (10.13)$$

Using $\varphi_i(1) = \sqrt{\frac{2i+1}{2}}$ and $\varphi_i(-1) = (-1)^i \sqrt{\frac{2i+1}{2}}$, we have

$$\begin{aligned} E_{ij} &= \frac{1}{2} \left(R_R^T R_R - R_L^T R_L \right)_{ij} \\ &= \frac{1}{2} (\varphi_i(1)\varphi_j(1) - \varphi_i(-1)\varphi_j(-1)) \\ &= \frac{1}{4} \left(1 - (-1)^{i+j} \right) \sqrt{(2i+1)(2j+1)} \\ &= \begin{cases} 0, & i+j \text{ even,} \\ \frac{1}{2} \sqrt{(2i+1)(2j+1)}, & i+j \text{ odd.} \end{cases} \end{aligned}$$

Hence, the stiffness matrix entries are

$$S_{ij} = Q_{ij} - E_{ij} = \begin{cases} \frac{1}{2} \sqrt{(2i+1)(2j+1)}, & i+j \text{ odd, } j > i, \\ -\frac{1}{2} \sqrt{(2i+1)(2j+1)}, & i+j \text{ odd, } i > j, \\ 0, & \text{otherwise.} \end{cases} \quad (10.14)$$

Similarly,

$$\begin{aligned} \frac{1}{2} \left(R_R^T R_L \right)_{ij} &= \frac{1}{4} (-1)^j \sqrt{(2i+1)(2j+1)}, \\ \frac{1}{2} \left(R_L^T R_R \right)_{ij} &= \frac{1}{4} (-1)^i \sqrt{(2i+1)(2j+1)}. \end{aligned} \quad (10.15)$$

Thus, $\frac{1}{2} R_R^T R_L = \left(\frac{1}{2} R_L^T R_R \right)^T$, so the off-diagonal coupling blocks are transposed negatives in the global operator.

From (10.14) it is immediate that S is skew-symmetric. Combining (10.14) and (10.15), we conclude that the global operator Q_g is skew symmetric. Consequently, with $\hat{H} = \mathbb{1}$, the global semi-discrete operator simplifies to

$$A = -\frac{2}{\Delta x} Q_g,$$

which is itself skew-symmetric. In this case,

$$\exp(At) = \exp\left(i \frac{A}{i} t\right),$$

and $\frac{A}{i}$ is Hermitian as desired. The dimension of the global discretisation operator is determined by the total number of elements and the local polynomial order:

$$N := n_{\text{elem}} (p+1).$$

Each of the n_{elem} elements contributes a local block of size $(p+1) \times (p+1)$.

Numerical Results

We solve the one-dimensional linear advection equation on $\Omega = [0, 1]$ with periodic boundary conditions, positive unit wave speed using the modal Legendre discretisation. The initial condition is

$$\mathcal{F}(x) = \sin(2\pi x),$$

whose modal coefficients on element κ are obtained by projection:

$$\hat{u}_i^{(\kappa)}(0) = \int_{-1}^1 \sin(2\pi \Phi_\kappa(\xi)) \varphi_i(\xi) d\xi, \quad i = 0, \dots, p. \quad (10.16)$$

The global semi-discrete system is advanced in time to $t = 1$ using a Krylov subspace based exponential integrators from the `ExponentialUtilities.jl` package [Sci25]. The exact solution is given by

$$\mathcal{U}(x, t) = \sin(2\pi(x - t)),$$

and the error is measured in the L^2 norm,

$$\begin{aligned} \|\mathcal{U} - u_h\|_{L^2} &= \left(\sum_{\kappa=1}^N \int_{\Omega_\kappa} (\mathcal{U}(x, t) - u_h^{(\kappa)}(x, t))^2 dx \right)^{1/2} \\ &= \left(\frac{\Delta x}{2} \sum_{\kappa=1}^N \int_{-1}^1 \left(\mathcal{U}(\Phi_\kappa(\xi), t) - \sum_{i=0}^p \hat{u}_i^{(\kappa)}(t) \varphi_i(\xi) \right)^2 d\xi \right)^{1/2}. \end{aligned} \quad (10.17)$$

Elementwise integrals are again approximated numerically with a tight absolute tolerance. Figure 10.3 reports $\|\mathcal{U} - u_h\|_{L^2}$ versus mesh spacing Δx for polynomial degrees $p = 1$ to 5. The rates are computed using a least-squares fit on the last three data points and are consistent with the expected high-order convergence for smooth periodic data.

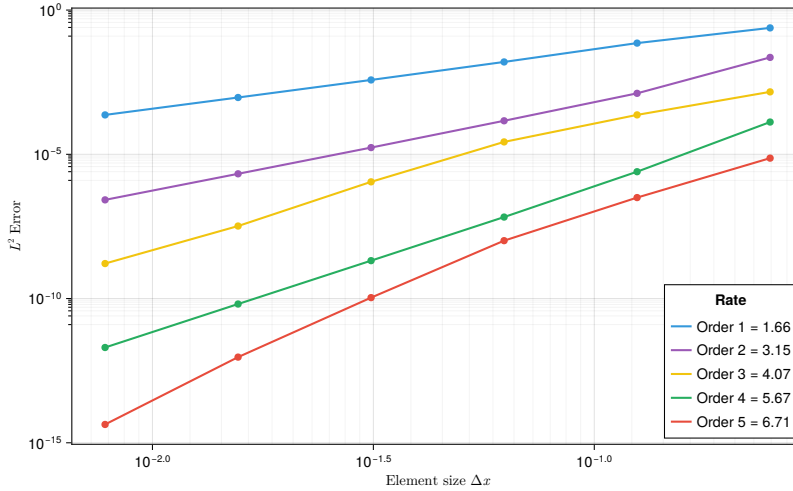


Figure 10.3: L^2 -error convergence of the classical SBP scheme for 1D linear advection problem.

10.2 Quantum Realisation of the Semi-Discrete System

In order to construct the quantum analogue of the semi-discrete system, we first map the numerical solution $u_h(t)$ obtained from the discretised PDE to

a quantum state and identify how the QSVT procedure acts on it. We use the notation $\mathcal{U}(T)$ for the exact PDE solution, $u_h(T)$ for its semi-discrete approximation, and $u_h^q(T)$ for the solution reconstructed from the quantum algorithm. For the semi-discrete system in (10.4), the evolution operator

$$U_T := \exp\left(i\frac{A}{i}T\right)$$

is unitary since A is skew-symmetric. Hence, the semi-discrete evolution preserves the discrete energy:²

$$\|u_h(T)\|_2 = \|u_h(0)\|_2.$$

² For any unitary U_T , we have $\|U_T u\|_2^2 = u^* U_T^* U_T u = \|u\|_2^2$.

State Preparation and Time Evolution

To initialise the quantum algorithm, the classical vector $u_h(0)$ is normalised and encoded as a quantum state,

$$|u_0\rangle := \frac{u_h(0)}{\|u_h(0)\|_2}.$$

This state is prepared using a **state preparation** procedure that maps the all-zero state to $|u_0\rangle$,³ $\text{PREP}|0^n\rangle = |u_0\rangle$. The QSVT circuit V_T implements an $(1, \cdot, \varepsilon_q)$ block-encoding of U_T . Let

³ $n = \log_2(N)$ qubits are required to represent an N -dimensional vector.

$$|\psi_T\rangle := V_T |u_0\rangle, \quad |\varphi_T\rangle := U_T |u_0\rangle = \frac{u_h(T)}{\|u_h(0)\|_2}.$$

By the operator-norm bound for QSVT approximations,

$$\| |\psi_T\rangle - |\varphi_T\rangle \|_2 = \|(V_T - U_T) |u_0\rangle\|_2 \leq \|V_T - U_T\| \| |u_0\rangle \|_2 \leq \varepsilon_q. \quad (10.18)$$

Reconstruction and Success Probability

Since U_T is unitary, the QSVT circuit involves no amplitude attenuation from the block-encoding normalisation: postselection succeeds with probability $1 - \mathcal{O}(\varepsilon_q^2)$, and no amplitude amplification is required. To recover the classical solution, we rescale by the known initial norm,

$$u_h^q(T) := \|u_h(0)\|_2 |\psi_T\rangle.$$

The reconstruction error is given by

$$\|u_h^q(T) - u_h(T)\|_2 \leq \|u_h(0)\|_2 \varepsilon_q.$$

Thus, QSVT contributes an **algorithmic** error proportional to the desired tolerance ε_q and the initial data norm.

Overall Error Bound

We want to quantify the error between $\mathcal{U}(T)$, $u_h(T)$, and $u_h^q(T)$. Using the triangle inequality, we have

$$\|\mathcal{U}(T) - u_h^q(T)\|_{L^2} \leq \|\mathcal{U}(T) - u_h(T)\|_{L^2} + \|u_h(T) - u_h^q(T)\|_{L^2}.$$

[HW07, p. 88] shows that for smooth periodic solution and initial data, the error from the modal Legendre discretisation for the advection equation follows

$$\|\mathcal{U}(T) - u_h(T)\|_{L^2} \leq C \Delta x^{p+1} T, \quad \Delta x = \frac{L}{n_{\text{elem}}}, \quad (10.19)$$

with $C > 0$ independent of p and n_{elem} . Thus, the overall error bound is

$$\|\mathcal{U}(T) - u_h^q(T)\|_{L^2} \leq \underbrace{C \left(\frac{L}{n_{\text{elem}}} \right)^{p+1} T}_{\varepsilon_d} + \underbrace{\|u_h(0)\| \varepsilon_q}_{\text{QSVT error}}. \quad (10.20)$$

Numerical Verification

Figure 10.4 shows convergence behaviour of the quantum algorithm, plotting the overall error $\|\mathcal{U}(T) - u_h^q(T)\|_{L^2}$ against the mesh spacing Δx for polynomial degrees $p = 1$ and 3. The QSVT tolerance is fixed at $\varepsilon_q = 10^{-13}$ to ensure that the total error is dominated by the classical discretisation term ε_d , and to remain within the numerical precision limits of the PYQSP package used to compute the QSVT phase factors. The observed rates (computed by least-squares fit on the last three data points) are consistent with the expected high-order convergence from the classical discretisation.

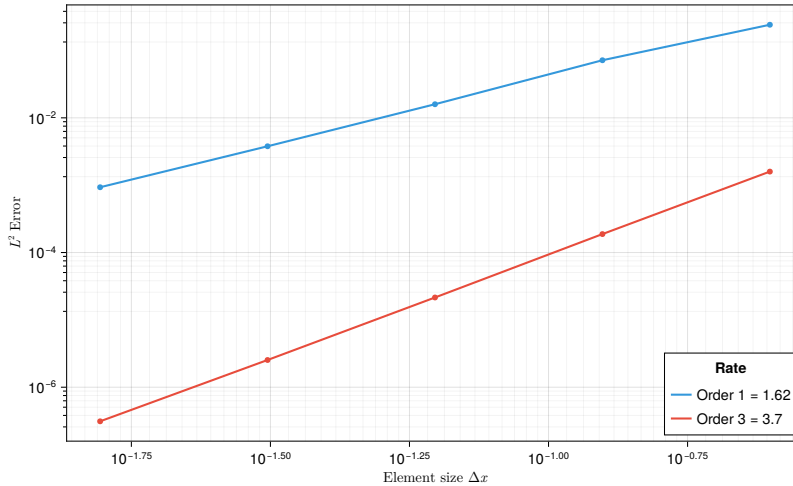


Figure 10.4: L^2 -error convergence of the quantum algorithm for polynomial degrees $p = 1$ and 3 with fixed QSVT tolerance $\varepsilon_q = 10^{-13}$.

10.3 Quantum Circuit Complexity

We now compute the quantum circuit complexity for implementing the time evolution operator $\exp(At)$ using QSVT. The overall complexity depends on the number of queries to a block-encoding of $\frac{A}{t}$, and the cost of each query. From [Theorem 9.2](#), the number of queries to block-encoding of A depends on the normalisation factor α and the evolution time t . For brevity, we use A to denote A for

Normalisation Factor

Recall from [Equation \(7.15\)](#) that the normalisation factor of the block-encoding is

$$\alpha = \sum_{l=0}^{s_0-1} |A_l|,$$

where the entries of A are partitioned into s_0 data items satisfying the offset condition of [Assumption 7.1](#). In particular, each data item corresponds to a set of nonzero entries whose row indices are related to the column index j by an arithmetic map $r_l(j) = j + k_l \pmod{N}$. Since the global operator is $A = -\frac{2}{\Delta x} Q_g$ with $\Delta x = \frac{L}{n_{\text{elem}}}$ where $L = b - a$, the overall normalisation factor scales as

$$\alpha_A = \frac{2}{\Delta x} \alpha_{Q_g} = \frac{2}{L} n_{\text{elem}} \alpha_{Q_g},$$

where α_{Q_g} denotes the contribution obtained from the block-encoding of Q_g alone.

The operator Q_g has a block-tridiagonal form with diagonal blocks S and off-diagonal coupling blocks given by $\frac{1}{2} R_R^\top R_L$ and $-\frac{1}{2} R_L^\top R_R$. Since these three local blocks repeat across every **block column** with a fixed offset, it suffices to consider a single “middle” block column when determining the set of distinct data items. The offset structure ensures that all repetitions are automatically accounted for. [Figure 10.5](#) shows the structure of one representative block column for polynomial degree $p = 3$.

Counting data items. Let $m := p+1$ denote the dimension of each block; throughout this section we adopt m as shorthand for $p+1$ to simplify notation.

- The local stiffness block S has a checkerboard sparsity: an entry is nonzero only when $i + j$ is odd. Thus, for even m (the case relevant here, since we assume m is a power of two),

$$\#\{(i, j) : S_{ij} \neq 0\} = \frac{m^2}{2}.$$

- Each of the two coupling blocks is dense, contributing m^2 nonzeros apiece.

$j \backslash i$	8	9	10	11
4	$\frac{1}{4}$	$-\frac{\sqrt{3}}{4}$	$\frac{\sqrt{5}}{4}$	$-\frac{\sqrt{7}}{4}$
5	$\frac{\sqrt{3}}{4}$	$-\frac{3}{4}$	$\frac{\sqrt{15}}{4}$	$-\frac{\sqrt{21}}{4}$
6	$\frac{\sqrt{5}}{4}$	$-\frac{\sqrt{15}}{4}$	$\frac{5}{4}$	$-\frac{\sqrt{35}}{4}$
7	$\frac{\sqrt{7}}{4}$	$-\frac{\sqrt{21}}{4}$	$\frac{\sqrt{35}}{4}$	$-\frac{7}{4}$
8		$\frac{\sqrt{3}}{2}$		$\frac{\sqrt{7}}{2}$
9	$-\frac{\sqrt{3}}{2}$		$\frac{\sqrt{15}}{2}$	
10		$-\frac{\sqrt{15}}{2}$		$\frac{\sqrt{35}}{2}$
11	$-\frac{\sqrt{7}}{2}$		$-\frac{\sqrt{35}}{2}$	
12	$-\frac{1}{4}$	$-\frac{\sqrt{3}}{4}$	$-\frac{\sqrt{5}}{4}$	$-\frac{\sqrt{7}}{4}$
13	$\frac{\sqrt{3}}{4}$	$\frac{3}{4}$	$\frac{\sqrt{15}}{4}$	$\frac{\sqrt{21}}{4}$
14	$-\frac{\sqrt{5}}{4}$	$-\frac{\sqrt{15}}{4}$	$-\frac{5}{4}$	$-\frac{\sqrt{35}}{4}$
15	$\frac{\sqrt{7}}{4}$	$\frac{\sqrt{21}}{4}$	$\frac{\sqrt{35}}{4}$	$\frac{7}{4}$

Figure 10.5: Representative block column of A for $p = 3$. Each colored square is a 4×4 local block: top = $\frac{1}{2} R_R^\top R_L$, middle = S , bottom = $-\frac{1}{2} R_L^\top R_R$. Gray cells in S indicate zero entries.

Hence one representative block column contains

$$s_0 = \frac{m^2}{2} + 2m^2 = \frac{5}{2}m^2$$

data items that satisfy the offset condition. Thus, the normalisation factor α is simply the sum of the absolute values of all entries in one block column.

The absolute values of the nonzero entries are

$$|S_{ij}| = \frac{1}{2} \sqrt{(2i+1)(2j+1)} \quad (i+j \text{ odd}),$$

$$\left| \frac{1}{2} (R_R^\top R_L)_{ij} \right| = \left| \frac{1}{2} (R_L^\top R_R)_{ij} \right| = \frac{1}{4} \sqrt{(2i+1)(2j+1)}.$$

Thus, the normalisation factor is

$$\alpha_{Q_g} = \frac{1}{2} \sum_{i,j=0}^{m-1} \sqrt{(2i+1)(2j+1)} + \frac{1}{2} \sum_{\substack{i,j=0 \\ i+j \text{ odd}}}^{m-1} \sqrt{(2i+1)(2j+1)}.$$

Lemma 10.1. *The normalisation factor α_A satisfies $\alpha_A \in \Theta(n_{\text{elem}} m^3 / L)$.*

Proof.

Introduce

$$S_1 = \sum_{i=0}^{m-1} \sqrt{2i+1}, \quad S_2 = \sum_{i=0}^{m-1} (-1)^i \sqrt{2i+1}.$$

Then

$$\begin{aligned} \sum_{i,j=0}^{m-1} \sqrt{(2i+1)(2j+1)} &= S_1^2 \\ &\leq \left(\int_0^m \sqrt{2x+1} dx \right)^2 \\ &= \left(\frac{(2m+1)^{3/2} - 1}{3} \right)^2 \in \Theta(m^3). \end{aligned}$$

Similarly,

$$\sum_{\substack{0 \leq i,j \leq m-1 \\ i+j \text{ odd}}} \sqrt{(2i+1)(2j+1)} = \frac{1}{2} S_1^2 - S_2^2.$$

The sum S_2 can be factorised as

$$\begin{aligned} S_2 &= \sum_{i=0}^{m-1} (-1)^i \sqrt{2i+1} \\ &= \sum_{i=0}^{m-1} \left(\sqrt{4i+1} - \sqrt{4i+3} \right) \\ &= - \sum_{i=0}^{m-1} \frac{2}{\sqrt{4i+3} + \sqrt{4i+1}}. \end{aligned}$$

Bounding the summand on both sides gives:

$$-\frac{1}{\sqrt{4i+1}} \leq -\frac{2}{\sqrt{4i+3} + \sqrt{4i+1}} \leq -\frac{1}{\sqrt{4i+3}}.$$

Approximating the sums by integrals yields

$$-\int_0^{m-1} \frac{1}{\sqrt{4x+1}} dx \leq S_2 \leq -\int_0^{m-1} \frac{1}{\sqrt{4x+3}} dx, \implies S_2 \in \Theta(\sqrt{m}).$$

Combining these estimates, we find that $\alpha_{Q_g} \in \Theta(m^3)$. Hence, the overall normalisation factor α_A scales as

$\alpha_A \in \Theta(n_{\text{elem}} m^3 / L).$

■

Block-Encoding Cost

Having determined that each representative block column contributes $s_0 = \frac{5}{2}m^2$ distinct data items, we now quantify the qubit requirements of the block-encoding in terms of the polynomial degree p and the number of elements n_{elem} . The data index $l \in \{0, \dots, s_0 - 1\}$ labels each distinct nonzero entry corresponding to a shift map $r_l(j) = j + k_l \pmod{N}$. To address all possible l , the ancilla register must be large enough to encode s_0 items. Let

$$n_a := \lceil \log s_0 \rceil = \left\lceil \log \left(\frac{5}{2}m^2 \right) \right\rceil$$

denote the number of ancillary qubits required to index the data items. The data register indexes the rows and columns of the global operator $A \in \mathbb{C}^{N \times N}$, where $N = n_{\text{elem}} m$. Accordingly, the data register requires

$$n_d := \lceil \log N \rceil = \lceil \log(m n_{\text{elem}}) \rceil$$

qubits to address all matrix indices. Unless otherwise stated, all logarithms are base 2. We now quantify the cost of implementing the block-encoding oracles.

PREP/UNPREP oracle. Recall that the state-preparation oracles PREP and UNPREP require $\mathcal{O}(s)$ two-qubit gates, where $s = 2^{n_a}$. We do **not** consider any optimised state-preparation methods for PREP/UNPREP in this thesis. Since s is the next power of two above s_0 , we have

$$s_0 \leq s < 2s_0 \implies \mathcal{O}(s) = \mathcal{O}(m^2). \quad (10.21)$$

Accounting for both PREP and UNPREP, the combined data-loader cost is therefore

$$C_{\text{PREP}} = \mathcal{O}(m^2). \quad (10.22)$$

O_r oracle. Recall that O_r implements the arithmetic maps

$$r_l(j) = j + k_l \pmod{N}, \quad l = 0, \dots, s_0 - 1,$$

where each offset k_l corresponds to one of the distinct data items defined previously. Since we have $s_0 = \frac{5}{2}m^2$ data items, we must implement s_0 controlled addition circuits, each performing a shift by a constant offset k_l .

Note 10. In practice, our Gray-code-based construction yields a substantially lower empirical cost, scaling as $\mathcal{O}(m \log m)$ rather than $\mathcal{O}(m^2)$. This improvement depends on implementation details such as the chosen ESOP minimisation strategy, ordering heuristics, and the internal structure of the Boolean synthesis solver. For our complexity analysis, we adopt the conservative $\mathcal{O}(m^2)$ bound to ensure consistency across different implementations.

Each offset gate for n -qubit register with c controls uses $\mathcal{O}(c + n \log(n))$ CX-gates if one dirty ancilla qubit is available. We can use the target qubit of O_{rg} as a dirty ancilla for O_r . In our setting,

$$n = n_d = \left\lceil \log(m n_{\text{elem}}) \right\rceil, \quad c = n_a = \left\lceil \log\left(\frac{5}{2}m^2\right) \right\rceil.$$

Summing over all s_0 shifts, the total cost of the O_r oracle is $s_0(n_a + n_d \log n_d)$, i.e.,

$$C_{O_r} = \mathcal{O}\left(m^2(\log m + (\log n_{\text{elem}} + \log m) \log(\log n_{\text{elem}} + \log m))\right). \quad (10.23)$$

O_{rg} oracle. The O_{rg} oracle has one MCX gate per data item, so there are a total of s_0 such gates. Each data item requires only a single MCX because the item follows a structured, repeating activation pattern that is simplified by ESOP minimisation. Each MCX_n -gate requires $12n - 18$ CX-gates if one dirty ancilla is available.

Not counting further optimisations, the n_a ancilla qubits (indexing the data items) act as controls exactly as in O_r . On the data register, due to the ESOP simplification the predicate evaluates only the residue class $j \bmod m$, rather than the full index j . Hence the data-side controls reduce to

$$n_m := \left\lceil \log m \right\rceil,$$

instead of using all n_d bits. Therefore, each MCX is controlled on

$$n = n_a + n_m = \left\lceil \log\left(\frac{5}{2}m^2\right) \right\rceil + \left\lceil \log m \right\rceil$$

qubits. With one MCX per data item, the total two-qubit gate cost is

$$\begin{aligned} s_0(12n - 18) &= \frac{5}{2}m^2 \left[12 \left(\left\lceil \log\left(\frac{5}{2}m^2\right) \right\rceil + \left\lceil \log m \right\rceil \right) - 18 \right] \\ &= \mathcal{O}\left(m^2 \log(m)\right). \end{aligned} \quad (10.24)$$

Combining (10.21), (10.23), and (10.24), the overall CX cost for **one application** of the block-encoding is

$$C_{\text{BE}} = \mathcal{O} \left(m^2 (\log m + (\log n_{\text{elem}} + \log m) \log(\log n_{\text{elem}} + \log m)) \right). \quad (10.25)$$

Thus, the dominant cost arises from the O_r oracle. Empirically, our Gray-code optimisation often reduces the **effective** number of data items to $s_0^{\text{eff}} = \mathcal{O}(m \log m)$, which correspondingly tightens the cost of both O_r and O_{rg} . Figure 10.6 plots the measured CX counts versus the global matrix size $N = n_{\text{elem}}(p+1)$ where, **for each curve**, the polynomial order p is fixed (with $p+1 = 2^k$) and the number of elements n_{elem} is doubled along the x -axis. Each curve shows the expected (poly)logarithmic growth in N at fixed order, while the vertical separation across curves reflects the m^2 prefactor in (10.25).

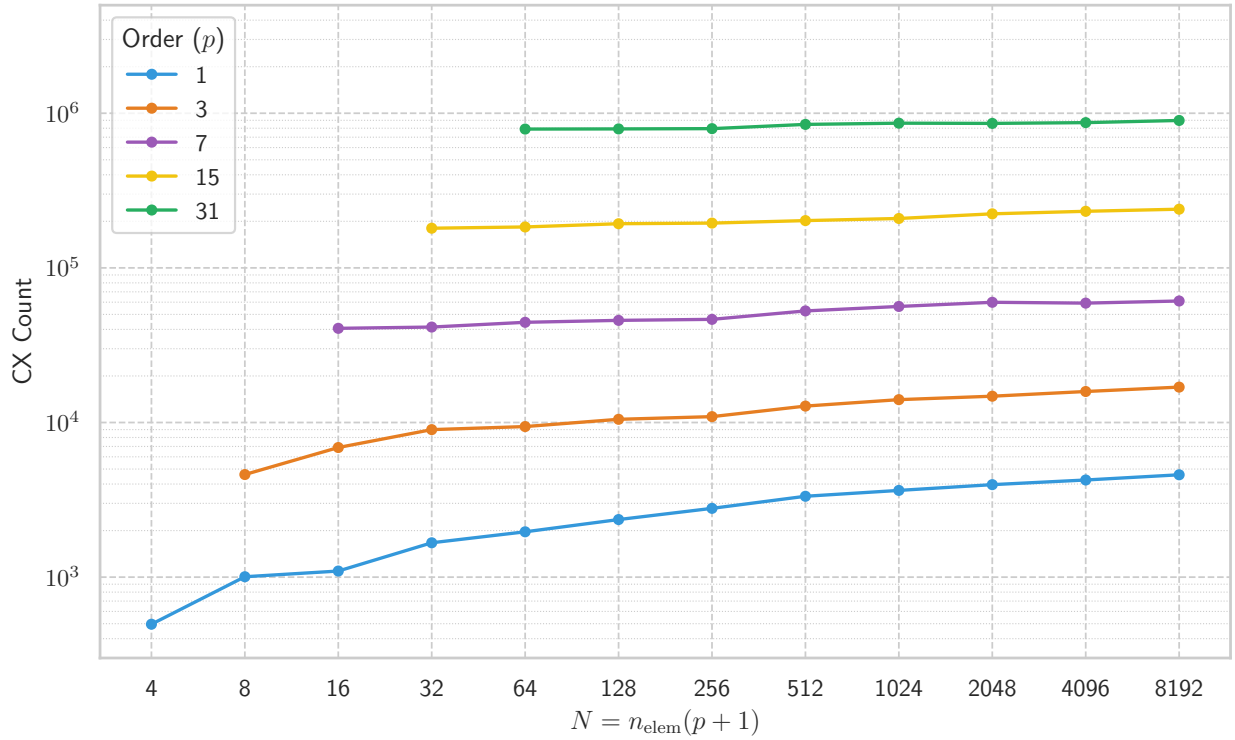


Figure 10.6: Measured CX-gate counts versus global matrix size $N = n_{\text{elem}}(p+1)$.

Overall QSVT Complexity

We can now summarise the quantum circuit complexity for implementing the time evolution operator $\exp(AT)$ using QSVT for $T > 0$. From Theorem 9.2,

the total number of queries to the block-encoding of A is

$$Q = \Theta \left(\alpha T + \frac{\log(1/\varepsilon_q)}{\log(e + \frac{\log(1/\varepsilon_q)}{\alpha T})} \right) \quad (10.26)$$

and additional two-qubit gate overhead of $\mathcal{O}(\alpha T a)$ where a is the number of ancilla qubits used in the block-encoding. In our case, $a = \lceil \log s_0 \rceil = \Theta(\log m)$.

In the asymptotic complexity (10.26), the first term dominates whenever $\alpha T \gg \log(1/\varepsilon_q)$. This regime is precisely the one of practical interest for large-scale scientific and engineering problems. The block-encoding normalisation factor scales as $\alpha \in \Theta(n_{\text{elem}} m^3)$, so even for moderate time horizons $T = \mathcal{O}(1)$, the product αT rapidly exceeds $\log(1/\varepsilon_q)$ for practically relevant accuracies (e.g., $\varepsilon_q \sim 10^{-6}$ – 10^{-12}). Typical applications such as global atmospheric dynamics or high-Reynolds-number turbulence often involve $n_{\text{elem}} \sim 10^6$ – 10^9 , yielding $\alpha T \sim 10^6$ – 10^9 while $\log(1/\varepsilon_q) \lesssim 20$. Hence the precision-dependent term contributes negligibly in these compute-dominated regimes. Consequently, for the purpose of determining the optimal discretisation parameters p and n_{elem} , we may work under the dominant-term approximation

$$\Theta \left(\alpha T + \frac{\log(1/\varepsilon_q)}{\log(e + \frac{\log(1/\varepsilon_q)}{\alpha T})} \right) \approx \Theta(\alpha T),$$

which simplifies the optimisation analysis. After obtaining the optimal values p^* and n_{elem}^* , we substitute them back into the full QSVT complexity expression to recover the complete scaling with ε_q and T . As shown in Figure 10.7, as αT increases, the contours flatten, indicating that for sufficiently large problems the required polynomial degree becomes nearly independent of the target tolerance. This behaviour quantitatively supports the validity of this approximation in the compute-dominated regime.

Choosing p and n_{elem}

Having derived the quantum circuit complexity, we now select the polynomial degree p and the number of elements n_{elem} to achieve a desired accuracy ε . Since we are working under the dominant-term approximation, we fix the QSVT error tolerance $\varepsilon_q \leq 0.1 \varepsilon / \|u_h(0)\|_2$ and the discretisation tolerance $\varepsilon_d \leq 0.9 \varepsilon$ so that the overall error bound (10.20) is satisfied. A similar constant-fraction split was adopted in [PJKW25, Corollary 11] when optimising multiple query-dependent costs under a single global tolerance.

Based on (10.19), it suffices to choose (p, n_{elem}) such that

$$C \left(\frac{L}{n_{\text{elem}}} \right)^{p+1} T \leq \varepsilon_d \iff n_{\text{elem}} \geq L \left(\frac{CT}{\varepsilon_d} \right)^{1/(p+1)}.$$

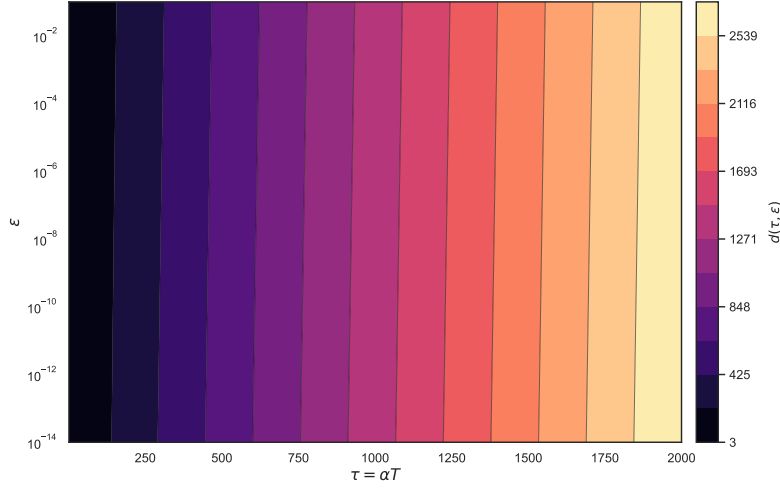


Figure 10.7: Isocontours of the required QSVT polynomial degree $d(\tau, \varepsilon)$.

For brevity set $R := CT/\varepsilon_d > 1$. Using the lower bound for n_{elem} ,

$$n_{\text{elem}} \geq L R^{1/m}, \quad \log n_{\text{elem}} = \log L + \frac{\log R}{m}, \quad m := p + 1.$$

Under the dominant-term approximation, the overall CX-gate complexity we seek to minimise is

$$C_{\text{QSVT}}(p, n_{\text{elem}}, T, \varepsilon_d) = Q \cdot C_{\text{BE}} + \mathcal{O}(\alpha T a).$$

where $Q = \mathcal{O}(\alpha T)$. Substituting the expressions for α and C_{BE} , we obtain

$$C_{\text{QSVT}}(p, n_{\text{elem}}, T, \varepsilon_d) = \mathcal{O}\left(n_{\text{elem}}/L T m^5 [\log m + (\log n_{\text{elem}} + \log m) \log(\log n_{\text{elem}} + \log m)]\right). \quad (10.27)$$

Substituting $n_{\text{elem}} = L R^{1/m}$ into (10.27) and collecting slowly varying poly-logarithmic terms into

$$\Lambda(m, L, R) := \log m + \left(\log L + \log m + \frac{\log R}{m}\right) \log\left(\log L + \log m + \frac{\log R}{m}\right),$$

the cost simplifies to

$$\mathcal{O}\left(T m^5 R^{1/m} \Lambda(m, L, R)\right). \quad (10.28)$$

We now seek to choose $m > 1$ to minimise the cost expression $f(m) := m^5 R^{1/m} \Lambda(m, L, R)$. We instead minimise its logarithm,

$$g(m) = \log f(m) = 5 \log m + \frac{\log R}{m} + \log \Lambda(m, L, R).$$

Note that since $\Lambda(m, L, R) = \mathcal{O}(\log m \log(\log m))$, its logarithm grows as doubly-logarithmically in m . Thus, the dominant behaviour is captured by the first two factors, giving $g(m) := \log f(m) \approx 5 \log m + \frac{\log R}{m}$. Stationarity $g'(m) = 0$ yields

$$\frac{5}{m} - \frac{\log R}{m^2} = 0 \implies m^* = \frac{\log R}{5}, \quad p^* = \frac{\log R}{5} - 1.$$

With $R = CT/\varepsilon_d$ we obtain

$$m^* = \frac{1}{5} \log\left(\frac{CT}{\varepsilon_d}\right), \quad p^* = \frac{1}{5} \log\left(\frac{CT}{\varepsilon_d}\right) - 1, \quad n_{\text{elem}}^* \approx L R^{1/m^*} = L e^5.$$

Resulting complexity at the optimum. At $m^* = \frac{1}{5} \log(CT/\varepsilon_d)$, we have $R^{1/m^*} = e^5 = \Theta(1)$ and hence $n_{\text{elem}}^* = L e^5$. The per-query block-encoding cost becomes

$$C_{\text{BE}}(m^*) = \mathcal{O}\left((m^*)^2 \Lambda(m^*, L, R)\right), \quad m^* = \frac{1}{5} \log\left(\frac{CT}{\varepsilon_d}\right).$$

Since $\alpha^* T = \Theta(T(m^*)^3)$, the total two-qubit complexity is the sum of the dominant αT -term and the precision-driven term:

$$\begin{aligned} C_{\text{QSVT}}(m^*) &= \underbrace{\mathcal{O}\left(T (m^*)^5 \Lambda(m^*, L, R)\right)}_{\text{dominant } \alpha T \text{ term}} \\ &\quad + \underbrace{\mathcal{O}\left(\frac{\log(1/\varepsilon_q)}{\log\left(e + \frac{\log(1/\varepsilon_q)}{\Theta(T (m^*)^3)}\right)} (m^*)^2 \Lambda(m^*, L, R)\right)}_{\text{precision term}}. \end{aligned} \tag{10.29}$$

Absorbing constant multiplicative factors into big- $\mathcal{O}$, we may take $R = T/\varepsilon$ and equivalently write

$$\begin{aligned} C_{\text{QSVT}}(m^*) &= \mathcal{O}\left(T \log^5\left(\frac{T}{\varepsilon}\right) \Lambda\left(\frac{1}{5} \log\left(\frac{T}{\varepsilon}\right), L, \frac{T}{\varepsilon}\right)\right) \\ &\quad + \mathcal{O}\left(\frac{\log(1/\varepsilon)}{\log\left(e + \frac{\log(1/\varepsilon)}{T \log^3(T/\varepsilon)}\right)} \log^2\left(\frac{T}{\varepsilon}\right) \Lambda\left(\frac{1}{5} \log\left(\frac{T}{\varepsilon}\right), L, \frac{T}{\varepsilon}\right)\right). \end{aligned} \tag{10.30}$$

Recall that $\Lambda(m, L, R)$ grows only polylogarithmically with m , L , and $R = \frac{CT}{\varepsilon}$. Thus, the total two-qubit gate count grows **linearly** with the simulation time T and **polylogarithmically** with $1/\varepsilon$.

10.4 Limitations and Future Extensions

Limitations

The proposed framework makes several simplifying assumptions that restrict its applicability to broader classes of problems. First, we assume access to an

oracle for state preparation corresponding to the initial condition. Preparing arbitrary initial states remains an open challenge in quantum algorithms, and likewise, this work does not address the issue of state readout. Second, the approach relies on the skew-symmetry of the semi-discrete operator, which holds for certain classes of periodic boundary conditions. For such operators, QSVT realises eigenvalue transformations, but this assumption limits the framework to a subset of PDEs and boundary conditions. Finally, the current implementation assumes a uniform computational mesh. Many practical PDE problems often require non-uniform or curvilinear meshes to capture local solution features. In such cases, the discrete norm matrix A is no longer proportional to the identity, necessitating distinct data loads for each element and thereby increasing the block-encoding cost.

Future Extensions

Extensions to Non-Skew-Symmetric Operators A natural next step is to extend the framework beyond skew-symmetric operators. Several recent works propose alternatives for simulating non-Hermitian dynamics on quantum computers [FLT23, ACLY24, HJLZ24]. One promising approach is the Linear Combination of Hamiltonian Simulation (LCHS) algorithm [ACLY24], which decomposes a general operator as

$$A = L + iH, \quad L = \frac{A + A^\dagger}{2}, \quad H = \frac{A - A^\dagger}{2i},$$

and expresses the matrix exponential via a contour integral,

$$\exp(-At) = \int_{\mathbb{R}} \frac{f(k)}{1 - ik} \exp(-it(kL + H)) dk,$$

where $f(k)$ is chosen from a family of kernel functions. The integral is then approximated using a quadrature rule, and each term in the quadrature rule involves simulating operator $kL + H$ using Hamiltonian-simulation techniques such as QSVT. It was recently proven in [LS25] that LCHS achieves asymptotically optimal scaling for matrix exponentiation; within a constant factor of three of pure QSVT further motivating QSVT-based algorithms for solving PDEs numerically.

Another line of extension involves quantum linear system algorithms (QLSAs), which can be used to implement implicit time-stepping schemes. While recent comparisons suggest that LCHS can outperform QLSA by $10^2 - 10^3$ in two-qubit gate counts for time-dependent problems [PJKW25], QLSAs compute the full intermediate state at each time step, which can be advantageous for certain applications. Both approaches ultimately require (i) a block-encoding of the discretised operator, (ii) a QSVT-based polynomial approximation of the desired function, and (iii) efficient state preparation and readout-making the block-encoding scheme developed in this work directly reusable in both contexts.

Curvilinear Meshes Extending the framework to curvilinear meshes represents another key direction. In the current formulation, the discretisation is performed classically before block-encoding the resulting operator. For non-uniform or curved meshes, the elementwise discretisation stencils vary across the domain, requiring distinct data items for each element and increasing both the number of data-loading operations and the normalisation factor. A promising strategy is to separate the encoding of mesh geometry and discretisation coefficients. Curvilinear meshes are often represented as polynomial mappings from reference element coordinates to physical space; these mappings and their derivatives could themselves be approximated using QSVT, allowing the loading of only the minimal geometric data (e.g., nodal coordinates in reference space).

State Preparation and Readout Finally, recent work [MGB22, OS25] has explored using QSVT for state preparation, where the statevector is given as function values on a discrete grid. By loading grid coordinates along a diagonal matrix and applying QSVT to transform the diagonal entries into polynomial approximations of the target function, one can prepare the desired initial state. Beyond state preparation, recent studies, such as [LWJ⁺25], have shown that in certain PDE contexts, linear functionals of the solution vector can be estimated exponentially faster on a quantum computer. Adapting this idea to the present framework would enable practical readout strategies focused on physically relevant observables.

Conclusions and Recommendations

11.1 Conclusions

This thesis establishes a significant step toward a unified framework for translating classical high-order discretisations with provable stability into efficient quantum algorithms for linear PDEs. By starting from well-posed PDEs and constructing stable semi-discrete formulations, we have shown that these stability principles can be preserved at the semi-discrete level within the quantum circuit model through block-encoding and time evolution via QSVT. The result is a quantum algorithmic formulation that retains the discrete energy estimates, conservation properties, and operator structures of its classical analogue while admitting an efficient realisation in terms of quantum gates.

[Chapter 6](#) provides the quantum foundations required for implementing these discretisations efficiently on quantum hardware via block-encoding. It develops the arithmetic and control primitives underlying block-encoding, which serve as the building blocks later assembled in [Chapter 7](#) into an algorithm that embeds structured matrices as sub-blocks of larger unitaries in an **efficient** and **automated** manner. Together, these chapters form the bridge between classical discretisation theory and quantum circuit design, providing the foundation for automated synthesis of block-encodings.

[Chapter 8](#) and [Chapter 9](#) develop the polynomial-transformation machinery that forms the mathematical basis of the main quantum algorithm. QSP is introduced as a single-qubit protocol for implementing polynomial functions through alternating phase sequences, and QSVT extends this principle to arbitrary block-encoded matrices.

[Chapter 10](#) brings everything together by realising the SBP–SAT semi-discrete formulation of the linear advection equation as a complete quantum algorithm. It derives explicit relationships between classical discretisation and quantum approximation errors and establishes complexity bounds on block-encoding normalisation, QSVT query complexity, and the optimal trade-off between polynomial degree and mesh resolution. This final chapter highlights the central contribution of the thesis: demonstrating, through a

concrete example, that high-order energy-stable discretisations can be implemented as quantum circuits while maintaining their theoretical guarantees. Our construction provides a blueprint for automating the synthesis of discretisation-faithful quantum algorithms for a broad class of linear PDEs and advanced numerical methods.

Contributions

The main contributions of this thesis can be summarised as follows:

- **Ancilla-efficient automated block-encodings.** Design of an automated circuit-synthesis framework for block-encoding sparse, structured matrices with minimal ancilla requirements. The approach avoids the overhead of large ancillary registers typically required by general-purpose schemes, making it more suitable for near-term and early fault-tolerant quantum computers.
- **High-order stable discretisations on quantum hardware.** Most existing quantum PDE approaches rely on first-order finite-difference discretisations. This thesis extends the paradigm to high-order SBP element-based schemes, showing that classical energy-stable discretisations retain their semi-discrete stability and conservation properties when realised as quantum circuits.
- **Cross-disciplinary synthesis and expository treatment.** The thesis brings together tools from numerical analysis, quantum algorithms, quantum circuit synthesis, Boolean logic synthesis, and software engineering within a single, coherent framework. Equal emphasis is placed on exposition and reproducibility: each component is developed from first principles, with explicit constructions, illustrative examples, and supporting software utilities. This combination of methodological depth and pedagogical clarity provides a self-contained resource for researchers seeking to apply quantum computing techniques to state-of-the-art numerical simulation problems.
- **Explicit and verified quantum circuits.** Construction of explicit quantum circuits implementing time evolution via block-encoding and QSVT, modulo state-preparation oracles. All circuit components were verified through classical simulation, confirming the correctness of the block-encoded operators, QSVT transformations, and overall algorithmic behaviour. These simulations provide empirical support for the theoretical analysis.
- **SBPLite.jl.** Development of a lightweight Julia package for high-order Summation-by-Parts discretisations. The package enables rapid prototyping of classical discretisations, automatic generation of operator matrices, and numerical verification of energy stability on both uniform and

curvilinear meshes. It serves as the classical foundation for the quantum implementations presented later in the thesis.

11.2 Recommendations

Some recommendations for future work are listed below.

- **Design SBP operators optimised for block-encoding.** Exploit the intrinsic flexibility in the definition of SBP operators to construct new variants whose algebraic structure minimises block-encoding cost. This reverses the usual perspective: instead of adapting block-encoding to a given discretisation, design discretisations whose coefficient patterns and sparsity are inherently compatible with efficient quantum realisation.
- **Increase ancilla budget.** Explore block-encodings that trade circuit depth for additional ancillary qubits, enabling efficient reversible arithmetic such as long division and modulus checks [WLC24].
- **Support non-power-of-two dimensions.** Implement zero-padding and efficient permutation circuits [DLX25] to handle matrices whose sizes are not exact powers of two, reducing structural constraints on discretised operators.
- **Adopt dual-oracle block-encodings.** Extend the automated synthesis pipeline to support frameworks using both row and column oracles [SCC24], broadening the class of efficiently encodable matrices.
- **Generalise projector constructions.** Investigate the use of arbitrary control patterns and non-diagonal projectors within the block-encoding design to enable more flexible operator structures.
- **Extend beyond skew-symmetric operators.** Incorporate QSVT-based algorithms for non-Hermitian systems, including Linear Combination of Hamiltonian Simulation (LCHS) [ACLY24, LS25] and Quantum Linear System Algorithms (QLSAs) [PJKW25].
- **Integrate curvilinear geometries.** Develop techniques for encoding curvilinear meshes where geometric mappings are computed using QSVT, reducing per-element data loading.
- **Leverage QSVT for state preparation and readout.** Apply recent QSVT-based methods [MGB22, OS25, LWJ⁺25] for preparing initial conditions and extracting physically relevant observables efficiently.

Bibliography

- [ABO99] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate, 1999. (cit. on p. [48](#))
- [ACLY24] Dong An, Andrew M. Childs, Lin Lin, and Lexing Ying. Laplace transform based quantum eigenvalue transformation via linear combination of Hamiltonian simulation, November 2024. (cit. on pp. [140](#) and [145](#))
- [AGo4] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5), November 2004. (cit. on p. [48](#))
- [BBC⁺95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. (cit. on p. [48](#))
- [BCC⁺14] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse hamiltonians. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. ACM, May 2014. (cit. on p. [71](#))
- [BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE, October 2015. (cit. on p. [73](#))
- [BGB⁺18] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4), October 2018. (cit. on p. [91](#))
- [BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation, 2002. (cit. on p. [71](#))
- [BK22] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, May 2022. (cit. on p. [111](#))
- [Cal65] D.K Callebaut. Generalization of the cauchy-schwarz inequality. *Journal of Mathematical Analysis and Applications*, 12(3):491–494, 1965. (cit. on p. [88](#))
- [CDRFC19] Jesse Chan, David C. Del Rey Fernández, and Mark H. Carpenter. Efficient entropy stable gauss collocation methods. *SIAM Journal on Scientific Computing*, 41(5):A2938–A2966, 2019. (cit. on p. [37](#))
- [CGA94] Mark H. Carpenter, David Gottlieb, and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, 1994. (cit. on p. [20](#))

- [CHDRF⁺18] Jared Crean, Jason E. Hicken, David C. Del Rey Fernández, David W. Zingg, and Mark H. Carpenter. Entropy-stable summation-by-parts discretization of the Euler equations on general curved elements. *Journal of Computational Physics*, 356:410–438, March 2018. (cit. on pp. 1, 19, 27, 28, 29, 30, and 31)
- [CLO21] Andrew M. Childs, Jin-Peng Liu, and Aaron Ostrander. High-precision quantum algorithms for partial differential equations. *Quantum*, 5:574, November 2021. (cit. on p. 2)
- [Cra15] Gidney Craig. Constructing large controlled nots, 2015. (cit. on pp. 54 and 55)
- [Cra17] Gidney Craig. Proxy phasing and computed phasing, 2017. (cit. on pp. 66 and 69)
- [CS17] Tianheng Chen and Chi-Wang Shu. Entropy stable high order discontinuous galerkin methods with suitable quadrature rules for hyperbolic conservation laws. *Journal of Computational Physics*, 345:427–461, 2017. (cit. on p. 22)
- [CW00] Richard Cleve and John Watrous. Fast parallel circuits for the quantum fourier transform, 2000. (cit. on pp. 51 and 62)
- [DBZ14] David C. Del Rey Fernández, Pieter D. Boom, and David W. Zingg. A generalized framework for nodal first derivative summation-by-parts operators. *Journal of Computational Physics*, 266:214–239, June 2014. (cit. on p. 13)
- [DHZ14] David C. Del Rey Fernández, Jason E. Hicken, and David W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Computers & Fluids*, 95:171–196, 2014. (cit. on pp. 1, 13, 14, and 15)
- [DLX25] Dekuan Dong, Yingzhou Li, and Jungong Xue. Products between block-encodings, 2025. (cit. on pp. 100 and 145)
- [DMWL21] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. Efficient phase-factor evaluation in quantum signal processing. *Physical Review A*, 103(4), April 2021. (cit. on p. 109)
- [Dra00] Thomas G. Draper. Addition on a quantum computer, 2000. (cit. on p. 59)
- [DRFHZ18] David C. Del Rey Fernández, Jason E. Hicken, and David W. Zingg. Simultaneous Approximation Terms for Multi-dimensional Summation-by-Parts Operators. *Journal of Scientific Computing*, 75(1):83–110, April 2018. (cit. on pp. 20 and 21)
- [Eva10] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010. (cit. on p. 7)
- [Fef06] Charles L. Fefferman. Existence and smoothness of the navier-stokes equation. In *The Millennium Prize Problems*, pages 57–67. American Mathematical Society, 2006. (cit. on p. 7)
- [FLT23] Di Fang, Lin Lin, and Yu Tong. Time-marching based quantum solvers for time-dependent linear differential equations. *Quantum*, 7:955, March 2023. (cit. on p. 140)
- [FT82] Edward Fredkin and Tommaso Toffoli. Conservative logic. *International journal of theoretical physics*, 21(3-4):219–253, 1982. (cit. on pp. 47 and 48)
- [Gan60] F.R. Gantmacher. *The Theory of Matrices*. Number v. 1 in The Theory of Matrices. Chelsea Publishing Company, 1960. (cit. on p. 112)
- [Gid18] Craig Gidney. Factoring with $n+2$ clean qubits and $n-1$ dirty qubits, 2018. (cit. on pp. 58, 59, 60, 153, and 154)
- [Gid21] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. (cit. on p. 48)

- [GKNÖ23] Jan Glaubitz, Simon-Christian Klein, Jan Nordström, and Philipp Öffner. Multi-dimensional summation-by-parts operators for general function spaces: Theory and construction. *Journal of Computational Physics*, 491:112370, October 2023. (cit. on p. 19)
- [GKO13] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time-Dependent Problems and Difference Methods*, volume 123 of *Pure and applied mathematics*. John Wiley & Sons, Incorporated, Newark, second edition. edition, 2013. (cit. on pp. 7 and 8)
- [Got97] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997. (cit. on p. 48)
- [GR09] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. (cit. on p. 37)
- [Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996. (cit. on pp. 2 and 104)
- [GSLW19] 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*, pages 193–204. ACM, June 2019. (cit. on pp. 2, 3, 75, 76, 78, 79, 85, 105, 106, 107, 109, 113, 117, 118, 121, and 122)
- [Guso8] Bertil Gustafsson. *High Order Difference Methods for Time Dependent PDE*, volume 38 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 1. aufl. edition, 2008. (cit. on p. 9)
- [Hado3] J. Hadamard. *Lectures on Cauchy’s Problem in Linear Partial Differential Equations*. Dover phoenix editions. Dover Publications, 2003. (cit. on p. 7)
- [HDRFZ16] Jason E. Hicken, David C. Del Rey Fernández, and David W. Zingg. Multidimensional summation-by-parts operators: General theory and application to simplex elements. *SIAM Journal on Scientific Computing*, 38(4):A1935–A1958, 2016. (cit. on p. 19)
- [HDRFZ25] Jason E. Hicken, David C. Del Rey Fernández, and David W. Zingg. Summation-by-parts methods for the numerical solution of partial differential equations: Theory and practice. In preparation, 2025. (cit. on pp. 17 and 19)
- [HH15] John H. Hubbard and Barbara Burke Hubbard. *Vector calculus, linear algebra, and differential forms : a unified approach*. Matrix Editions, Ithaca, NY, 5th edition. edition, 2015. (cit. on p. 23)
- [HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), October 2009. (cit. on p. 2)
- [Higo8] Nicholas J. Higham. *Functions of Matrices*. Society for Industrial and Applied Mathematics, 2008. (cit. on p. 112)
- [HJ13] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, 2nd ed. edition, 2013. (cit. on p. 112)
- [HJLZ24] Junpeng Hu, Shi Jin, Nana Liu, and Lei Zhang. Quantum Circuits for partial differential equations via Schrödingerisation, March 2024. (cit. on p. 140)
- [HRS17] Thomas Häner, Martin Roetteler, and Krysta M. Svore. Factoring using $2n + 2$ qubits with toffoli based modular multiplication. *Quantum Info. Comput.*, 17(7–8):673–684, June 2017. (cit. on p. 62)
- [HW07] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer New York, 2007. (cit. on pp. 32 and 131)

- [ICK⁺16] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum Circuits for Isometries. *Physical Review A*, 93(3):032318, March 2016. (cit. on pp. 46, 47, 58, and 91)
- [JATK⁺24] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024. (cit. on p. 59)
- [Jul25] JuliaMath. Quadgk.jl: Adaptive numerical gauss-kronrod integration in julia. <https://github.com/JuliaMath/QuadGK.jl>, 2025. GitHub repository. (cit. on p. 126)
- [KG25] Tanuj Khattar and Craig Gidney. Rise of conditionally clean ancillae for efficient quantum circuit constructions. *Quantum*, 9:1752, May 2025. (cit. on pp. x, xi, 55, 57, 58, and 89)
- [KMM13] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by clifford and t gates, 2013. (cit. on p. 48)
- [Kot14] Robin Kothari. Efficient algorithms in quantum query complexity, August 2014. PhD thesis, University of Waterloo. (cit. on p. 73)
- [KS74] H.-O. KREISS and G. SCHERER. Finite element and finite difference methods for hyperbolic partial differential equations. In Carl de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 195–212. Academic Press, 1974. (cit. on pp. 1 and 19)
- [KSB⁺20] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(Volume 11, 2020):369–395, 2020. (cit. on p. 44)
- [LC17] Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, Jan 2017. (cit. on pp. 2 and 105)
- [LC19] Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, July 2019. (cit. on p. 105)
- [Llo22] Seth Lloyd. Quantum polar decomposition – IPAM at UCLA, January 2022. (cit. on p. 4)
- [LS25] Guang Hao Low and Rolando D. Somma. Optimal quantum simulation of linear non-unitary dynamics, August 2025. (cit. on pp. 140 and 145)
- [LW60] Peter Lax and Burton Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13(2):217–237, 1960. (cit. on p. 34)
- [LWJ⁺25] Margot Lockwood, Nathan Wiebe, Connah Johnson, Johannes Mülmenstädt, Jaehun Chun, Gregory Schenter, Margaret S. Cheung, and Xiangyu Li. Quantum Algorithm for Subcellular Multiscale Reaction-Diffusion Systems, September 2025. (cit. on pp. 141 and 145)
- [LYC16] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Methodology of resonant equiangular composite quantum gates. *Phys. Rev. X*, 6:041067, Dec 2016. (cit. on pp. 3 and 105)
- [MC13] M.M. Mano and M.D. Ciletti. *Digital Design: With a Introduction to the Verilog Hdl*. Always learning. Pearson Prentice Hall, 2013. (cit. on p. 44)
- [MGB22] Sam McArdle, András Gilyén, and Mario Berta. Quantum state preparation without coherent arithmetic, October 2022. (cit. on pp. 141 and 145)
- [MP01] Alan Mishchenko and Marek Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In *Proceedings of the Reed-Muller Workshop (RM’2001)*, August 2001. (cit. on p. 94)

- [MRTC21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. A Grand Unification of Quantum Algorithms. *PRX Quantum*, 2(4):040203, December 2021. (cit. on pp. 2 and 106)
- [MZ22] André L. Marchildon and David W. Zingg. Unisolvency for polynomial interpolation in simplices with symmetrical nodal distributions. *Journal of Scientific Computing*, 92(2):50, August 2022. (cit. on p. 21)
- [NCo0] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000. (cit. on pp. 2, 41, 46, and 48)
- [Nor15] Jan Nordström. Well posed problems and boundary conditions in computational fluid dynamics. 06 2015. (cit. on p. 8)
- [NZS24] Junhong Nie, Wei Zi, and Xiaoming Sun. Quantum circuit for multi-qubit toffoli gate with optimal resource, 2024. (cit. on p. 55)
- [OBB24] Giuseppe Orlando, Tommaso Benacchio, and Luca Bonaventura. Impact of curved elements for flows over orography with a discontinuous galerkin scheme. *Journal of Computational Physics*, 519, December 2024. (cit. on p. 27)
- [Ode13] J.T. Oden. *Finite Elements of Nonlinear Continua*. Dover Civil and Mechanical Engineering. Dover Publications, 2013. (cit. on p. 26)
- [OS25] Oliver O’Brien and Christoph Sünderhauf. Quantum state preparation via piecewise QSVT. *Quantum*, 9:1786, July 2025. (cit. on pp. 141 and 145)
- [PJKW25] Matthew Pocrnic, Peter D. Johnson, Amara Katarwa, and Nathan Wiebe. Constant-Factor Improvements in Quantum Algorithms for Linear Differential Equations, June 2025. (cit. on pp. 137, 140, and 145)
- [Ric24] Cleve Richard. Lecture notes on quantum information processing: Quantum algorithms. <https://cleve.iqc.uwaterloo.ca/qic710/lecture-notes/>, 2024. (cit. on p. 49)
- [RPGE17] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. Quantum arithmetic with the quantum fourier transform. *Quantum Information Processing*, 16(6), April 2017. (cit. on p. 59)
- [RV05] Yvan Van Rentergem and Alexis De Vos. Optimal design of a reversible full adder. *Int. J. Unconv. Comput.*, 1:339–355, 2005. (cit. on p. 60)
- [SBM06] Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. Synthesis of Quantum Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, June 2006. (cit. on p. 91)
- [SCC24] Christoph Sünderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. *Quantum*, 8:1226, January 2024. (cit. on pp. 84, 85, 99, 100, and 145)
- [Sci25] SciML. ExponentialUtilities.jl: High-performance matrix exponentiation and products. <https://github.com/SciML/ExponentialUtilities.jl>, 2025. GitHub repository. (cit. on p. 129)
- [Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. (cit. on p. 2)
- [SZ20] Siavosh Shadpey and David W. Zingg. Entropy-Stable Multidimensional Summation-by-Parts Discretizations on hp-Adaptive Curvilinear Grids for Hyperbolic Conservation Laws. *Journal of Scientific Computing*, 82(3):70, March 2020. (cit. on p. 22)
- [Tre13] L.N. Trefethen. *Approximation Theory and Approximation Practice*. Other Titles in Applied Mathematics. Siam, 2013. (cit. on pp. 108 and 109)

- [Wan07] Z.J. Wang. High-order methods for the euler and navier-stokes equations on unstructured grids. *Progress in Aerospace Sciences*, 43(1):1–41, 2007. (cit. on p. 1)
- [WLC24] Siyi Wang, Eugene Lim, and Anupam Chattopadhyay. Boosting the efficiency of quantum divider through effective design space exploration, 2024. (cit. on pp. 100 and 145)
- [YYL⁺24] Chunlin Yang, Hongmei Yao, Zexian Li, Zhaobing Fan, Guofeng Zhang, and Jianshe Liu. Block encoding of sparse structured matrices coming from ocean acoustics in quantum computing, July 2024. (cit. on pp. 81, 82, 84, 85, 86, 99, and 100)
- [ZN16] Philip Zwanenburg and Siva Nadarajah. Equivalence between the energy stable flux reconstruction and filtered discontinuous galerkin schemes. *Journal of Computational Physics*, 306:343–369, 2016. (cit. on p. 26)

Appendix A: Supplementary Material for Chapter 6

This appendix provides supplementary material for Chapter 6, including detailed circuit constructions and explicit gate counts for the quantum circuits discussed therein.

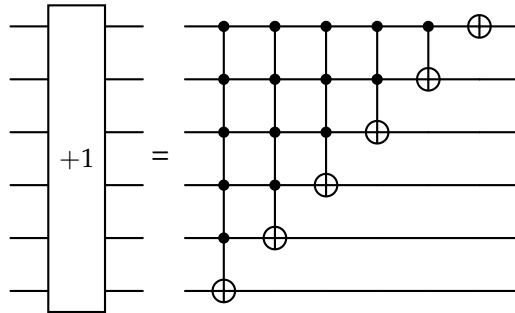
Quantum Incrementer

Naive Incrementer The quantum incrementer INC_n can be expressed as an ordered product of MCX-gates:

$$\text{INC}_n := \prod_{j=n-1}^0 \text{MCX}_j,$$

where MCX_j targets qubit j and is controlled on qubits $0, 1, \dots, j-1$ ¹.

¹ The gate with the largest index acts first; $\text{MCX}_0 := X^{(0)}$ is a single X gate.



Each gate implements the propagation of a carry bit: starting from the LSB, a bit flips iff all less significant bits are in the $|1\rangle$ state. Thus, the circuit forms a cascade of MCX-gates, each applying a conditional flip determined by the lower bits. This naive design is shown in Figure 1. Decomposing every MCX_j with the constructions from Chapter 6 leads to a quadratic cost, since each MCX_j requires $\mathcal{O}(j)$ CX gates. Hence,

Naive incrementer cost: $\mathcal{O}(n^2)$ CX gates.

A more efficient alternative, due to Gidney [Gid18, Figure 19], achieves

Figure 1: Naive incrementer using a series of MCX gates for $n = 6$.

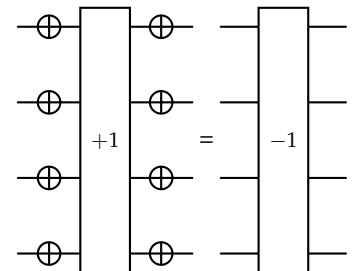


Figure 2: Surrounding an increment gate with X gates turns it into a decrement gate.

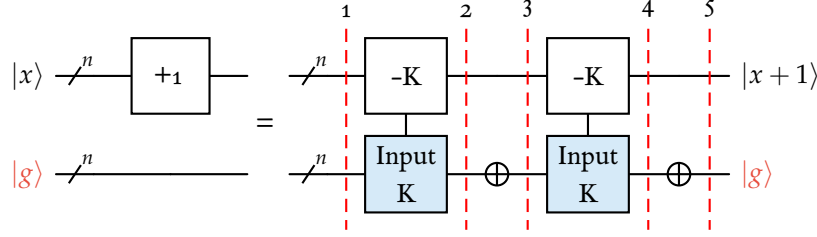


Figure 3: Incrementer circuit for n -qubit registers using n dirty ancilla qubits in register $|g\rangle$.

the same operation using two's-complement arithmetic with a dirty ancilla register as detailed next.

Gidney Incrementer: n -dirty ancillae Let $g \in \{0, \dots, 2^n - 1\}$ denote the two's-complement value held in a dirty ancilla register, and $\bar{g}$ its bitwise complement. Then

$$\bar{g} = (-g - 1) \bmod 2^n \implies g + \bar{g} = -1 \bmod 2^n, \quad -g - \bar{g} = 1 \bmod 2^n.$$

This identity, illustrated in Figure 2, implies that adding 1 can be realised by subtracting both a value and its complement:

$$|x\rangle \mapsto |x - g - \bar{g}\rangle = |(x + 1) \bmod 2^n\rangle,$$

for any unknown g , enabling increment with only dirty ancillae. Figure 3 shows the corresponding circuit. The evolution of the quantum state proceeds as:

1. Initialize: $|x\rangle |g\rangle$
2. Subtract the dirty register: $|(x - g) \bmod 2^n\rangle |g\rangle$
3. Apply bitwise negation to the ancilla: $|(x - g) \bmod 2^n\rangle |\bar{g}\rangle$
4. Subtract the complement: $|(x - g - \bar{g}) \bmod 2^n\rangle |\bar{g}\rangle = |(x + 1) \bmod 2^n\rangle |\bar{g}\rangle$
5. Uncompute the negation: $|(x + 1) \bmod 2^n\rangle |g\rangle$

Both subtraction steps use the Van Rentergem adder applied in reverse. The circuit therefore consists of two adders plus $2n$ X gates for the bitwise negations, yielding

Total CX count = $32n - 30$.

This achieves a linear gate count, representing a substantial improvement over the naive $\mathcal{O}(n^2)$ construction.

Gidney Incrementer: 1-dirty ancilla Building on the previous adder, we now describe a construction requiring only a **single** dirty ancilla qubit, following [Gid18, Figure 20].

Let $m = 2n - 1$ and partition the m -qubit target register as

$$|x\rangle = |x_H\rangle \otimes |x_L\rangle, \quad |x_H\rangle \in \mathcal{H}_{n-1}, \quad |x_L\rangle \in \mathcal{H}_n,$$

where $|x_L\rangle$ holds the least significant bits. We also have a dirty ancilla $|g\rangle \in \mathcal{H}_1$, prepended to $|x_H\rangle$ to form the extended high register

$$|\hat{x}_H\rangle := |g\rangle |x_H\rangle \in \mathcal{H}_n.$$

The goal is to implement the transformation

$$|x_L\rangle |x_H\rangle \mapsto |(x+1) \bmod 2^m\rangle,$$

using only one dirty qubit. The construction uses two interacting increments:

- **Increment the low half unconditionally:** Increment x_L using the double-subtraction method from before, treating $\hat{x}_H$ as the dirty register.
- **Increment the high half conditionally:** Increment x_H only when $x_L = |1^n\rangle$. This is achieved by sandwiching an addition of x_L to $\hat{x}_H$ between two MCX gates controlled on x_L . When the control condition fails, the addition and subtraction cancel; when true, the addition becomes another subtraction, yielding

$$|\hat{x}_H\rangle \mapsto |\hat{x}_H - 2x_L\rangle = |(\hat{x}_H + 2) \bmod 2^n\rangle,$$

since $x_L = -1 \bmod 2^n$ in two's-complement arithmetic².

² Two's-complement identity: $x_L = |1^n\rangle \iff x_L = -1 \bmod 2^n$.

We now trace the evolution through the circuit shown in [Figure 4](#):

1. Initialize: $|x_L\rangle |\hat{x}_H\rangle$
2. Subtract x_L from $\hat{x}_H$: $|x_L\rangle |(\hat{x}_H - x_L) \bmod 2^n\rangle$
3. Conditionally complement $\hat{x}_H$ if $x_L = |1^n\rangle$:

$$|x_L\rangle |f_{x_L}(\hat{x}_H - x_L)\rangle \quad \text{where} \quad f_{x_L}(z) = \begin{cases} \bar{z} & \text{if } x_L = |1^n\rangle, \\ z & \text{otherwise} \end{cases}$$

4. Add x_L in place: $|x_L\rangle |f_{x_L}(\hat{x}_H - x_L) + x_L\rangle$. When $x_L = |1^n\rangle$,

$$f_{x_L}(\hat{x}_H - x_L) + x_L = \overline{\hat{x}_H - x_L} + x_L = -\hat{x}_H - 3.$$

5. Uncompute the conditional complement:

$$|x_L\rangle |(\hat{x}_H + \delta - 1) \bmod 2^n\rangle, \quad \delta = \begin{cases} 3, & x_L = |1^n\rangle, \\ 0, & \text{otherwise.} \end{cases}$$

6. Increment x_L unconditionally using the double-subtraction trick:

$$|(x_L + 1) \bmod 2^n\rangle |(\hat{x}_H + \delta - 1) \bmod 2^n\rangle.$$

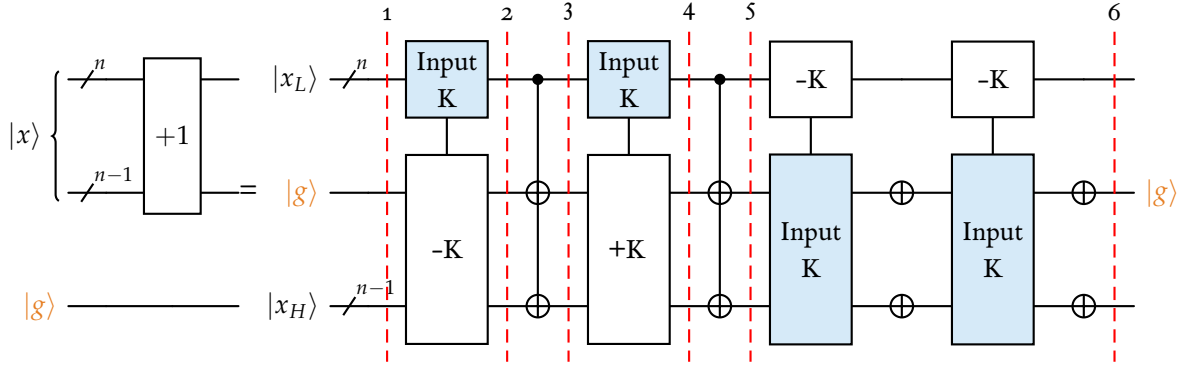


Figure 4: Incrementer circuit for an n -qubit register with odd n , using a single dirty ancilla qubit $|g\rangle$.

If $x_L = |1^n\rangle$, then $|(x_L + 1) \bmod 2^n\rangle = |0\rangle$ and,

$$|(\hat{x}_H + \delta - 1) \bmod 2^n\rangle = |(\hat{x}_H + 2) \bmod 2^n\rangle = |g\rangle |(x_H + 1) \bmod 2^{n-1}\rangle,$$

yielding the final transformation

$$|x\rangle |g\rangle \mapsto |(x + 1) \bmod 2^m\rangle |g\rangle,$$

thus realising a full increment using only a single dirty ancilla.

The construction described so far applies only to registers of odd length. To extend it to even-length registers, we isolate the least significant bit and perform a controlled increment on the remaining $(m - 1)$ -bit subregister conditioned on the LSB, followed by a final CX on the LSB. The increment then decomposes into two steps:

1. Apply a controlled-increment to the upper $(m - 1)$ bits, conditioned on the LSB being $|1\rangle$.
2. Apply a CX gate to the LSB to complete the increment.

Intuitively, this works because toggling the LSB increments it modulo 2, while the conditional increment propagates the carry when the LSB overflows from $|1\rangle$ to $|0\rangle$. The resulting circuit is shown in Figure 5. A detailed gate-count analysis yields:

$$\text{Total CX count} = \begin{cases} 44n - 62, & n \text{ even,} \\ 42n - 40, & n \text{ odd.} \end{cases}$$

For small registers ($n \leq 8$), the naive incrementer remains cheaper due to constant factors. However, in block-encoding contexts, 2^n corresponds to matrix dimension, so n is typically much larger.

Controlled Gidney Incrementer: 1-dirty ancilla The controlled version of the single-dirty-ancilla incrementer leverages the self-inverse structure of the original circuit, which consists of a subtractor and adder separated by layers of X and MCMT gates. To introduce control, all MCMT gates are promoted

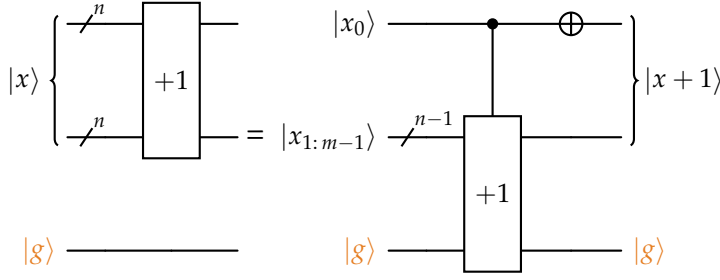


Figure 5: Incrementer circuit for an n -qubit register with even n , using a single dirty ancilla qubit $|g\rangle$.

to be conditioned on the control register, and the intervening X gates are replaced by their controlled counterparts. The resulting circuit is shown in Figure 6. When the control condition is inactive, the leading and trailing pairs of MCMT gates do not trigger, and the subtractor-adder sequence cancels exactly. A subtle modification ensures this cancellation still holds: the final subtractor in the uncontrolled design is replaced by an adder conjugated by X gates, maintaining reversibility when the control is off. When the control is active, the circuit executes identically to the original incrementer, achieving the desired controlled operation. The CX-gate complexity is bounded as³

$$\text{Total CX count} = \begin{cases} 60c + 50n - 112, & n \text{ even,} \\ 48c + 50n - 92, & n \text{ odd.} \end{cases}$$

³ Bounds are stated for the regime $c > n$. In block-encoding contexts we typically have $n \gg c$, resulting in smaller actual counts.

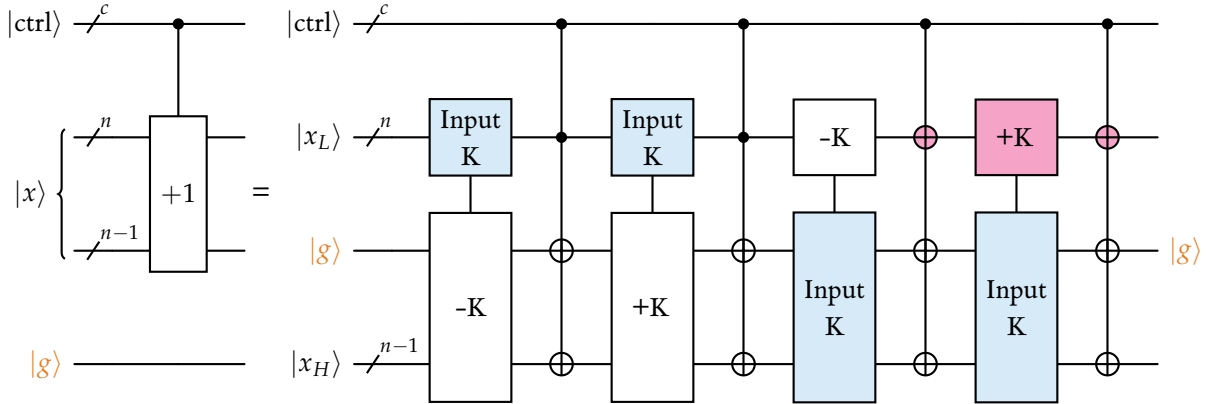


Figure 6: Controlled version of Figure 4. The pink gates indicate the modification where the final subtractor is replaced by an adder conjugated by X-gates.

Constant Adders

Controlled Draper Constant Adder The **controlled Draper constant adder** is obtained by promoting each phase gate in the $\text{CADD}_n(a)$ circuit to be additionally controlled on an external register. The key insight is that when the control condition is not met, no controlled-phase rotations are applied,

causing the QFT_n and $\text{QFT}_n^\dagger$ layers to cancel exactly. Consequently, the target register remains unchanged, while if the control is active, the adder performs the full modular addition. The corresponding circuit is shown in Figure 7.

Constant-Specific Optimisations for HRS Adder Additional circuit-level optimisations can be performed by exploiting the structure of the classical constant a . If a is divisible by 2^k , then the lowest k qubits of the quantum register $|x\rangle$ remain unchanged under addition modulo 2^n . Consequently, the adder operates only on the upper $n - k$ qubits, reducing both recursion depth and total gate count. Formally, writing $a = 2^k a'$ gives

$$|x\rangle \mapsto |x_L\rangle \otimes |(x_H + a') \bmod 2^{n-k}\rangle,$$

where $|x_L\rangle$ are the lowest k qubits and $|x_H\rangle$ the remaining $n - k$. The optimisation is implemented by detecting the largest power-of-two factor of a and omitting the corresponding least-significant qubits in the addition. For example, $a = 12 = 1100_2 = 2^2 \cdot 3$, so the two least significant qubits of $|x\rangle$ remain fixed. The recursive adder thus acts only on the top $n - 2$ qubits with constant 3. This truncation shortens the recursion and yields

Total CX count with shift optimisation: $\mathcal{O}((n - k) \log(n - k))$,

where k is the number of trailing zeros in the binary expansion of a .

Carry Gate Implementation The CARRY_n^a gate is implemented through reversible carry propagation using dirty ancillae. Let x_i denote the i^{th} qubit of the input register, a_i the i^{th} bit of the constant a , and g_i the dirty ancilla carrying overflow from bit i to $i + 1$. A carry is generated whenever $(x_i \wedge a_i)$, $(x_i \wedge g_{i-1})$, or $(a_i \wedge g_{i-1})$ holds, giving

$$g_i \mapsto g_i \oplus [(x_i \wedge a_i) \oplus (x_i \wedge g_{i-1}) \oplus (a_i \wedge g_{i-1})].$$

This Boolean rule is implemented reversibly as:

- If $a_i = 1$, apply $\text{CX}(x_i \mapsto g_i)$ and an X on x_i before the Toffoli to handle the case $x_i = 0, g_{i-1} = 1$.
- Always apply a Toffoli $\text{CCX}(x_i, g_{i-1}, g_i)$.

Because g_{i-1} may initially be $|1\rangle$, we must trigger the Toffoli only when g_{i-1} was toggled by the current addition rather than its initial value. This is ensured via the **double-Toffoli trick**: insert identical $\text{CCX}(x_i, g_{i-1}, g_i)$ gates before and after the operations that may flip g_{i-1} .

This guarantees correct carry propagation:

- If g_{i-1} is unchanged, the two Toffolis cancel.
- If g_{i-1} toggles, the control configuration differs and g_i is flipped once.

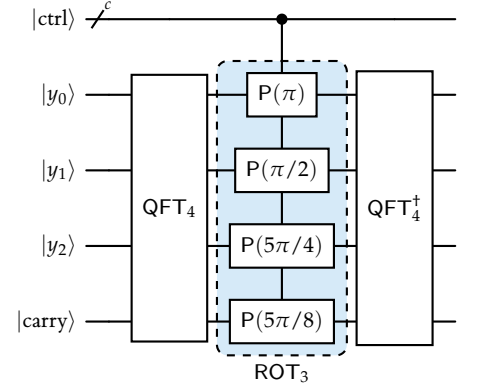
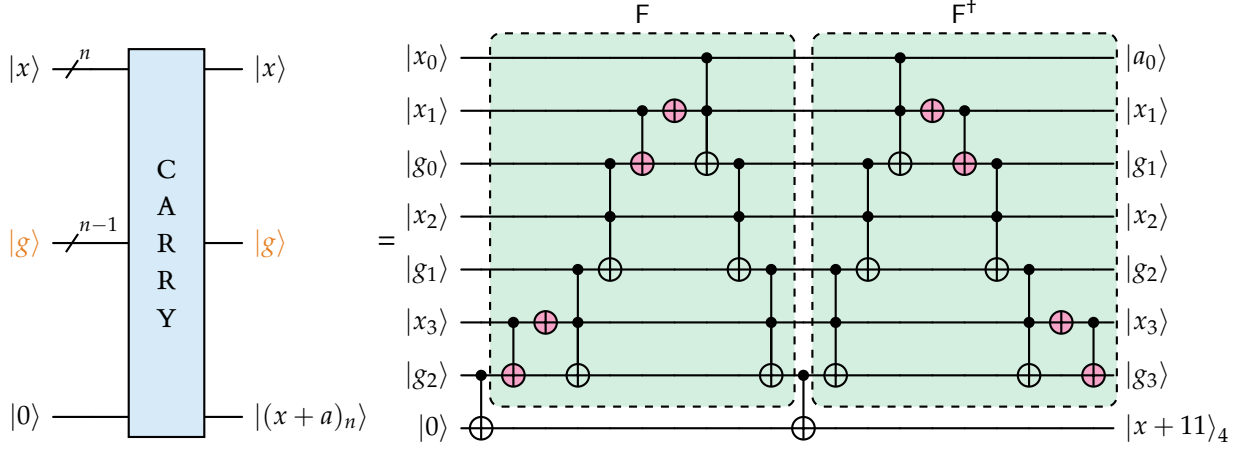


Figure 7: Controlled Draper constant adder circuit CADD_3 for $n = 3$ and $a = 5 = 101_2$.

Assuming all $a_i = 1$, the worst-case CX count is

Carry gate cost: $14n - 18$ CX gates.

The first ancilla g_0 can be removed by directly controlling g_1 on x_0 . If $a_0 = 0$, all operations on g_0 cancel; if $a_0 = 1$, the pair reduces to a single $\text{CCX}(x_0 \mapsto g_1)$.

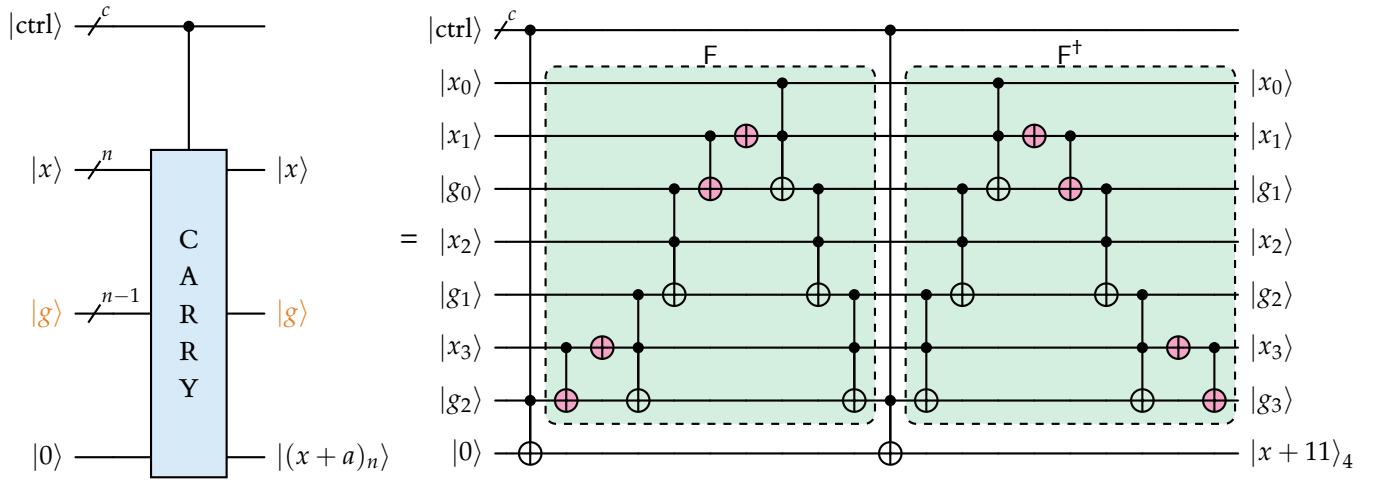


Controlled Carry Gate The controlled CARRY_n^a gate, shown in Figure 9, extends the original carry logic (Figure 8) to the controlled setting. The circuit follows a standard compute-uncompute pattern consisting of subcircuits F and $F^\dagger$, enclosed by two CX gates. To obtain the controlled version, the control condition is applied only to these two outer CX gates. When the control is inactive, the controlled CX gates act as identities, and the F and $F^\dagger$ blocks cancel, leaving the carry register unchanged. When the control is active, the circuit behaves identically to the original CARRY_n^a and computes the correct overflow bit. The CX-gate complexity is:⁴

Total CX count: $24c + 14n - 44$.

Figure 8: Example circuit for computing the final carry bit of $x + 11$ using dirty ancilla qubits g . The pink gates acting on qubit i are present only if the i -th bit of the constant a is 1. For the constant $a = 11 = 1011_2$, the gates on qubit index 2 are omitted, since $a_2 = 0$.

⁴ This bound assumes $c > 2n$, which is rare in our block-encoding applications.


 Figure 9: Controlled CARRY_n^a gate.

Appendix B: Supplementary Material

for Chapter 7

Commutativity of Block-Encoding Oracles

Lemma .1. *The oracle O_{data} commutes with both O_{rg} and O_r .*

Proof.

We verify commutativity case by case.

- **Commutation with O_r .** Consider

$$\begin{aligned}
 & (O_{\text{data}} \otimes \mathbb{1}_{N+2})(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \begin{cases} (O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |r_l(j)\rangle, & l \in [0, s_0 - 1], \\ (O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1], \end{cases} \\
 &= \begin{cases} \left(\frac{A_l}{\|A\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |r_l(j)\rangle, & l \in [0, s_0 - 1], \\ |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1]. \end{cases} \\
 & \hspace{15em} (1)
 \end{aligned}$$

Reversing the order yields the same action:

$$\begin{aligned}
 & (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r)(O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \begin{cases} (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r) \left(\frac{A_l}{\|A\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], \\ (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes O_r) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1], \end{cases} \\
 &= \begin{cases} \left(\frac{A_l}{\|A\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{|A_l|}{\|A\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |r_l(j)\rangle, & l \in [0, s_0 - 1], \\ |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1]. \end{cases} \\
 & \hspace{15em} (2)
 \end{aligned}$$

Hence, O_{data} commutes with O_r .

- **Commutation with O_{rg} .** Similarly,

$$\begin{aligned}
 & (O_{\text{data}} \otimes \mathbb{1}_{N+2})(\mathbb{1}_2 \otimes O_{\text{rg}}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \begin{cases} (O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], j \in \mathcal{C}_l, \\ (O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |1\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1] \text{ or } j \notin \mathcal{C}_l, \end{cases} \\
 &= \begin{cases} \left(\frac{A_l}{\|\mathbf{A}\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{A_l}{\|\mathbf{A}\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], j \in \mathcal{C}_l, \\ |0\rangle_{\text{data}} |1\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1] \text{ or } j \notin \mathcal{C}_l, \end{cases}
 \end{aligned} \tag{3}$$

Again, reversing the order yields the same expression:

$$\begin{aligned}
 & (\mathbb{1}_2 \otimes O_{\text{rg}})(O_{\text{data}} \otimes \mathbb{1}_{N+2}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle \\
 &= \begin{cases} (\mathbb{1}_2 \otimes O_{\text{rg}}) \left(\frac{A_l}{\|\mathbf{A}\|} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{A_l}{\|\mathbf{A}\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], \\ (\mathbb{1}_2 \otimes O_{\text{rg}}) |0\rangle_{\text{data}} |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1], \end{cases} \\
 &= \begin{cases} \left(\frac{A_l}{\|\mathbf{A}\|_{\max}} |0\rangle_{\text{data}} + \sqrt{1 - \left(\frac{A_l}{\|\mathbf{A}\|_{\max}} \right)^2} |1\rangle_{\text{data}} \right) |0\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [0, s_0 - 1], j \in \mathcal{C}_l, \\ |0\rangle_{\text{data}} |1\rangle_{\text{flag}} |l\rangle |j\rangle, & l \in [s_0, s - 1] \text{ or } j \notin \mathcal{C}_l. \end{cases}
 \end{aligned} \tag{4}$$

Hence, O_{data} commutes with O_{rg} .

■

Algorithm 4: Canonical Offset

Input: Integers $i, j \in [0, N - 1]$, modulus $N > 0$.

Output: Canonical representative $k \in [-N + 1, N - 1]$ satisfying $k \equiv (i - j) \pmod{N}$.

```

1: diff  $\leftarrow (i - j) \pmod{N}$ 
2: if diff = 0 then
3:   return 0
4: end if
5:  $k_1 \leftarrow \text{diff}$  ▷ positive representative
6:  $k_2 \leftarrow \text{diff} - N$  ▷ negative representative

7: function  $v_2(m)$ 
8:    $m \leftarrow |m|$ 
9:   return  $\text{bitlength}(m \ \& \ -m) - 1$  ▷ number of trailing zeros in binary representation
10: end function

11:  $t_1 \leftarrow v_2(k_1), \quad t_2 \leftarrow v_2(k_2)$ 
12: if  $t_1 \neq t_2$  then
13:   return  $\arg \max_{k \in \{k_1, k_2\}} v_2(k)$ 
14: else if  $|k_1| \neq |k_2|$  then
15:   return  $\arg \min_{k \in \{k_1, k_2\}} |k|$ 
16: else
17:   return  $\max(k_1, k_2)$  ▷ tie: prefer positive representative
18: end if

```

Appendix C: Supplementary Material

for Chapter 10

Integral Identity for Legendre Polynomials

Lemma .2. *For orthonormalised Legendre polynomials,*

$$\int_{-1}^1 \varphi_i(\xi) \varphi_j'(\xi) d\xi = \begin{cases} \sqrt{(2i+1)(2j+1)}, & i+j \text{ odd}, j > i, \\ 0, & \text{otherwise.} \end{cases}$$

Proof.

Using $\varphi_i(\xi) = \sqrt{\frac{2i+1}{2}} P_i(\xi)$, we obtain

$$\int_{-1}^1 \varphi_i(\xi) \varphi_j'(\xi) d\xi = \frac{\sqrt{(2i+1)(2j+1)}}{2} \int_{-1}^1 P_i(\xi) P_j'(\xi) d\xi.$$

Integration by parts gives

$$I_{ij} = \int_{-1}^1 P_i(\xi) P_j'(\xi) d\xi = [P_i(\xi) P_j(\xi)]_{-1}^1 - \int_{-1}^1 P_i'(\xi) P_j(\xi) d\xi.$$

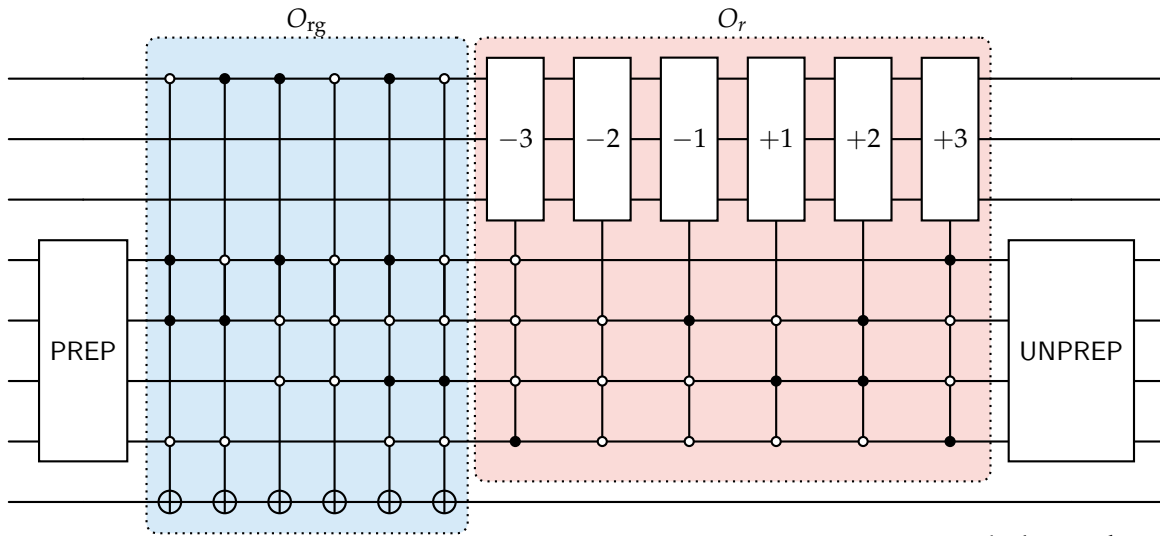
The boundary term is $1 - (-1)^{i+j}$, and we have

$$I_{ij} + I_{ji} = 1 - (-1)^{i+j} = \begin{cases} 0, & i+j \text{ even}, \\ 2, & i+j \text{ odd.} \end{cases}$$

If $j \leq i$, then P_j' has degree $\leq j-1 < i$, so by orthogonality $I_{ij} = 0$. Similarly, if $j > i$, then $I_{ji} = 0$. Thus, we have

$$I_{ij} = \begin{cases} 2, & i+j \text{ odd}, j > i, \\ 0, & \text{otherwise.} \end{cases}$$

The result follows. ■

Quantum Circuit for Block-Encoding Q_g

 Figure 10: Block-encoding Q_g matrix for $p = 1$.