Imperial College London

Development and Analysis of Variational Quantum Algorithms to Solve Engineering Differential Equations

---

By

Joséphine Hunout

Supervised by

Prof Sylvain Laizet and Prof Lorenzo Iannucci

Thesis submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Aeronautics of Imperial College London

Department of Aeronautics

Faculty of Engineering

Imperial College London

September 2025

Declaration of Originality

I hereby declare that all the work and material presented in this thesis is composed and reported as my own. All information and data obtained from external sources are properly acknowledged and cited in text with the full list of references given in the final bibliography section.

Joséphine Hunout

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to the many people who have supported me throughout my PhD journey.

First and foremost, I wish to thank the late Professor Lorenzo Iannucci, who chose me for this project and believed in my potential. Although he will unfortunately never see its completion, I hope that the successful accomplishment of this thesis would make him proud. I am also deeply grateful to my second supervisor, Professor Sylvain Laizet, who stepped in midway through the project and offered invaluable guidance and support.

I am profoundly grateful to all the incredible colleagues and friends I met along the way. Your companionship, encouragement, and the countless moments we shared made my experience at Imperial College London truly special and unforgettable.

Finally, I would like to take this opportunity to thank my family, and in particular my parents, who have supported me unwaveringly throughout ten years of study. They may never have fully understood the nature of my work, yet their trust and faith in me have been a constant source of strength.

Abstract

Differential equations underpin a vast range of scientific and engineering domains, including structural mechanics, fluid dynamics, and financial modelling. Their intricate nature often renders classical numerical methods computationally demanding, requiring fine discretization or large basis expansions, which can lead to prohibitive costs. Quantum computing offers a fundamentally different paradigm, leveraging superposition and entanglement with the promise of computational speedup. However, in the current Noisy Intermediate-Scale Quantum (NISQ) era, fully fault-tolerant quantum computers remain decades away, motivating the development of hybrid strategies such as Variational Quantum Algorithms (VQAs), which train parametrised quantum circuits using classical optimizers.

This work investigates spectral VQAs for solving differential equations, extending the Chebyshev-encoded VQA of Kyriienko *et al.* and introducing a novel quantum spectral approach: the Lagrange-encoded VQA. This approach shifts the paradigm by incorporating entanglement directly within the encoding block while maintaining a shallow variational ansatz. Inspired by the Hadamard test structure, the design enables an efficient representation of the complete set of Lagrange polynomial basis functions using a limited number of gates and facilitates the use of the Hadamard test differentiation method, thereby reducing the computational load. VQA performance is analysed through the accuracy of the result in combination with the gate count and circuit count metrics, which serve as proxies for runtime and noise accumulation on NISQ devices.

The proposed VQA is benchmarked on multiple problems: (i) a first-order linear ODEs and (ii) the second-order ODEs damped mass-spring system, both comparing Lagrange- and Chebyshev-encoded VQAs, (iii) the Poisson equation, comparing the Lagrange-encoded VQA with the discretized approach of Sato *et al.*, and (iv) a first-order ODE with nonlinear coefficients addressed with the Lagrange-encoded VQA. Across these cases, the Lagrange-encoded VQA consistently achieves comparable or improved accuracy while significantly reducing circuit depth and computational overload, making it a promising candidate for NISQ devices.

Contents

1 Introduction **1**

1.1 Motivations 1

1.2 Research Journey 2

1.3 Objectives 3

1.4 Contributions 4

1.5 Thesis Outline 6

2 Quantum Computing Context **9**

2.1 The objectives of Quantum Computing 9

2.2 Computational promises of Quantum Computing 10

2.2.1 Quantum Computing paradigm 10

2.2.2 Quantum gates and circuits 14

2.2.3 Quantum algorithms demonstrating a potential quantum advantage 16

2.3 Technical challenges of Quantum Computing 21

2.3.1 Fault-tolerant quantum computing versus Noisy Intermediate-Scale Quantum systems 21

2.3.2 Quantum Noise 22

2.3.3 Current Quantum Hardware Technologies 24

2.4 Quantum Circuit Runtimes . . . 27
2.4.1 Readout and measurement times . . . 28
2.4.2 Parallelism and Circuit steps . . . 28
2.4.3 Connectivity . . . 29
2.4.4 Quantum Memory . . . 30
2.5 Summary . . . 31

3 Literature Review **33**

3.1 Differential equations . . . 33
3.1.1 Types of differential equations . . . 34
3.1.2 Existence of Solutions . . . 36
3.2 Classical Computing for solving differential equations . . . 37
3.3 Quantum Computing for solving differential equations . . . 38
3.3.1 Fully quantum algorithms . . . 38
3.3.2 Hybrid quantum-classical algorithms . . . 40
3.4 Variational Quantum Algorithms . . . 40
3.4.1 Variational Quantum Circuit . . . 42
3.4.2 Cost Operator and Loss Function . . . 42
3.4.3 Classical optimizers . . . 42
3.4.4 Barren Plateau . . . 46
3.4.5 Derivative Quantum Circuits . . . 46
3.4.6 Variational Ansatz . . . 48
3.4.7 Challenges and Limitations in development of VQA . . . 50
3.5 Existing VQAs for solving differential equations . . . 51

3.5.1 Kyriienko *et al.* Algorithm . . . 51

3.5.2 Sato *et al.* Algorithm . . . 55

3.6 Summary . . . 59

4 Variational Quantum Algorithms Based on Polynomial Encoding 61

4.1 Algorithms Workflow . . . 62

4.1.1 Classical Optimizer . . . 63

4.1.2 Loss Function . . . 64

4.1.3 Initial and Boundary Conditions . . . 65

4.1.4 Regularization . . . 66

4.1.5 Cost Operator . . . 66

4.2 Derivatives and Gradients . . . 67

4.2.1 Differentiability . . . 68

4.3 Iterative Process . . . 68

4.4 Quantum Circuits Structures . . . 70

4.4.1 Chebyshev-encoded VQA . . . 71

4.4.2 Proposed new Lagrange-encoded VQA . . . 71

4.5 Summary . . . 75

5 Analysis of the Polynomial-based Variational Quantum Algorithms 77

5.1 Chebyshev-encoded VQA . . . 77

5.1.1 Function circuits . . . 77

5.1.2 Derivative circuits . . . 82

5.2 Lagrange-encoded VQA . . . 87

5.2.1 Function circuit 87
5.2.2 Quantum feature map circuit structures 91
5.2.3 Complete parametrised circuit 93
5.2.4 Entangling layers in the variational ansatz 94
5.2.5 Derivative circuits 97
5.3 Variational Ansatz 102
5.3.1 Gradients with respect to the variational parameters 103
5.4 Summary 105

6 Quantum Circuit Simulation and Complexity Estimation 107

6.1 Quantum Circuit Simulation 107
6.1.1 Summary 118
6.2 Derivative Quantum Circuit Simulation 119
6.2.1 Derivatives of Chebyshev-encoded VQCs 120
6.2.2 Derivatives of Lagrange-encoded VQCs 122
6.2.3 Summary 125
6.3 Variational Quantum Algorithm Runtime 127
6.3.1 Runtime of VQAs on quantum computers 127
6.3.2 Runtime of simulated VQAs 129
6.4 Complexity Estimation - Gate Count 131
6.4.1 Chebyshev-encoded VQA 131
6.4.2 Lagrange-encoded VQA 132
6.4.3 Sato *et al.* VQA 133
6.5 Summary 135

7 Results **137**

7.1 First-order Linear Ordinary Differential Equation 137

7.1.1 Problem definition 137

7.1.2 Analytical solution 138

7.1.3 Quantum resolution of the homogeneous ODE 138

7.1.4 Quantum resolution of the complete ODE 149

7.1.5 Summary 154

7.2 Damped Mass-Spring System 156

7.2.1 Problem definition 156

7.2.2 Analytical solution 157

7.2.3 Quantum resolution 157

7.2.4 Trainability and Expressivity 167

7.3 Poisson Equation 170

7.3.1 Problem definition 170

7.3.2 Analytical solution 170

7.3.3 Quantum resolution 172

7.4 First-order linear ODE with nonlinear coefficients 182

7.4.1 Problem definition 182

7.4.2 Quantum resolution 183

7.5 Summary 187

8 Conclusions and Future Work **189**

8.1 Summary and Conclusions 189

8.2 Future Work 191

8.2.1 VQA Encoding Feasibility 191

8.2.2 Alternative Polynomials and Trainable Encoding 191

8.2.3 Extended Applications 192

Bibliography **192**

List of Figures

2.1 Photograph of the Sycamore chip [7]. 10

2.2 Visualisation of a classical bit and a qubit 11

2.3 Symbol of the quantum measurement gate. 13

2.4 Symbols of the quantum Pauli gates X, Y and Z. 14

2.5 Symbols of the quantum Clifford gates H, S and CNOT. 15

2.6 Symbols of the quantum rotational gates R_Y, R_Y and R_Z. 15

2.7 Quantum circuit used to create the Bell state $|\Phi^+\rangle$, starting from a 2-qubit system initialised in $|00\rangle$ and applying the unitary $\text{CNOT}_{01}, (H \otimes I)$. 16

2.8 Quantum subroutine in Shor's algorithm. 18

2.9 Quantum subroutine in Grover's algorithm. 19

2.10 Quantum subroutine in the HHL algorithm. 21

2.11 a) Photograph of the IBM *Heron* processor [42]. b) Photograph of the IBM *Quantum system two* quantum computer [42]. 25

2.12 Illustration of quantum circuit execution steps. Each step corresponds to a set of gates executed in parallel. The total runtime depends on the number of steps and the time required for the slowest gate in each layer. Two-qubit gates impose constraints that limit parallelism compared to single-qubit gates. 29

2.13 a) Typical connectivity graph of a superconducting circuit quantum processor. b) Star-like connectivity of trapped ion qubits. c) Single 2D layer of optically trapped neutral atoms, showing long-range interactions (blue shading) with the highlighted yellow atom in an excited Rydberg state. Figure from [37]. 30

3.1 Schematic representation of a variational quantum algorithm and its hybrid quantum classical workflow. 41

3.2 Parameter Shift Rule. a) Original quantum circuit, b) Derivative quantum circuits. α and β are parameters of Pauli gates, representing rotation angles. 47

3.3 Hadamard test differentiation. a) Hadamard Test, b) Hadamard Test Differentiation. . 48

3.4 Hardware Efficient Ansatz quantum circuit. 49

3.5 Quantum circuit structure of the Kyriienko *et al.* algorithm. 52

3.6 Results obtained by Kyriienko *et al.* for the discontinuous first-order differential equation using VQAs. a), b): solutions for $\lambda = 8$ and $\lambda = 20$, respectively. c): comparison of different circuit depths for $\lambda = 20$ with Chebyshev tower encoding. d), e), f): corresponding evolution of the overall loss function and quality function across iterations. Figure from [4]. 54

3.7 Application of the VQA by Kyriienko *et al.* to the stationary Navier–Stokes equations. a) First optimization stage, b) second optimization stage, c) evolution of the solution quality and the loss function across iterations. Figure from [4]. 55

3.8 Quantum circuit structure of the Sato *et al.* algorithm, reproducing the state $|s, \psi_\theta\rangle$ defined in Equation (3.18). 56

3.9 Solutions obtained with the VQA of Sato *et al.* for each boundary condition, compared to the classical computation of $A^{-1}s$ as a function of the number of qubits. Figure from [5]. 58

4.1 Workflow of the proposed variational quantum algorithm, building on the general schematic presented in Fig. 3.1. The quantum computation part is shown in red, whereas all other stages are performed classically. 62

4.2 Quantum circuit structure of the Lagrange-encoded algorithm with an extended structure. 71

4.3 Circuit Structure of the Lagrange-encoded algorithm with a reduced structure. 72

5.1 One-qubit Chebyshev quantum feature map. 78

5.2 Comparison between quantum observables and Chebyshev polynomials of the first kind: a) Single qubit measurement $\langle \hat{Z}_j \rangle$ and corresponding Chebyshev polynomials. b) Total Z measurement $\langle \hat{Z}_{tot} \rangle$ and sum of the Chebyshev polynomials. 79

5.3 Entangled two-qubits Chebyshev encoding. 79

5.4 Two-qubits Chebyshev encoding and two-layers HEA 80

5.5 Comparison between parameter shift rule outcomes and analytical derivatives of Chebyshev polynomials with respect to the encoding function φ_j. a) $\langle \hat{Z}_j \rangle$ measurements on each qubit. b) Sum over all qubits of both quantum-derived and analytical derivatives. 83

5.6 Gradient of the quantum expectation value with respect to the input variable x, obtained via automatic differentiation. a) $\langle \hat{Z}_j \rangle$ measurements on each qubit, differentiated with respect to x. b) Sum across all qubits, compared to the theoretical derivative of the full encoded polynomial. 84

5.7 a) Plot of the encoding functions φ_j applied to each qubit, for $j = 1$ to 4. b) Corresponding analytical derivatives $\frac{d\varphi_j}{dx}$, used for the automatic differentiation. 85

5.8 a) Second-order derivatives $\frac{d^2\varphi_j}{dx^2}$ for $j = 1$ to 4, highlighting the divergence as $x \to 1^-$. b) Zoom-in on the region $x \in [0.7, 1]$, showing the small differences between the second-order derivatives. 87

5.9 Comparison of the three types of sets of encoding points for $N_{\text{points}} = 10$. 89

5.10 Hadamard Test circuit written with the Lagrange encoding function. 90

5.11 Extended quantum feature map circuit for $n = 2$ encoding points. 91

5.12 Reduced quantum feature map circuit for $n = 2$ encoding points. 92

5.13 Extended quantum feature map circuit for $n = 3$ encoding points. 92

5.14 Reduced quantum feature map circuit for $n = 3$ encoding points. 93

5.15 Comparison between quantum observables and Lagrange basis polynomials: a) Expectation values $\langle \hat{Z}_j \rangle$ from single-qubit measurements and corresponding degree-2 polynomials. b) Scaled observables $\frac{1}{\rho_j} \langle \hat{Z}_j \rangle$ and associated Lagrange basis functions $L_j(x)$. . . 93

5.16 Reduced quantum parametrised circuit with Lagrange encoding for $n = 3$ encoding points. 94

5.17 Lagrange encoding for $n = 3$ followed by an entanglement layer. 95

5.18 Lagrange encoding for $n = 3$ followed by a layer of hardware efficient ansatz. 96

5.19 Comparison between the output of the function circuit for a random variational parameter over a set of five Chebyshev nodes of the first type (Equation (5.20)) and the theoretical Lagrange interpolation polynomial, as well as their first order derivatives. a) Function f arising from the cost function $\hat{C}$ and corresponding Lagrange polynomial P. b) Derivative $\frac{d}{dx} f$ obtained from the total cost $\sum_k \hat{C}_k$ and derivative of the polynomial $\frac{d}{dx} P$. 99

5.20 a) Plot of the encoding functions φ_j applied to each qubit, for $j = 0$ to 4. b) Corresponding analytical derivatives $\frac{d\varphi_j}{dx}$, used for the automatic differentiation. 100

6.1 Evolution of the order of magnitude of the number of shots required to reach a given accuracy for different measured value magnitudes. a) for values between 0.1 and 2, b) for values between 1 and 2. 109

6.2 Version 1 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *Aer-Simulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value. a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit. 110

6.3 Version 2 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *Aer-Simulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value. a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit. 112

6.4 Reduced six-qubit function circuit with third-version Lagrange encoding 113

6.5 Version 3 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *Aer-Simulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value. a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit. 113

6.6 Comparison of the three versions (V1, V2, V3) of the Lagrange quantum feature map circuit simulated on the IBM Qiskit *AerSimulator* with 50 thousands shots and the statevector value. a) Z-measurement of the first qubit. b) Z-measurement of the second qubit. 114

6.7 Comparison of the third version of the Lagrange quantum feature map circuit simulated on noise-free *AerSimulator* and noisy *FakeNairobiV2* simulator using 50 thousand shots, referenced to the statevector result. a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit. 115

6.8 Impact of noise on the Lagrange encoded function circuit for two HEA, evaluated under the two measurement settings. Simulations were performed with 5 and 10 thousand shots on both the noise-free *AerSimulator* and the noisy *FakeNairobiV2* simulator, in comparison to the statevector value a) Single-gate sequence $\hat{R}_X$, and b) Two-gate sequence $\hat{R}_Z - \hat{R}_X$. 115

6.9 Impact of noise on the Lagrange encoded function circuit for two HEA, evaluated under the two measurement settings. Simulations were performed with 50 thousand shots on both the noise-free *AerSimulator* and the noisy *FakeNairobiV2* simulator, in comparison to the statevector value a) Single-gate sequence $\hat{R}_X$, and b) Two-gate sequence $\hat{R}_Z - \hat{R}_X$. 116

6.10 Comparison of a Chebyshev-encoded circuit implemented with two hardware-efficient ansätze, evaluated under two measurement settings. Simulations were performed on the IBM Qiskit noise free *AerSimulator* and noisy *FakeNairobiV2*, using 5 and 10 thousand shots, and compared to the Statevector value. a) Single-gate HEA sequence $\hat{R}_X$, b) Three-gate HEA sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$. 117

6.11 Comparison of a Chebyshev-encoded circuit implemented with two hardware-efficient ansätze, evaluated under two measurement settings. Simulations were performed on the IBM Qiskit noise free *AerSimulator* and noisy *FakeNairobiV2*, using 50 thousand shots, and compared to the Statevector value. a) Single-gate HEA sequence $\hat{R}_X$, b) Three-gate HEA sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$. 117

6.12 Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Forward Difference Approximation (FDA) from ideal statevector measurements, applied to a random function constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and FDA. b) Mean Square Error (MSE) between PSR and FDA derivatives. 120

6.13 Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Forward Difference Approximation (FDA) from sampling-based measurements on a noise-free simulator, as well as PSR from the ideal statevector measurement. The test function is constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and FDA. b) Mean Square Error (MSE) between PSR and FDA derivatives. 121

6.14 Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Central Difference Approximation (CDA) from sampling-based measurements on a noise-free simulator, as well as PSR from the ideal statevector measurement. The test function is constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and CDA. b) Mean Square Error (MSE) between PSR and CDA derivatives. 122

6.15 Comparison between derivatives obtained using the Hadamard test differentiation (HT) and the Forward Difference Approximation (FDA) from ideal statevector measurements, applied to a random function constructed with a Lagrange-encoded parameterized quantum circuit (with the second version of the encoding function) with $n = 4$ qubits. a) Derivative with respect to x computed using HT and FDA. b) Mean Square Error (MSE) between HT and FDA derivatives. 122

6.16 Comparison between derivatives obtained using the Hadamard test differentiation method (HT) and the Forward Difference Approximation (FDA) from sampling-based measurements on a noise-free simulator, as well as HT from the ideal statevector measurement. The test function is constructed with a Lagrange-encoded parameterized quantum circuit ($n = 4$ qubits, version 2). a) Derivative with respect to x computed using HT and FDA. b) Mean Square Error (MSE) between HT and FDA derivatives. 123

6.17 Comparison from ideal statevector measurements and the *AerSimulator* sampling measurements, for, a) the function circuit f, b) the derivative of the function circuit using the Hadamard test differentiation method 124

6.18 Comparison between derivatives obtained using the Hadamard test differentiation method (HT) and the Central Difference Approximation (CDA) from sampling-based measurements on a noise-free simulator, and HT from the ideal statevector measurement. The function is constructed with a Lagrange-encoded parameterized quantum circuit ($n = 4$ qubits, version 2). a) Derivative with respect to x computed using HT and CDA. b) Mean Square Error (MSE) between HT and CDA derivatives. 125

6.19 Runtime of the Chebyshev-encoded circuit measured with total Z measurement for 20 points with a varying number of qubits. a) Evolution of the runtime with a depth increase from 3 to 6, with 1 gate sequence $\hat{R}_X$, b) Evolution of the runtime with a HEA gate sequence increase from $\hat{R}_X$ to $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, with a depth of 3. 129

6.20 Runtime of the Chebyshev-encoded circuit measured with total Z measurement for 20 points with a varying number of qubits. a) Evolution of the runtime between statevector measurement and noise-free simulation using IBM AerSimulator, with a depth of 3 and HEA sequence of $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, b) Evolution of the runtime with a depth increase from 3 to 6, with 1 gate sequence $\hat{R}_X$, measured by the noise-free simulator IBM AerSimulator. 130

7.1 Overall loss from the differential equation term $\mathcal{L}_{DE}$ for each run of the algorithm (y-axis), shown on a logarithmic scale, as a function of the number of training points (x-axis) and the type of encoding points. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits. 140

7.2 Average number of circuits (left axis) and average overall loss over five runs (right axis) for each configuration. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits. 141

7.3 a) Circuit function and b) first-order derivative for the best result obtained in each configuration. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits. 141

7.4 Comparison between the three approaches. a) Overall loss from the differential equation term $\mathcal{L}_{DE}$ for each run of the algorithm (y-axis), plotted on a logarithmic scale. b) Average overall loss over five runs (right axis) and average number of circuits (left axis) for each configuration. The results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6, 7$ encoding points and applying the optimizer in one step or two steps. 143

7.5 a) Circuit function, b) first-order derivative for the best result obtained in each configuration. The results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6, 7$ encoding points and applying the optimizer in one step or two steps. 143

7.6 a) Overall loss $\mathcal{L}_{DE}$ for each run of the algorithm across different ansatz depths. b) Average overall loss and corresponding circuit count as a function of depth. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits. 145

7.7 Solutions from the best run for each depth studied, showing both the intermediate result after the first optimization step (10 training points) and the final result after the second step (20 training points). a) Function f, b) First-order derivative $\dot{f}$. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits. 146

7.8 Final solutions from the best run for each depth studied, obtained after the second optimization step (20 training points). a) Function f, b) First-order derivative $\dot{f}$. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits. 147

7.9 Loss evaluated across the interval for the best run corresponding to each depth, shown on a logarithmic scale. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits. 147

7.10 Comparison of the approximated solutions of the homogeneous first-order linear ODE obtained from Chebyshev- and Lagrange-encoded VQAs. a) Function f, b) First-order derivative $\dot{f}$. The Chebyshev encoding uses 6 qubits, a tower structure, and a hardware-efficient ansatz of depth 5 with a three-gate sequence, resulting in 90 variational parameters. The Lagrange encoding uses 7 interpolation points, corresponding to 8 qubits and 7 variational parameters. 148

7.11 Comparison of Chebyshev-encoded and Lagrange-encoded VQAs for the homogeneous first-order linear ODE. a) Average circuit counts (left axis) and average overall loss (right axis), b) Loss function across the interval for the best-performing instance of each VQA. The Chebyshev encoding uses 6 qubits, tower encoding, and a hardware-efficient ansatz of depth 5 with a three-gate sequence, for 90 variational parameters. The Lagrange encoding uses 7 interpolation points, corresponding to 8 qubits and 7 variational parameters. 149

7.12 Comparison of Lagrange-encoded VQA performance for 6, 7, and 8 encoding points. a) Overall loss over five runs per configuration. b) Average circuit count (left axis) and corresponding average overall loss (right axis). 150

7.13 Best Lagrange-encoded VQA solutions for 6, 7, and 8 encoding points. a) Function $f(x)$, b) First-order derivative $\dot{f}(x)$. 151

7.14 Loss function $\mathcal{L}_{DE}$ over the interval for the best result obtained with 6, 7, and 8 encoding points in the Lagrange-encoded VQA, compared with the analytical solution. 151

7.15 Chebyshev-encoded VQA applied to the complete ODE: comparison of five runs. a) Overall loss for each run, plotted on a logarithmic scale. b) Loss function over the interval, showing the best result in red and the variability of other runs in the green area. 152

7.16 Chebyshev-encoded VQA applied to the complete ODE: comparison of five runs with the analytical solution. a) Function $f(x)$. b) First-order derivative $\dot{f}(x)$. The red curve corresponds to the best run, while the green area shows the variability of the remaining runs. 152

7.17 Comparison of the best outcomes of the Chebyshev and Lagrange encoded VQAs applied to the complete ODE, plotted against the analytical solution. a) Function $f(x)$. b) First-order derivative $\dot{f}(x)$. The Chebyshev encoding uses 6 qubits, a hardware-efficient ansatz with three gates and depth 5. The Lagrange encoding uses 8 encoding points, corresponding to 9 qubits. Both are trained using a two-step optimization procedure. . 153

7.18 Comparison of the performance best outcomes of the Chebyshev- and Lagrange-encoded VQAs applied to the complete ODE solution. a) Average circuit count (left axis) and average overall loss (right axis). b) average loss over the interval, plotted on logarithmic scale. The Chebyshev encoding uses 6 qubits, a hardware-efficient ansatz with three gates and depth 5. The Lagrange encoding uses 8 encoding points, corresponding to 9 qubits. Both are trained using a two-step optimization procedure. The Chebyshev-encoded VQA performance plotted exclude the outlier 154

7.19 Average total loss and gate complexity for the Chebyshev-encoded VQA applied to the most efficient VQC structures with varying training set lengths over 5 tries. $N5L2I2$ stands for the structure of 5 qubits and 2 layers across a training set of the second type, while $N4L3I1$ stands for the structure of 4 qubits and 3 layers across a training set of the first type. 159

7.20 Results are presented for each of the most efficient VQC structures, illustrating variability and the best outcome achieved across five attempts for: a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale. $N5L2I2$ stands for the structure of 5 qubits and 2 layers across a training set of the second type, while $N4L3I1$ stands for the structure of 4 qubits and 3 layers across a training set of the first type. 160

7.21 Results of the Lagrange-encoded VQA for both parts of the optimization process, achieved across five attempts. a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale. $I1$ stands for the first type of training set of points while $I2$ stands for the second. 163

7.22 Results comparison between the Lagrange-encoded VQA and the best outcomes from this work version of the Chebyshev-encoded VQA for both types of training sets of points, for : a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale. 164

7.23 DE and BC loss comparison between the Lagrange-encoded VQA and the best outcomes of this work version of the Chebyshev-encoded VQA for both types of training sets of points, plotted on a logarithmic scale. 165

7.24 Gate complexity comparison between the Lagrange-encoded VQA and the best outcomes of this work version of the Chebyshev-encoded VQA for both types of training sets of points. 167

7.25 Interpolation results for the Chebyshev and Lagrange encodings applied to the damped mass-spring system. a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$, and d) the loss function of the differential equation $\mathcal{L}_{DE}(x)$ across the interval. 169

7.26 a) Source term s used for the Poisson equation application. b) Iterative evolution of the Lagrange-encoded solution for Neumann boundary conditions. 172

7.27 a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with Neumann boundary conditions, plotted on a logarithmic scale. 174

7.28 Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the Neumann boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$. 174

7.29 a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with periodic boundary conditions, plotted on a logarithmic scale. 176

7.30 Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the periodic boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$. 176

7.31 a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm.. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with Dirichlet boundary conditions, plotted on a logarithmic scale. 177

7.32 Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the Dirichlet boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$. 177

7.33 Overall loss value of the Lagrange-encoded algorithm applied to the different boundary conditions, plotted on a logarithmic scale. 178

7.34 Gate complexity estimation of the Lagrange-encoded algorithm, averaged over five attempts per boundary conditions, compared with the Sato *et al.* algorithm [5], plotted on a logarithmic scale. a) Basic gates per iteration, b) Total basic gates. 179

7.35 Circuit complexity estimation of the Lagrange-encoded algorithm, averaged over five attempts per boundary conditions, compared with the Sato *et al.* algorithm [5]. a) Circuits per iteration, b) Total Circuits. 181

7.36 Illustration of the different types of points. Red indicates forbidden values x_f, where the DE expression diverges towards $\pm\infty$; blue indicates special values where the DE admits an different solution; green points represent training points in the window $[\pi/6\lambda, \pi/3\lambda]$ modulo π. 184

7.37 Evolution of the solution during training of the first-order discontinuous DE using a Lagrange-encoded VQA with six encoding points: a) function $f(x)$, b) first-order derivative $\dot{f}(x)$. 184

7.38 Evolution of training metrics over the three steps: a) circuit count, b) derivative loss $\mathcal{L}_{DE}$. 185

7.39 Approximation of the solution of the first-order discontinuous DE using a Lagrange-encoded VQA with six encoding points, over five runs: a) function $f(x)$, b) derivative $\dot{f}(x)$. 185

7.40 Circuit count and derivative loss for the five runs of the VQA solution: a) circuit count, b) derivative loss $\mathcal{L}_{DE}$. 186

List of Tables

2.1 Typical performance metrics of superconducting, trapped-ion, and neutral-atom quantum processors, as reported in a 2025 study [37], based on data from [27], [38], [39]. . 24

6.1 Comparison of three Lagrange encoding versions with scaled MSE for $\langle \hat{Z}_0 \rangle$ and $\langle \hat{Z}_1 \rangle$, and improvement factors over Version 1. 114

6.2 Summary of runtime components. Here d denotes the circuit depth, N_{sequence} the number of single-qubit gates per hardware-efficient layer, T_1 and T_2 the execution times of single- and two-qubit gates respectively, T_m the measurement time, and n the number of encoding points. 128

Glossary

DE Differential Equation, an equation involving derivatives of a function with respect to one or more variables.

DMSS Damped Mass Spring System, a classical mechanical system consisting of a mass attached to a spring with damping, often modelled by a second-order differential equation.

DQC Derivative Quantum Circuit, a quantum circuit used to evaluate the derivative of a quantum state with respect to its parameters.

HHL Harrow Hassidim Lloyd quantum algorithm for solving linear systems of equations.

NISQ Noisy Intermediate-Scale Quantum, referring to quantum devices with a moderate number of qubits that are prone to noise.

ODE Ordinary Differential Equation, a DE involving functions of a single independent variable and their derivatives.

PDE Partial Differential Equation, a DE involving functions of multiple independent variables and their partial derivatives.

PSR Parameter Shift Rule, a technique for computing derivatives of parametrised quantum circuits.

QML Quantum Machine Learning, the use of quantum algorithms or quantum-inspired methods to perform machine learning tasks.

VQA Variational Quantum Algorithm, a class of hybrid quantum-classical algorithms that use parameterized quantum circuits and classical optimization.

VQC Variational Quantum Circuit, a parameterized quantum circuit used within variational algorithms.

Chapter 1

Introduction

1.1 Motivations

Solving equations, and differential equations in particular, is central to many areas of science. In aerospace engineering, they are essential to model and understand the dynamic behaviour of physical systems in structural analysis or fluid dynamics. In medicine, climate science, and finance, they appear in studies of pharmacokinetics, weather prediction, or investment risks.

Their solutions capture a wide range of phenomena, yet even numerical approaches can be challenging, especially for the most interesting of cases, which often involve many variables or nonlinear interactions. Research continues to explore new methods for tackling these complex differential equations, particularly nonlinear systems with multiple variables, which remain among the most difficult and captivating problems. As a result, this field remains active in classical computation and, over the last decade, has also attracted interest in quantum computing.

In classical computation, differential equations are typically solved using methods such as finite differences, finite element or volume techniques, or spectral methods based on polynomial approximations. Achieving high accuracy with these approaches often requires a fine spatial discretisation or a large basis set, both of which can incur substantial computational costs. In engineering applications, although the equations usually involve one to three dimensions, the need for a very fine mesh in all spatial directions is common, resulting in significant memory and computational demands.

Quantum computation, by contrast, offers a fundamentally different paradigm. Recent research has focused on exploiting the inherent properties of quantum systems, such as entanglement and superposition, to develop alternative approaches to classical methods, with the aim of achieving computational speedups or enabling novel applications. Quantum computing remains in its early stages. Fully fault-tolerant machines are not yet available, though many imperfect processors already exist. This raises a key question: how can we make practical use of current devices?

Initially, the goal of quantum computing was to operate independently and surpass classical methods. Today, given current hardware limitations, purely quantum approaches are no longer the only focus. Among the most promising strategies are hybrid quantum-classical algorithms, known as Variational Quantum Algorithms (VQAs), which train parameterized quantum circuits using classical optimizers.

This PhD focuses on applying VQAs to solve differential equations. From the outset of this project, these algorithms appeared to be a promising avenue for harnessing both present and future quantum processors. Because VQAs are designed for the current generation of Noisy Intermediate-Scale Quantum (NISQ) devices, they represent an early step toward practical quantum computing. While they do not require fully error-corrected, large-scale machines, their implementation still demands careful management of noise and the use of emerging error mitigation techniques.

The main objective of this research is to study and develop VQAs that are both theoretically grounded and soon practically implementable, bridging the gap between today's hardware and tomorrow's quantum advantage.

1.2 Research Journey

In this work, I studied Variational Quantum Algorithms for solving differential equations.

A review of the literature revealed that much of the existing research focused on discretized approaches for solving differential equations with quantum algorithms, often targeting fault-tolerant quantum computers, which are not available today and may remain out of reach in the foreseeable future. This motivated my focus on quantum algorithms suitable for near-term devices: VQAs.

The primary VQA that captured my attention is the algorithm proposed by Kyriienko *et al.*, due

to its fully spectral approach, which exploits the fact that quantum circuits can be analytically differentiated. Their method solves first-order differential equations using a polynomial-based VQA, employing Chebyshev polynomials and a large set of variational parameters. This work was the first to propose a fully spectral approach based on polynomial approximations of the solution, revealing both a promising framework and an opportunity for further contributions.

Initially, my research aimed to extend the Kyriienko *et al.* VQA to higher-order equations and a broader class of differential equations. However, further study revealed several limitations of the algorithm, including high computational demands and difficulties in representing smooth functions, which manifested as noticeable oscillations in the solution.

These observations motivated the exploration of strategies to reduce computational cost and mitigate oscillations, which were limiting the precision and performance of the VQA. I first considered reducing the number of variational parameters and computing gradients via finite differencing. However, these strategies did not lead to significant improvements in the Chebyshev-encoded VQA for second-order differential equations.

Consequently, I investigated a more fundamental aspect of the algorithm: the encoding itself. This led to the development of a new spectral VQA based on an alternative polynomial encoding, designed to improve both computational efficiency and solution accuracy: the Lagrange-encoded VQA.

1.3 Objectives

The objectives of this PhD are:

- Study and understand the potential application of quantum algorithms to problems of relevance in engineering;
- Explore the use of variational quantum algorithms for solving differential equations, as they are foundational across many scientific and industrial domains;
- Extend existing algorithms to broader classes of differential equations, and investigate how their structure could be optimised;

- Develop a new quantum algorithm to reduce quantum circuit depth and complexity, in order to improve practical implementability;
- Build a foundation for future work on noise-aware implementations and potential integration with fault-tolerant architectures.

1.4 Contributions

The novel technical contributions presented in this thesis include:

- **Extension of the Kyriienko *et al.* (2021) variational quantum algorithm framework to second-order differential equations.** The original method proposed by Kyriienko *et al.* was limited to first-order differential equations. This work generalizes the approach to encompass second-order systems by reformulating the cost function and ansatz structure, opening the door to a broader class of physically relevant problems such as harmonic oscillators.
- **Design of a novel variational quantum algorithm based on Lagrange polynomial encoding, inspired by Hadamard test differentiation techniques.** This algorithm introduces an original encoding strategy using Lagrange basis polynomials to approximate the solutions of differential equations. The method draws inspiration from the Hadamard test, primarily for gradient computation and secondarily as a conceptual tool for constructing efficient quantum circuits with a minimal number of gates and straightforward encoding function.
- **Development of a complexity evaluation framework for variational quantum algorithms based on gate count and circuit count.** This framework introduces two key metrics: the gate count, which estimates the total number of quantum operations required to implement a given algorithm, and the circuit count, which quantifies the number of distinct quantum circuits that must be executed to evaluate the cost function and its gradients. While these quantities do not yet translate directly into physical running times due to current hardware and compilation variability, they serve as proxies for quantum-classical communication overhead and algorithmic scaling. Together, they enable a principled comparison between different VQAs, offering early insight into the feasibility and efficiency of various quantum strategies under real-world constraints.

Presentations

- **Poster**: "Quantum Algorithms Applied to Engineering Problems", Imperial College of London Aero-showcase, London, UK, 5th July 2022.
- **Poster**: "Quantum Algorithms Applied to Engineering Differential Equations", Quantum Techniques in Machine Learning Conference hosted by Università degli Studi di Napoli Federico II, Naples, Italy, 7th November 2022.
- **Poster**: "Quantum Algorithms Applied to Engineering Differential Equations", Imperial College of London, Women in Engineering, London, UK, 20th June 2023.
- **Poster**: "Variational Quantum Algorithms to solve Poisson Equation", Quantum Techniques in Machine Learning Conference hosted by CERN, Geneva, Switzerland, 19th November 2023.
- **Poster**: "Solving DEs with Variational Quantum Algorithms", Imperial College of London Aero-showcase, London, UK, 4th September 2024.
- **Presentation**: "Variational Quantum Algorithms to solve Differential Equations", QEVEC / QuANDiE Workshop hosted by Cardiff University, Cardiff, Wales, 7th March 2025.
- **Presentation**: "Variational Quantum Algorithms to solve Differential Equations", International Conference on Applied Quantum Methods in Computational Science and Engineering hosted by RWTH Aachen University, Aachen, Germany, 9th October 2025.

Publications

The content of Chapters 4, 7 are based on the paper:

Variational quantum algorithm based on Lagrange polynomial encoding to solve differential equations. Josephine Hunout, Sylvain Laizet, and Lorenzo Iannucci. Phys. Rev. A, 111, 062404. https://doi.org/10.1103/PhysRevA.111.062404 [1].

Alongside my PhD research, I led the research, analysis, and writing of two Nature Reviews Physics Comments as part of the Center for Quantum Engineering, Science and Technology (QuEST) at Imperial College of London. The first article analyzed the recruitment landscape of the United Kingdom

quantum sector and evaluated the extent to which current hiring trends support the objectives of the National Quantum Missions [2]. The second article examined quantum teaching at Imperial and across the country and showed that although substantial quantum content already exists, its limited visibility on public platforms shapes the range of students who recognise these opportunities and may therefore enter the field [3]. Taken together, these studies trace how workforce development and educational visibility interact, and how both factors will influence the composition and readiness of the next generation of quantum specialists.

1.5 Thesis Outline

This thesis is structured into the following chapters. Each chapter concludes with a summary highlighting the main points and takeaways.

- **Chapter 2: Quantum Computing Context** This chapter introduces the main concepts of quantum computing, covering both the theoretical foundations and the computational properties that define the quantum paradigm. It also reviews the current development of quantum hardware, its challenges and limitations, and addresses the overarching question driving the field: What is the objective of quantum computing?

- **Chapter 3: Literature Review**
  This chapter presents a comprehensive review of differential equations, their importance, and the challenges they pose, along with current research on quantum computing approaches to these problems. Several VQAs are introduced and discussed, including their mechanisms, advantages, and limitations. In particular, the spectral approach of Kyriienko *et al.* [4], which inspired this work, and the discretized approach of Sato *et al.* [5], which serves as a comparison, are analysed in detail.

- **Chapter 4: Variational Quantum Algorithms Based on Polynomial Encoding**
  This chapter focuses on the specific class of VQAs studied in this work: polynomial-encoded, fully spectral algorithms for solving differential equations. It details the definition of each component required for these VQAs, presenting the Chebyshev-encoded algorithm of Kyriienko *et al.*, its

extension to higher orders developed in this work and the novel Lagrange-encoded VQA proposed in this thesis.

- **Chapter 5: Analysis of Polynomial-based Variational Quantum Algorithms**

  This chapter examines the algorithms at the circuit level and from a theoretical perspective. It explains how the encoding operates and how the quantum circuits are differentiated, for the Chebyshev and Lagrange-encoded VQAs.

- **Chapter 6: Quantum Circuit Simulation and Complexity Estimation**

  This chapter addresses the practical feasibility of implementing these VQAs. It examines the challenges of simulating quantum algorithms, including sampling and measurement issues, and highlights how algorithm design must account for these factors to remain implementable on real devices. These considerations lead to improvements in the Lagrange-encoded VQA. Furthermore, NISQ-specific challenges such as runtime and noise are analysed and quantified through circuit depth and gate count metrics.

- **Chapter 7: Results**

  This chapter presents the application of the Lagrange-encoded VQA to several differential equations: linear first-order ODEs, second-order ODEs, and linear first-order DEs with nonlinear coefficients. Results are compared to the two other VQAs discussed in Chapter 3: the spectral approach of Kyriienko *et al.* [4] and the discretized approach of Sato *et al.* [5].

- **Chapter 8: Conclusions and Future Work**

  The final chapter summarizes the key findings of this thesis, highlights its main contributions, and outlines potential directions for future research to further improve the Lagrange-encoded VQA and explore its applications.

Chapter 2

Quantum Computing Context

Before presenting the details of this PhD research, it is essential to provide some context, beginning with a question that continues to shape the trajectory of the field:
What are we really trying to achieve with quantum computing?

2.1 The objectives of Quantum Computing

In the early days of quantum computing, the community set its sights on a bold and provocative objective: achieving what was defined as "*Quantum Supremacy*". This term, introduced by Preskill in 2012 [6], referred to the possibility that a quantum computer might one day solve a problem super-polynomially faster than any classical counterpart. The claim took centre stage in 2019, when Google published a study in *Nature* asserting that its 53-qubit processor, Sycamore, had reached that milestone [7]. Their quantum processor had performed a sampling task (Random Circuit Sampling) based on a pseudo-random quantum circuit, which, they argued, would require approximately 10,000 years for a classical supercomputer to reproduce.

However, the celebration was short-lived. IBM swiftly challenged this assertion [8], demonstrating that the same task could in fact be simulated on a classical system in just 2.5 days with higher fidelity. Beyond the technical rebuttal, IBM raised a deeper concern regarding the term "Quantum Supremacy". It implied, misleadingly, that quantum machines would entirely replace classical computing. This misconception, they warned, risked distorting the public and scientific discourse. Given

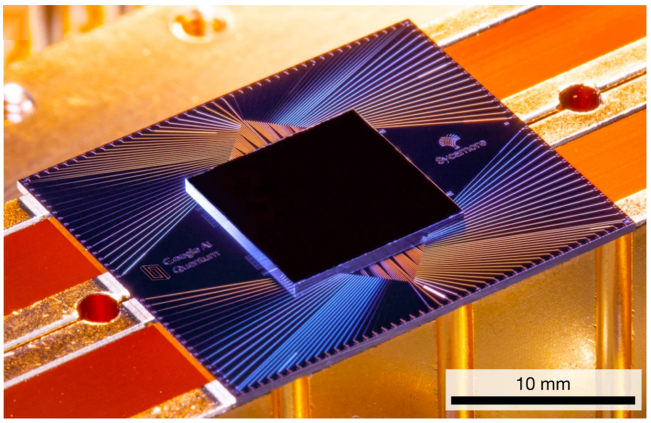


Figure 2.1: Photograph of the Sycamore chip [7].

the subtlety of benchmarking and the complexity of comparing fundamentally different computational paradigms, any claim of definitive supremacy demanded rigorous scrutiny.

Indeed, the comparison between classical and quantum approaches is fraught with ambiguity. Google's demonstration showcased mainly quantum resources, such as entanglement and superposition, yet it downplayed the vast arsenal of classical capabilities, including hierarchical memory architectures, high-precision hardware, mature software frameworks, and decades of algorithmic development [8].

In light of these debates, the field has undergone a conceptual shift. The term "*Quantum Advantage*" has emerged as a replacement, partly to avoid the historical and semantic weight of "supremacy" [9]. But more importantly, it signals a maturing vision. The ambition is no longer to dominate classical computing, but rather to identify problems where quantum algorithms can offer a genuine, practical benefit, collaborating with classical systems. This reorientation reflects a deeper understanding of what the future of computation may realistically look like.

2.2 Computational promises of Quantum Computing

2.2.1 Quantum Computing paradigm

Quantum computing is an emerging field that leverages the principles of quantum mechanics to process information in fundamentally new ways. Central to this framework are the concepts of the qubit, superposition, and entanglement, each foundational to the operation and potential of quantum computational systems.

Qubit

A "quantum bit", or "*qubit*", serves as the fundamental unit of quantum information. Unlike a classical bit, which exists definitively in one of two states (0 or 1), a qubit resides within a two-dimensional Hilbert space and can exist in a superposition of basis states, as shown in the figure 2.2. Mathematically, the state of a qubit can be expressed as:

$$|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

where $|0\rangle$ and $|1\rangle$ denote the computational basis states, and $\alpha, \beta \in \mathbb{C}$ are complex probability amplitudes satisfying the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

Upon measurement in the computational basis, the qubit is projected into to state $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$. This probabilistic nature is governed by the Born rule and underscores the departure of quantum systems from deterministic classical behaviour.

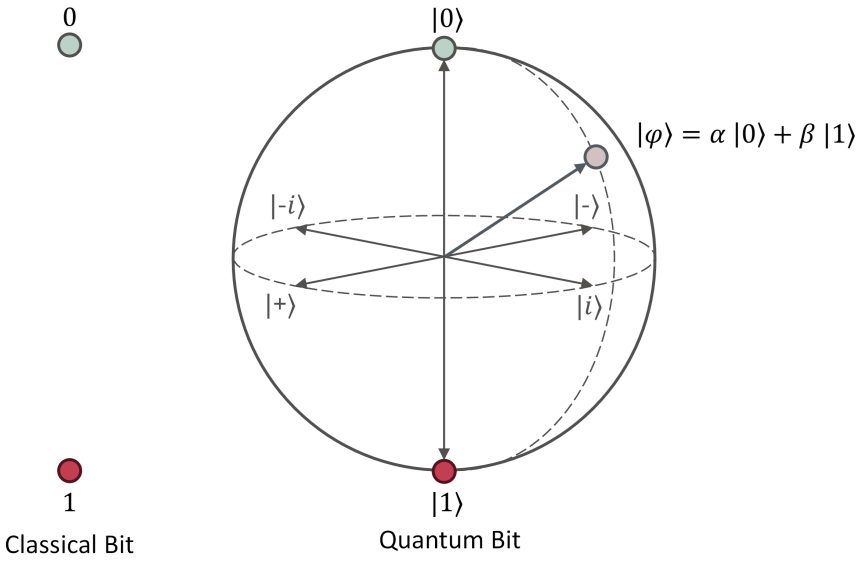


Figure 2.2: Visualisation of a classical bit and a qubit .

Bloch Sphere

A pure quantum state of a qubit and their transformations in the three basis can be represented using the Bloch sphere, a unit sphere in $\mathbb{R}^3$ as illustrated in the figure 2.2, where the three axes are the Pauli axes:

- The axis with $|0\rangle$ and $|1\rangle$ is the Pauli Z axis, which refers to the computation basis. A measurement in that basis is called a Z measurement.
- The axis with $|+\rangle$ and $|-\rangle$ is the Pauli X axis,
- and the axis with $|+i\rangle$ and $|-i\rangle$ is the Pauli Y axis.

The Bloch sphere not only illustrates the concept of superposition but also provides insight into the effects of quantum gates. For instance, single-qubit gates correspond to rotations of the state vector around specific axes of the sphere. This visualization aids in understanding the dynamics of qubit manipulations and the design of quantum algorithms. While pure states are represented as points on the surface, mixed states (statistical mixtures of pure states) are represented by points inside the sphere. The centre of the sphere corresponds to a completely mixed state, where the qubit has equal probabilities of being in $|0\rangle$ or $|1\rangle$.

Superposition

Superposition is a fundamental principle of quantum mechanics, asserting that a quantum system can exist simultaneously in multiple states until an observation collapses it into one of the possible configurations. For a qubit, this means it can be in a linear combination of $|0\rangle$ and $|1\rangle$, as shown above. This property allows quantum computers to process a vast number of potential outcomes concurrently, providing a form of parallelism that is exponentially more powerful than classical systems for certain computational problems.

Entanglement

Entanglement is a uniquely quantum mechanical phenomenon in which the states of two or more qubits become interdependent such that the state of each qubit cannot be described independently of the others, regardless of the spatial separation between them. For example, consider the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.3}$$

In this entangled state, measurement of one qubit instantaneously determines the state of the other, a non-classical correlation that defies classical intuitions about locality and separability. Entanglement is a critical resource in quantum computing, enabling phenomena such as quantum teleportation and is essential for the operation of many quantum algorithms.

Quantum Measurement

When a quantum state is measured, it collapses into a definite classical value. In the computational basis, the definite values would be either $|0\rangle$ or $|1\rangle$. As presented before in the qubits definition, this collapse is inherently probabilistic: the probability of each outcome is determined by the square of the amplitude of the state's projection onto the measurement basis. To retrieve or guess what the superposition was, quantum measurements are typically repeated multiple times, a process referred to as executing multiple shots. Each shot involves preparing the same quantum state, performing the measurement, and saving the outcome. By aggregating the results over many shots, the probabilities $|\alpha|^2$ and $|\beta|^2$ can be statistically estimated. However, a projective measurements in the Z-basis alone provide information only about the magnitudes $|\alpha|$ and $|\beta|$ of the superposition coefficients, not their relative phase.

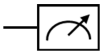


Figure 2.3: Symbol of the quantum measurement gate.

While measurements are often performed at the conclusion of quantum circuits to read out results, they can also be strategically integrated during computation, notably in quantum error correction techniques.

2.2.2 Quantum gates and circuits

Quantum gates are the basic units of quantum circuits, performing reversible unitary operations on one or more qubits. Represented by unitary matrices, they manipulate qubit statevectors and are applied in sequence over a quantum register, forming a quantum circuit.

Pauli Gates

The Pauli gates are a set of single-qubit quantum gates represented by the Pauli matrices. They correspond to 180-degree rotations about the respective axes of the Bloch sphere. These gates are Hermitian and unitary, with eigenvalues of ± 1. They form the basis for more complex quantum operations.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.4}$$

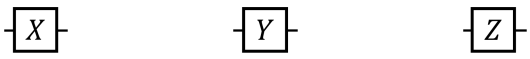


Figure 2.4: Symbols of the quantum Pauli gates X, Y and Z.

Clifford Gates

Clifford gates are a set of quantum gates that normalize the Pauli group, meaning they map Pauli operators to other Pauli operators under conjugation. These gates are are essential in quantum error correction and can be efficiently simulated on classical computers due to the Gottesman-Knill theorem. The Clifford group is generated by the Hadamard (H), Phase (S), and Controlled-NOT (CNOT) gates. The Controlled-NOT gate is a two-qubit gate that flips the state of the target qubit if and only if the control qubit is in the $|1\rangle$ state and is instrumental in creating entanglement between qubits.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.5}$$



Figure 2.5: Symbols of the quantum Clifford gates H, S and CNOT.

Rotational Gates

Rotational gates perform rotations of qubit states around the Bloch sphere axes by a specified angle θ. These gates are crucial for implementing arbitrary single-qubit rotations and are fundamental in constructing universal quantum gate sets. They are defined as:

$$R_X(\theta) = \exp\left(-i\frac{\theta}{2}X\right) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \tag{2.6}$$

$$R_Y(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \tag{2.7}$$

$$R_Z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.8}$$

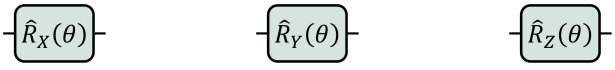


Figure 2.6: Symbols of the quantum rotational gates R_Y, R_Y and R_Z.

Example of the Bell State

Circling back to the example of the Bell state $|\Phi^+\rangle$ introduced earlier to illustrate entanglement, it can be generated with a simple two qubit circuit. Starting from a quantum state initialised on $|00\rangle$, a Hadamard gate is applied to the first qubit, followed by a CNOT_{01} gate, meaning that the first qubit acts as the control and the second as the target. In this circuit, the quantum state is built following:

$$\begin{aligned} \text{CNOT}_{01}(H \otimes I)\,|00\rangle &= \text{CNOT}_{01}\big(H\,|0\rangle \otimes |0\rangle\,\big) \\ &= \text{CNOT}_{01}\big(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle\,\big) \\ &= \frac{1}{\sqrt{2}}\,(|00\rangle + |11\rangle) \\ &= |\Phi^+\rangle\,. \end{aligned} \tag{2.9}$$

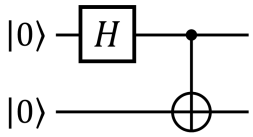


Figure 2.7: Quantum circuit used to create the Bell state $|\Phi^+\rangle$, starting from a 2-qubit system initialised in $|00\rangle$ and applying the unitary $\text{CNOT}_{01}, (H \otimes I)$.

Remark. In equations, the right most matrix acts first, however, quantum circuit diagrams follow the opposite convention, where the left most gate acts first.

2.2.3 Quantum algorithms demonstrating a potential quantum advantage

Finally, the promises of quantum computation come from leveraging the quantum properties such as superposition and entanglement to reimagine problem-solving strategies, aiming for significant speed-ups over classical methods. Over the last 30 years, few fully quantum algorithms have been proposed that theoretically surpass their classical counterparts. While demonstrating promising theoretical advantages, the practical implementation of these algorithms remains constrained by the current limitations of quantum hardware. The following section provides an overview of three seminal quantum algorithms: Shor's algorithm [10], Grover's algorithm [11], and the Harrow-Hassidim-Lloyd (HHL) algorithm [12].

Shor's Algorithm

Among all quantum algorithms that have demonstrated a potential computational advantage, Shor's algorithm is arguably the most impactful. Proposed by Peter Shor in 1994 [10], it addresses the problem of integer factorisation, decomposing a composite number into its prime factors. This breakthrough not only challenged long-standing assumptions in classical cryptography but also provided the first concrete example where a quantum computer could significantly outperform its classical counterpart on a useful real world problem.

Classically, factoring large integers is a computationally demanding task, with the most efficient known algorithms, such as the general number field sieve, operating in sub exponential time. In contrast, Shor's algorithm achieves this in polynomial time, representing an exponential speed up over classical approaches. This capability has profound implications for cryptography, particularly regarding the RSA encryption scheme, which relies on the presumed hardness of factoring large integers.

The efficiency of Shor's algorithm originates from its quantum subroutine based on the Quantum Phase Estimation (QPE). This procedure estimates the phase associated with an eigenstate, or a superposition of eigenstates, of a unitary operator $\hat{\mathcal{U}}$, defined through $\hat{\mathcal{U}}\left|\psi\right\rangle = e^{2i\pi j}\left|\psi\right\rangle$ and encodes an approximation of j into a binary representation in the control register by applying the inverse Quantum Fourier Transform (QFT).

The Quantum Fourier Transform is the quantum analogue of the classical discrete Fourier transform. It performs a change of basis from the computational basis $\left(\left|0\right\rangle, \left|1\right\rangle\right)^{\otimes n}$ to the Fourier basis. Formally, the QFT acts on a computational basis state $\left|x\right\rangle$, where $x \in \{0, \ldots, N-1\}$, as follows:

$$\text{QFT} : \left|x\right\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{xk} \left|k\right\rangle, \quad \text{with} \quad \omega_N = e^{2\pi i/N}. \tag{2.10}$$

Shor's algorithm proceeds in two main stages. In the first stage, the integer factorisation problem is classically reduced to an order finding problem. Given a composite integer N to be factorised and an integer $a \in \{2, \ldots, N-1\}$ that is coprime with N, there exists a positive integer r, the multiplicative order of a modulo N, such that $a^r \equiv 1 \mod N$, with r defined as the smallest integer satisfying this congruence.

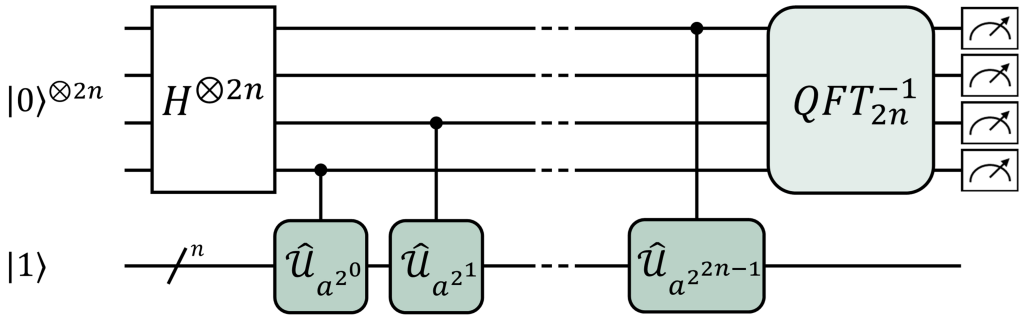


Figure 2.8: Quantum subroutine in Shor's algorithm.

The second stage consists of determining the order r, which is achieved by the quantum subroutine of Shor's algorithm, illustrated in Figure 2.8. This subroutine applies QPE to the unitary operator $\hat{\mathcal{U}}_a$, defined by $\hat{\mathcal{U}}_a |x\rangle = |ax \mod N\rangle$, acting on the quantum state $|1\rangle$ encoded on n qubits, with $n = \lceil \log_2 N \rceil$. The circuit therefore employs substantial qubit resources and considerable circuit depth, with a control register of size $2n$ qubits and a work register of size n qubits.

For any integer $k \in \mathbb{N}^*$, the unitary satisfies $\hat{\mathcal{U}}_a^k |x\rangle = |a^k x \mod N\rangle$ and, in particular, $\hat{\mathcal{U}}_a^r |x\rangle = |a^r x \mod N\rangle \equiv |x \mod N\rangle$, which implies the eigenvalues are the r-th roots of unity. When applied to the state $|1\rangle$, which belongs to the invariant subspace generated by the eigenstates of $\hat{\mathcal{U}}_a$, the QPE outputs an estimate of the phase of the form $\frac{s}{r} 2^{2n}$, with s a random integer in $\{0, \ldots, r-1\}$. The value of r can then be recovered through classical post processing, typically using the continued fractions algorithm.

If r is even, the non-trivial solution of the factorisation problem is $g = \gcd(a^{r/2} - 1, N)$ and N/g. In the case where r is odd, the algorithm does not yield to a non-trivial factor and must be repeated with a different coprime integer a.

The quantum subroutine of Shor's algorithm requires deep, coherent quantum circuits and highly precise gate operations, rendering the algorithm currently infeasible on Noisy Intermediate-Scale Quantum (NISQ) devices. Nevertheless, Shor's algorithm remains a leading candidate for achieving quantum advantage in the era of fault-tolerant quantum computing. In this regime, machines equipped with thousands of stable, error-corrected qubits would be capable of executing such routines reliably and at scale. Other quantum algorithms that rely on the Quantum Fourier Transform or quantum phase estimation are also expected to become practical under the assumption of fully fault-tolerant quantum hardware.

As mentioned previously, the original formulation of Shor algorithm requires at least $3n$ qubits, which represents a substantial resource overhead. As a result, reducing the qubit count has remained an active area of research. Several works have demonstrated that the algorithm can be implemented using $2n$ plus a small number of ancillary qubits [13], [14], [15], and even as few as $1.5n$ qubits [16]. The most advanced result to date is due to Chevignard *et al.*, who prove that Shor algorithm can be executed using $(0.5 + \epsilon)n$ logical qubits [17], which implies that factoring a 2048 bit semiprime would require approximately 1730 logical qubits.

Grover's Algorithm

Two years later, in 1996, Lov Grover introduced a quantum algorithm that speeds up unstructured search from $\mathcal{O}(N)$ to $\mathcal{O}(\sqrt{N})$ queries [11]. Rather than accessing data directly, it uses an oracle $\hat{\mathcal{U}}_\omega$ to identify solutions, turning search into a constraint satisfaction problem.

The algorithm amplifies the probability amplitude of the correct solution through repeated applications of the Grover operator, a combination of the oracle and a diffusion operator. For a search space of size $N = 2^n$, the algorithm requires n qubits to represent the superposition of all possible states, plus ancillary qubits for the oracle and diffusion operations, typically resulting in a total qubit count on the order of $n + \mathcal{O}(1)$. It is optimal for black-box search problems and has found applications in cryptanalysis, optimization, and machine learning [18], [19].

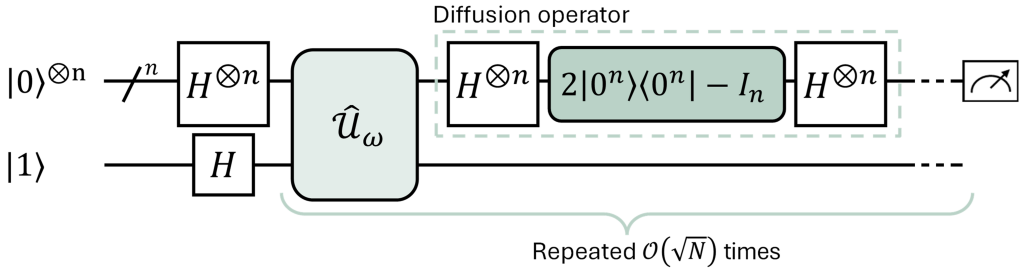


Figure 2.9: Quantum subroutine in Grover's algorithm.

However, Grover's quadratic speed-up, while meaningful in theory, is often too small to outweigh the noise and overhead of current quantum hardware. Efficient oracle construction remains problem-specific and non-trivial, and will mainly be useful as a subroutine where the main task provides the oracle. As a result, Grover's algorithm is better suited for future fault-tolerant machines than for today's NISQ devices.

HHL Algorithm

Another quantum algorithm demonstrating theoretical quantum advantage is the Harrow-Hassidim-Lloyd (HHL) algorithm [12]. It solves linear systems $A\vec{x} = \vec{b}$ by preparing a quantum state proportional to the solution vector $\vec{x}$, achieving exponential speed-up under certain conditions:

- Sparse: Most elements of A are zero, enabling efficient storage and computation.
- Well-conditioned: The condition number $\kappa(A)$, defined as the ratio between the largest and smallest singular values of A, is low. This ensures numerical stability and reliable solutions.
- Efficiently implementable: The unitary operator e^{iAt} can be simulated with a manageable number of quantum gates.
- Hermitian: The matrix satisfies $A = A^\dagger$, meaning it equals its conjugate transpose.

In many practical cases, A is not Hermitian. To handle this, A can be embedded into a larger Hermitian matrix of size $2N \times 2N$:

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}, \tag{2.11}$$

which allows applying the HHL algorithm unchanged. This transformation doubles the system size but, interestingly, only increases the qubit requirement by one, preserving the overall scaling.

The HHL circuit involves several quantum registers: one n-qubit register to encode $\vec{b}$, another n-qubit register for quantum phase estimation, and an ancilla qubit for postselection through a controlled rotation. Thus, for a system size $N = 2^n$, the minimal qubit count is approximately $2n + 1$. If A is non-Hermitian and the embedding is used, one additional qubit is required, raising the total to about $2n + 3$ qubits.

Despite its promise, the algorithm presents several practical challenges. Preparing the input state $|b\rangle$ is nontrivial unless $\vec{b}$ has a simple structure. The core subroutine, based on quantum phase estimation and Hamiltonian simulation, also requires fault-tolerant circuits. Finally, the output is a quantum state, accessing the full solution vector would require repeated measurements, which should be taken into account in the evaluation of the overall speed-up.

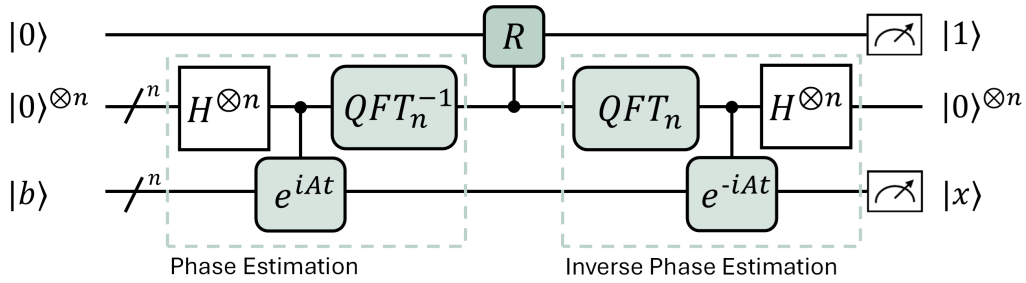


Figure 2.10: Quantum subroutine in the HHL algorithm.

These limitations, as noted by Aaronson [20], mean that HHL is unlikely to be practical on NISQ devices. Nonetheless, it remains foundational and has inspired many other approaches including in solving differential equations, described Section 3.3.

2.3 Technical challenges of Quantum Computing

Despite the promising theoretical quantum advantage demonstrated by previously presented quantum algorithms, current quantum hardware is yet not ready for practical implementation and large-scale deployment. When addressing the technical challenges of building the quantum computers of tomorrow, the primary focus lies on scaling up the number of qubits, improving gate fidelity, suppressing noise, as well as managing power consumption, thermal constraints, and integration with classical electronics.

This distinction underpins the difference between tomorrow's fault-tolerant quantum computing (FTQC) and current Noisy Intermediate-Scale Quantum (NISQ) devices.

2.3.1 Fault-tolerant quantum computing versus Noisy Intermediate-Scale Quantum systems

Currently available quantum devices are predominantly classified as Noisy Intermediate-Scale Quantum systems. These processors operate with on the order of 50 to several hundred qubits, but remain highly susceptible to errors arising from decoherence, imperfect gate operations, and cross talk between qubits. Such errors are particularly difficult to mitigate on NISQ devices, as quantum error correction requires both a substantial overhead in additional qubits and sufficiently low physical er-

ror rates. In practice, the cost of implementing error correction on noisy hardware often exceeds its benefits, rendering fault tolerant operation infeasible on present day NISQ machines.

This stands in contrast with the theoretical model of fault tolerant quantum computing, in which logical qubits are encoded into large numbers of physical qubits using quantum error correction codes to suppress noise. Provided that the physical error rate remains below a critical threshold, known as the threshold theorem, the application of quantum error correction allows the logical error rate to be reduced to arbitrarily low values [21]. For commonly studied error correction schemes, this threshold is typically estimated to be on the order of 1% [22], [23].

Recently, physical error rates low enough to enable the first demonstrations of QEC have been achieved for a limited number of hardware platforms. For less mature technologies, such as trapped ions and neutral atoms, qubit counts remain comparatively small. In contrast, more mature platforms with larger qubit numbers, notably superconducting qubits, have recently demonstrated successful error correction, as illustrated by the Google quantum processor Willow [24].

2.3.2 Quantum Noise

Quantum noise refers any element that disturb a quantum system and causes errors in computations. Quantum information is very fragile and can be affected in many ways: hardware sensitivities like thermal fluctuations or electromagnetic interference, imperfect quantum gates, or interactions with the environment. The main types of errors are:

- **Bit-flip errors (X errors)**: the qubit flips from $|0\rangle$ to $|1\rangle$ (or the other way around).
- **Phase-flip errors (Z errors)**: the phase of the qubit flips, from $|+\rangle$ to $|-\rangle$ (or the other way around).
- **Bit-phase-flip errors (Y errors)**: a mix of bit and phase flips.
- **Leakage errors**: the qubit escapes the computational space, which normal error correction cannot fix.
- **Cross-talk and correlated errors**: when acting on one qubit accidentally affects its neighbors, often spatially connected [25], [26].

Additionally, quantum gates are imperfect, meaning their fidelity is not exactly 100%. However, in the last few years, the gate fidelity has improved and is now up to 99,999% for a single qubit gate and 99,8% on a two-qubit gate for a trapped-ion quantum processor [27]. Finally, quantum processors have a limited coherence time, meaning the quantum circuit must be computed before decoherence destroys the quantum information.

Once decoherence occurs, qubits lose their superposition and entanglement, and the information encoded in their quantum states is lost, which strongly limits the depth and complexity of quantum circuits that can actually be run. Even small amounts of noise can induce decoherence, which place quantum decoherence as a major challenge for achieving FTQC [28].

Quantum Error Correction

Achieving fault tolerance requires implementing quantum error correction, which encodes one logical qubit into many physical qubits to detect and correct errors. Among error-correcting codes, the surface code is the most studied, particularly suitable for two-dimensional qubit arrays with nearest-neighbor coupling. It features a high error threshold (around 1%), making it a leading candidate for scalable fault tolerance [29].

Importantly, quantum error correction can only be effective if the underlying physical qubits already possess sufficiently low error rates. When physical error rates exceed the fault-tolerance threshold (roughly 1%), error correction amplifies noise instead of reducing it [29], [30].

In 2023, Google reported the first experimental realization of a surface code demonstrating that logical qubit performance improves as the code size increases [31]. This experiment showed for the first time that error correction can provide a practical benefit.

Quantum Error Mitigation

Quantum noise can also be mitigated, through post-processing or circuit modification, without additional qubits and overhead imposed by full QEC, allowing meaningful results from NISQ devices [32], [33], [34], [35]. Main approaches include:

- **Zero-noise extrapolation**: runs circuits at different noise levels and extrapolates results to an ideal zero-noise case.

- **Probabilistic error cancellation**: reconstructs the inverse noise map through quasi probability decomposition.

- **Measurement error mitigation**: corrects readout errors to improve measurement accuracy.

2.3.3 Current Quantum Hardware Technologies

Since the first quantum computer, a 2-qubit nuclear magnetic resonance device invented in 1998 [36], many other quantum hardware systems have been developed and studied, all sharing the common goal of researchers and industry actors to build a fault-tolerant large-scale quantum computer. Today's quantum hardware platforms differ significantly in their qubit implementations, coherence times, scalability objectives, and operational fidelities. Each technology faces limitations related not only to qubit number and quality, but also to power consumption, thermal management, and integration with classical control electronics, all critical for building large-scale quantum processors.

In the context of solving nonlinear differential equations, the three most promising quantum hardware technologies seem to be the trapped-ion, superconducting, and neutral-atom quantum processors, for being the most advanced quantum technologies for their number of qubits and gate fidelity [37].

The recent typical performances of this hardware types are summarized in the Table 2.1.

Table 2.1: Typical performance metrics of superconducting, trapped-ion, and neutral-atom quantum processors, as reported in a 2025 study [37], based on data from [27], [38], [39].

	Superconducting	Trapped-ion	Neutral-atom
Qubit Lifetime	$\sim 100\mu$s	Several minutes	Several seconds
Single-qubit gate fidelity	0.9999	0.999999	$0.996 - 0.999$
Two-qubit gate fidelity	0.99	0.998	$0.955 - 0.995$
Gate execution time	$\sim 10 - 100$ns	$\sim 10 - 100\mu$s	400ns to 2μs
Connectivity	$4:1$	$40:1$	$10:1$ to $20:1$
Physical qubits	~ 1000	~ 40	~ 1000

Superconducting Qubits

Superconducting qubits are fabricated using Josephson junctions integrated into superconducting circuits and are controlled by microwave pulses. These devices enable rapid gate operations on the order of tens of nanoseconds and benefit from established semiconductor fabrication techniques, allowing scalability to over 100 qubits. However, coherence times remain limited to microseconds which corresponds to only hundreds of two-qubit gate operations, and operation demands dilution refrigeration at millikelvin temperatures, presenting significant engineering challenges.

The development of superconducting quantum hardware has been driven by major industrial players, notably IBM, a pioneer of gate based quantum processors, and D-Wave in the context of quantum annealing. Google Quantum AI entered the field more recently with the *Sycamore* processor in 2019, introduced earlier in Figure 2.1. In 2024, Google announced Willow, a 105-qubit superconducting processor claimed to operate below the quantum error correction threshold [24]. They repeated the Random Circuit Sampling (RCS) benchmark used with *Sycamore*, completing it in five minutes. They estimated this would take over 10 septillion years on the fastest classical supercomputers [40]. The result is technically impressive but remains open to interpretation. The RCS task, while classically hard, is not practically useful [41]; it is not yet clear how these achievements translate to solving real-world problems.

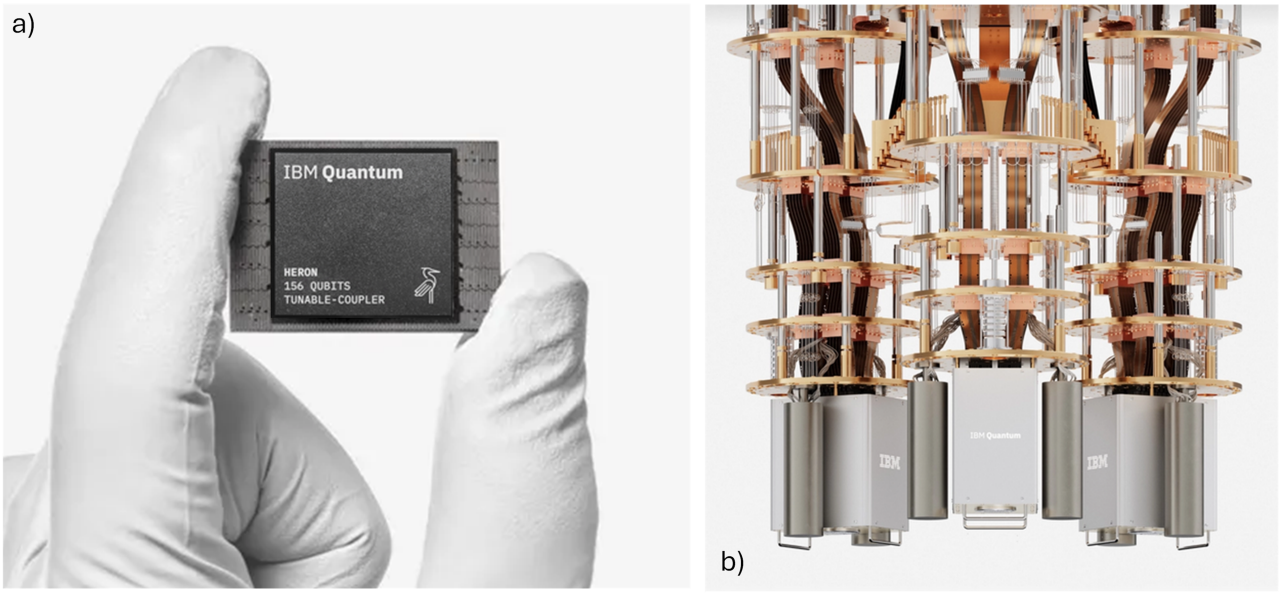


Figure 2.11: a) Photograph of the IBM *Heron* processor [42]. b) Photograph of the IBM *Quantum system two* quantum computer [42].

But Google is far from alone in this race, other companies, such as IBM and Rigetti, are also advancing superconducting quantum processors. IBM made headlines in 2023 by surpassing the symbolic 1000-qubit barrier with its *Connor* processor [43]. In 2024, they introduced *Heron*, a 156-qubit device they describe as their most performant to date. At the time of writing this thesis, three Heron processors are integrated within the IBM Quantum System Two architecture [42].

Trapped Ion Qubits

Trapped ion qubits are based on individual ions confined in electromagnetic traps within vacuum environments, manipulated by laser pulses. Main industrial actors are Quantinuum, with their 56 fully-connected qubit processor *System Model H2* released in 2023 [44], and IonQ with the *Forte Enterprise* 36-qubit fully-connected processor [45]. These processors present exceptionally long coherence times of up to several minutes and support high-fidelity gates with all-to-all connectivity. The primary limitations are slower gate speeds compared to other quantum processors and the complexity of the laser control infrastructure required for scaling. As shown in Table 2.1, trapped ion qubits exhibit very long coherence times, corresponding to several million two-qubit gate operation times.

Neutral Atom Qubits

Neutral atom qubits are individual atoms trapped in tweezer arrays or optical lattices, manipulated by laser fields. They offer high scalability and long coherence times, with recent advances enabling processors with hundreds or even thousands of qubits. For instance, Pasqal's *Orion Gamma* processor (2025) contains over 140 qubits [46], while Atom Computing's *AC1000* features 1200 physical qubits with all-to-all connectivity [47]. Finally, similarly to trapped ion qubits, neutral atom qubits exhibit coherence times corresponding to several million two qubit gate operation times.

Other quantum hardware technologies

In recent years, several other quantum processor technologies have emerged. These approaches are promising but remain at an early stage, with limited qubit counts and immature hardware.

Silicon spin qubits These qubits use the spin of electrons in silicon quantum dots, benefiting from compatibility with existing semiconductor fabrication and relatively long coherence times. Recent demonstrations, such as Grover's algorithm on three-qubit silicon devices [48], highlight the progress of this technology. Intel is a major player in this field, with their 12-qubit processor *Tunnel Falls* [49].

Photonic qubits Photonic quantum processors encode information in single photons travelling through optical circuits. Operating at room temperature for most components, this technology is highly promising due to low decoherence and potential for scalable, integrated photonic chips. However, the manipulation and entanglement of photons remain technically challenging, as they require precise control of optical elements, and photon losses during propagation through fibers or other media can significantly degrade the performance of quantum operations over long distances [50], [51].

2.4 Quantum Circuit Runtimes

The execution time of a quantum circuit depends primarily on the number of sequential operations, often referred to as "steps", required to implement the entire circuit. In practice, multiple gates can be applied in parallel, provided that they act on different qubits and that all relevant qubits are in the appropriate state. Traditionally, circuit runtime has been approximated using the circuit depth expressed in terms of basic gates. However, the actual execution time is influenced not only by the algorithm itself but also by the hardware architecture and the qubit connectivity.

Estimating and minimizing quantum circuit runtimes is critical for two main reasons. First, as is evident, reducing runtime helps improve algorithmic efficiency and accelerates the overall computation. Second, and more importantly, shorter runtimes mitigate the risk of decoherence, ensuring that all quantum operations are completed before qubits lose their quantum properties [52]. Reducing execution time through circuit optimization also allows for more complex computations to be executed within the coherence time of the processor, which remains one of the most challenging aspects of quantum hardware scalability.

2.4.1 Readout and measurement times

As discussed in previous sections, the execution times of quantum gates vary across hardware platforms. Single-qubit gates are typically the fastest operations, while two-qubit gates are slower by an order of magnitude. However, the most time-consuming operation in most architectures is the measurement stage. For superconducting qubits, single shot measurements typically require on the order of 500ns [53], which is approximately five times longer than a two-qubit gate operation. In contrast, for trapped ion qubits, the same measurement can take around 46μs [54], corresponding to roughly half the duration of a two-qubit gate. For neutral atom platforms, measurement operations are particularly costly, with durations on the order of milliseconds [55], that is, about thousand times longer than a two-qubit gate operation.

When comparing these measurement times, gate times and coherence times, it becomes evident that while measurements dominate the latency, their contribution is relatively small compared to the available coherence window. However, this implicitly assumes ideal physical qubits that do not require any form of quantum error correction. In practice, once the additional gates and mid circuit measurements required by error correction are included, the effective circuit depth increases significantly, reducing the apparent coherence window of the qubits.

Finally, even when all operations fit within the coherence time, long circuits increase exposure to noise and error accumulation, potentially degrading the fidelity of the final measurement. This makes runtime minimisation an essential aspect of quantum algorithm design.

2.4.2 Parallelism and Circuit steps

The runtime of a quantum circuit can be approximated by the number of execution steps it contains, which is closely related to its depth. Counting these steps corresponds to determining into how many layers the circuit can be partitioned. However, as previously discussed, gate durations vary depending on their type and must therefore be taken into account [52]. A more accurate approximation consists in considering the duration of the slowest operation within each step. As illustrated in Figure 2.12, single-qubit gates are often highly parallelisable, whereas two-qubit gates introduce additional scheduling constraints, as they generally require specific connectivity and cannot be applied simultaneously on

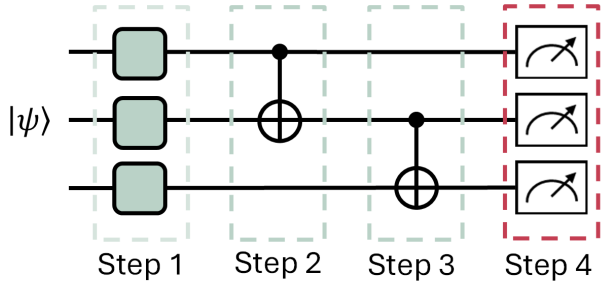


Figure 2.12: Illustration of quantum circuit execution steps. Each step corresponds to a set of gates executed in parallel. The total runtime depends on the number of steps and the time required for the slowest gate in each layer. Two-qubit gates impose constraints that limit parallelism compared to single-qubit gates.

qubits sharing an interaction. Consequently, execution time tends to increase with the number of two-qubit gates. The final measurement stage also contributes significantly to runtime, as it is typically among the slowest operations.

In the context of variational quantum algorithms (VQAs), which will be discussed in the next chapter, this relationship is of particular relevance. VQAs require iterative evaluations of parameterized circuits, and each evaluation involves multiple circuit executions. Therefore, minimizing the number of circuits involved directly minimize the overall computational cost. Optimising for minimal depth, or equivalently minimal execution steps, is a critical objective in quantum algorithm design. This optimization reduces runtime and improves resilience against decoherence and noise accumulation. However, hardware constraints, such as limited qubit connectivity, often complicate this process.

2.4.3 Connectivity

The connectivity of quantum processors describes how qubits are arranged and linked to each other. In simpler terms, it determines whether a two-qubit gate can be applied directly between two qubits. When qubits are not directly connected, SWAP gates (which can be decomposed into three CNOT gates) must be inserted to mediate the operation, allowing the two-qubit gate to be performed. These additional layers can significantly increase the number of sequential operations and consequently the overall circuit depth. As discussed earlier, connectivity varies depending on the type of quantum processor. In the study by Tennie *et al.*. [37], which examined the three most promising quantum hardware technologies for solving nonlinear differential equations, the authors illustrated the differing connectivity patterns for each technology, as shown in Figure 2.13.

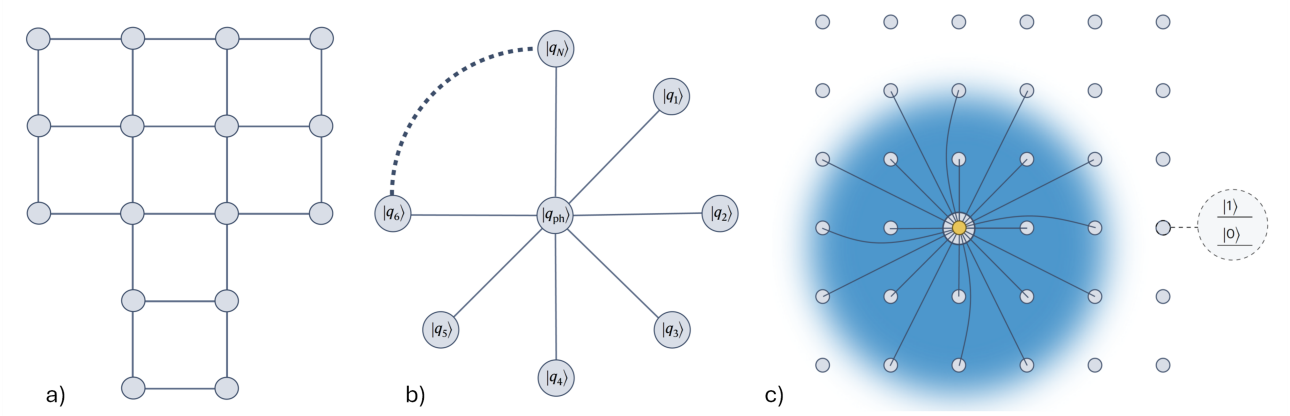


Figure 2.13: a) Typical connectivity graph of a superconducting circuit quantum processor. b) Starlike connectivity of trapped ion qubits. c) Single 2D layer of optically trapped neutral atoms, showing long-range interactions (blue shading) with the highlighted yellow atom in an excited Rydberg state. Figure from [37].

While indirect connectivity does not prevent the implementation of quantum algorithms, it often requires numerous SWAP gates, which, as mentioned, can substantially increase not only the runtime but also the total number of gates, consequently raising the system's exposure to noise.

2.4.4 Quantum Memory

What is missing in quantum computing is the ability to store a quantum state. Developing such a capability would represent a major breakthrough, and this concept is known as Quantum Random Access Memory (QRAM). QRAM would enable efficient and accelerated storage and retrieval of data within quantum systems. Unlike classical RAM, which stores information in bits incompatible with quantum computation, QRAM would natively handle quantum data, potentially leading to significant speedups in various quantum algorithms [56].

However, the current state of quantum hardware, combined with the challenges of fault-tolerant error correction, makes the implementation of a reliable and large-scale QRAM unfeasible at present [57]. These difficulties include limited scalability, restricted applicability, and susceptibility to noise [56]. Nevertheless, QRAM has been an active area of research since 2008, and proposed architectures have evolved significantly over the past two decades, showing promising progress [58], [59], [60], [61], [62].

2.5 Summary

In this chapter, we provided the quantum context at the time of writing this thesis. We introduced the fundamental concepts of quantum computing and outlined the objectives of the field. We discussed the state of current hardware and its limitations, particularly in terms of the number of qubits. While several platforms with a large number of physical qubits are emerging, the availability of logical qubits remains limited. We presented the different state-of-the-art hardware platforms, highlighting their specific challenges as well as the common issue they all share: noise.

Finally, we established the context that quantum computing alone is no longer the ultimate goal. Given the current hardware capabilities, we will see in the next chapter that one of the most promising approaches that has emerged are the hybrid quantum-classical algorithms, also called Variational Quantum Algorithms (VQAs).

Chapter 3

Literature Review

3.1 Differential equations

In mathematics, a differential equation (DE) is an equation involving one or more unknown functions and their derivatives. The order of a DE is determined by the highest order of the unknown function derivatives.

Solving differential equations is a foundational task in many scientific and engineering disciplines. In aerospace engineering, DEs are central to understanding the dynamic behaviour of physical systems, from structural analysis to fluid dynamics. Classical computational methods for solving DEs include finite-difference, finite element, and spectral techniques such as polynomial approximations. These methods often require high spatial resolution or large basis sets to achieve high accuracy, which can result in substantial computational and memory demands. Even when the dimensionality of the DEs is low (typically 1D to 3D in engineering applications), fine discretization in all spatial directions is common, which can result in high memory and computational requirements.

3.1.1 Types of differential equations

Ordinary and partial differential equations

Differential equations are classified according to the number of independent variables. If the unknown function depends on a single independent variable, the equation is an ordinary differential equation (ODE); if it depends on several, it is a partial differential equation (PDE). A classic example of a PDE is the Laplace equation, which models steady-state heat conduction, electrostatics, and potential fluid flow:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0. \tag{3.1}$$

In one spatial dimension, it reduces to:

$$\frac{d^2 f}{dx^2} = 0, \tag{3.2}$$

illustrating how a PDE can be simplified to an ODE under symmetry or dimensional constraints.

Initial value problem and boundary value problem

In applications, a differential equation is complemented by additional conditions that define the specific scenario. These are typically of two types:

- Initial value problems (IVPs): all data are specified at a single point, often representing the initial state of a time-dependent system, for instance, the initial position and velocity of a moving object.
- Boundary value problems (BVPs): conditions are imposed at two or more distinct points, usually at the spatial limits of a domain, as in steady-state heat distribution.

IVPs predict future states from a known starting point, whereas BVPs determine equilibrium or steady-state configurations.

Although these conditions are intended to secure a unique solution, it is essential to recognise that a differential equation, even when paired with initial or boundary conditions, may fail to admit any

solution, for instance when the prescribed conditions conflict with the underlying mathematical model or physical constraints.

Linear and nonlinear differential equation

Another key distinction is between linear and nonlinear equations, defined by whether the unknown function and its derivatives appear linearly. Formally, a differential operator $\mathcal{D}$ acting on a function f is linear if it satisfies the superposition principle: for any functions f_1, f_2 in the domain of $\mathcal{D}$ and any scalar $\lambda \in \mathbb{R}$,

$$\mathcal{D}(f_1 + \lambda f_2) = \mathcal{D}(f_1) + \lambda \mathcal{D}(f_2). \tag{3.3}$$

In other words, the unknown function and its derivatives appear only to the first power, without multiplication or nonlinear composition.

In contrast, a nonlinear differential equation involves nonlinear combinations of the unknown function and its derivatives and does not satisfy Equation (3.3). Exact analytical methods for solving such equations are scarce and typically rely on the presence of specific symmetries. Over large intervals, nonlinear differential equations can exhibit complex and unpredictable behavior, sometimes even leading to chaotic behaviour. But they are extensively found in science and engineering, such as the ubiquitous Navier-Stokes equation.

The existence, uniqueness, and extendability of solutions remain open questions in many cases, especially for nonlinear partial differential equations. The mathematical stability and solvability of problems with given initial and boundary conditions is itself a major research challenge. Nevertheless, when such equations correctly model an underlying physical phenomenon, one generally expects a meaningful solution to exist.

In practice, nonlinear differential equations can be approximated by linear differential equations, valid under certain conditions. This process is called linearization and is widely used to approximate the solution of a nonlinear differential equation.

A classical example is the nonlinear dynamics of a pendulum:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0, \tag{3.4}$$

where $\theta(t)$ denotes the angular displacement, L the pendulum length, and g the gravitational acceleration. For small angles, $\sin\theta \approx \theta$, reducing the system to the linear harmonic oscillator:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0. \tag{3.5}$$

Linearisation Methods

Several techniques approximate a nonlinear differential equation by a linear model, each valid under specific assumptions and serving distinct purposes. Most common approaches are the Taylor series linearisation, using the first-order terms of the Taylor expansion of the nonlinear function around an equilibrium point; the piecewise linearisation, dividing the nonlinear system into a set of linear subsystems, each valid within a specific region; and the Carleman method, widely used in numerical analysis, which turns nonlinear ODEs into finite-dimensional linear systems by introducing higher-order terms.

3.1.2 Existence of Solutions

Before trying to solve DEs, it is essential to determine whether a solution actually exists and whether it is unique, given the equation and the initial or boundary conditions. This question is particularly relevant in engineering, where differential equations model physical systems under specific constraints. While a closed-form analytical solution expressed in terms of standard functions may not always exist, the differential equation can still admit a solution, which in practice is often approximated numerically over a given interval.

The existence and/or uniqueness of solutions can, in some cases, be established using theorems such as the Picard–Lindelöf theorem and the Peano theorem for continuous first-order differential equations, but these results do not apply to all DEs. Moreover, the occurrence of a phenomenon in real life does not guarantee that its mathematical model is well-posed; it may lack a solution or admit multiple solutions, even when the described physical system is deterministic.

3.2 Classical Computing for solving differential equations

Differential equations can be solved using finite difference methods, which discretize derivatives on a fine mesh. This approach allows high-order derivatives to be computed with good accuracy and is widely used in aerospace engineering to study flows over simple geometries. However, its application becomes challenging for complex geometries or when the mesh is coarse.

For structural analysis in aerospace, standard methods are based on finite element methods (FEM), which are used to investigate stress, vibration, and fatigue by solving linear and nonlinear ordinary and partial differential equations with high fidelity. Its main advantage lies in its geometric flexibility and its ability to handle complex boundary conditions. The main limitation, however, is its computational cost. Achieving high accuracy requires a fine mesh, which often necessitates reducing the size of the system studied or simplifying the problem.

An alternative approach is the spectral method, which represents the solution of the differential equation with a global or piecewise polynomial expansion, typically using orthogonal Chebyshev polynomials or high-order Lagrange polynomials over non-uniformly spaced nodes. Spectral methods require smooth functions, which can limit their applicability. Finite difference and spectral methods can also be combined in the spectral finite element method, leveraging the strengths of both approaches. This hybrid approach has shown promise in engineering applications, particularly in fluid dynamics [63], [64], and continues to be investigated for problems such as stress prediction and fatigue analysis in composite aerospace structures [65], [66], [67].

In recent years, machine learning methods have been applied to PDEs, including approaches based on deep neural networks and discretisation free, parallelised strategies to reduce the cost of solving PDEs [68], deep neural networks built on spectral methods [69], and physics informed deep learning [70]. Naturally, this has encouraged the application of machine learning to fluid problems [71], showing very promising results compared to traditional classical approaches [72].

Machine learning applied to fluid dynamics has demonstrated the ability to capture nonlinear relationships between input and output variables, for example in turbulence super resolution [73] or navigation problems [74].

This enthusiasm for machine learning approaches is further supported by the availability of powerful GPUs in computing centres [75]. In addition, it is expected that QPUs (Quantum Processing Units, i.e., quantum computers) will be integrated into these centres to unlock new computational paradigms, including a new generation of machine learning approaches, as described by Biamonte *et al.* [76], with applications to fluid dynamics [77].

3.3 Quantum Computing for solving differential equations

As presented in Chapter 2, while significantly more challenging to design, quantum computers can run specialised algorithms that can scale better than their classical counterparts, sometimes exponentially faster. It is therefore natural to investigate the potential of quantum computers for solving DEs.

Quantum algorithms for tackling DEs fall into two distinct groups: fully quantum algorithms, and hybrid quantum-classical algorithms, also called variational quantum algorithms (VQAs).

3.3.1 Fully quantum algorithms

Fully quantum approaches offer certain advantages in solving linear differential equations, primarily because quantum operators act linearly on quantum states. However, evaluating the performance and determining the quantum advantage of these algorithms remains unclear, as these metrics depend strongly on the complexity of the problem, and practical testing on real quantum hardware is still challenging. Fully quantum algorithms aim to solve a problem using a single quantum circuit. For these algorithms to be practically relevant, they require a large and deep circuit with many logical qubits, which means they depend on fault-tolerant quantum computing and quantum random access memory. These resources are currently unavailable, as discussed in Chapter 2.

Nevertheless, fully quantum algorithms were the first approaches proposed for quantum differential equation solvers, and although they are not yet applicable at a meaningful scale, they represent the long-term goal. After the success of Shor's and Grover's algorithms, which demonstrated exponential and quadratic speedups, there was a surge of interest in developing quantum algorithms for various problems, including solving differential equations.

Many early quantum solvers, developed in the 2000s, were based on amplitude encoding and built on Quantum Amplitude Amplification and Quantum Amplitude Estimation algorithms (QAAA and QAEA) [78]. These solvers used discretization-based methods to represent differential equations over a grid of variables. A notable example is the Kacewicz algorithm (2006) [79], designed to solve initial value problems (IVPs), hence ordinary differential equations with initial conditions. This algorithm applied QAAA and QAEA to estimate the mean value of the ODE across multiple point sets. Later, the Kacewicz method was adapted for simulating solutions to more complex PDEs, including the steady-state Navier–Stokes equations [80], the Burgers' equation [81], and the heat equation [82].

Another major milestone was the HHL algorithm [83] (2009), presented in Chapter 2, for solving systems of linear equations using a matrix-based approach that outputs a quantum state proportional to the solution vector. Because many differential equations can be discretized and rewritten as linear systems using finite difference schemes, HHL inspired several quantum solvers for DEs.

One of the first HHL-based solver, proposed by Leyton *et al.*, targeted nonlinear differential equations [84]. Later, Berry *et al.* improved the approach but restricted to linear problems which appeared to be more suitable for quantum computing due to the inherent linearity of quantum operators [85]. Since then, numerous HHL-based algorithms have been proposed for solving linear DEs, employing methods such as: finite difference discretization [86], [87], [88], [89], finite element discretization [90], [91], high-order discretization methods and precision improvements [85], [92], [93].

HHL had been further improved into the Quantum Linear Systems Algorithm (QLSA) [94], showing an exponential improvement on the precision. Later, QLSA served as the basis for a quantum spectral method for differential equations [95], which approximate the solutions of the DE using Chebyshev polynomials and pseudo-spectral method to rewrite the problem as a linear system suitable for QLSA. Quantum solvers have also been developed for nonlinear ODEs and PDEs using linearization techniques, such as the Carleman method [84], [96], [97], [98], [99], [100].

Although this field is advancing rapidly and methods continue to improve, most of these fully quantum approaches target the fault-tolerant era of quantum computing, which remains decades away. Consequently, a new class of algorithms emerged over the past decade: hybrid quantum-classical approaches, also known as variational quantum algorithms (VQAs). These methods are designed to work on NISQ (Noisy Intermediate-Scale Quantum) devices that are not yet fault tolerant or fully error corrected.

3.3.2 Hybrid quantum-classical algorithms

The development of VQAs started in 2014, notably with the Variational Quantum Eigensolver (VQE) [101], designed to find the ground-state energy of a given system, with applications primarily in quantum chemistry and materials science [102], [103]. Building on this, researchers proposed a Variational Quantum Linear Solver (VQLS), which variationally finds a quantum state proportional to the solution of a linear system of equations [104]. Thus, VQAs have been rapidly studied to solve linear and nonlinear ODEs and PDEs, often using linearization techniques such as the Carleman method [5], [105], based on quantum machine learning techniques [106], [107].

The method studied in this PhD builds upon this line of research: the variational spectral approach of Kyriienko *et al.* [4]. Unlike previous methods that estimated a direction vector, this approach trains a polynomial-encoded quantum circuit to approximate the solution of a differential equation across its entire domain. In their work, they demonstrated the approach on nonlinear first-order equations without linearization. This paradigm shift opens new possibilities for quantum solvers.

However, VQAs face significant challenges, particularly regarding trainability. Gradient-based optimization often suffers from the barren plateau phenomenon, where gradients vanish exponentially with system size. Strategies such as judicious choices of cost functions and observables, efficient gradient computation techniques, and limiting circuit depth provide promising avenues to mitigate this issue [108], [109].

3.4 Variational Quantum Algorithms

Variational quantum algorithms (VQAs) have been actively studied over the past few years with the aim of achieving quantum advantage on NISQ devices. In 2021, just a few months after the start of this thesis, the comprehensive review by Cerezo *et al.* [110] was published. It provided a detailed analysis of the challenges, strategies, trainability, and feasibility of VQAs, and highlighted their "exciting prospect" for enabling quantum advantage on near-term hardware. The strategy outlined was clear: for a VQA to be implementable on NISQ devices, the parameterised quantum circuit must have low depth and require minimal use of ancilla qubits.

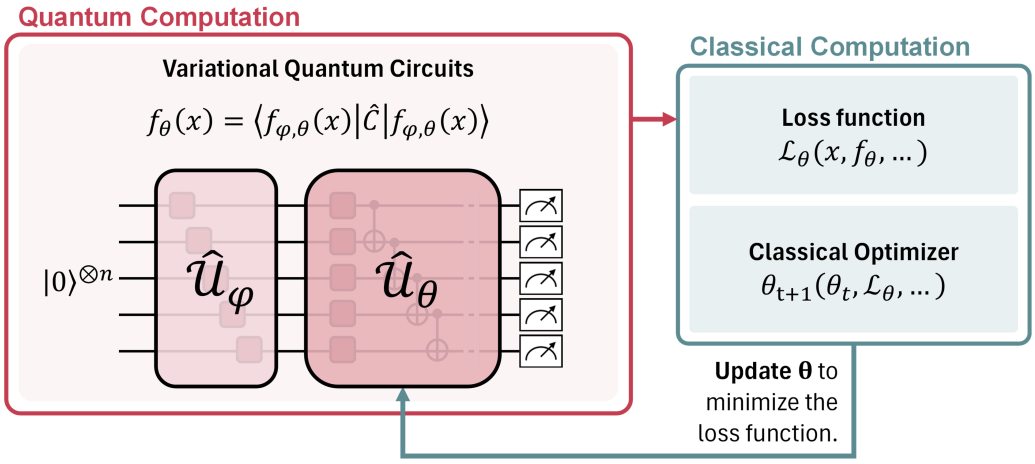


Figure 3.1: Schematic representation of a variational quantum algorithm and its hybrid quantum classical workflow.

This constraint is directly tied to the limitations of NISQ hardware, which remains highly sensitive to gate noise, constrained by a limited number of qubits, and subject to short coherence times. These features stand in contrast to the assumptions underlying fault-tolerant quantum algorithms, which presume idealised, error-corrected quantum operations.

The objective of a VQA is to ally the strengths of both classical and quantum computing paradigms, by training a parametrised quantum circuit with a classical optimizer. Although VQAs have been explored across diverse applications, they may feature different algorithmic approaches and quantum circuit architectures, but they all share a general definition using common key elements [110]: a quantum circuit built on a set of variational parameters $\{\theta_i\}_i$ (collectively denoted by the vector θ), a cost operator $\hat{C}$, a loss function $\mathcal{L}$ and a classical optimizer.

These key elements, together with the overall functioning of a VQA, are illustrated in Figure 3.1. The quantum part, shown on the left in red, consists of a variational quantum circuit parameterised by the vector of variational parameters θ, which is evaluated with respect to an observable $\hat{C}$ defining the cost function. The output of the quantum circuit is then passed to the classical part, shown on the right in green, where a loss function is computed and used by a classical optimizer to update the parameters θ so as to minimise the loss. This procedure corresponds to a single iteration of the algorithm. A VQA proceeds by repeating this hybrid loop until convergence towards an optimal set of parameters θ.

3.4.1 Variational Quantum Circuit

In the VQAs studied in this work, the Variational Quantum Circuits (VQCs) are composed of two parts: an encoding block, sometimes referred to as a quantum feature map circuit, and a variational ansatz, sometimes directly referred to as the variational quantum circuit itself. While the encoding block is used to initialise the circuit with fixed parameters, the variational ansatz is built on the set of variational parameters $\{\theta_i\}_i$ later optimised throughout the algorithm.

3.4.2 Cost Operator and Loss Function

One of the core features of a VQA is to select the architecture of the VQC and a suitable cost operator $\hat{C}$ for a given problem. In this work, the cost operator is distinguished from the loss function $\mathcal{L}$. The cost operator maps observables to the VQCs, while the loss function guides the optimization process by quantifying the distance between the algorithm's outputs and the desired solution.

3.4.3 Classical optimizers

Finally, the Variational Quantum Circuit is trained by a classical optimizer. As the loss function may exhibit multiple local extrema, training a VQC is recognised as an NP-hard problem [111], meaning that while verifying a given solution can be fast, finding the optimal solution can be extremely difficult and time-consuming. It also faces additional challenges arising from the stochastic nature of quantum measurements, noise in quantum hardware, and the presence of barren plateau (discussed in the next section 3.4.4) [110]. Consequently, the choice of an appropriate classical optimizer remains an active research topic.

Adam optimizer

Algorithm 1 Adam optimizer [112]: Default parameters values are: $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and, $\varepsilon = 10^{-8}$. All operations on vectors are element-wise, and β_1^t and β_2^t denotes β_1 and β_2 to the power t.

Require: θ_0: initial parameters
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function
$m_0 \leftarrow 0$ ▷ first moment vector
$v_0 \leftarrow 0$ ▷ second moment vector
$t \leftarrow 0$ ▷ Initialize timestep
while not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ ▷ compute stochastic gradient
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ ▷ update first moment
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t$ ▷ update second moment
$\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$ ▷ bias corrected first moment
$\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$ ▷ bias corrected second moment
$\theta_t \leftarrow \theta_{t-1} - \alpha \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$ ▷ parameter update elementwise
end while

One of the first classical optimizers adopted for VQAs is the Adam (ADAptive Moment estimation) optimizer, which has been widely used in classical machine learning, particularly for training neural networks, due to its ease of implementation, computational efficiency, and low memory requirements [112]. It is an advanced variant of Stochastic Gradient Descent (SGD) that leverages first-order gradients to estimate both the first and second moments of the gradient.

This approach allows Adam to adapt the learning rate for each parameter dynamically throughout the optimization process, making it more robust to noisy gradient estimates. In the context of VQAs for differential equations, this thesis primarily focused on the spectral approach introduced by Kyriienko *et al.* in 2021 [4], which employs the Adam optimizer to guide the optimization of the variational parameters.

BFGS optimizer

Algorithm 2 BFGS Quasi-Newton Method

Require: $x_0 \in \mathbb{R}^n$: initial point
Require: B_0: initial Hessian approximation (positive definite)
  $k \leftarrow 0$
  while Stopping criteria not met **do**
    $g_k \leftarrow \nabla f(x_k)$ ▷ compute gradient
    $B_k p_k = -g_k$ ▷ to solve, to compute search direction
    α_k ▷ choose step length via line search satisfying Wolfe conditions
    $s_k \leftarrow \alpha_k p_k$ ▷ Compute the step vector
    $x_{k+1} \leftarrow x_k + s_k$ ▷ Update the point
    $g_{k+1} \leftarrow \nabla f(x_{k+1})$ ▷ Compute the new gradient
    $y_k \leftarrow g_{k+1} - g_k$ ▷ Compute the difference of gradients
    $B_{k+1} \leftarrow B_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k}$
    $k \leftarrow k + 1$ ▷ update Hessian approximation
  end while

The second classical optimizer discussed in this thesis is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which was employed in the VQA proposed by Sato *et al.* [5] in the same year. This approach aims to solve differential equations using a discretised formulation. Unlike the Adam optimizer, which is specifically designed to handle sparse gradients and noisy environments, BFGS targets general unconstrained nonlinear optimization problems. It belongs to the family of quasi-Newton methods, which approximate the Hessian matrix of the loss function rather than computing it explicitly, hence reducing computational complexity. The BFGS method constructs an iterative update of an approximation to the inverse Hessian, built from first-order gradients or approximations of the gradient, leading to superlinear convergence under suitable conditions.

Is there "best" optimizer for VQAs?

Over the past few years, the choice of an optimizer for VQAs has been extensively investigated and remains an active area of research. The performance of these optimizers appears to strongly depend on the structure of the VQC, its size, and the specific form of the loss function. A comparative study of the Adam optimizer, BFGS, and natural gradient descent used for the Variational Quantum Eigensolver (VQE) [113] indicated that BFGS tends to fail to converge as the system size becomes too large (more than 20 variational parameters). Adam, on the other hand, often becomes trapped in local minima,

and the natural gradient descent demonstrated more stable performance, but with a significantly higher computational cost. An empirical review from 2022 further suggested that Adam remains the most effective gradient-based optimizer for VQAs [114], and regarding gradient-free approaches, the Simultaneous Perturbation Stochastic Approximation (SPSA) method [115] was identified as the most promising. SPSA is considered gradient-free because it estimates the gradient using a single finite-difference approximation along a randomly chosen direction, rather than computing gradients explicitly.

A comparison between gradient-based and gradient-free optimizers [116] for the Variational Linear Quantum Solver [104] over 3 to 5 qubits with realistic noise did not highlight an optimizer type that outperformed the other. Among gradient-based optimizers, the best performances were obtained with the BFGS optimizer, while the overall best performances were obtained with the gradient-free optimizer SPSA [115]. The SPSA is expected to be an efficient method for VQAs due to its reduced complexity and better performance compared to a SGD approach. However, other gradient-free optimizers such as COBYLA (Constrained Optimization BY Linear Approximation) [117] performed poorly in the presence of noise compared to other optimizers [116].

Since these latter classical optimizers were initially proposed for classical computation, other methods have been developed recently for quantum hybrid computation, with, for example, the meta-learner [118], a SGD-based optimizer, which adapts the learning rate of the optimization process by training a neural network based on the optimization history of similar problems. Another adaptive and SGD-based approach for VQA is to vary the number of shots required to determine the loss function gradient, that is, its precision, throughout the optimization process to significantly reduce the complexity of the optimization [119], [120]. One more gradient-based approach for VQAs is the quantum natural gradient descent method, which instead evaluates the gradient of the loss function in terms of the Euclidean norms, it uses a metric tensor that quantifies the sensitivity of the quantum circuit to variation of parameters [121]. In the case of a loss function defined as a sum of trigonometric functions, a gradient-free approach can be used where all the local parameters are sequentially updated [122], [123].

Finally, current research on gradient-free methods continues to explore strategies for improving their performance [124], [125], while parallel efforts for gradient-based optimizers aim to reduce their com-

putational cost [126] or enhance the efficiency of the Adam optimizer [127].

3.4.4 Barren Plateau

The Barren plateau phenomenon refers to the problem of vanishing gradients in VQAs. It severely limits the trainability of these algorithms and poses a challenge for optimization problems in quantum systems. This phenomenon continues to receive a significant attention from researchers, to comprehensively understand the phenomenon, ascertain the reasons behind its occurrence, and explore effective strategies to surmount its challenges.

The first cause identified for the Barren plateau phenomenon is the use of deep randomly initialised hardware efficient ansatzes [128]. Consequently, attention to overcome the barren plateau was initially directed toward a clever initialisation of the VQC parameters [129], [130], [131], [132] or a clever VQC structure [133]. Later, Cerezo *et al.* [108] studied the correlation between the depth of the circuit as well as the use of a local or global measurement and the occurrence of barren plateau. Their findings revealed that even shallow VQCs could lead to a barren plateau, highlighting the selection of the cost function as a main cause of the phenomenon. Specifically, opting for local cost functions over global cost functions was shown to mitigate the barren plateau, resulting in a polynomial gradient decay with circuit size instead of an exponential decay. Recently, one more approach proposed counteracting the emergence of the barren plateau by computing the gradient and variance of the cost function using Clifford approximant quantum circuits [109].

3.4.5 Derivative Quantum Circuits

For VQAs using a gradient-based optimizer, the gradient of a VQC with respect to its variational parameters can be computed analytically. In the spectral approach considered in this thesis, we extend this idea to also compute derivatives with respect to the differential equation variables. In practice, analytical derivatives of quantum circuits with respect to Pauli gate parameters can be obtained directly by inserting an appropriate rotational gate, depending on the circuit structure.

Parameter Shift Rule Method

The Parameter Shift Rule method is widely used to differentiate quantum circuits. This method requires the building of two modified circuits to obtain the derivative with respect to one Pauli gate parameter [106]. In the example of a circuit of two qubits as described in Figure 3.2.a, the derivative with respect to the parameter gate α is obtained by the difference of two modified versions where the parameter gate α is shifted to $\pm\pi/2$, as described in Figure 3.2.b.

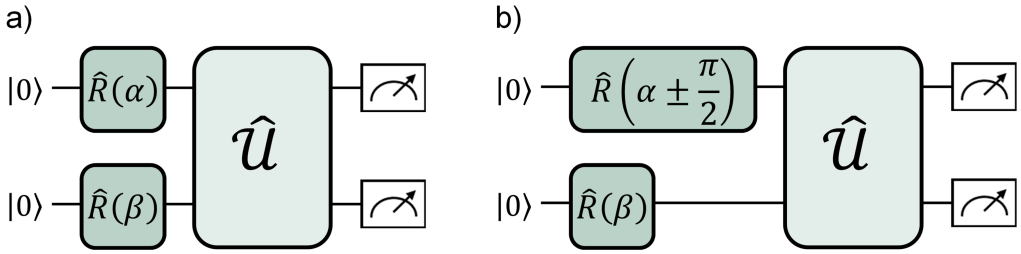


Figure 3.2: Parameter Shift Rule. a) Original quantum circuit, b) Derivative quantum circuits. α and β are parameters of Pauli gates, representing rotation angles.

Thus, the derivative of the original quantum circuit in figure 3.2.a with respect to α is:

$$\frac{\partial}{\partial\alpha}\langle f_\alpha|\hat{C}|f_\alpha\rangle = \frac{1}{2}\left(\langle f_{\alpha^+}|\hat{C}|f_{\alpha^+}\rangle - \langle f_{\alpha^-}|\hat{C}|f_{\alpha^-}\rangle\right), \tag{3.6}$$

where $|f_\alpha\rangle$ is the state vector of the original quantum circuit, $|f_{\alpha^+}\rangle$ and $|f_{\alpha^-}\rangle$ are the state vectors of the quantum circuit where the parameter α has been shifted respectively of $\pi/2$ and $-\pi/2$. $\hat{C}$ is a chosen cost function.

Hadamard Test Differentiation

A second method to derive quantum circuits is based on the Hadamard test, originally used to extract the real part of an expectation value by measuring the magnetisation of the ancilla qubit a, as shown in Figure 3.3.a: $\langle a|\,\hat{Z}\,|a\rangle = Re\,\langle\psi|\,\hat{\mathcal{U}}\,|\psi\rangle$. This structure can be reused to obtain the derivative of the expectation value with respect to a Pauli gate parameter θ_j (Figure 3.3.b) [134]:

$$\langle a|\,\hat{Z}\,|a\rangle = \frac{\partial}{\partial\theta_j}\langle\psi|\,\hat{\mathcal{U}}\,|\psi\rangle\,, \tag{3.7}$$

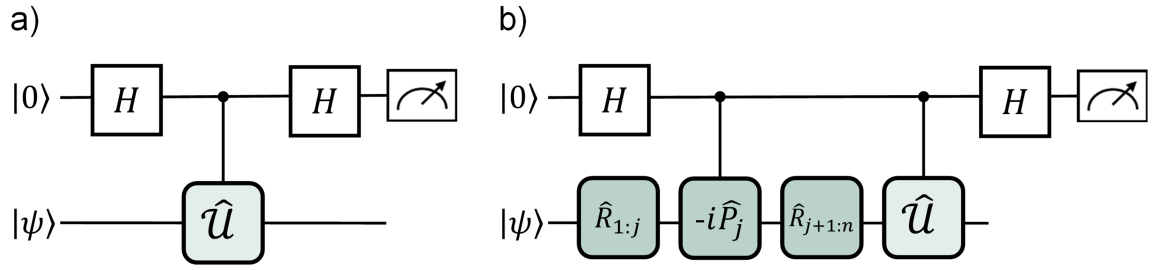


Figure 3.3: Hadamard test differentiation. a) Hadamard Test, b) Hadamard Test Differentiation.

where $\hat{R}_j = \exp\left(\frac{-i}{2}\theta_j \hat{P}_j\right)$, and $\hat{P}$ is a tensor of Pauli matrices. With this differentiation method, only one circuit is required to obtain the derivative of the expectation value.

3.4.6 Variational Ansatz

An important aspect of a variational quantum algorithm is the design of the variational ansatz, which organises and distributes the impact of each parameter θ_j in the circuit. The ansatz defines how quantum states are explored during training, thereby determining both the expressivity and the trainability. While the structure of the ansatz can, in principle, be tailored to the specific problem at hand, one architecture has become widely adopted in the field: the hardware-efficient ansatz [110].

Hardware Efficient Ansatz

The Hardware Efficient Ansatz (HEA) originates from the work of Kandala *et al.*, published in 2017 [135], where they introduced a hardware-tailored variational quantum eigensolver (VQE) designed for small molecules and quantum magnets. In this study, they proposed a parametrised quantum circuit adapted to the native gate set of superconducting qubit devices, demonstrating the experimental optimization of molecular Hamiltonians involving up to six qubits and over one hundred Pauli terms. Their main contribution lies in the use of a variational ansatz whose structure reflects the physical connectivity and constraints of the underlying hardware.

This approach has since become a standard in the field of VQAs, widely adopted due to its generality and compatibility with NISQ devices. Moreover, recent work suggests promising performance even when implemented on alternative platforms such as trapped ions [136]. The structure of the HEA consists of alternating layers of single-qubit rotations and entangling operations. Each layer contains

a block of parameterised rotation gates such as $\hat{R}_X, \hat{R}_Y$ or $\hat{R}_Z$ and a block of entangling gates such as CNOT gates. The overall circuit, illustrated in Figure 3.4 can be expressed as:

$$\hat{\mathcal{U}}(\theta) = \bigotimes_j^n [\, \hat{\mathcal{U}}^{j,d}(\theta)] \times \hat{\mathcal{U}}_{\text{ent}} \times \bigotimes_j^n [\, \hat{\mathcal{U}}^{j,d-1}(\theta)] \times \cdots \times \hat{\mathcal{U}}_{\text{ent}} \times \bigotimes_j^n [\, \hat{\mathcal{U}}^{j,0}(\theta)]. \tag{3.8}$$

where n denotes the number of qubits, d the circuit depth (number of layers), $\hat{\mathcal{U}}^{j,l}(\theta)$ the rotation block applied to qubit j at layer $l \in [0, d]$, and $\hat{\mathcal{U}}_{\text{ent}}$ the entangling block.

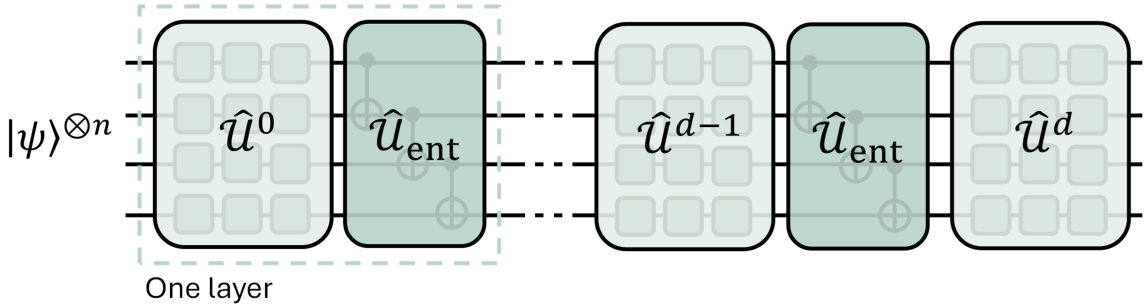


Figure 3.4: Hardware Efficient Ansatz quantum circuit.

In the original implementation by Kandala *et al.* [135], each single-qubit rotation block was composed of three consecutive gates:

$$\hat{\mathcal{U}}^{j,l}(\theta) = \hat{R}_Z(\theta_2^{j,l})\hat{R}_X(\theta_1^{j,l})\hat{R}_Z(\theta_0^{j,l}), \tag{3.9}$$

where $\hat{R}_X$ and $\hat{R}_Z$ denote rotations around the X and Z axes of the Bloch sphere. This particular sequence of rotations allows reaching any point on the Bloch sphere. This structure results in a total of $n(3d + 2)$ single-qubit rotation gates, and therefore the same number of independent variational parameters. For each layer d, there are three rotational gates per qubit, while in the last parameterized layer, which is not followed by an entangling layer, the sequence can be reduced to $\hat{\mathcal{U}}^{j,d}(\theta) = \hat{R}_X(\theta_1^{j,d})\hat{R}_Z(\theta_0^{j,d})$, if it is immediately followed by a Z-basis measurement.

For the entangling operations, the authors exploited the native two-qubit interactions present in superconducting qubit devices. These are described by a drift Hamiltonian H_0, and the entangling unitary is implemented via natural time evolution under this interaction:

$$\hat{\mathcal{U}}_{\text{ent}} = \exp(-iH_0\tau), \tag{3.10}$$

With τ the evolution time.

Following its adoption into standard quantum software libraries such as InQuanto [137], the ansatz has been generalised to allow greater flexibility in the design of the rotation blocks. A common simplification is to use a single parameterised gate per qubit, along with a linear entangling layer of CNOT gates. In this case, the building blocks become:

$$\hat{\mathcal{U}}^{j,\ell}(\theta) = \hat{R}_X(\theta^{j,\ell}) \tag{3.11}$$

$$\hat{\mathcal{U}}_{\text{ent}} = \prod_{j=1}^{n-1} \text{CNOT}_{j,j+1}. \tag{3.12}$$

This structure requires only dn variational parameters and $(n-1)d$ entangling gates, with a linear entanglement pattern compatible with nearest-neighbour hardware. Although the rotation block can be reduced to a single gate, any combination of one to three elementary rotations,$\hat{R}_X$, $\hat{R}_Y$, $\hat{R}_Z$, can be used, depending on the specific encoding and expressivity requirements of the problem.

Throughout this manuscript, sequences of rotational gates appearing in the hardware efficient ansatz or other variational ansätze are denoted using dashes. This notation denotes the ordered application of quantum gates in the circuit. For example, in the original implementation by Kandala *et al.* [135], the single-qubit rotation block given in (3.9) can be written as the sequence $\hat{R}_Z(\theta_0) - \hat{R}_X(\theta_1) - \hat{R}_Z(\theta_2)$ in the circuit.

Finally, when measurements are performed in the computational basis (that is, along the z axis), any trailing $\hat{R}_Z$ gate in the final rotation layer can be omitted, as it commutes with the measurement operator and does not influence the outcome.

3.4.7 Challenges and Limitations in development of VQA

Despite their advantages, VQAs faces some limitations. While VQAs are designed for NISQ devices, access to such hardware remains limited. Some cloud platforms, such as IBM, provide access to real devices or simulators, but the available qubit counts are still low, and the queuing systems make variational algorithms challenging to run [138]. This situation is evolving and may improve soon, but access typically comes at a cost [139], [140].

Additionally, during the early stages of the development of a quantum algorithm, it can be necessary to have a visualisation of the statevector, which cannot be observed on real machines, making classical simulations of quantum circuits useful. But, simulating a quantum circuit classically is not as efficient: an n-qubits quantum circuit requires storing a vector of 2^n complex values. The memory requirement grows exponentially with the number of qubits, and even a few dozen qubits may require specialized hardware [141].

Some VQAs have already been executed on real quantum processors. For example, IBM Heron has been used to predict mRNA sequences up to 60 nucleotides using several hundred qubits [142]. However, this is still considered a relatively simple problem. At the same time, it does not seem feasible to explore far beyond current hardware capabilities, since simulating a VQA such as a 30 qubit VQE with a classical quantum simulator would already require 15 hours on 1024 FUJITSU Processor A64FX units, each delivering 3.1 TFLOPS [143]. Classical simulation rapidly becomes computationally prohibitive.

3.5 Existing VQAs for solving differential equations

3.5.1 Kyriienko *et al.* Algorithm

The first quantum algorithm studied in this work is a polynomial-encoded VQA developed by Kyriienko *et al.* [4], which trains VQCs to approximate the solution of DEs by polynomial fitting. In their paper, Kyriienko *et al.* applied their algorithm exclusively to first-order DEs. However, they suggested the possibility of extending its use to higher-order DEs through repeated circuit differentiation. In this work, we propose an extension of the algorithm's application to second-order DEs.

Circuit Structure

The Kyriienko *et al.* algorithm [4] aims to represent an approximation of the solution of a differential equation by a parameterised quantum circuit divided into two parts : a quantum feature map $\hat{\mathcal{U}}_\varphi(x)$ and a variational ansatz $\hat{\mathcal{U}}_\theta$.

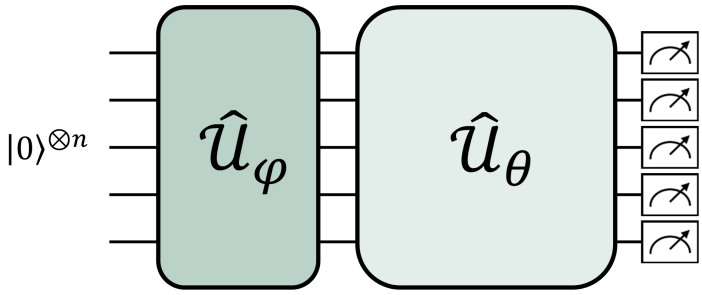


Figure 3.5: Quantum circuit structure of the Kyriienko *et al.* algorithm.

The first part encodes the equation variable x through a fitting set of nonlinear Chebyshev polynomials of the first and second kind over $]0, 1[$. Encoding the variable x through functions allows a high expressivity of the quantum circuit while ensuring its derivability. The quantum feature map $\hat{\mathcal{U}}_{\varphi}(x)$ is composed of Y-rotational gates $\hat{R}_Y$, parameterised by the functions $\varphi_j(x) = 2j \arccos(x)$, where the degree j of the Chebyshev polynomials grows with the number of qubits n [4]. $\hat{\mathcal{U}}_{\varphi}(x)$ can be expressed as:

$$\hat{\mathcal{U}}_{\varphi}(x) = \bigotimes_{j=1}^{n} \hat{R}_{Y,j} \left[2j \arccos(x)\right] \tag{3.13}$$

This quantum feature map encodes the Chebyshev polynomials of the first and second kind as follow:

$$\hat{R}_{Y,n}\big(\varphi(x)\big) |0\rangle = T_n(x) |0\rangle + \sqrt{1 - x^2}\, U_{n-1}(x) |1\rangle , \tag{3.14}$$

where the Chebyshev polynomials of the first kind is defined by $T_n = \cos(n \arccos(x))$ and of the second kind by $U_{n-1}(x) = \sin(n \arccos(x))/\sin(\arccos(x))$ [4]. The chosen variational ansatz $\hat{\mathcal{U}}_{\theta}$ is the hardware efficient ansatz [144], composed of multiple layers of a sequence of three rotational gates and CNOT gates.

Algorithm

In Kyriienko *et al.* [4], the resulting quantum state $|f_{\varphi,\theta}(x)\rangle = \hat{\mathcal{U}}_{\theta}\, \hat{\mathcal{U}}_{\varphi}(x) |0\rangle^{\otimes n}$ is evaluated by the total magnetisation of the system $\hat{C} = \sum_{j=1}^{n} \hat{Z}_j$. The derivatives with respect to x and θ are determined by the parameter shift rule method and the chosen classical optimizer is Adam [112]. Finally, the loss function is defined as the mean square of the DE with all terms on one side, as expressed in Equation (4.2).

Results

The work of Kyriienko *et al.* [4] addressed different types of first-order differential equations, including a nonlinear equation and the stationary state of the Navier–Stokes convergent–divergent nozzle problem.

First-order differential equation with nonlinear coefficients The first-order nonlinear differential equation studied in their work is expressed as:

$$\frac{df}{dx} + \lambda f(\kappa + \tan(\lambda x)) = 0, \tag{3.15}$$

with $\lambda = 8$ or 20, $\kappa = 0.1$, and $f(0) = 1$. This equation admits the following analytical solution:

$$u(x) = e^{-\kappa\lambda x}\cos(\lambda x) + C, \tag{3.16}$$

where $C = 0$ under the given initial condition.

In this application, the Chebyshev-encoded VQC was constructed using 6 qubits. The HEA consisted of a three-gate sequence $\hat{R}_Z, \hat{R}_X, \hat{R}_Z$ with a depth of 5. The initial condition was enforced through an automatic shift of the function at each iteration, and training was performed using the Adam optimizer over 20 equidistant points in the interval $[0, 0.9]$.

This example served to illustrate the performance of different encoding schemes and to highlight the efficiency of the Chebyshev encoding with degree 1 on each qubit, as well as the Chebyshev tower encoding, where the degree of the Chebyshev polynomial basis increases with the number of qubits. Additionally, the authors examined how the depth of the HEA impacts the quality of the approximation and the behaviour of the loss function.

As shown in Figure 3.6(a) and (b), for different values of λ, corresponding to distinct oscillatory behaviours, the Chebyshev-encoded VQA produced solutions that closely matched the analytical results. Since details regarding the learning rate or the initial parameter vector are not specified, we assume that the Adam optimizer uses the default learning rate and that the parameter vector is initialized with random angles.

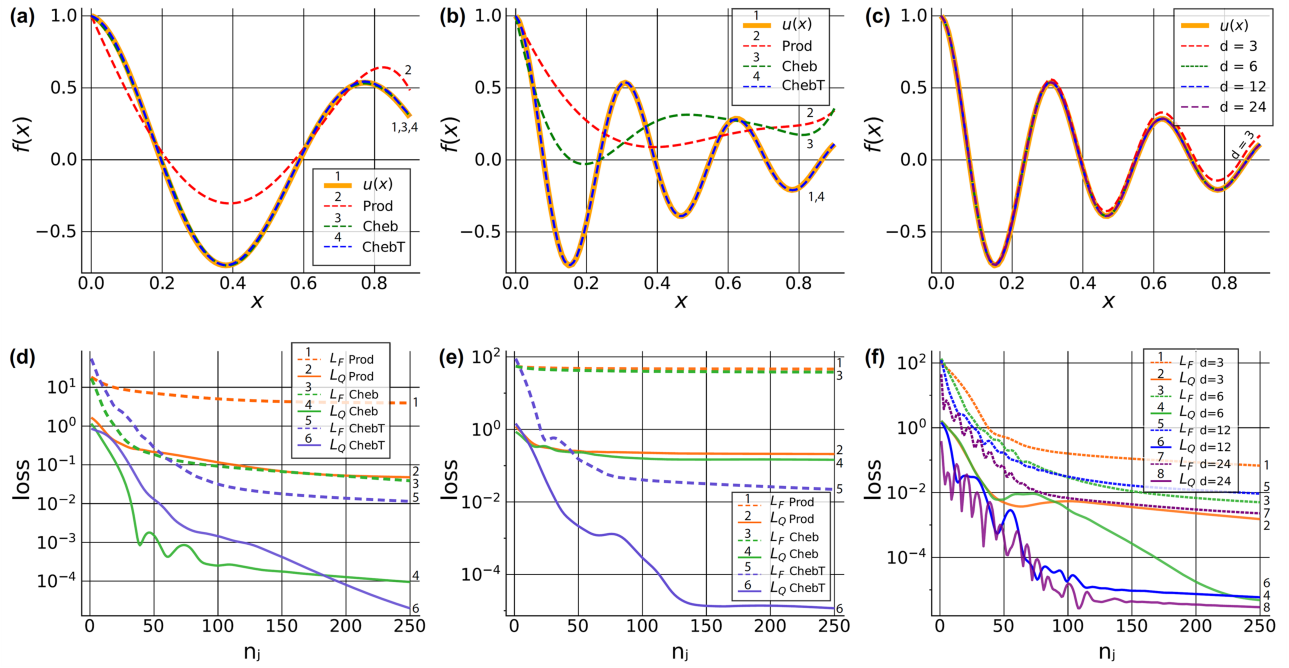


Figure 3.6: Results obtained by Kyriienko *et al.* for the discontinuous first-order differential equation using VQAs. a), b): solutions for $\lambda = 8$ and $\lambda = 20$, respectively. c): comparison of different circuit depths for $\lambda = 20$ with Chebyshev tower encoding. d), e), f): corresponding evolution of the overall loss function and quality function across iterations. Figure from [4].

Stationary Navier–Stokes equation Another significant result obtained using their method concerns the solution of the stationary Navier–Stokes equations for a convergent–divergent nozzle. In this case, they employed three 6-qubit VQCs using the simple Chebyshev encoding and an HEA composed of the three-gate sequence $\hat{R}_Z - \hat{R}_X, -\hat{R}_Z$ with a depth of 6. These VQCs were trained to represent, respectively, the three functions $\rho(x)$, $T(x)$, and $V(x)$, in the problem formulated as:

$$\frac{\partial \rho}{\partial x} = \frac{\rho V^2 d_x(\ln(A))}{T - V^2}, \quad \frac{\partial T}{\partial x} = \frac{TV^2(\gamma - 1) d_x(\ln(A))}{T - V^2}, \quad \frac{\partial V}{\partial x} = -\frac{TV d_x(\ln(A))}{T - V^2}, \tag{3.17}$$

where $A(x) = 1 + 4.95(2x - 1)^2$ defines the nozzle shape, with initial conditions $\rho(0) = T(0) = 1$ and $V(0) = 0.1$, and $\gamma = 1.4$ is the heat capacity ratio.

This problem is particularly challenging because of the subsonic–supersonic transition occurring near the throat of the nozzle (i.e. the middle of the interval), which induces a sharp change in behaviour: the velocity V increases while the temperature T and density ρ decrease. The training was therefore performed in two stages. In the first stage, the three VQCs were trained over the interval $[0, 0.4]$, which includes the initial region but avoids the transition zone. The Adam optimizer was used with

a learning rate of 0.01 and 20 equidistant training points.

In the second stage, the training was extended to the region $[0.6, 0.9]$, still avoiding the transition point. This step reused the 20 points from the first stage as regularization points, ensuring consistency with the previously learned solution, and added 20 new points from the second region. Furthermore, 5 additional regularization points were introduced in the supersonic region, introducing a bias toward the expected behaviour. The optimizer was again Adam, but with a reduced learning rate of 0.005.

The results of this procedure are shown in Figure 3.7, illustrating the first optimization stage, the second optimization stage, and the evolution of the solution quality and loss functions across iterations.

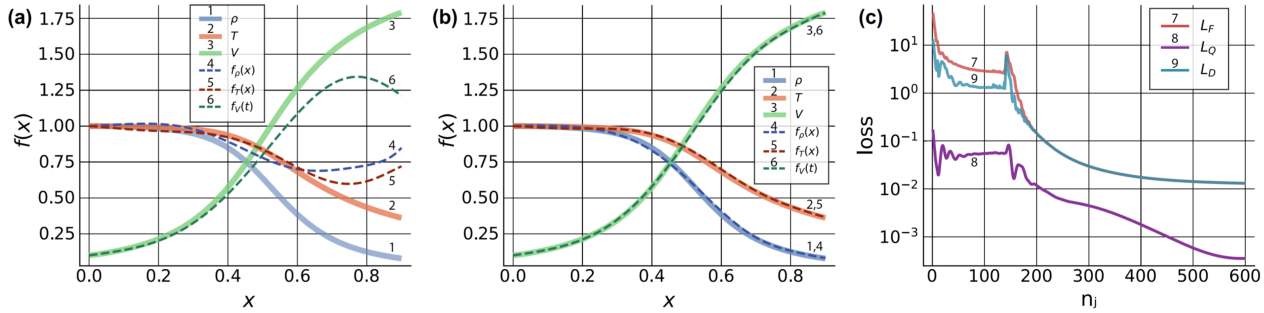


Figure 3.7: Application of the VQA by Kyriienko *et al.* to the stationary Navier–Stokes equations. a) First optimization stage, b) second optimization stage, c) evolution of the solution quality and the loss function across iterations. Figure from [4].

The results indicate an excellent overlap between the VQA solution and the analytical solution. However, it must be noted that the method required prior knowledge of the solution for the second part of the interval in order to select appropriate training points. The paper does not provide details regarding the exact positions of these points or their respective weights in the optimization process.

3.5.2 Sato *et al.* Algorithm

The second VQA used in this work for comparison purposes is the Sato *et al.* algorithm [5], which was designed to solve the Poisson equation. This algorithm is based on a discretised approach in which the VQC is used to approximate the unitary vector of the solution of the 1D Poisson equation, which is relevant for determining pressure in incompressible flow.

Circuit Structure

The circuit structure has been built to solve a 1D second order differential equation of the form : $d^2f(x)/dx^2 + s(x) = 0$, where s is the source term encoded through its amplitudes by $\hat{\mathcal{U}}_s$ and the direction vector of the solution f is approximated by the variational ansatz $\hat{\mathcal{U}}_\theta$. This structure using an ancilla gives the following state :

$$|s, \psi_\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle |s\rangle + |1\rangle |\psi_\theta\rangle) \tag{3.18}$$

where the state vector $|\psi_\theta\rangle$ encodes the unitary vector of the discretised vector that approximates the solution f.

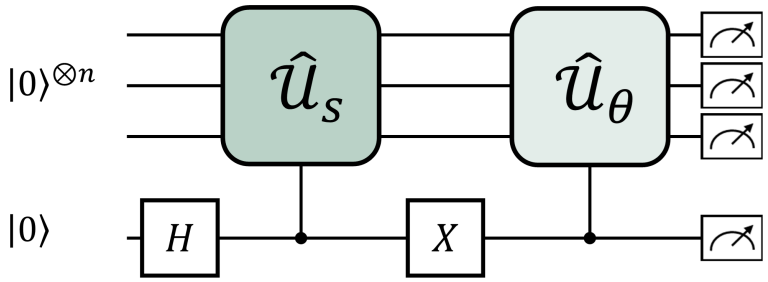


Figure 3.8: Quantum circuit structure of the Sato *et al.* algorithm, reproducing the state $|s, \psi_\theta\rangle$ defined in Equation (3.18).

Loss function

The approach of Sato *et al.* [5] is based on the minimisation of the total potential energy defined as:

$$E = \frac{1}{2}\int_\Omega \nabla v^* \cdot \nabla v \; d\Omega - \frac{1}{2}\int_\Omega v^* s \; d\Omega - \frac{1}{2}\int_\Omega s^* v \; d\Omega, \tag{3.19}$$

where Ω denotes the domain, s the source function, and v an element of the Sobolev space $H^1(\Omega)$ satisfying the boundary conditions. Minimising the total potential energy with respect to v yields the state field that satisfies the Poisson equation. After discretisation, the total potential energy becomes:

$$E_h = \frac{1}{2}r^2 \langle\psi_\theta| A |\psi_\theta\rangle - r \langle s, \psi_\theta| X \otimes I^{\otimes n} |s, \psi_\theta\rangle , \tag{3.20}$$

where $|\psi_\theta\rangle$ and $|s, \psi_\theta\rangle$ are the quantum states described earlier, n is the number of qubits, and r is a real coefficient satisfying $|v\rangle = r_\theta |\psi_\theta\rangle$ and can be defined in function of θ, such as:

$$r(\theta) = \frac{\langle s, \psi_\theta | X \otimes I^{\otimes n} | s, \psi_\theta \rangle}{\langle \psi_\theta | A | \psi_\theta \rangle}. \tag{3.21}$$

For each boundary condition, the matrix A, which is positive-definite and symmetric, becomes:

$$A_{\text{Periodic}} = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix},$$

$$A_{\text{Dirichlet}} = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 2 \end{bmatrix}, \; A_{\text{Neumann}} = \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix}. \tag{3.22}$$

These matrices can be written using a shift operator P:

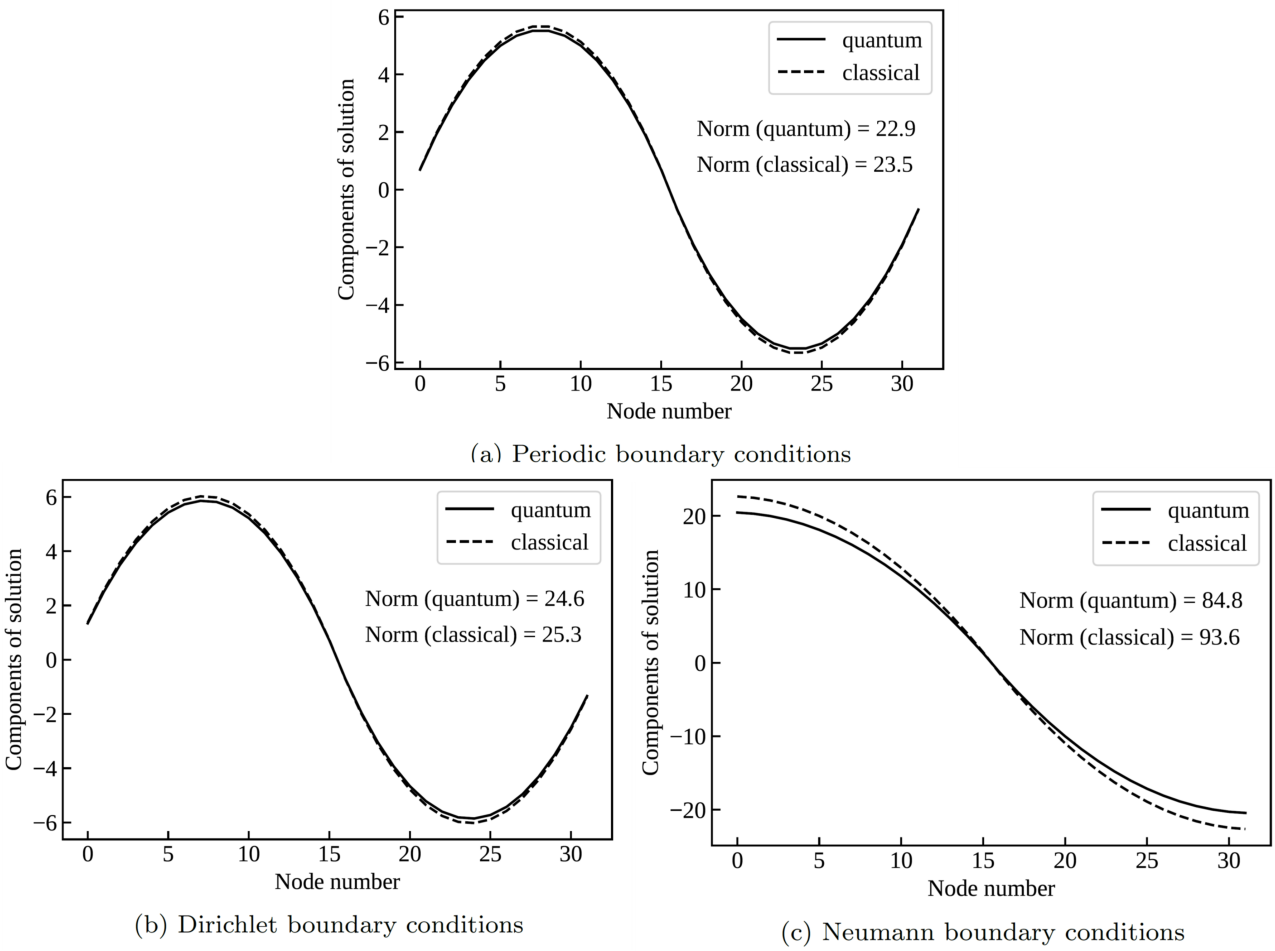
$$\begin{aligned} A_{\text{Periodic}} &= I^{\otimes n-1} \otimes (I - X) \quad + P^{-1}\big(I^{\otimes n-1} \otimes (I - X)\big)P, \\ A_{\text{Dirichlet}} &= A_{\text{Periodic}} + P^{-1}\big(I_0^{\otimes n-1} \otimes X\big)P, \\ A_{\text{Neumann}} &= A_{\text{Periodic}} - P^{-1}\big(I_0^{\otimes n-1} \otimes (I - X)\big)P. \end{aligned} \tag{3.23}$$

Algorithm

This approach evaluates the VQC described in Figure 3.8 with multiple cost operators depending on the boundary conditions to quantify the loss function, which here is the potential energy of the DE. As this algorithm is a discretised approach, derivatives with respect to the variable x are obtained with finite-differencing. In contrast, the gradient of the loss function $\nabla_\theta \mathcal{L}_\theta$ requires differentiating the VQC, using a method based on the Hadamard test differentiation [5]. The classical optimizer chosen

by Sato *et al.* is the BFGS optimizer, described in Section 3.4.3.

Results



(a) Periodic boundary conditions

(b) Dirichlet boundary conditions

(c) Neumann boundary conditions

Figure 3.9: Solutions obtained with the VQA of Sato *et al.* for each boundary condition, compared to the classical computation of $A^{-1}s$ as a function of the number of qubits. Figure from [5].

In their paper, Sato *et al.* applied their VQA to the Poisson equation with a step function s ranging from $1/2^{n/2}$ to $-1/2^{n/2}$, for $n = 5$, corresponding to $n + 1 = 6$ qubits over $2^n = 32$ points. Their results for each boundary condition, Neumann, Periodic, and Dirichlet, are shown in Figure 3.9. The VQA performs well for the Periodic and Dirichlet boundary conditions. For the Neumann condition, there is a slight discrepancy between the classical and quantum computations. This appears to be primarily due to imprecision in the norm estimation, while the direction of the solution vector remains reasonably accurate.

3.6 Summary

In this chapter, we have shown that solving differential equations is a fundamental topic in science and engineering, which can be computationally expensive using classical methods. Research continues to explore new approaches to solve a wide variety of differential equations, particularly nonlinear systems with multiple variables, which remain among the most interesting and challenging. Consequently, this field remains active in classical computation and, over the last decade, has emerged as an area of interest in quantum computing.

In this thesis, we focused on variational quantum algorithms. At the beginning of the PhD and still at the time of writing, these algorithms appeared to be a promising use of current, and eventually future, quantum computers.

The primary VQA that captured our attention is the Kyriienko *et al.* algorithm, due to its spectral approach, which leverages the fact that quantum circuits can be analytically differentiated. As mentioned in Chapter 1, the research for this PhD initially aimed to expand the application of the Kyriienko *et al.*. VQA to higher-order equations and a broader range of differential equations. However, this work revealed several limitations and challenges, including high computational demands and difficulties in representing smooth functions, which led to noticeable oscillations.

From these observations arise the idea of investigating whether an alternative encoding could improve one or both of these aspects, leading to the development of a new spectral VQA encoding based on different polynomials: the Lagrange-encoded VQA.

Chapter 4

Variational Quantum Algorithms Based on Polynomial Encoding

In this chapter, we present an overview of the workflow and key components of spectral VQAs for solving differential equations: polynomial-encoded VQAs.

The workflow draws inspiration from the work of Kyriienko *et al.* [4], with was extended in this work to broaden the applicability of their Chebyshev-encoded VQA. We also introduce an novel variational quantum circuit, based on a novel encoding strategy, or quantum feature map, which embeds the Lagrange polynomial basis into the circuit. This new encoding, inspired by the Hadamard test structure, shifts the VQA paradigm by incorporating entanglement directly within the encoding block while maintaining a shallow variational ansatz. The underlying concept remains the same: a variational quantum circuit, parameterized by the variable of the differential equation and variational parameters, is evaluated with a cost function and optimized using a classical optimizer.

The motivation for developing this new encoding arises from the limitations observed with the Chebyshev-based approach. While effective in many cases, it faced challenges in representing smooth functions accurately and incurred high computational costs for higher order differential equations. These limitations suggested that exploring alternative polynomial bases could improve both the expressivity of the variational circuit and the stability of the solutions, guiding the development of the Lagrange-encoded VQA.

A detailed discussion of the variational quantum circuits, the differences in encoding strategies, and their implementation is provided in Chapter 5.

4.1 Algorithms Workflow

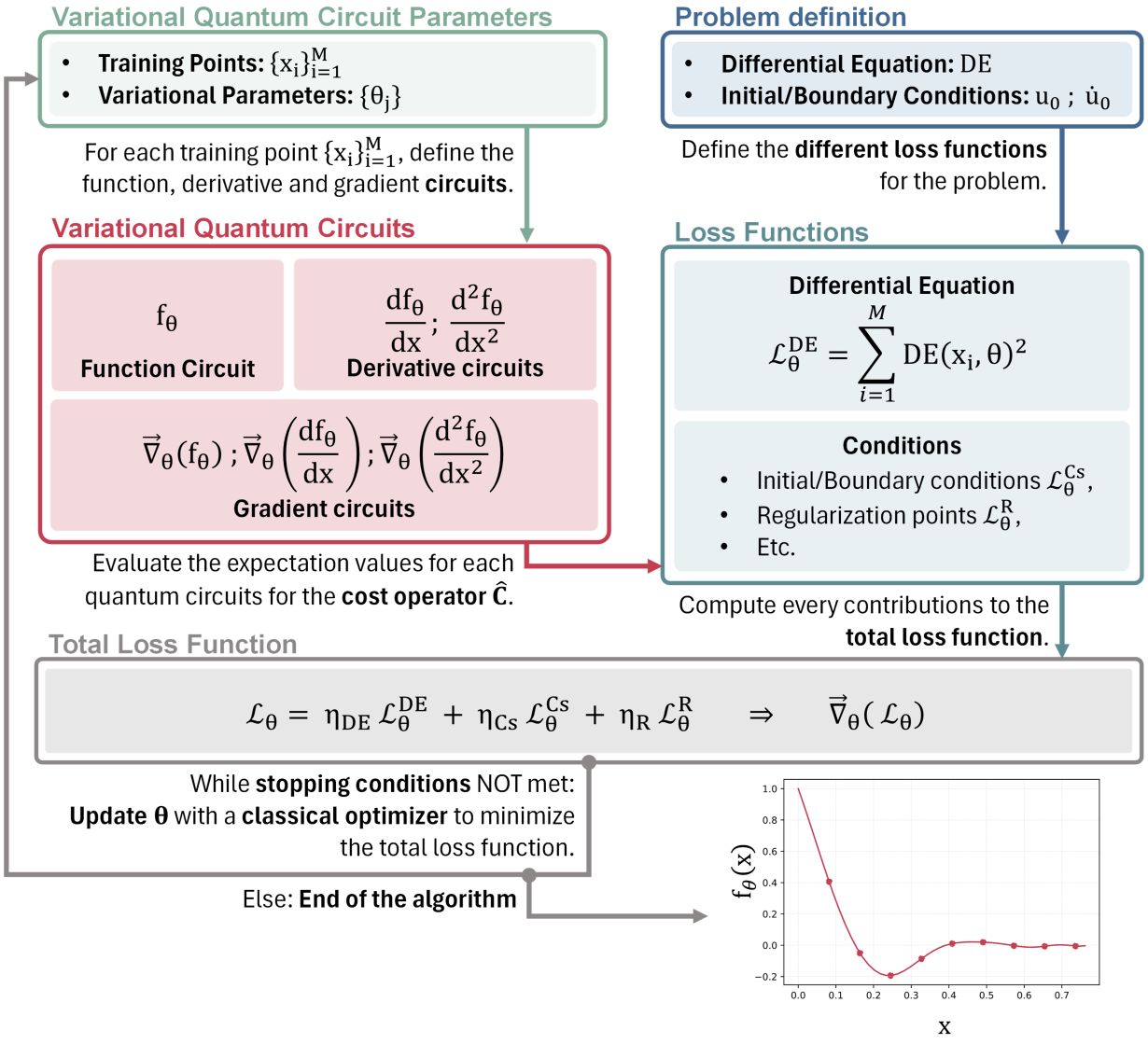


Figure 4.1: Workflow of the proposed variational quantum algorithm, building on the general schematic presented in Fig. 3.1. The quantum computation part is shown in red, whereas all other stages are performed classically.

For both polynomial-encoded VQAs studied here, the quantum circuits are aimed to approximate the solution of a DE. To do so, the algorithms follow a workflow inspired by Kyriienko *et al.* [4],

summarised in Figure 4.1. First, an initial set of quantum circuits and their derivatives with respect to x is built for each point x_i in the chosen interval of interest, with a random vector of variational parameters $\{\theta_i\}$: this is the initial state of the VQA.

The value of the resulting function f_θ is read as an expectation value through a cost operator $\hat{C}$ applied to the quantum state $|f_{\varphi,\theta}(x)\rangle$ of the circuit:

$$f_\theta(x) = \langle f_{\varphi,\theta}(x)|\hat{C}|f_{\varphi,\theta}(x)\rangle\,, \tag{4.1}$$

where φ denotes the encoding function of the variable x.

The classical optimizer then updates the variational parameters based on the evaluated loss. The loss function quantifies how well the VQC output satisfies the differential equation, as well as additional constraints such as initial or boundary conditions and any regularization points.

This work specifically focuses on enhancing the VQC structure to not only encode other polynomials but also to effectively determine the gradient and derivatives at a lower computation cost by comparison to existing approach while not compromising on the quality of the solution, thereby making this new algorithm attractive.

4.1.1 Classical Optimizer

In this work, the selected classical optimizer is Adam [112], presented in the section 3.4.3, while not being perfect, but has demonstrated satisfactory performances in previous VQA studies, including in the Chebyshev-encoded VQA this thesis investigates [4].

What makes Adam interesting, as shown in Algorithm 1, is its adaptive stepsize. The first iteration starts with a step size of α, regardless of the gradient value. The stepsize in then, in the following iterations, increased or decreased, based on the evolution of the gradient, rather than its absolute value. This is a key difference compared to standard gradient descent. Consequently, Adam keeps the step size within a controlled range, which is particularly relevant for the variational parameters used here, which are rotation angles. A standard gradient descent update would follow $\theta_{t+1} = \theta_t - \alpha\nabla_\theta(\mathcal{L})$, and if this update happens to be large enough to wrap around by 2π, the parameter effectively remains

unchanged, leading to redundant updates. The Adam optimizer is used here as it avoids this problem by regulating the magnitude of each step.

As mentioned in the section 3.4.3, the main issue that has been raised with the Adam optimizer [112] is its tendency to be trapped in local minima, especially when dealing with highly parameterized VQCs. Knowing this limitation, efforts were made to work around it by reducing the number of variational parameters, by tuning the optimizer hyperparameters such as the step size, or by running the optimization multiple times, which allows the "resetting" of moment memory, restore the initial step size, and eventually escape from the local minima.

As this optimizer relies on gradient descent optimization, derivative quantum circuits with respect to the variational parameters θ must be calculated. Finally, the vector of variational parameters θ is updated and all circuits are recalculated in a subsequent loop until the resulting function converges to the desired function.

4.1.2 Loss Function

With a similar approach to the Kyriienko *et al.* algorithm [4], the loss function $\mathcal{L}_{DE}$ quantifies the distance between the DE, written with all terms on one side, and zero, over a set of M points within the chosen interval I. This loss function is defined as

$$\mathcal{L}_{\theta}^{DE}\left[\frac{\partial^2 f_\theta}{\partial x^2}, \frac{\partial f_\theta}{\partial x}, f_\theta, x\right] = \frac{1}{M}\sum_{i=1}^{M} L\left(DE\left[\frac{\partial^2 f_\theta}{\partial x^2}(x_i), \frac{\partial f_\theta}{\partial x}(x_i), f_\theta(x_i), x_i\right], 0\right), \tag{4.2}$$

where the distance L is evaluated with the mean square error $L(a,b) = (a-b)^2$. Other distance evaluations can be used, such as the mean absolute error $L(a,b) = |a-b|$. This loss function can be completed with additional functions that quantify the satisfaction of initial or boundary conditions $\mathcal{L}_{Cs}$ or regularization points $\mathcal{L}_R$ [4]. Arbitrary weights η can also be used to prioritise a certain condition over the others, as expressed in the equation

$$\mathcal{L}_\theta = \eta_{DE}\ \mathcal{L}_\theta^{DE} + \eta_{Cs}\ \mathcal{L}_\theta^{Cs} + \eta_R\ \mathcal{L}_\theta^{R}. \tag{4.3}$$

4.1.3 Initial and Boundary Conditions

In the present work, the differential equation can be either an initial value problem (IVP) or a boundary value problem (BVP). In both cases, the associated initial or boundary conditions represent constraints that the solution should satisfy.

These conditions can be taken into account in the optimization loop by contributing to the loss function, or can also be handled separately as an automatic shift of the function read out from the quantum circuit as proposed by Kyriienko *et al.* [4].

Contribution to the loss function

For a second-order DE, the contribution of the initial or boundary conditions to the loss function can be described as follows:

$$\mathcal{L}_\theta^{Cs} = L\left(f_\theta(x_0), u_0\right) + L\left(\frac{\partial f_\theta}{\partial x}(x_0), \dot{u}_0\right), \tag{4.4}$$

where u_0 and $\dot{u}_0$ are the value of the sought function and its derivative at $x = x_0$. For a lower- or higher-order DE, the corresponding terms must be removed or added.

Floating boundary handling

As an alternative approach, a shift term f_{Cs} can be calculated at each iteration to automatically match the initial or boundary condition regardless of the value of the variable parameters θ [4]. Hence, the contribution to the loss function is null ($\mathcal{L}_\theta^{Cs} = 0$), and we have:

$$f_\theta(x) = \langle f_{\varphi,\theta}(x)|\hat{C}|f_{\varphi,\theta}(x)\rangle + f_{Cs}\ , \tag{4.5}$$

$$f_{Cs} = u_0 - \langle f_{\varphi,\theta}(x_0)|\hat{C}|f_{\varphi,\theta}(x_0)\rangle\,. \tag{4.6}$$

This approach is more suitable for the function itself than for its derivatives. Introducing a shift term for each derivative can be counterproductive, as it would require integration to propagate the

shift correctly to the function, potentially leading to inconsistencies. Consequently, while the method guarantees that the boundary or initial condition is satisfied for the function, it can overconstrain the variational quantum circuit and make it challenging to apply conditions to the derivatives.

Additionally, as discussed in Chapter 6, running a quantum circuit on a NISQ device is inherently noisy and can introduce a systematic shift in the function. In this context, the floating boundary handling proposed by Kyriienko *et al.* would provide an effective correction for such shifts.

4.1.4 Regularization

To help the convergence, for equations where certain values of the desired function u_r are known, an additional condition can be included to guide the solution toward these values. This contribution to the loss function can be defined as:

$$\mathcal{L}_\theta^R = \frac{1}{M_R} \sum_{r=1}^{M_R} \left(L(f_\theta(x_r), u_r) \right), \tag{4.7}$$

where M_R is the number of regularization points and u_r the value of the sought function at those points.

A regularization procedure can also be implemented for derivatives values [4]. To minimise the complexity of the algorithm, regularization can be added to the loop by dividing the set of points $\{x_i\}_{i=1}^M$ into subsets of points that will, one by one, be used as training points for the differential equation and then as regularization points once the algorithm converged on this subset. This approach can save an important amount of computing resources but can in certain cases decrease the overall accuracy of the algorithm output.

4.1.5 Cost Operator

Importantly, the value of the approximated function and its derivatives from the different quantum circuits built throughout the algorithm are read out using a cost operator $\hat{C}$.

$$f(x) = \langle f_{\varphi,\theta}(x)| \, \hat{C} \, |f_{\varphi,\theta}(x)\rangle \,, \tag{4.8}$$

where $|f_{\varphi,\theta}(x)\rangle$ is the quantum state of the quantum circuit evaluated at x within the chosen interval. The choice of the cost operator impacts the flexibility and trainability of the algorithm. Most common cost functions are the magnetisation of a single qubit j : $\hat{C} = \hat{Z}_j$, which can be used for the measurement of an ancillary qubit for example, or the total magnetisation of the system, where all qubits are measured: $\hat{C} = \sum_{j=1}^{n} \hat{Z}_j$.

The notation $\hat{Z}_j$, refers to the tensor product $I^{\otimes j} \otimes Z \otimes I^{\otimes(n-j-1)}$.

In this study, the cost functions employed are centred around the total magnetisation of the system, or variations of it.

4.2 Derivatives and Gradients

The two structures of the new VQA proposed in this work aim to represent the function solution of the DE through a quantum circuit. To do so, the derivatives with respect to the variable x are necessary to evaluate the loss function. Both quantum feature maps encode the variable x using a set of functions $\{\varphi_j\}_{j=0}^{n}$ where n is the number of gates used for the quantum feature map encoding. Hence, the first- and second-order derivatives are defined as follow:

$$\frac{df}{dx} = \sum_{j=0}^{n} \frac{\partial}{\partial \varphi_j} \langle f_{\varphi,\theta}| \hat{C} |f_{\varphi,\theta}\rangle \ \frac{d\varphi_j}{dx}, \tag{4.9}$$

$$\frac{d^2 f}{dx^2} = \sum_{j=0}^{n} \frac{\partial}{\partial \varphi_j} \langle f_{\varphi,\theta}| \hat{C} |f_{\varphi,\theta}\rangle \ \frac{d^2\varphi_j}{dx^2} + \sum_{j=0}^{n} \Big(\sum_{h=0}^{n} \frac{\partial^2}{\partial \varphi_j \varphi_h} \langle f_{\varphi,\theta}| \hat{C} |f_{\varphi,\theta}\rangle \ \frac{d\varphi_h}{dx} \Big) \frac{d\varphi_j}{dx}. \tag{4.10}$$

However, in the variational ansatz, the parameter θ is encoded directly into the parameterized rotational gates. Consequently, there is no need to apply the chain rule or rely on automatic differentiation to compute gradients with respect to θ. In this work, all gradients ∇_θ of polynomial-encoded VQAs are evaluated using the Parameter Shift Rule, as defined in Equation (3.6).

In the case of the Lagrange encoding, the derivatives with respect to the variable x can also be obtained using a directly differentiated encoding, which eliminates the constraints imposed by the chain rule. This method is described in more detail in the next chapter, in Section 5.2.5.

4.2.1 Differentiability

In this work, we operate with final results expressed as polynomials, which belong to the class $\mathcal{C}^\infty$. Consequently, they can be differentiated arbitrarily many times without restriction. However, we must also account for the nonlinear encoding functions. The polynomial basis is mapped through a nonlinear encoding function φ, and it is essential to verify the existence and differentiability of this function. Not every function qualifies as an encoding map under these requirements.

For the variational quantum algorithms considered in this work, we use encoding functions φ based on the arccos function, which is well-defined and differentiable over the interval $]-1, 1[$. We also need φ to support the differentiation required for gradient-based optimization. The function arccos belongs to the class $\mathcal{C}^\infty$ on its domain, so it admits derivatives of any order.

That being said, the behaviour of the encoding function, even when it exists, can impact the training process and the interval to consider. The behaviour of the encoding function for both Chebyshev- and Lagrange-encoded VQAs are discussed further in the next chapter. Finally, regarding gradients with respect to the variational parameters θ, here only first-order derivatives are required. Since the trigonometric functions cos and sin are also in $\mathcal{C}^\infty$, they admit differentiation of any order, making higher-order optimization schemes theoretically feasible on that aspect.

4.3 Iterative Process

The iterative process used to train the VQAs in this work relies on the classical optimizer *Adam* [112]. As outlined in the Chapter 3, Adam is a gradient-based optimization algorithm that combines the advantages of momentum and adaptive learning rates to perform efficient parameter updates. It minimises the loss function iteratively by adjusting the trainable parameters θ based on estimates of the first- and second-order moments of the gradients.

The optimization procedure requires a convergence criterion or stopping condition. Since Adam adapts the learning rate according to the behavior of the gradients, the gradient magnitude itself provides information about how much the parameters should change to approach the optimum, which is defined here as minimizing the loss function.

However, relying solely on the gradient is insufficient because it does not capture Adam's adaptive mechanism. For this reason, multiple stopping conditions were implemented to guarantee that the optimization neither runs indefinitely nor wastes computational resources performing iterations with negligible improvements. These criteria ensure both stability (by preventing unnecessary computations once the solution has stabilised) and accuracy (by ensuring the loss function reaches an acceptable value). The stopping conditions are defined as follows:

1. A threshold on the value of the loss function, which is referred to as the precision criterion. If the loss reaches a sufficiently small value, the solution is considered accurate:

$$|\mathcal{L}_\theta| \leq \varepsilon_{\mathcal{L}}. \tag{4.11}$$

2. A condition on the gradient magnitude, ensuring meaningful updates:

$$\left|\frac{\partial \mathcal{L}}{\partial \theta_i}\right| \leq \varepsilon_{\partial \mathcal{L}} \text{ , for each } \theta_i \in \{\theta_i\}_n. \tag{4.12}$$

3. A condition on the variation of the parameters between two successive iterations, to capture the evolution of the step size of Adam and prevents useless updates:

$$\alpha_{i,t} = \Delta\theta_{i,t} = |\theta_{i,t} - \theta_{i,t-1}| \leq \varepsilon_\alpha \text{ , for each } \theta_i \in \{\theta_i\}_n. \tag{4.13}$$

4. Finally, a maximum iteration limit is imposed to handle cases where convergence might never occur, for example, due to the VQC structure (such as the number of qubits or parameters):

$$t \leq t_{\max}. \tag{4.14}$$

The threshold values were selected empirically, based on experience with the differential equations investigated in this work. The values that provided satisfactory convergence in practice are: $\varepsilon_{\mathcal{L}} = 1 \cdot 10^{-4}$, $\varepsilon_{\partial \mathcal{L}} = 1 \cdot 10^{-4}$, $\varepsilon_\alpha = 2 \cdot 10^{-3}$.

4.4 Quantum Circuits Structures

For the two new VQA architectures, the variable function x is encoded through a quantum feature map circuit $\hat{\mathcal{U}}_{\varphi}(x)$ and the vector of variational parameters θ is encoded through a variational ansatz $\hat{\mathcal{U}}_{\theta}$. The VQA inspired by the work of Kyriienko *et al.* [4] is based on two unitaries acting on the same register, one after the other, and all qubits in the register are measured, as shown in Figure 3.5. For the newly designed VQA, two different structures have been studied (Figures 4.2 and 4.3) and in both cases the circuit is inspired by the Hadamard test structure (Figure 3.3), over the ancilla register of one or multiple qubits.

The development of the Lagrange-encoded VQA has been motivated by two main factors. As mentioned in Chapter 1, the initial objective of this PhD was to extend the Chebyshev-encoded VQA to higher-order differential equations. However, in the application to a second-order differential equation, as discussed in Section 7.2, we identified areas for improvement.

Firstly, the computational demands of the parameter-shift rule were significant. Evaluating the first and second derivatives with respect to x, along with their gradients with respect to θ, involved a substantial gate complexity. Secondly, the application to a second-order differential equation revealed oscillations, likely caused by the use of excessively high-degree polynomials for interpolation tasks.

These observations motivated the design of a VQC structure that reduces the computational overhead associated with evaluating derivatives and gradients while implementing Lagrange interpolation, without compromising solution quality or accuracy. Lagrange interpolation is particularly well-suited for smooth functions, and by carefully selecting the encoding points, it is possible to construct interpolating polynomials of minimal degree, thereby avoiding the oscillations observed with higher-degree Chebyshev polynomials.

The first approach of the Lagrange-encoded VQA was initially designed across two quantum registers of the same sizes. However, this extended structure is suboptimal as it necessitates a high number of qubits. Therefore, the Lagrange-encoded structure was re-designed to minimise the required number of qubits, while retaining the same cost function and differentiation method. Intermediate structures where the number of qubits required for the encoding is between 1 and n could also have been considered, but the study of such structure is left for a future work.

4.4.1 Chebyshev-encoded VQA

The first polynomial-encoded studied in this work is the algorithm developed by Kyriienko *et al.*[4]. In this work, we applied it to other first-order differential equations and extended its scope to second-order equations, as introduced in Equation (4.10). The hardware-efficient ansatz originally proposed consists of a sequence of three rotational gates, $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$. In addition, we investigated a simplified sequence based on a single rotational gate, $\hat{R}_X$.

4.4.2 Proposed new Lagrange-encoded VQA

The new algorithm designed and tested in this work has been developed based on the Hadamard test structure (Figure 3.3) but extended to a multiple-qubit register, to approximate the solution of the differential equation by Lagrange interpolation. In this work, two different structures of the quantum feature map have been proposed.

Quantum feature map

First, an extended version (Figure 4.2) has been developed where an encoding block $\hat{\mathcal{U}}_\varphi$, with $\varphi_j(x) = \arccos(x - x_j)$, is applied in between two networks of CNOT gates, over a quantum register of n qubits, leading to:

$$\hat{\mathcal{U}}_\varphi(x) = \bigotimes_{j=1}^{n} \hat{R}_{Y,j}[(\arccos(x - x_j)]. \tag{4.15}$$

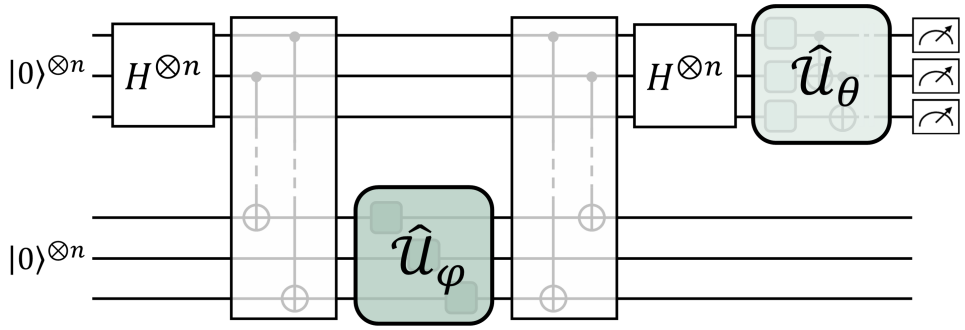


Figure 4.2: Quantum circuit structure of the Lagrange-encoded algorithm with an extended structure.

A second version with a reduced structure (Figure 4.3) has been developed, where only one ancilla qubit is used for the encoding. Here, the variable x is encoded through Y-rotational gates parameterised by

the same encoding function φ_i defined in the extended structure but within the networks of CNOT gates:

$$\hat{\mathcal{U}}_{\varphi}(x) = \Big(\prod_{i=1}^{n-1} \mathrm{CNOT}_{i,a}\ \hat{R}_{Y,a}(\varphi_{i+1}(x))\Big)\mathrm{CNOT}_{n,a}\Big(\prod_{i=1}^{n-1} \mathrm{CNOT}_{i,a}\ \hat{R}_{Y,a}(\varphi_i(x))\Big)\mathrm{CNOT}_{n,a}\ , \tag{4.16}$$

where $\mathrm{CNOT}_{i,a}$ denotes a controlled-NOT gate with control on the i-th qubit and target on the ancilla qubit, and $\hat{R}_{Y,a}(\varphi)$ represents a rotation around the Y-axis applied to the ancilla qubit, defined as $\hat{R}_{Y,a} = I^{\otimes n} \otimes R_Y$.

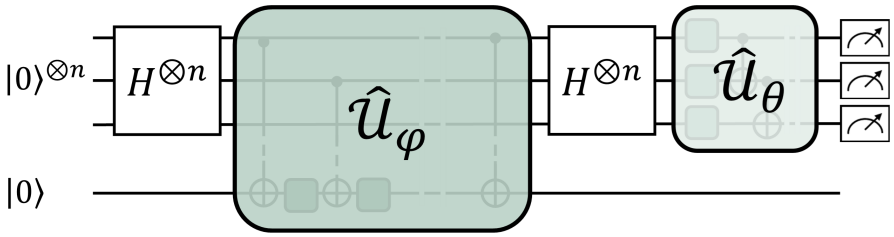


Figure 4.3: Circuit Structure of the Lagrange-encoded algorithm with a reduced structure.

In both structures, the set of points $\{x_j\}_{j=1}^{n}$ included in the interval I used for the polynomial interpolation is directly linked to the size of the first quantum register. While the second register can be built over n qubits (Figure 4.2) or only one qubit (Figure 4.3), for both structures, only the first register is measured where the observable corresponds to the magnetisation of the first n qubits, expressed as a sum of Z_i operators, leading to local measurements. As discussed in section 3.4.4, this allows to delay the appearance of the barren plateau phenomenon.

Additionally, determining whether to use a reduced or extended circuit structure might depend on the quantum device being employed. The extended circuit structure requires more qubits but fewer gates in comparison to its reduced counterpart, described in Chapter 5. Specifically, the reduced circuit structure saves $n-1$ qubits, but requires additional steps to be computed. Consequently, they would potentially both have different limitations and advantages when applied to real quantum devices. While the extended would allow a lower number of steps, it requires a linear connectivity and involves a significantly larger number of CNOT gates, which have lower fidelities than single-qubit gates. In contrast, the reduced version requires more steps and a one-to-all connectivity, it uses fewer CNOT gates but more single-qubit gates. In the realm of quantum circuit simulations, the preference leans towards the reduced circuit structure due to its lower number of qubits and consequently lower computational resource demands (as both circuit provides the same solution on noise-free qubits).

Variational ansatz

In this VQA, the variational ansatz $\hat{\mathcal{U}}_\theta$ is composed of one layer.

Cost operator

For both circuit structures to be read as a Lagrange interpolating polynomial, the first register is evaluated by the same cost operator:

$$\hat{C} = \sum_{j=1}^{n} \frac{1}{\rho_j} \hat{Z}_j, \text{ with } \rho_j = \langle f_\varphi(x_j)| \hat{Z}_j |f_\varphi(x_j)\rangle . \tag{4.17}$$

Here, $|f_\varphi(x_j)\rangle$ denotes the quantum state obtained by applying only the quantum feature map.

The resulting function is a Lagrange interpolating polynomial:

$$f(x) = \sum_{j=1}^{n} \left(\alpha_j(\theta) \prod_{i \neq j}^{n} \frac{x - x_i}{x_j - x_i} \right), \tag{4.18}$$

where α_θ is a coefficient determined by the variational ansatz. The detail of the calculation is given in the Chapter 5.

Encoding points

The choice of encoding points, also referred to as interpolation points, will be analysed and discussed in the next two chapters. When working with interpolation, the selection of the interpolation abscissas $\{x_j\}_n$ is at least as important as the number of points, and consequently the degree of the polynomial, for achieving a good fit and accuracy. One of the well-known issues in polynomial interpolation is the Runge phenomenon, where oscillations appear at the interval boundaries. A classical solution to mitigate this problem is to use Chebyshev nodes as interpolation points.

In this work, we considered different sets of encoding points to determine which configuration provides the best performance for a given number of points. Beyond this, one could also imagine optimizing the positions of the encoding points dynamically, particularly during the final stage of training, to further improve the fit and tailor the variational quantum circuit to the specific problem. Some works

in the literature have explored similar ideas by training the quantum feature map, such as the work of Jaderberg *et al.* [145].

Training the encoding points would not be computationally demanding on the quantum side, as we can reuse the partial derivatives already computed for the first-order derivative with respect to x. By applying the chain rule, we obtain:

$$\frac{df}{dx_j} = \frac{\partial f}{\partial \varphi_j} \cdot \frac{d\varphi_j}{dx_j}, \tag{4.19}$$

$$\frac{d\varphi_j}{dx_j} = -\frac{d\varphi_j}{dx}. \tag{4.20}$$

This allows the construction of the gradient of the loss function with respect to the encoding points to optimize them. However, this approach can increase the burden on the classical optimization routine.

4.5 Summary

In this chapter, we provided an overview of polynomial-encoded VQAs for solving differential equations using a quantum spectral method, describing their workflow and key components. We presented the Chebyshev-encoded approach introduced by Kyriienko *et al.* [4], along with the alternative encoding developed in this work, which implements the Lagrange polynomial basis into the quantum state.

Although both approaches share several conceptual aspects, the Lagrange-encoded VQA (in its extended or reduced structure) differs fundamentally in the design of the quantum feature map and the variational ansatz. In this method, strong entanglement occurs within the feature map, whereas the ansatz introduces no entanglement. On the contrary, in the Chebyshev-encoded VQA, the entanglement is introduced by the hardware-efficient ansatz and not by the feature map.

The Lagrange-encoded VQA developed in this work was designed with two main objectives. The first is to reduce the computational overhead by leveraging the Hadamard-test differentiation technique for evaluating derivatives and by decreasing the number of variational parameters, thereby reducing both circuit depth and the computational cost of gradient evaluation during optimization. The second objective is to exploit the properties of Lagrange interpolation, which tends to generate interpolating polynomials of minimal degree. This characteristic can help mitigate oscillations near the boundaries of the domain, as observed when studying the application of the Chebyshev-encoded approach to a second-order differential equation in section 7.2.

The next chapter presents a detailed comparison of the two encoding strategies at the circuit level.

Chapter 5

Analysis of the Polynomial-based Variational Quantum Algorithms

In this work, the focus in on variational quantum algorithms implementing a polynomial approximation. As presented in Chapters 3 and 4, these algorithms can be divided into two parts: an encoding block and a variational block. While the encoding part will set a polynomial basis for the polynomial approximation of the solution of a considered differential equation, the variational block can be understood as a quantum neural network, trained throughout the algorithm via a classical optimizer. This research has been motivated by the promising results from Kyriienko *et al.* [4], using Chebyshev polynomials for the encoding.

5.1 Chebyshev-encoded VQA

5.1.1 Function circuits

In their work, Kyriienko *et al.* [4] introduced quantum feature maps encodings based on Chebyshev polynomials. They highlighted the efficiency of two specific strategies: one where each qubit encodes a Chebyshev polynomial of the same degree, and another where the degrees increase across qubits, which they refer to as the *Chebyshev tower* encoding.

The encoding function used here, $\varphi_j(x) = 2j \arccos(x)$, is a nonlinear, smooth, and strictly decreasing

function defined on the interval $[-1, 1]$. It maps real inputs to angles in $[0, 2j\pi]$, where $j \in \mathbb{N}^*$ indexes the qubit or the corresponding encoding gate, since each qubit is associated with a distinct encoding.

As presented in Section 3.5.1, their encoding block is constructed using the encoding function $\varphi_j(x)$, applied to each qubit via a rotation around the Y-axis, denoted by $\hat{R}_Y$. This rotation maps the input x onto a quantum state whose amplitudes correspond to the Chebyshev polynomials of the first and second kind, respectively $T_j(x)$ and $U_{j-1}(x)$. Explicitly, the one-qubit quantum state $|\psi\rangle$ shown in the Figure 5.1 is:

$$\begin{aligned} |\psi\rangle = \hat{R}_Y\big(\varphi_j(x)\big)\,|0\rangle = \;& \cos(j\arccos(x)) && |0\rangle + && \sin(j\arccos(x))\;|1\rangle \\ =\; & T_j(x) && |0\rangle + && \sqrt{1-x^2}\,U_{j-1}(x)\;|1\rangle\,. \end{aligned} \tag{5.1}$$

Finally, a Z measurement of the quantum state $|\psi\rangle$ of the encoded qubit gives :

$$\begin{aligned} \langle\psi|\,\hat{Z}\,|\psi\rangle &= \cos(j\arccos(x))^2 - \sin(j\arccos(x))^2 \\ &= \cos(2j\arccos(x)) \\ &= T_{2j}(x). \end{aligned} \tag{5.2}$$

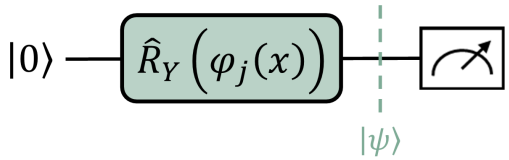


Figure 5.1: One-qubit Chebyshev quantum feature map.

For multiple qubits, the Chebyshev feature map unitary $\hat{\mathcal{U}}_\varphi(x)$ with increasing degrees is defined in Chapter 3, in Equation (3.13). The cost function used to evaluate the encoding from the quantum circuit is the total magnetisation $\hat{C} = \sum_{j=1}^{n} \hat{Z}_j$ of the circuit $|f_\varphi(x)\rangle = \hat{\mathcal{U}}_\varphi\,|0\rangle^{\otimes n}$.

While the quantum state is composed of a superposition of two polynomials of degree j, the $\langle\hat{Z}\rangle$ measurement will be of degree $2j$. For example, in a Chebyshev Tower quantum feature map encoding block over $n = 4$ qubits, the measurement of the j^{th} qubit $\langle\hat{Z}_j\rangle = \langle f_\varphi|\,\hat{Z}_j\,|f_\varphi\rangle$ will give $T_{2(j+1)}(x)$, as illustrated in the Figure 5.2 a), and hence the total measurement of the circuit $\langle\hat{Z}_{tot}\rangle = \sum_{j=1}^{n}\langle\hat{Z}_j\rangle$.

To approximate the solution of a given differential equation, the strategy consists of introducing a set

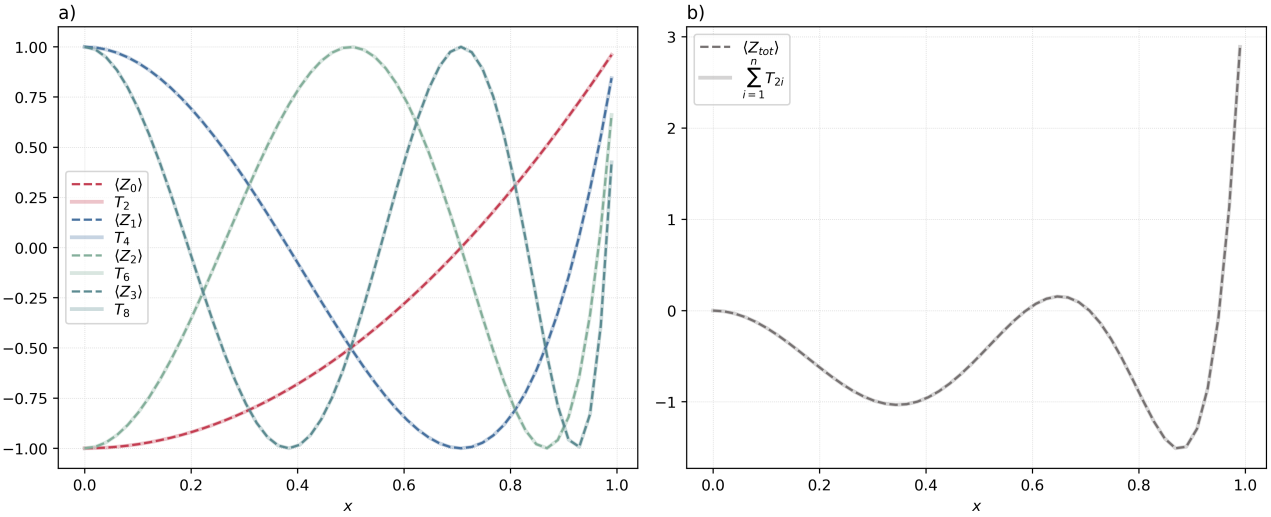


Figure 5.2: Comparison between quantum observables and Chebyshev polynomials of the first kind:
a) Single qubit measurement $\langle \hat{Z}_j \rangle$ and corresponding Chebyshev polynomials.
b) Total Z measurement $\langle \hat{Z}_{tot} \rangle$ and sum of the Chebyshev polynomials.

of weights prior to measurement, enabling an interpolation over the encoded polynomial basis. This is precisely the role of the variational ansatz.

Entangled Chebyshev quantum feature map

The Chebyshev Tower quantum feature map over n qubits encodes the Chebyshev polynomial basis of the polynomial ring of $\mathbb{R}_n[x]$, excluding the scalar term, which is addressed separately when implementing boundary conditions. As illustrated in Figure 5.2 b), the total measurement of the quantum feature map spans only the subspace of even polynomials in $\mathbb{R}_{2n}[x]$. Beyond introducing trainable weights, the hardware-efficient ansatz also enables polynomial multiplication through the use of CNOT gates. In an example of a two qubits Chebyshev encoding with a CNOT gate, the circuit would be:

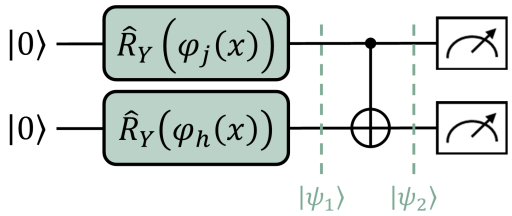


Figure 5.3: Entangled two-qubits Chebyshev encoding.

Prior to the CNOT gate, the statevector $|\psi_1\rangle$ is simply defined as:

$$|\psi_1\rangle = \left(T_j(x)\,|0\rangle + \sqrt{1-x^2}\;U_{j-1}(x)\,|1\rangle\right) \otimes \left(T_h(x)\,|0\rangle + \sqrt{1-x^2}\;U_{h-1}(x)\,|1\rangle\right), \tag{5.3}$$

And consequently, as explained earlier, the Z-measurements of the first and second qubit give respectively, $T_{2j}(x)$ and $T_{2h}(x)$.

However, after entanglement through the CNOT gate, the quantum state becomes:

$$\begin{aligned} |\psi_2\rangle = T_j(x)\;T_h(x) \qquad & |00\rangle + T_j(x)\sqrt{1-x^2}\;U_{h-1}(x)\,|01\rangle \\ + (1-x^2)\;U_{j-1}(x)U_{h-1}(x) \qquad & |10\rangle + T_h(x)\sqrt{1-x^2}\;U_{j-1}(x)\,|11\rangle\,, \end{aligned} \tag{5.4}$$

and gives the following Z-measurement: $\langle \hat{Z}_0 \rangle = T_{2j}(x)$ and $\langle \hat{Z}_1 \rangle = T_{2j}(x)\;T_{2h}(x)$. This illustrates how entanglement enriches the set of accessible functions by generating products of Chebyshev polynomials.

Hardware Efficient Ansatz on the Chebyshev quantum feature map

The final component of the circuit and of the variational ansatz, is the parameterised rotational gates. The function circuit becomes $|f_{\varphi,\theta}\rangle = \hat{\mathcal{U}}_\theta\;\hat{\mathcal{U}}_\varphi(x)\,|\varnothing\rangle$, and the Z-measurement of the j^{th} qubit $\langle \hat{Z}_j \rangle = \langle f_{\varphi,\theta}|\,\hat{Z}_j\,|f_{\varphi,\theta}\rangle$.

Continuing with the two-qubit example, a two-layer hardware-efficient ansatz using $\hat{R}_X$ gates only is illustrated in Figure 5.4.

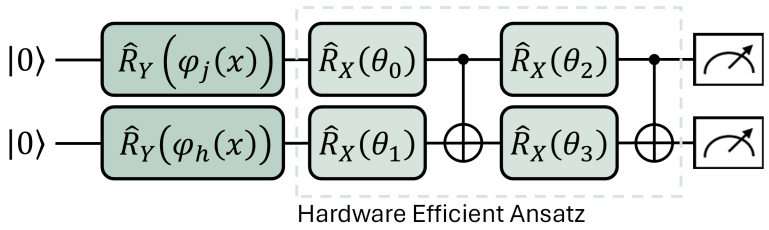


Figure 5.4: Two-qubits Chebyshev encoding and two-layers HEA

This circuit gives the following measurements:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= T_{2j}(x)\Big(\cos(\theta_0)\cos(\theta_2)\Big) - T_{2j}(x)\,U_{2h-1}(x)\sqrt{1-x^2}\Big(\sin(\theta_0)\sin(\theta_2)\Big) \\ \langle \hat{Z}_1 \rangle &= T_{2h}(x)\Big(\cos(\theta_2)\cos(\theta_1+\theta_3)\Big) - T_{2h}(x)\,U_{2j-1}(x)\sqrt{1-x^2}\Big(\sin(\theta_2)\sin(\theta_1+\theta_3)\Big).\end{aligned} \tag{5.5}$$

The variational parameters introduced through the rotational gates act as weights for the family of polynomials generated by the encoding, as well as their products. In the specific case of measuring the expectation value of the Pauli-Z operator on the first qubit, denoted by $\langle \hat{Z}_0 \rangle$, the output comprises a sum of two terms: one involving a Chebyshev polynomial of the first kind, $T_{2j}(x)$, and the other a product of Chebyshev polynomials of both the first and second kinds, $T_{2j}(x)U_{2h-1}(x)\sqrt{1-x^2}$. The resulting expression is thus a polynomial whose terms have degrees $2j$ and $4jh$ respectively. A similar structure appears when measuring $\langle \hat{Z}_1 \rangle$ on the second qubit, leading to terms of degree $2h$ and $4jh$.

As a consequence, the outcome of the circuit, given by the sum of the Z-expectation values across all qubits:

$$f(x) = \langle f_{\varphi,\theta} | \hat{C} | f_{\varphi,\theta} \rangle = \sum_{j=1}^{n} \langle \hat{Z}_j \rangle = \langle \hat{Z}_{tot} \rangle \,, \tag{5.6}$$

results in a polynomial that belongs to a specific subset of the four terms identified in Equation (5.5).

Notably, the presence of entanglement between the qubits alters the degree structure of the resulting polynomial. The encoding of one qubit influences the output measured on the other, a direct consequence of the shared variational parameters and their associated trigonometric dependencies. In particular, some of these weights, derived from the sine and cosine functions in the HEA, appear symmetrically in both measurements due to the entangling operations. For instance, when $\theta_2 = 0$, the sine term vanishes, effectively removing the entangling operation and thereby eliminating the higher-degree contributions. The expressions then reduce to:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= T_{2j}(x)\cos(\theta_0), \\ \langle \hat{Z}_1 \rangle &= T_{2h}(x)\cos(\theta_1+\theta_3).\end{aligned} \tag{5.7}$$

This reduction can also be interpreted from the structure of the circuit: the removal of the intermediate rotation between the two CNOT gates causes the entire entangling sequence to collapse, as the CNOT

gate is its own inverse, $\text{CNOT}^\dagger = \text{CNOT}$.

Hence, this encoding followed by a hardware-efficient ansatz is designed not to span the full ring of polynomials up to a given degree, but rather to interpolate the solution of the differential equation over a carefully structured subset of polynomials. This subset is determined by the encoding, the number of qubits, the depth of the variational circuit, and the specific values taken by the variational parameters.

5.1.2 Derivative circuits

Once the function circuit is established, the idea is now to obtain the its derivatives and gradients to evaluate the differential equation over a chosen set of training points. As presented in Section 3.5.1, the circuit derivatives for Chebyshev-encoded quantum circuits can be obtained using the parameter shift rule, as described by Kyriienko *et al.*[4]. Consequently, the derivative with respect to the input variable x is computed in two steps.

First, the parameter shift rule yields the derivative of the circuit output with respect to the encoding functions $\{\varphi_j(x)\}_{j=1}^n$. Then, the total derivative with respect to x is obtained by applying automatic differentiation. This two-step differentiation process is illustrated in Figures 5.5, 5.6.

The parameter shift rule produces the derivative of the circuit with respect to the given encoding function φ_j. Here, j denotes the index of the encoding function and h the index of the measured qubits:

$$\frac{\partial}{\partial\varphi_j}\langle f_{\varphi,\theta}|\,\hat{Z}_h\,|f_{\varphi,\theta}\rangle = \frac{1}{2}\left(\langle f_{\varphi_j^+,\theta}|\,\hat{Z}_h\,|f_{\varphi_j^+,\theta}\rangle - \langle f_{\varphi_j^-,\theta}|\,\hat{Z}_h\,|f_{\varphi_j^-,\theta}\rangle\right). \tag{5.8}$$

To avoid confusion, the Z-measurement of the h^{th} qubit in the function circuit will be denoted as $\langle\hat{Z}_h\rangle = \langle f_{\varphi,\theta}|\,\hat{Z}_h\,|f_{\varphi,\theta}\rangle$. In contrast, the measurements of modified circuits used in the parameter-shift rule are written explicitly as $\langle f_{\varphi_j^\pm,\theta}|\,\hat{Z}_h\,|f_{\varphi_j^\pm,\theta}\rangle$ where the subscript $\varphi_j^\pm$ indicates that the gate encoding φ_j has been shifted by $\pm\pi/2$, as specified by the sign.

First-Order derivative of the encoding circuit

For the block encoding only, and without entanglement, for each $j \neq h$, $\frac{\partial}{\partial \varphi_j} \langle f_\varphi | \hat{Z}_h | f_\varphi \rangle = 0$. Consequently, as shown in Figure 5.5,

$$\frac{\partial}{\partial \varphi_j} \langle f_\varphi | \hat{Z}_j | f_\varphi \rangle = \frac{\partial T_{2(j+1)}}{\partial \varphi_j}. \tag{5.9}$$

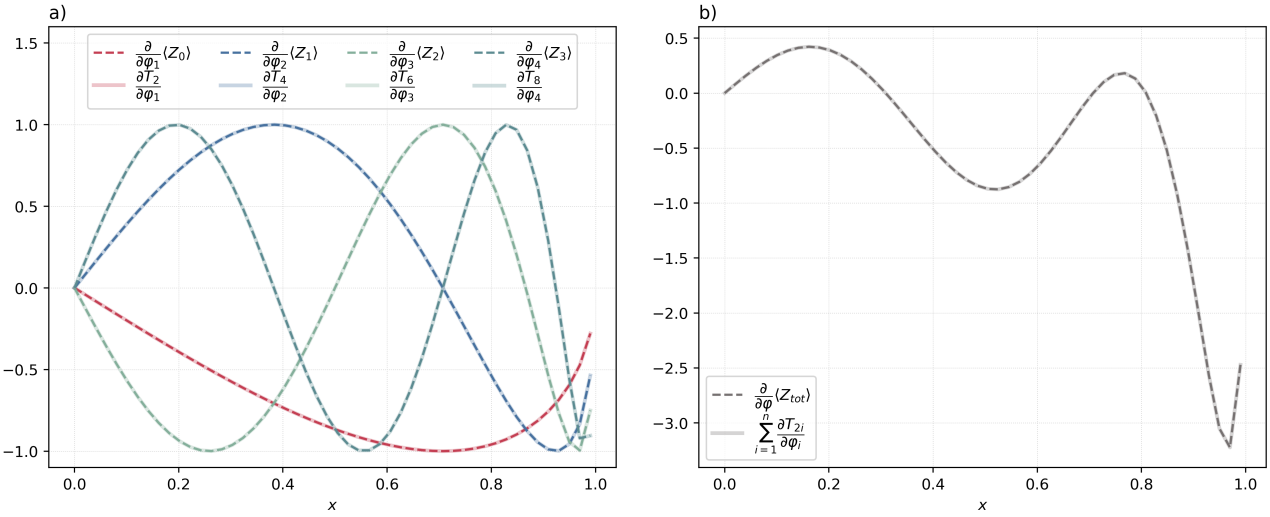


Figure 5.5: Comparison between parameter shift rule outcomes and analytical derivatives of Chebyshev polynomials with respect to the encoding function φ_j.
a) $\langle \hat{Z}_j \rangle$ measurements on each qubit.
b) Sum over all qubits of both quantum-derived and analytical derivatives.

Consequently, applying the automatic differentiation to the parameter shift rule outputs gives the total derivative of the encoded polynomial with respect to x:

$$\frac{d}{dx} \langle f_\varphi | \hat{Z}_j | f_\varphi \rangle = \frac{\partial}{\partial \varphi_j} \langle f_\varphi | \hat{Z}_j | f_\varphi \rangle \frac{d\varphi_j}{dx} = \frac{dT_{2(j+1)}}{dx}. \tag{5.10}$$

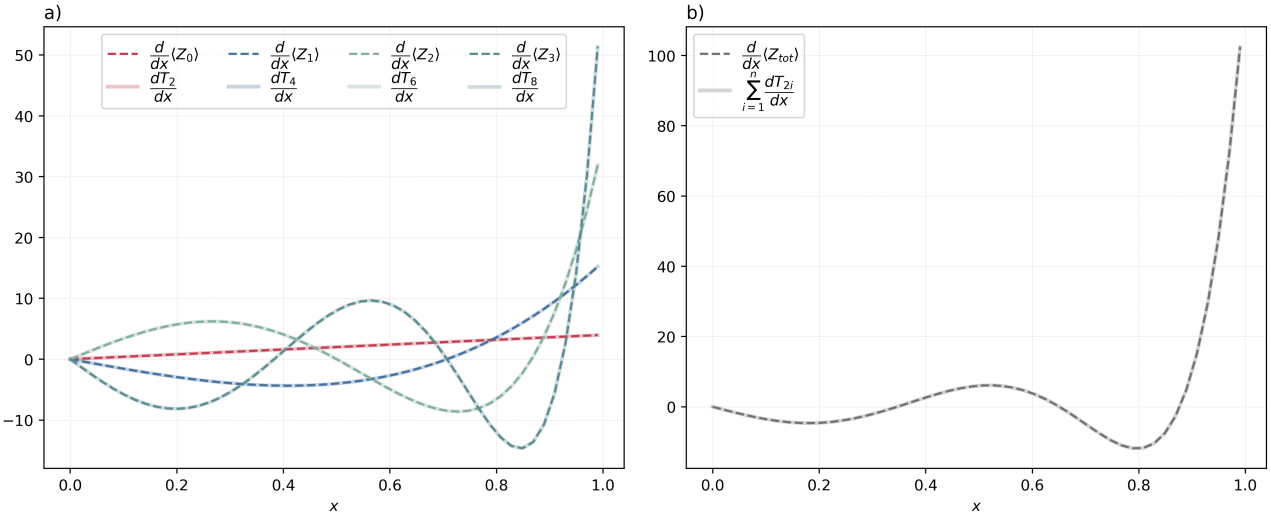


Figure 5.6: Gradient of the quantum expectation value with respect to the input variable x, obtained via automatic differentiation.
a) $\langle \hat{Z}_j \rangle$ measurements on each qubit, differentiated with respect to x.
b) Sum across all qubits, compared to the theoretical derivative of the full encoded polynomial.

First-Order derivative of the parametrised circuit

However, Equation (5.10), illustrated in Figure 5.6, only captures the differentiation of the encoding. As shown earlier with the function circuit, once entanglement is introduced, the encoding of a single qubit can influence the measurement outcomes of the other qubits. Consequently, for each $j \neq h$, $\frac{\partial}{\partial \varphi_j} \langle f_{\varphi,\theta}| \hat{Z}_h |f_{\varphi,\theta}\rangle$ is not necessarily zero. The total derivative of the full parameterized circuit can therefore be expressed as:

$$\frac{df}{dx} = \sum_{j=1}^{n} \left[\sum_{h=0}^{n-1} \frac{\partial}{\partial \varphi_j} \langle \hat{Z}_h \rangle \right] \frac{d\varphi_j}{dx} = \sum_{j=1}^{n} \frac{\partial}{\partial \varphi_j} \langle \hat{Z}_{tot} \rangle \ \frac{d\varphi_j}{dx} \tag{5.11}$$

Behaviour of the encoding function derivative

Additionally, the encoding function and its derivatives have to be taken into account in the realisation of the VQA, even if handled on the classical part of the algorithm. Figure 5.7 displays the set of encoding functions $\{\varphi_j(x)\}_{j=1}^{n}$, which vary across qubits in the Chebyshev tower encoding, alongside their analytical derivatives:

$$\frac{d\varphi_j}{dx}(x) = \frac{-2j}{\sqrt{1-x^2}}. \tag{5.12}$$

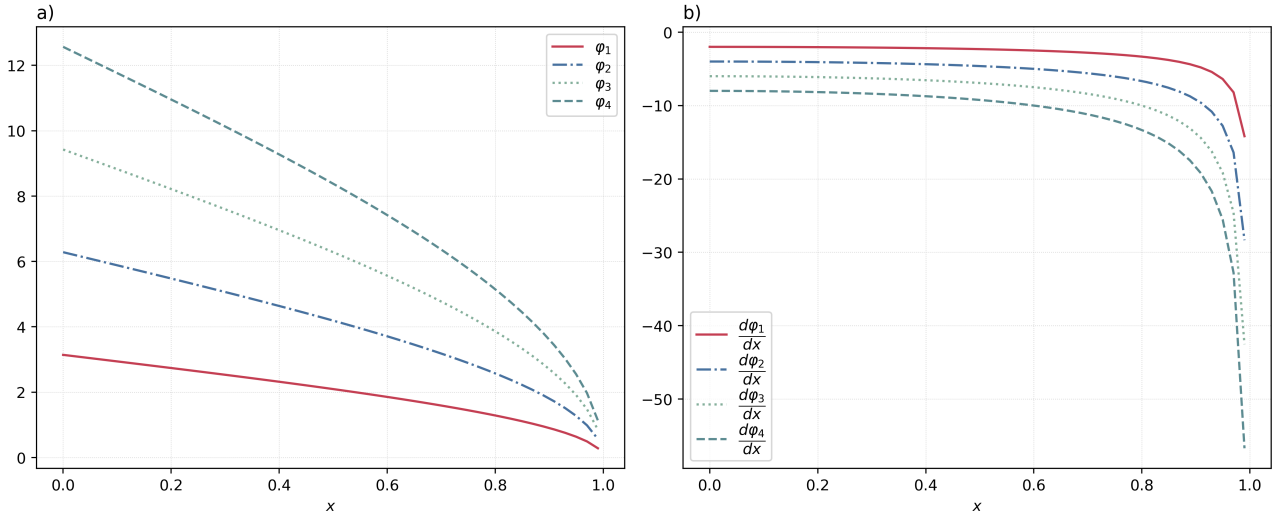


Figure 5.7: a) Plot of the encoding functions φ_j applied to each qubit, for $j = 1$ to 4.
b) Corresponding analytical derivatives $\frac{d\varphi_j}{dx}$, used for the automatic differentiation.

The Figure illustrates both the spatial variation of the encoding functions and the divergence of the derivatives as x approaches 1:

$$\forall\ j \in \mathbb{N}^*,\ \lim_{x \to 1^-} \frac{d\varphi_j}{dx}(x) = -\infty. \tag{5.13}$$

This behaviour shows an important phenomenon: although any outcomes of $\langle \hat{Z}_j \rangle$ measurements are confined to the interval $[-1, 1]$, and in this context only the positive part $[0, 1]$ is relevant, automatic differentiation remains sensitive to the divergence of the encoding function's derivative. In particular, near $x = 1$, the sharp decrease in $\frac{d\varphi_j}{dx}$ introduces numerical instability. Consequently, while the quantum measurement outcomes themselves do not exhibit divergent behaviour, the derivatives of encoding functions may compromise the accuracy of the result and affect the training dynamics of the variational circuit. For this reason, the training is realised over the interval $[0, 0.9]$ in this work and in the paper that introduced this method [4].

Higher Order derivatives

In their work [4], Kyriienko *et al.* applied their algorithm to first-order differential equations. In this work, we extend the approach to second-order differential equations. The higher-order derivatives can be obtained by reapplying the parameter shift rule and automatic differentiation. For the second-order

derivative of the overall function represented by the parametrised circuit f, the equation becomes:

$$\begin{aligned}\frac{d^2f}{dx^2} &= \sum_{j=1}^{n}\left[\sum_{h=0}^{n-1}\frac{\partial}{\partial\varphi_j}\langle\hat{Z}_h\rangle\right]\frac{d^2\varphi_j}{dx^2} \qquad + \sum_{j=1}^{n}\sum_{k=0}^{n-1}\left[\sum_{h=0}^{n-1}\frac{\partial^2}{\partial\varphi_j\partial\varphi_k}\langle\hat{Z}_h\rangle\right]\frac{d\varphi_j}{dx}\frac{d\varphi_k}{dx} \\ &= \sum_{j=1}^{n}\frac{\partial}{\partial\varphi_j}\langle\hat{Z}_{tot}\rangle\frac{d^2\varphi_j}{dx^2} \qquad + \sum_{j=1}^{n}\sum_{k=0}^{n-1}\frac{\partial^2}{\partial\varphi_j\varphi_k}\langle\hat{Z}_{tot}\rangle\frac{d\varphi_j}{dx}\frac{d\varphi_k}{dx}. \end{aligned} \tag{5.14}$$

The second-order partial derivatives of the expectation values $\langle\hat{Z}_h\rangle$ are computed by applying the parameter shift rule twice. For (j,k) indices of the encoding gates and h index the measured qubit,

$$\begin{aligned}\frac{\partial^2}{\partial\varphi_k\partial\varphi_j}\langle\hat{Z}_h\rangle &= \frac{\partial}{\partial\varphi_j}\left(\frac{\partial}{\partial\varphi_k}\langle\hat{Z}_h\rangle\right) \\ &= \frac{\partial}{\partial\varphi_j}\left(\frac{1}{2}\left(\langle f_{\varphi_k^+,\theta}|\,\hat{Z}_h\,|f_{\varphi_k^+,\theta}\rangle - \langle f_{\varphi_k^-,\theta}|\,\hat{Z}_h\,|f_{\varphi_k^-,\theta}\rangle\right)\right) \\ &= \frac{1}{4}\Big(\langle f_{\varphi_{jk}^{++},\theta}|\,\hat{Z}_h\,|f_{\varphi_{jk}^{++},\theta}\rangle - \langle f_{\varphi_{jk}^{+-},\theta}|\,\hat{Z}_h\,|f_{\varphi_{jk}^{+-},\theta}\rangle \\ &\quad - \langle f_{\varphi_{jk}^{-+},\theta}|\,\hat{Z}_h\,|f_{\varphi_{jk}^{-+},\theta}\rangle + \langle f_{\varphi_{jk}^{--},\theta}|\,\hat{Z}_h\,|f_{\varphi_{jk}^{--},\theta}\rangle\Big). \end{aligned} \tag{5.15}$$

And the expression is symmetric in the indices:

$$\frac{\partial^2}{\partial\varphi_k\partial\varphi_j}\langle\hat{Z}_h\rangle = \frac{\partial^2}{\partial\varphi_j\partial\varphi_k}\langle\hat{Z}_h\rangle. \tag{5.16}$$

In the special case where $j=k$, the same gate is shifted twice by $\pm\pi/2$, yielding:

$$\frac{\partial^2}{\partial\varphi_j^2}\langle\hat{Z}_h\rangle = \frac{1}{4}\Big(\langle f_{\varphi_j^{+\pi},\theta}|\,\hat{Z}_h\,|f_{\varphi_j^{+\pi},\theta}\rangle - 2\langle\hat{Z}_h\rangle + \langle f_{\varphi_j^{-\pi},\theta}|\,\hat{Z}_h\,|f_{\varphi_j^{-\pi},\theta}\rangle\Big). \tag{5.17}$$

And the second-order derivative of the encoding function is:

$$\frac{d^2\varphi_j}{dx^2}(x) = \frac{-2jx}{(1-x^2)^{3/2}} \tag{5.18}$$

Just like the first-order derivative of the encoding function, and as illustrated in the Figure 5.8, the second-order derivative of φ_j diverges as x approaches 1:

$$\forall\, j\in\mathbb{N}^*,\ \lim_{x\to1^-}\frac{d^2\varphi_j}{dx^2}(x) = -\infty. \tag{5.19}$$

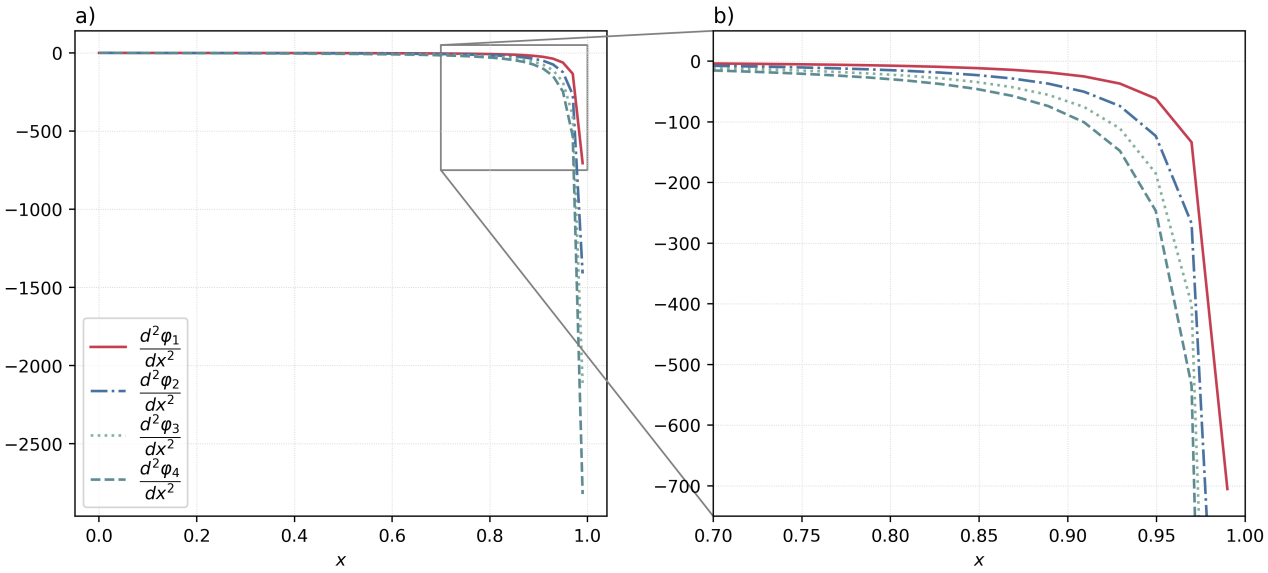


Figure 5.8: a) Second-order derivatives $\frac{d^2\varphi_j}{dx^2}$ for $j = 1$ to 4, highlighting the divergence as $x \to 1^-$. b) Zoom-in on the region $x \in [0.7, 1]$, showing the small differences between the second-order derivatives.

5.2 Lagrange-encoded VQA

In addition to the study of the Chebyshev-encoded VQA proposed by Kyriienko *et al.* [4], this thesis investigates how alternative encoding strategies and entanglement patterns could enhance the previously introduced approach. As outlined in the preceding chapter, we developed two circuit architectures that, although theoretically equivalent in terms of output, may offer distinct advantages or drawbacks when implemented on actual quantum hardware. These differences arise from variations in qubit count, circuit depth, gate complexity, and connectivity.

5.2.1 Function circuit

Encoding function

The encoding function is defined as $\varphi_j(x) = \arccos(x - x_j)$, where x_j denotes a centre parameter specific to each qubit or encoding gate. This function is nonlinear, smooth, and strictly decreasing in x, with domain $x \in [-1, 1]$. However, here x is restricted in the interval $[0, 0.9]$, to ensure consistency between the two polynimial-based approaches. The argument of the arccosine, $(x - x_j)$, is thus

constrained to the interval $[-0.9, 0.9]$. To compare the behaviour of different input distributions for encoding, three types of point sets over the interval $[0, 0.9]$ are considered: equidistant points, Chebyshev nodes of the first kind, and a variant using only positive Chebyshev nodes.

The first set consists of equispaced points, defined as n uniformly distributed values between 0 and 0.9. Although simple to construct, this distribution is known to suffer from large interpolation errors at the boundaries due to Runge's phenomenon, which appears when interpolating over equispaced points induces oscillations near the endpoints [146].

The remaining two sets are based on Chebyshev nodes, recognised as optimal for avoiding the Runge phenomenon [146]. These nodes are derived from the extrema and roots of Chebyshev polynomials, and provide a distribution that naturally clusters more points toward the boundaries.

The first Chebyshev set, $\{x_i^{(1)}\}_i$, corresponds to the classical Chebyshev nodes (originally defined on $[-1, 1]$), rescaled to an arbitrary interval $[a, b]$, here chosen as $a = 0$ and $b = 0.9$ for both encoding constraints and numerical stability. It is defined as:

$$x_k^{(1)} = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k-1}{2N_{\text{points}}}\pi\right), \quad 0 \leq k < N_{\text{points}}. \tag{5.20}$$

The second Chebyshev set, $\{x_i^{(2)}\}_i$, includes only the positive Chebyshev nodes scaled to the interval $[0, b]$, again with $b = 0.9$. This yields:

$$x_k^{(2)} = b \cos\left(\frac{2k-1}{4(N_{\text{points}}-1)}\pi\right), \quad 0 \leq k < N_{\text{points}}. \tag{5.21}$$

These three sets produce visibly distinct point distributions, as soon as $n \geq 4$. This is illustrated in Figure 5.9 for $N_{\text{points}} = 10$, highlighting the boundary clustering characteristic of Chebyshev-based configurations compared to the uniform spacing of the equidistant case.

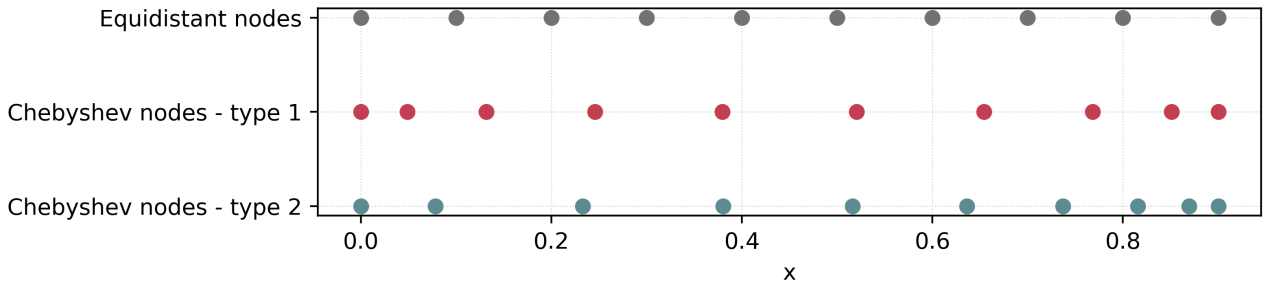


Figure 5.9: Comparison of the three types of sets of encoding points for $N_{\text{points}} = 10$.

Inspiration from the Hadamard test structure

The Lagrange polynomial encoding employed in both circuit structures is inspired by the Hadamard test, illustrated in Figure 3.3.a. This approach is particularly advantageous due to its differentiation techniques that require only a minimal number of circuit evaluations.

In the classical Hadamard test, the expectation value of the Pauli-Z operator measured on the control qubit corresponds to $Re(\langle a|\hat{\mathcal{U}}|a\rangle)$. To adapt this principle for Lagrange encoding, the controlled unitary is replaced by a network of CNOT and $\hat{R}_Y$ gates, allowing an efficient encoding of the Lagrange basis while avoiding redundancy of gates. The underlying structure of the quantum feature map shown in Figure 5.10, can be made explicit by rewriting the Hadamard test as follows:

$$|q_0, a\rangle = (H \otimes I) \cdot \text{CNOT}_{0,a} \cdot (I \otimes \hat{R}_Y(\varphi_j(x)) \cdot \text{CNOT}_{0,a} \cdot (H \otimes I) \ \ |0, a\rangle \ \ , \tag{5.22}$$

where $\text{CNOT}_{0,a}$ denotes a controlled-NOT gate with control on the first qubit and target on the ancilla qubit. This circuit first prepares a Bell state and then introduces the interpolation point x_j through the function $\varphi_j = \arccos(x - x_j)$. The resulting quantum state is :

$$|q_0, a\rangle = \cos(\varphi_j/2)\,|00\rangle + \sin(\varphi_j/2)\,|11\rangle\,. \tag{5.23}$$

A measurement of the Pauli-Z observable on the first qubit then yields:

$$\langle q_0, a|\,\hat{Z}_0\,|q_0, a\rangle = (\cos(\varphi_j/2))^2 - (\sin(\varphi_j/2))^2 = \cos(\varphi_j) = (x - x_j). \tag{5.24}$$

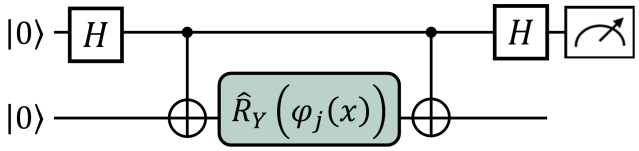


Figure 5.10: Hadamard Test circuit written with the Lagrange encoding function.

Lagrange basis polynomials encoding

This idea is extended across n qubits and following the entanglement scheme of one of the two structures presented in Chapter 4, the quantum feature map encodes on each qubit in the first register, the following polynomial :

$$\langle f_\varphi(x)|\, \hat{Z}_j\, |f_\varphi(x)\rangle = \prod_{i\neq j}^{n-1} (x - x_i), \tag{5.25}$$

which is proportional to the numerator of the Lagrange basis polynomials L_j:

$$L_j(x) = \prod_{i\neq j}^{n-1} \frac{(x - x_i)}{(x_j - x_i)}. \tag{5.26}$$

To recover the full Lagrange basis polynomials, the denominator is obtained by evaluating the quantum feature map at the interpolation node $x = x_j$, yielding a constant denoted by ρ_j as defined in Equation (4.17). Notably, ρ_j is independent of the variational parameters and can therefore be precomputed and stored for reuse, assuming storage resources are available. This constant also compensates any prefactor introduced by the measurement of the quantum feature map, ensuring that the encoded polynomial amplitudes are correctly scaled.

Lagrange interpolation polynomial

The final Lagrange interpolation polynomial is assembled by applying a variational ansatz on top of the Lagrange basis states. In this formulation, trainable weights are introduced via the variational parameters θ:

$$\langle f_{\varphi,\theta}(x)|\, \hat{Z}_j\, |f_{\varphi,\theta}(x)\rangle = \prod_{i\neq j}^{n} \alpha_{i,j}(\theta) \cdot (x - x_i). \tag{5.27}$$

Summing the contributions of all qubits in the first register, the quantum circuit evaluates the complete interpolated function:

$$f(x) = \sum_{j=0}^{n-1} \frac{\langle f_{\varphi,\theta}(x)| \hat{Z}_j |f_{\varphi,\theta}(x)\rangle}{\langle f_{\varphi}(x_j)| \hat{Z}_j |f_{\varphi}(x_j)\rangle} = \sum_{j=0}^{n-1} \alpha_j(\theta) \prod_{i\neq j}^{n} \frac{(x - x_i)}{(x_j - x_i)} = \sum_{j=0}^{n-1} \alpha_j(\theta)\ L_j(x). \tag{5.28}$$

5.2.2 Quantum feature map circuit structures

In the case of two encoding points x_j and x_h, corresponding to $n = 2$, the extended quantum feature map circuit that constructs the associated Lagrange polynomial basis takes the following form shown in Figure 5.11. A Z-measurement on the first register yields two degree-1 polynomials, which, as described previously, form the Lagrange basis over the two points $\{x_j, x_h\}$:

$$\begin{aligned} \langle \hat{Z}_0 \rangle &= (x - x_h) \\ \langle \hat{Z}_1 \rangle &= (x - x_j). \end{aligned} \tag{5.29}$$

These theoretical measurement outcomes can also be obtained using a reduced version of the circuit, shown in Figure 5.12.

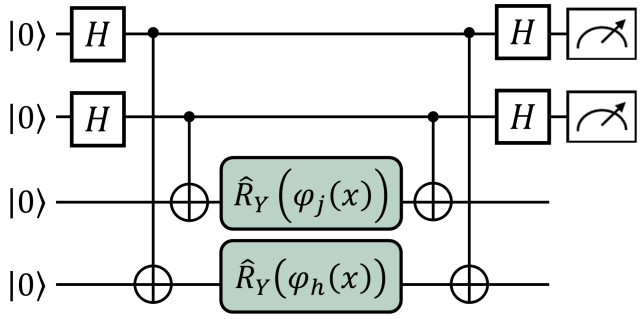


Figure 5.11: Extended quantum feature map circuit for $n = 2$ encoding points.

The difference between the two architectures lies in the number of qubits, and obviously how they are entangled. The number of encoding and CNOT gates remains the same, so the distinction between the two versions is not significant in this particular case.

In the case of three encoding points x_j, x_h, x_k, the difference between the extended and reduced quantum feature map circuits becomes more pronounced. These two architectures are shown respectively in Figures 5.13 and 5.14.

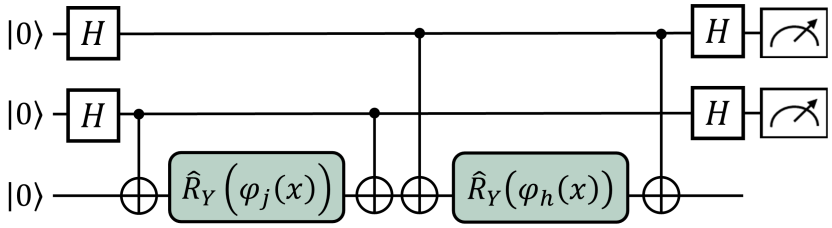


Figure 5.12: Reduced quantum feature map circuit for $n = 2$ encoding points.

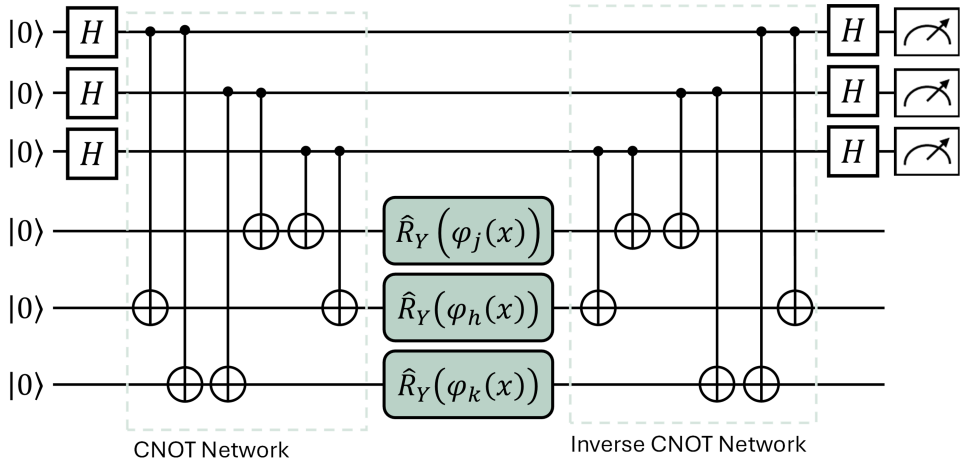


Figure 5.13: Extended quantum feature map circuit for $n = 3$ encoding points.

A Z-measurement on the first register of both circuits yields the following degree-2 polynomials:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= (x - x_h)(x - x_k) \\ \langle \hat{Z}_1 \rangle &= (x - x_k)(x - x_j) \\ \langle \hat{Z}_2 \rangle &= (x - x_j)(x - x_h).\end{aligned} \tag{5.30}$$

With the corresponding coefficients $\rho_j = \langle f_\varphi(x_j)| \hat{Z}_j |f_\varphi(x_j)\rangle$, the Lagrange basis is built, with $L_j(x) = \frac{1}{\rho_j} \langle \hat{Z}_j \rangle$, as shown in the Figure 5.15, for three equispaced encoding points $x_j, x_h, x_k = 0, 0.45, 0.9$.

The difference in qubit count is substantial: the reduced circuit uses only $n + 1$ qubits, whereas the extended version requires $2n$. In addition, the entangling network of CNOT gates scales more favourably in the reduced circuit. In the extended version, the number of CNOTs grows roughly as $2n + (n - 1)$, while in the reduced version, it scales linearly as $2n$.

Furthermore, in the case of three encoding points, the extended circuit requires just three encoding gates—one per input. In contrast, the reduced architecture introduces redundancy: due to the cyclic structure of the entanglement pattern needed to reconstruct the full Lagrange basis, some encoding

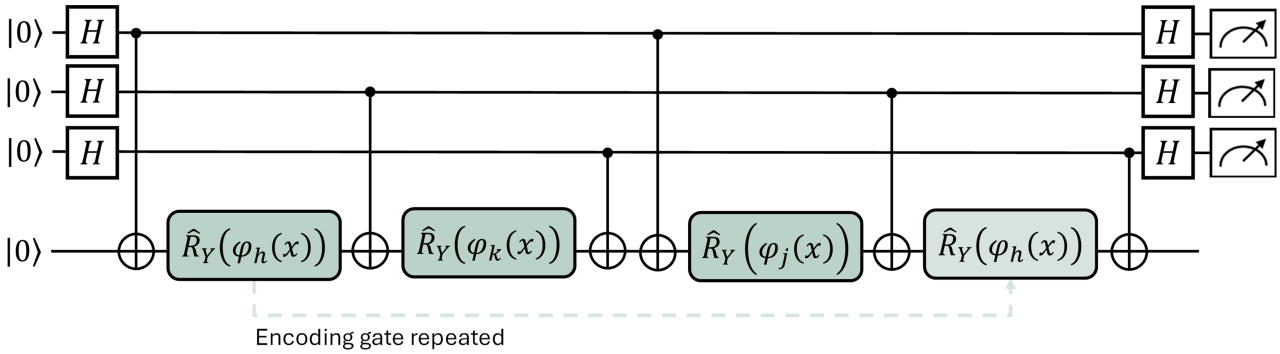


Figure 5.14: Reduced quantum feature map circuit for $n = 3$ encoding points.

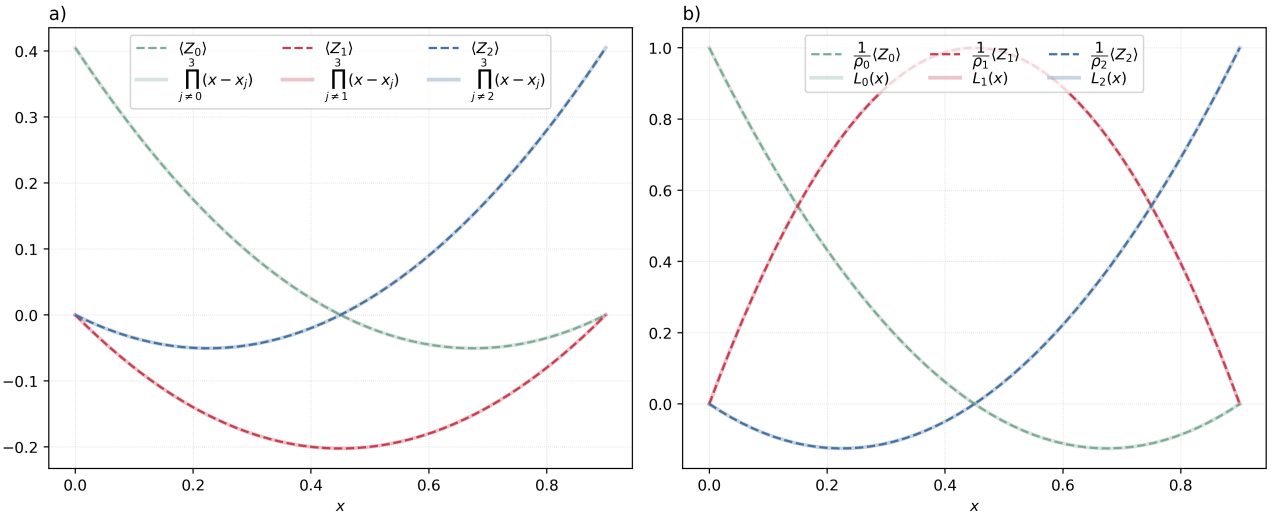


Figure 5.15: Comparison between quantum observables and Lagrange basis polynomials:
a) Expectation values $\langle \hat{Z}_j \rangle$ from single-qubit measurements and corresponding degree-2 polynomials.
b) Scaled observables $\frac{1}{\rho_j}\langle \hat{Z}_j \rangle$ and associated Lagrange basis functions $L_j(x)$.

gates must be applied multiple times. Overall, the reduced circuit may offer a more resource-efficient implementation, particularly in terms of qubit and gate counts, while maintaining equivalent theoretical outputs.

5.2.3 Complete parametrised circuit

Finally, parametrised $\hat{R}_X$ quantum gates are added to the end of the quantum feature map, yielding the complete Lagrange interpolation polynomial: $f(x) = \sum_{j=0}^{n-1} \alpha_j(\theta)\ L_j(x)$ (Equation (5.28)). Returning to the circuit with three encoding points, and now introducing variational parameters $\{\theta_j\}_{j=0}^{2}$, a Z-

measurement on the first register yields the following degree-2 polynomials, as shown in Figure 5.16:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= (x - x_h)(x - x_k)\cos(\theta_0) \\ \langle \hat{Z}_1 \rangle &= (x - x_k)(x - x_j)\cos(\theta_1) \\ \langle \hat{Z}_2 \rangle &= (x - x_j)(x - x_h)\cos(\theta_2).\end{aligned} \tag{5.31}$$

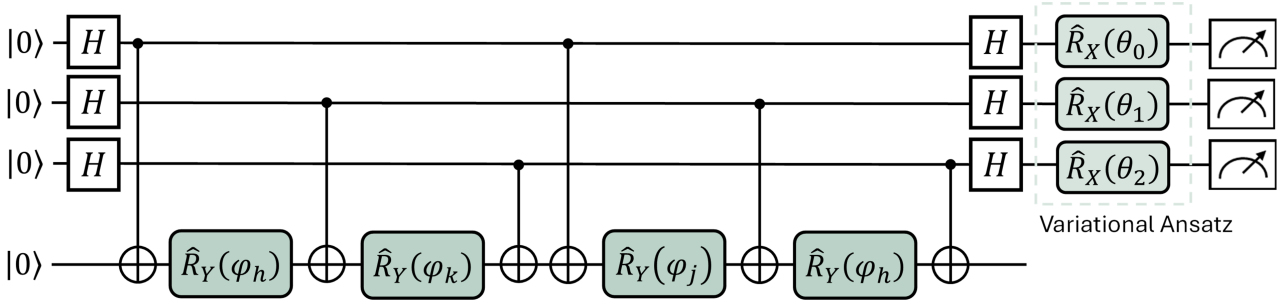


Figure 5.16: Reduced quantum parametrised circuit with Lagrange encoding for $n = 3$ encoding points.

5.2.4 Entangling layers in the variational ansatz

As detailed in the quantum feature map subsection, this approach embeds entanglement directly within the encoding. Even without additional entangling layers in the variational ansatz, the resulting circuit yields not only a polynomial of degree $n-1$, but specifically an interpolating polynomial expressed in the Lagrange basis over the ring of polynomials $\mathbb{R}_{n-1}[X]$, given known encoding points x_j and variable $x \in [0, 0.9]$, with coefficient bounds $-1 \leq \alpha_j(\theta) \leq 1$. Here, the upper bound of 0.9 is chosen to match the interval used in the Chebyshev encoded VQA, so that both approaches can be compared using the same problem definition.

In other words, any real polynomial of degree $n-1$, mapping the interval $[0, 0.9]$ to $[-n, n]$, can be represented by this circuit through interpolation. But what happens if we now include entangling layers in the variational ansatz? Returning to the previous example with three encoding points, the

circuit then takes the form illustrated in Figure 5.17. This circuit leads to the Z-expectations:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= (x - x_h)(x - x_k) \\ \langle \hat{Z}_1 \rangle &= (x - x_j)(x - x_h) \\ \langle \hat{Z}_2 \rangle &= (x - x_h)^2. \end{aligned} \tag{5.32}$$

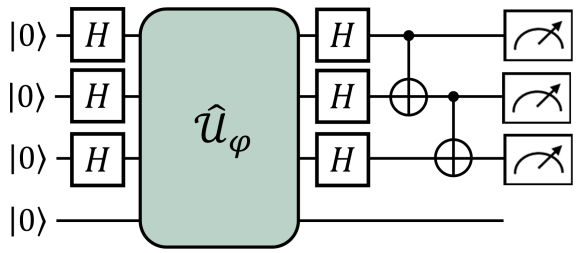


Figure 5.17: Lagrange encoding for $n = 3$ followed by an entanglement layer.

Adding an entangling layer after the encoding does not increase the degree of the polynomial, as it does in the case of Chebyshev encoding. Instead, it reorganises the contribution of each encoding point, reshuffling the indices. Here the structure of the cost function can be questioned, and the aim of the overall polynomial. By reshuffling the indices, and keeping the cost function defined earlier, the final polynomial will not be the exact Lagrange interpolation polynomial over the the n encoding points. However, by redefining the cost function coefficients ρ_j, the objective of the overall circuit, and the role of the cost function, can evolve. This perspective is further developed in subsection 5.2.4.

Hardware Efficient Ansatz applied on Lagrange encoding

Finally, if we combine all elements to create a single layer of a hardware-efficient ansatz (HEA), the circuit takes the form shown in Figure 5.18. The Z-expectation values on each qubit of the first register become:

$$\begin{aligned}\langle \hat{Z}_0 \rangle &= (x - x_h)(x - x_k)\cos(\theta_0) \\ \langle \hat{Z}_1 \rangle &= (x - x_j)(x - x_h)\cos(\theta_0)\cos(\theta_1) \\ \langle \hat{Z}_2 \rangle &= \big((x - x_h)^2 \cos(\theta_0 + \theta_2) + \sin(\theta_2)\sin(\theta_0)\big)\cos(\theta_1) \end{aligned} \tag{5.33}$$

While we still observe second-degree polynomials, we also obtain a scalar contribution (degree zero). This scalar acts as a vertical shift in the final function, which may prove useful when attempting to satisfy boundary conditions. Additionally, the parameter θ_0 appears particularly important. For $\theta_0 = \pm\pi/2$, we have $\cos(\theta_0) = 0$, which cancels the first two Z-expectations entirely. Thus, a single parameter can drastically alter the function's behaviour.

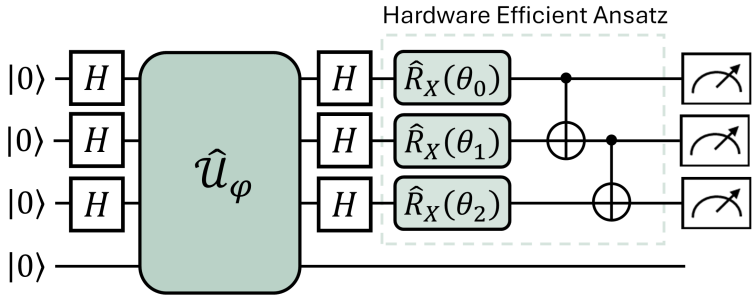


Figure 5.18: Lagrange encoding for $n = 3$ followed by a layer of hardware efficient ansatz.

Of course, this analysis is valid for a single HEA layer. Adding more layers increases the complexity both algebraically and in terms of parameter interaction. Regardless of how entanglement is introduced, the analytical tools for computing circuit derivatives remain valid. As long as encoding functions and variational parameters are introduced via Pauli rotations, as explained in Chapter 3, the circuit remains differentiable. This property makes such circuits highly versatile and attractive from both theoretical and practical perspectives.

As mentioned earlier, introducing entanglement in the variational part of the circuit modifies the objective of the variational quantum algorithm itself. While the reshuffling of indices breaks the strict Lagrange structure, it remains possible to retain a subset of the Lagrange polynomial basis by redefining the cost function coefficients as $\rho_j = \langle f_{\varphi,\vartheta}(x)| \hat{Z}j |f_{\varphi,\vartheta}(x)\rangle$, where ϑ is a vector of parameters that may be fixed or variational. As illustrated in the Z-measurements of the three-point Lagrange encoding followed by a hardware-efficient ansatz (Equation (5.33)), some values of ϑ can significantly impact the outcome. For instance, setting all components of ϑ to $\pm\pi/2$ cancels the Z-expectations entirely. Similarly, choosing $\vartheta = 0$ eliminates constant or cross terms and may discard information relevant to lower or mixed degrees.

From Lagrange interpolation to Hermite interpolation

The most intriguing scenario arises when the same index appears more than once in the expression, resulting in squared Lagrange basis functions. This phenomenon is visible, for instance, in the Z-measurement of the third qubit, $\langle \hat{Z}_2 \rangle$, in Equation (5.33). In classical interpolation, duplicating points falls outside the scope of Lagrange interpolation, which requires all nodes to be distinct, but opens up a new possibility: interpolating not only the values of the target function, but also its derivatives. This perspective brings us into the scope of Hermite interpolation. The goal of Hermite interpolation is to construct a polynomial of degree at most $2n-1$ over n distinct encoding points, such that both the function and its derivative are interpolated at each point. One standard method to construct such a polynomial is to use squares of Lagrange basis polynomials. This requires access to both the values of the target function $f(x_j)$, which we already estimate using the differential equation itself, and the values of its derivative $f'(x_j)$ at the same encoding points. The Hermite interpolating polynomial takes the form:

$$
\begin{aligned}
H(x) &= \sum_{j=0}^{n} f(x_j) H_j(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_j(x), \\
H_j(x) &= [1 - 2(x - x_j) L_j'(x_j)] L_j^2(x), \\
\hat{H}_j(x) &= (x - x_j) L_j^2(x).
\end{aligned}
\tag{5.34}
$$

However, while this approach appears promising, it falls outside the scope of the present PhD project and has therefore not been explored further in this work.

5.2.5 Derivative circuits

One of the main motivations for developing the Lagrange encoding of the parametrised circuit in a variational quantum algorithm designed to solve differential equations was to enable differentiation of the circuit with respect to the variable x via the Hadamard test differentiation method, as presented in the literature (see Figure 5.10). This technique allows the computation of the derivative with respect to x while requiring only half the number of circuit evaluations compared to the parameter shift rule.

First-Order Derivative of the encoding circuit

For the first-order derivative of the encoding circuit, applying the Hadamard test differentiation method to the encoding circuit, as depicted in Figure 3.3, yields the following result for indices $h \neq j$:

$$\frac{\partial}{\partial \varphi_h} \langle f_\varphi(x)| \hat{Z}_j |f\varphi(x)\rangle = \langle f_{\varphi_h^+}(x)| \hat{Z}_j |f\varphi_h^+(x)\rangle . \tag{5.35}$$

For the case $h = j$, since the j^{th} qubit is not encoded through the encoding function φ_h, its derivative is directly known to be zero:

$$\frac{\partial}{\partial \varphi_j} \langle f_\varphi(x)| \hat{Z}_j |f_\varphi(x)\rangle = 0. \tag{5.36}$$

Similarly to the Chebyshev encoding, the derivative with respect to the variable x is obtained through the chain rule:

$$\langle f_{\varphi_h^+}(x)| \hat{Z}_j |f_{\varphi_h^+}(x)\rangle \ \ \frac{d\varphi_h}{dx} = \prod_{\substack{k\neq j \\ k\neq h}}^{n-1} (x - x_k). \tag{5.37}$$

It can be shown that:

$$\begin{aligned} \sum_{\substack{h=0 \\ h\neq j}}^{n-1} \frac{1}{\rho_j} \langle f_{\varphi_h^+}(x)| \hat{Z}_j |f_{\varphi_h^+}(x)\rangle \ \ \frac{d\varphi_h}{dx} &= \sum_{\substack{h=0 \\ h\neq j}}^{n-1} \left[\prod_{\substack{k\neq h \\ k\neq j}}^{n-1} \frac{x - x_k}{x_j - x_k} \right] \frac{1}{x - x_h} \\ &= L_j(x) \cdot \sum_{\substack{h=0 \\ h\neq j}}^{n-1} \frac{1}{x - x_h} = L'_j(x), \end{aligned} \tag{5.38}$$

where ρ_j is defined as in Equation (4.17), $L_j(x)$ denotes the Lagrange basis polynomial, and $L'_j(x)$ its derivative. Consequently, the derivative of the function f with respect to x is expressed as:

$$\frac{df}{dx} = \sum_{j=0}^{n-1} \sum_{\substack{h=0 \\ h\neq j}}^{n-1} \frac{1}{\rho_j} \langle f_{\varphi_h^+,\theta}| \hat{Z}_j |f_{\varphi_h^+,\theta}\rangle \ \ \frac{d\varphi_h}{dx} = \sum_{j=0}^{n-1} \alpha_j(\theta) L'_j. \tag{5.39}$$

From this expression arises the cost function to be measured for each modified circuit $|f_{\varphi_k^+}(x)\rangle$ in the evaluation of the derivative with respect to x, which is defined as:

$$\hat{C}_k = \sum_{\substack{j=0 \\ j\neq k}}^{n-1} \frac{1}{\rho_j} \hat{Z}_j. \tag{5.40}$$

To illustrate, with a set of five Chebyshev nodes of the first kind as encoding points, and a random value of θ, the function arising from the cost function defined in Equation (4.17), as explicitly stated in Equation (5.28), as well as its derivative obtained from the overall cost function $\sum_k \hat{C}_k$ defined in Equation (5.40) and explicitly stated in Equation (5.39), has been plotted in Figure 5.19, where the theoretical Lagrange interpolation polynomial $P(x)$ and its derivative $\frac{d}{dx}P(x)$ are built over the set of points x_i, with $y_i = f(x_i)$, corresponding to the random function created by the quantum circuit.

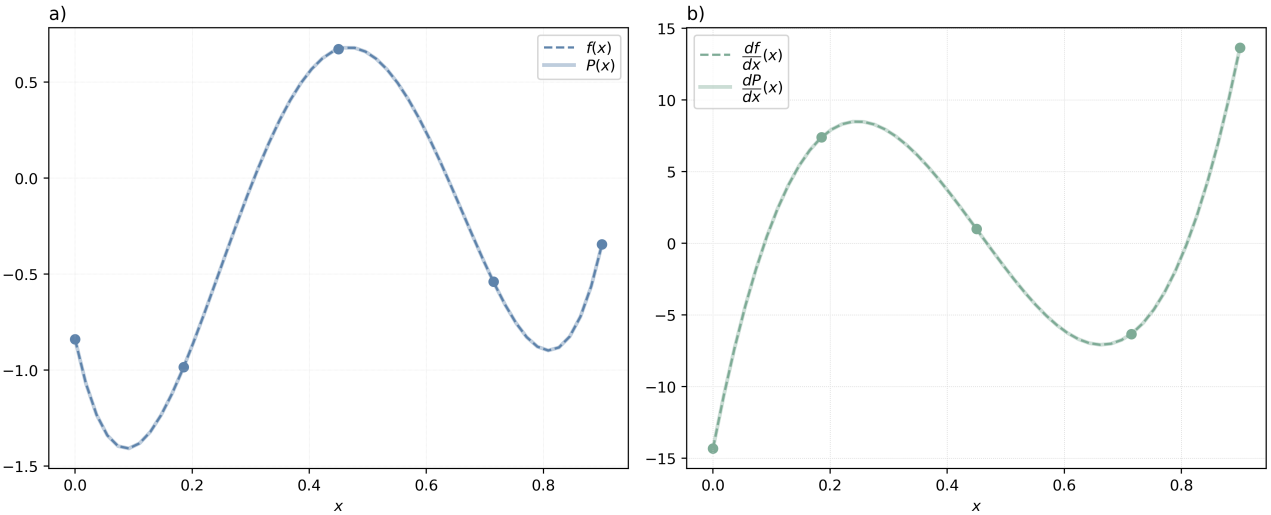


Figure 5.19: Comparison between the output of the function circuit for a random variational parameter over a set of five Chebyshev nodes of the first type (Equation (5.20)) and the theoretical Lagrange interpolation polynomial, as well as their first order derivatives.
a) Function f arising from the cost function $\hat{C}$ and corresponding Lagrange polynomial P.
b) Derivative $\frac{d}{dx}f$ obtained from the total cost $\sum_k \hat{C}_k$ and derivative of the polynomial $\frac{d}{dx}P$.

Behaviour of the encoding function over the interval

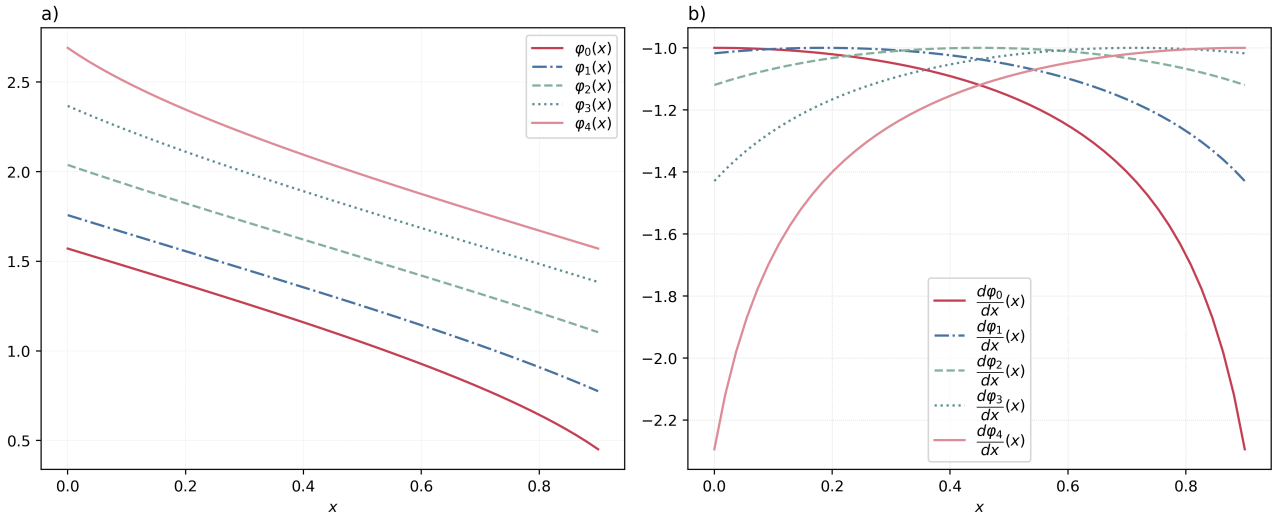


Figure 5.20: a) Plot of the encoding functions φ_j applied to each qubit, for $j = 0$ to 4.
b) Corresponding analytical derivatives $\frac{d\varphi_j}{dx}$, used for the automatic differentiation.

Regarding the encoding functions $\{\varphi_j\}_{j=0}^{n-1}$ and derivatives $\{\frac{d}{dx}\varphi_j\}_{j=0}^{n-1}$ (Equation (5.41)), which are used to compute the derivative of the function with respect to the variable x via automatic differentiation, they do not exhibit a fast divergent behaviour over the interval, as shown in Figure 5.20.

Additionally, what can be noticed is that, the highest amplitudes of the derivative are obtained with the encoding function of the first and last point of the interval φ_0, φ_{n-1}, which are the plain curves in Figure 5.20, and every other encoding function will be contained between those two curves.

$$\frac{d}{dx}\varphi_j(x) = \frac{-1}{\sqrt{1-(x-x_j)^2}}. \tag{5.41}$$

Higher order derivatives

Similarly to the case of Chebyshev encoding, higher order derivatives are obtained by repeatedly applying the same method described for the first order. In this context, the output for the derivative

satisfies the following equation:

$$\begin{aligned}
&\sum_{h\neq j}\sum_{l\neq(j,h)} \frac{1}{\rho_j} \langle f_{\varphi_{h,l}^{++}}(x)| \hat{Z}_j |f_{\varphi_{h,l}^{++}}(x)\rangle \frac{d\varphi_h}{dx}(x)\frac{d\varphi_l}{dx}(x) \\
&= \sum_{h\neq j}\sum_{l\neq(j,h)} \left[\prod_{\substack{k=0\\k\neq j}}^{n-1} \frac{x-x_k}{x_j-x_k}\right] \frac{1}{(x-x_h)(x-x_l)} \\
&= L_j(x)\cdot \sum_{h=1}^{n-1}\sum_{k=0}^{h-1} \frac{2}{(x-x_h)(x-x_l)} = L_j''(x),
\end{aligned} \tag{5.42}$$

Which yields to the expression of the second-order derivative:

$$\frac{d^2f}{dx^2} = \sum_{j=0}^{n-1}\sum_{h\neq j}\sum_{l\neq(j,h)} \frac{1}{\rho_j} \langle f_{\varphi_{h,l}^{++},\theta}| \hat{Z}_j |f_{\varphi_{h,l}^{++},\theta}\rangle \frac{d\varphi_h}{dx}\frac{d\varphi_l}{dx} = \sum_{j=0}^{n-1} \alpha_j(\theta) L_j''. \tag{5.43}$$

From this expression arises the cost function to be measured for each modified circuit $|f_{\varphi_{h,l}^{++},\theta}(x)\rangle$ in the evaluation of the second-order derivative with respect to x, which is defined as:

$$\hat{C}_{k,l} = \sum_{\substack{j=0\\j\neq k, j\neq l}}^{n-1} \frac{1}{\rho_j}\hat{Z}_j. \tag{5.44}$$

Differentiated encoding

Determining the derivative with respect to the variable x as described above relies on the chain rule, since the derivative circuits can only be differentiated with respect to their encoding function, as shown in Equation (5.39). However, although the arccosine function maps onto the interval $[0,1]$, its derivative maps onto the interval $[0,1[$, which restricts the window of training points and forces the amplitude of its argument to remain strictly below 1.

However, due to the nature of this encoding, one encoding gate gives :

$$\langle 0| \hat{R}_Y(\varphi_k(x))|\hat{Z}|\hat{R}_Y(\varphi_k(x)) |0\rangle = \cos(\varphi_k(x)) = x - x_k, \tag{5.45}$$

and looking at its derivative, we obtain:

$$\frac{d}{dx}\left(\cos(\varphi_k(x))\right) = \frac{d}{dx}\left(x - x_k\right) = 1 \tag{5.46}$$

$$= \frac{d}{d\varphi_k}\left(\cos(\varphi_k)\right) \cdot \frac{d\varphi_k}{dx}(x) \tag{5.47}$$

$$= -\sqrt{1-(x-x_k)^2} \cdot \frac{-1}{\sqrt{1-(x-x_k)^2}} = 1. \tag{5.48}$$

This shows that the chain rule imposes an unnecessary constraint on the value of $(x - x_k)$, since it is ultimately cancelled out between the two terms. Consequently, this allows us to rewrite the derivatives using a modified circuit $|f_{d\varphi_k,\theta}(x)\rangle$, where $\varphi_k(x)$ is replaced by $d\varphi_k(x) = \arccos(1) = 0$, which implies that $\hat{R}_Y(d\varphi_k(x)) = 1$, so the corresponding gate can be removed.

This method allows us to exploit a larger range of values and therefore a larger magnitude for the encoding output, since the argument of the arccosine function no longer needs to remain strictly below 1. It also removes one gate for the first order derivative and two gates for the second order derivative.

Although the resulting values are identical to those obtained with the Hadamard test differentiation method, combined with the chain rule, and both approaches require the same number of circuits. The differentiated encoding offers the crucial advantage of enabling a larger encoding amplitude, which is essential for practical implementation, as we discuss in the Chapter 6.

5.3 Variational Ansatz

As presented earlier in Chapter 3, the variational ansatz plays a central role in a variational quantum algorithm. It organises and distributes the influence of each parameter θ_j across the quantum circuit, shaping how the quantum state space is explored during training. This, in turn, determines both the expressivity and the trainability of the algorithm.

Most ansätze used in practice are based on the hardware-efficient ansatz, whose structure introduces entanglement through native gate patterns and encodes weights via parametrised rotations. This is, for instance, the approach taken in the work of Kyriienko *et al.* [4]. However, as we have discussed, in the case of Lagrange encoding, the quantum state is already entangled prior to the application of the variational ansatz. As a result, additional entangling layers may not always be necessary.

Furthermore, the rotational blocks used in the ansatz can be adapted to the specific problem. In the work of Kyriienko *et al.* [4], the authors used the same sequence as proposed by Kandala *et al.* [135] $\hat{\mathcal{U}}^{j,l}(\theta) = \hat{R}_Z(\theta_2^{j,l})\hat{R}_X(\theta_1^{j,l})\hat{R}_Z(\theta_0^{j,l})$, which, when applied to a general single-qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, (\alpha,\beta) \in \mathbb{C}^2$, yields:

$$\begin{aligned}\hat{R}_Z(\theta_2)\hat{R}_X(\theta_1)\hat{R}_Z(\theta_0)|\psi\rangle = &\left(\alpha e^{-i\theta_0/2}\cos\left(\frac{\theta_1}{2}\right) - i\beta e^{i\theta_0/2}\sin\left(\frac{\theta_1}{2}\right)\right)e^{-i\theta_2/2} \\ &+ \left(-i\alpha e^{-i\theta_0/2}\sin\left(\frac{\theta_1}{2}\right) + \beta e^{i\theta_0/2}\cos\left(\frac{\theta_1}{2}\right)\right)e^{i\theta_2/2}. \end{aligned} \tag{5.49}$$

This block structure is expressive, as it introduces three variational parameters per qubit per layer. However, it also comes with practical drawbacks. A higher number of parameters increases the risk of encountering trainability issues, including convergence to local minima and barren plateaus. Moreover, it increases the depth of the circuit and the total number of gates, which is problematic on NISQ devices due to short coherence times and hardware noise.

In addition, a large number of variational parameters leads to a higher number of partial derivatives to compute at each optimization step, thus requiring a significantly greater number of quantum circuit evaluations for gradient estimation. For this reason, when expressivity is not a limiting factor, it may be preferable to simplify the rotation block to a single gate, for instance:

$$\hat{\mathcal{U}}^{j,\ell}(\theta) = \hat{R}_X(\theta^{j,\ell}), \tag{5.50}$$

which, when applied to the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, yields:

$$\begin{aligned}\hat{R}_X(\theta)|\psi\rangle = &\alpha\cos\left(\frac{\theta}{2}\right) - i\beta\sin\left(\frac{\theta}{2}\right) \\ &- \alpha\sin\left(\frac{\theta}{2}\right) + \beta\cos\left(\frac{\theta}{2}\right). \end{aligned} \tag{5.51}$$

5.3.1 Gradients with respect to the variational parameters

As the chosen classical optimizer relies on gradient-based updates, the gradient of the cost function with respect to the variational parameters must be computed at each iteration. This is required not

only for the main function circuit $|f_{\varphi,\theta}\rangle$, but also for the additional circuits used to compute derivatives with respect to the input variable x.

To estimate these gradients efficiently, we use the parameter-shift rule, as introduced in the Chapter 4. For the function circuit, this yields:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} f_{\varphi,\theta} &= \left(\frac{\partial f_{\varphi,\theta}}{\partial \theta_0}, \frac{\partial f_{\varphi,\theta}}{\partial \theta_1}, \ldots, \frac{\partial f_{\varphi,\theta}}{\partial \theta_m} \right), \\ \frac{\partial f_{\varphi,\theta}}{\partial \theta_j} &= \frac{1}{2} \left(\langle f_{\varphi,\theta_j^+} | \hat{C} | f_{\varphi,\theta_j^+} \rangle - \langle f_{\varphi,\theta_j^-} | \hat{C} | f_{\varphi,\theta_j^-} \rangle \right) \end{aligned} \tag{5.52}$$

where $\theta_j^{\pm}$ denotes the vector θ_j with the j^{th} component shifted by $\pm\pi/2$, and $\hat{C}$ is the cost observable.

5.4 Summary

In this chapter, we examined in greater detail polynomial-encoded variational quantum algorithms. We observed that the Chebyshev-encoded VQA produces a polynomial whose degree grows rapidly with the number of qubits and the depth of the hardware-efficient ansatz. We clarified the role of entanglement and the construction of the derivative quantum circuit. This analysis highlighted a key limitation: the first- and second-order derivatives of the Chebyshev encoding function diverge near the end of the domain, suggesting that applying this method to higher-order differential equations may pose significant training challenges.

For the Lagrange-encoded VQA, we analysed the circuit structure and its ability to reproduce Lagrange interpolation, along with the evaluation of derivatives, which do not exhibit divergent behaviour. We also discussed how introducing entanglement after the encoding can modify the type of interpolation, for instance by redistributing the encoding points or encoding the squares of the Lagrange polynomials, which could be used in Hermite interpolation. This latter extension is a potential direction for future work and was not explored in the present study.

Finally, we investigated an alternative approach to determine the derivative with respect to x using differentiated encoding, which removes the constraints imposed by the chain rule on the values that the encoding function can take, allowing its amplitude to be maximised, a key factor in ensuring its feasibility, as discussed in the next chapter.

Chapter 6

Quantum Circuit Simulation and Complexity Estimation

6.1 Quantum Circuit Simulation

The application of the two polynomial-based VQAs presented earlier to an actual quantum device lies outside the scope of this thesis. Nevertheless, to better understand how the structure of a given VQA may affect its feasibility on real hardware, we investigate the impact of executing it on quantum processor simulators: we first employ the noise-free simulator *AerSimulator* and the noisy simulator *FakeNairobiV2* from IBM-Qiskit.

Although a detailed study of noise is not included in this work, it is instructive to have a brief idea of its influence on measurement outcomes, and therefore of how far we are from applying these VQAs to real devices, as well as what can be prioritised in their design to limit noise sources and ensure their feasability. The IBM's *FakeNairobiV2* simulator mimics a 7-qubits NISQ device based on calibration data from previous years. While this model does not perfectly reproduce the behaviour of today's most advanced devices, it provides a reasonable approximation of the average performance of a NISQ device.

Quantum circuit measurement

The measurement of a quantum circuit executed on quantum hardware is probabilistic. Each run of the circuit ends with a readout, where every qubit yields either 0 or 1, as the quantum state collapses with a probability given by the squared magnitude of the corresponding amplitude in the quantum statevector. Consequently, as discussed in the literature review, a single measurement cannot retrieve the full information encoded in the state. In a measurement performed in the computational basis, or Z-basis measurement, the goal is to estimate as accurately as possible the probabilities $|\alpha|^2$ and $|\beta|^2$ of the quantum state $|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

When working on the theoretical aspect of quantum algorithms, the use of statevectors to "measure" the circuit is preferred, as it enables the exact extraction of the values α and β, and consequently the amplitudes $|\alpha|^2$ and $|\beta|^2$, while remaining significantly more efficient to compute. However, when it comes to real devices or simulations, the measurement is achieved by running the circuit a certain number of times, referred to as shots, to estimate $p_0 = |\alpha|^2$ and $p_1 = |\beta|^2$ from the observed measurement statistics, to be able to perform $\langle \hat{Z} \rangle = p_0 - p_1$ as accurately as possible.

One can naturally think that increasing the number of shots improves the precision of the measurement. However, this improvement is not linear. Beyond a certain point, increasing the number of shots no longer yields a significant increase in accuracy, since the statistical uncertainty scales as $1/\sqrt{N_{\text{shots}}}$ with N_{shots} the number of shots, and increasing the number of shots would cost time and resources without particularly increasing the precision.

This statistical uncertainty can be expressed as a percentage of the measured value. Consequently, it is natural to expect that larger magnitudes of the expected value require fewer shots to reach a given target accuracy. For quantum systems, expectation values are confined within $[-1, 1]$, so the upper limit of the magnitude is 2. In this section, we illustrate how optimizing the encoding to increase the magnitude can improve the accuracy of the sampling measurement outcome for a fixed number of shots. As shown in Figure 6.1, for small magnitudes, achieving an uncertainty of 0.5 to 1% is impossible. However, when approaching the upper limit, significant accuracy can be obtained with only a few thousand shots. Consequently, simulating a circuit over N_{shots} and measuring the register yields, for each shot, a bitstring such as '101' for a three qubit register. After N_{shots}, each bitstring is associated to a count and probability. If we want to obtain individual probabilities for every qubit of an

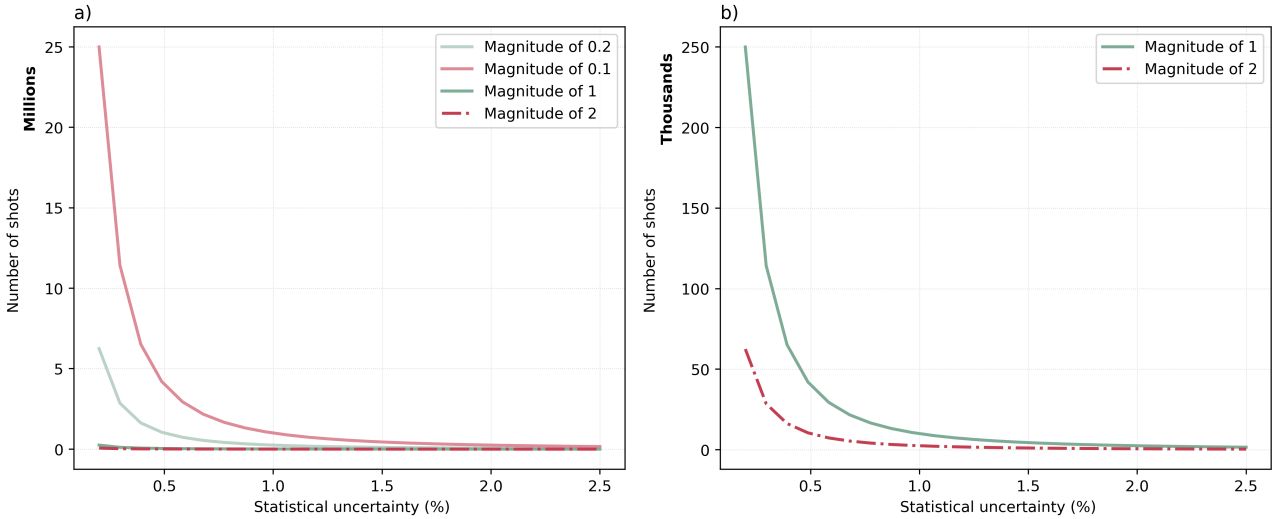


Figure 6.1: Evolution of the order of magnitude of the number of shots required to reach a given accuracy for different measured value magnitudes. a) for values between 0.1 and 2, b) for values between 1 and 2.

n qubit register, we can measure all qubits simultaneously and compute, for example, the probability of the first qubit being in state $|0\rangle$ as $p(q_0 = 0) = \sum_s p(0s_1s_2...s_{n-1})$, where s_k is the k^{th} bit. Hence, a run of the circuit gives access to the measurement statistics of all qubits in the register.

Quantum measurement of the Lagrange encoding

The Lagrange encoding developed in this work had the main objective to propose an encoding that would reduce the computational requirements of polynomial-based VQAs to solve differential equations. With this in mind, the key points were to limit the number of qubits, gates, depth, and circuits. Two structures of the same overall idea have been presented in Chapter 4, presenting the same encoding function $\varphi_j(x) = \arccos(x - x_j)$. In this section, we look at how slight changes in this function can improve its feasibility and potential application to a real device.

First encoding function First, this function has been studied on a theoretical level only, leading to the introduction of a coefficient $\frac{1}{2}$ to avoid the divergent behaviour of the arccosine function close to 1, at the extremity of the interval $[0, 1]$, as presented in our paper [1]. In Chapter 5, we have seen that, without the coefficient, this behaviour is not yet problematic for the first nor second order derivative. But additionally, while adding a coefficient does not impact the theoretical output as it will be compensated in the cost function $\hat{C}$ with the division by ρ_j (Equation (5.28)), the quick study

on the simulator highlighted its practical impact. Looking at the quantum feature map circuit, the original and naive version of the encoding function $\phi_j(x) = \arccos\left(\frac{x-x_j}{2}\right)$, resulted in the measurement of the j^{th} qubit of:

$$\langle f_{\phi,\theta}(x)| \hat{Z}_j |f_{\phi,\theta}(x)\rangle = \frac{1}{2^{n-1}} \prod_{j\neq k}^{n-1} (x - x_k), \tag{6.1}$$

The value becomes very small as the system size n increases. Since the product $\prod_{j\neq k}^{n-1}(x - x_j)$ has a maximum amplitude of 1, the prefactor $1/2^{n-1}$ causes the overall value to decrease with n. This has a direct impact on the feasibility of the circuit. The smaller the circuit outcome, the more difficult it becomes to measure it with acceptable accuracy. Eventually, the outcome will vanish, becoming negligible compared to the inherent sampling noise.

Version 1: Naive improvement of the encoding function The first idea to address this issue was simply to remove the coefficient, leading to the encoding function $\varphi_j(x) = \arccos(x - x_j)$. However, this still did not appear optimal, as it resulted in a value still too small to be measured efficiently given the number of shots used in the simulation. For a system of six qubits, the measurement of the first and second qubits of the quantum feature maps using the noise-free simulator gave the result shown in figure 6.2:

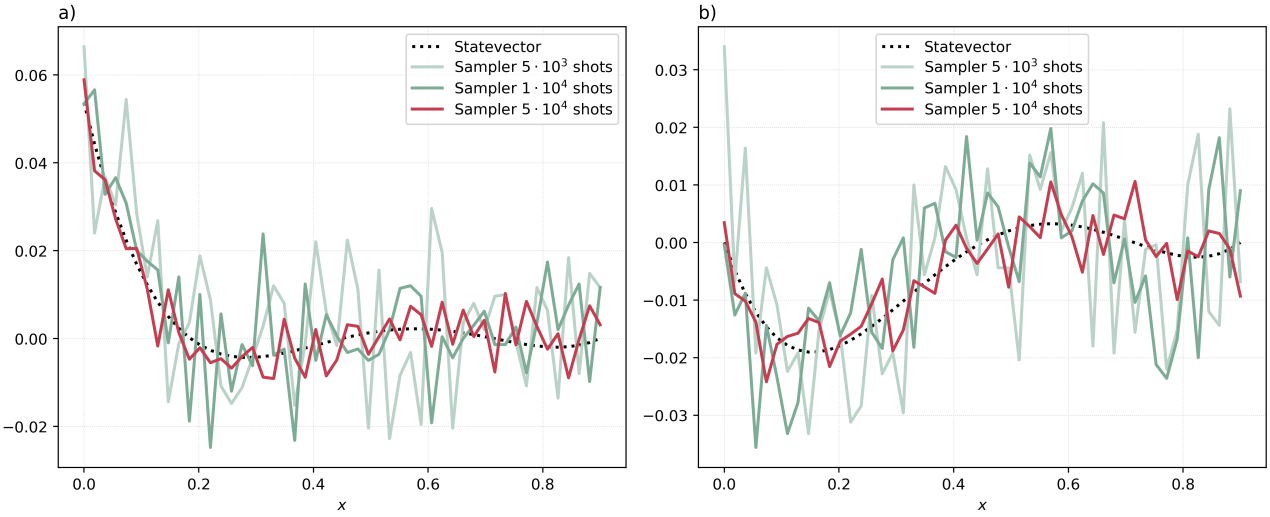


Figure 6.2: Version 1 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *AerSimulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value.
a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit.

Version 2: Introduction of smarter weights in the encoding function The next idea that comes naturally, would be to introduce a weight w_k in the arccos function of the encoding function to amplify the output amplitude of the circuit while respecting the domain of existence of the arccos function $[-1, 1]$, leading to the following proposition:

$$\psi_k = \arccos\left(\frac{x - x_k}{w_k}\right) \quad \text{with,} \tag{6.2}$$

$$w_k = \max(|x_k|, |0.9 - x_k|). \tag{6.3}$$

While not impacting the theoretical output as the weight is fixed for each gate and is cancelled out in the cost function, it greatly improved the readability of the circuit, as shown in the simulation results in the figure 6.3. This allowed the reduction of the average mean square error between simulated circuit reading and the output deducted from the statevector by a factor of 33 for the first qubit measurement and by a factor of 11 for the second qubit. The average mean square error is computed over $M = 50$ equispaced points with:

$$\text{MSE} = \frac{1}{M} \sum_{j=0}^{M-1} \left(f_{\text{sample}_v}(x_j) - f_{\text{statevector}_v}(x_j)\right)^2. \tag{6.4}$$

To allow a fair comparison across different versions, the MSE must be rescaled to account for the coefficient W_v introduced by the quantum feature map, which depends on the version v. This coefficient can be computed analytically at any point x_j as:

$$W_1 = 1, \quad \text{and for } v = 2, 3: \quad W_v = \frac{f_{\text{statevector}_0}(x_j)}{f_{\text{statevector}_v}(x_j)}. \tag{6.5}$$

The scaled MSE is then:

$$\text{MSE} = \frac{1}{M} \sum_{j=0}^{M-1} \left(W_v \; f_{\text{sample}_v}(x_j) - f_{\text{statevector}_1}(x_j)\right)^2. \tag{6.6}$$

On a side note, it is important to mention that the encoding function constructed here produces an argument of the arccosine that reaches 1 at the extremities of the interval. This implies that the differentiated encoding should be used instead of the automatic differentiation based on the chain rule, as explained in Section 5.2.5.

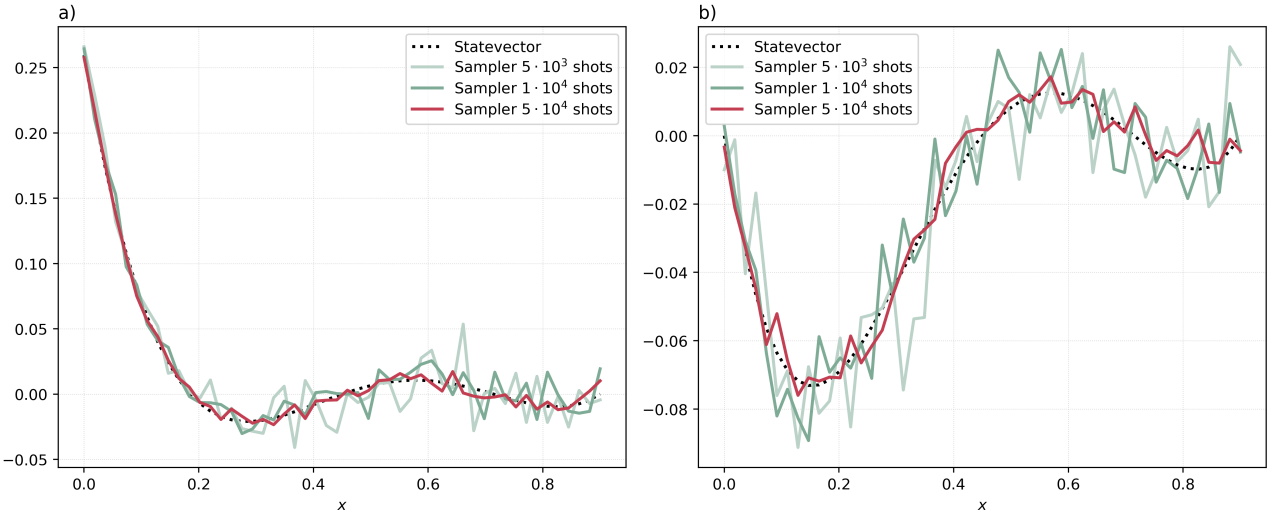


Figure 6.3: Version 2 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *AerSimulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value.
a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit.

Version 3: Optimized structure While the previous result can already be regarded as close to optimal, since with a sufficiently large number of shots the output approaches the expected statevector value, a few additional adjustments can still be introduced to improve readability, particularly for the measurement of the second qubit, whose amplitude is smaller. This reduced visibility arises from the fact that measurements of the intermediate qubits, meaning all qubits except the first and the last in the register, involve the product:

$$\frac{(x - x_0)}{w_0} \cdot \frac{(x - x_{n-1})}{w_{n-1}} = \frac{(x - x_0)(x - x_{n-1})}{0.9^2}. \tag{6.7}$$

This expression can be improved, since when x is near the beginning of the interval, $(x - x_0) \simeq 0$ and $(x - x_{n-1}) \simeq 0.9$, whereas at the end of the interval $(x - x_0) \simeq -0.9$ and $(x - x_{n-1}) \simeq 0$. Consequently, the structure of the quantum feature map circuit can be slightly reconfigured by combining the previous encoding function ψ_k with the following additional function:

$$\psi^*_{k,l} = \arccos\left(\frac{(x - x_k)(x - x_l)}{w_{k,l}}\right), \quad \text{with} \tag{6.8}$$

$$w_{k,l} = \max\left(|x_k x_l|, |(0.9 - x_k)(0.9 - x_l)|, |(x_k - x_l)/2|^2\right).$$

In the example here of 6 qubits (thus a register of 5 qubits measured), the circuit structure is slightly changed, as pictured in the figure 6.4.

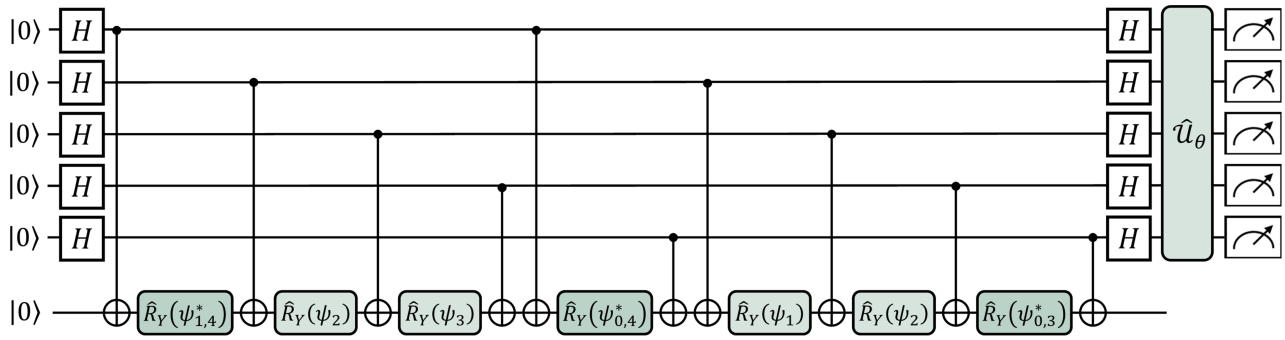


Figure 6.4: Reduced six-qubit function circuit with third-version Lagrange encoding

With this slightly modified structure, the average mean square error between the simulated circuit readings and the statevector-derived outputs is reduced by a factor of 592 for the first qubit and by a factor of 265 for the second qubit, which is quite significant and can be clearly identified in Figure 6.6.

Once again, the encoding function constructed here has an arccosine argument that reaches 1 at the extremities of the interval. Consequently, the differentiated encoding is preferred in this case over the automatic differentiation based on the chain rule, as explained in Section 5.2.5.

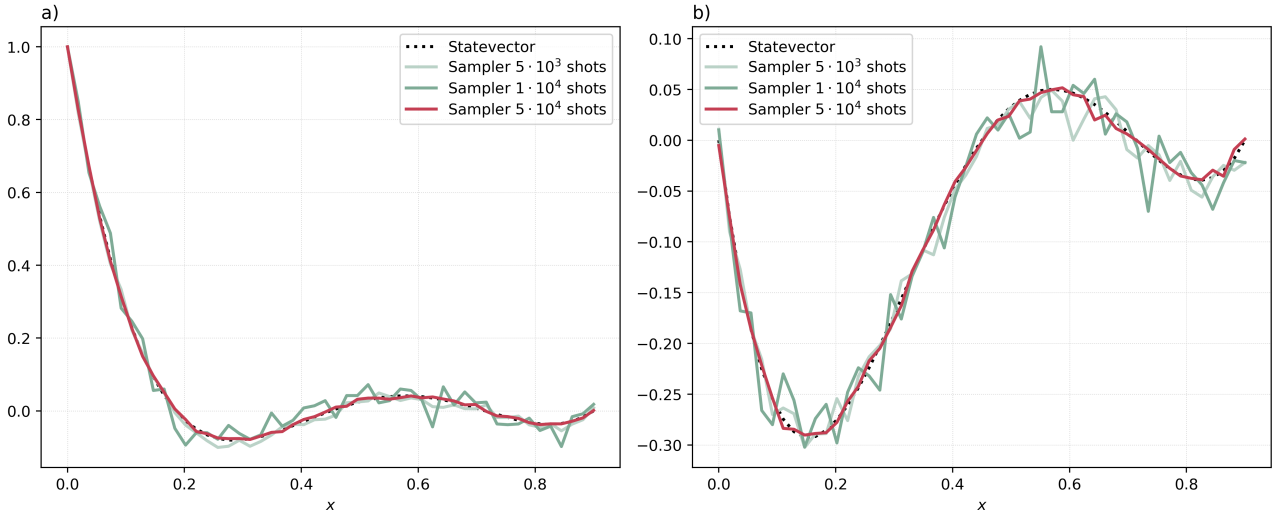


Figure 6.5: Version 3 - Lagrange quantum feature map circuit simulation on the IBM-Qiskit *AerSimulator*, using 5, 10 and 50 thousands shots, in comparison to the statevector value.
a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit.

Comparison of the three versions Finally, the best outputs of the three versions are compared in Fig. 6.6 and summarised in Table 6.1. In the figure, the results are rescaled using the fixed weights introduced earlier, enabling direct comparison in terms of mean square error (MSE) and allowing them to be plotted together.

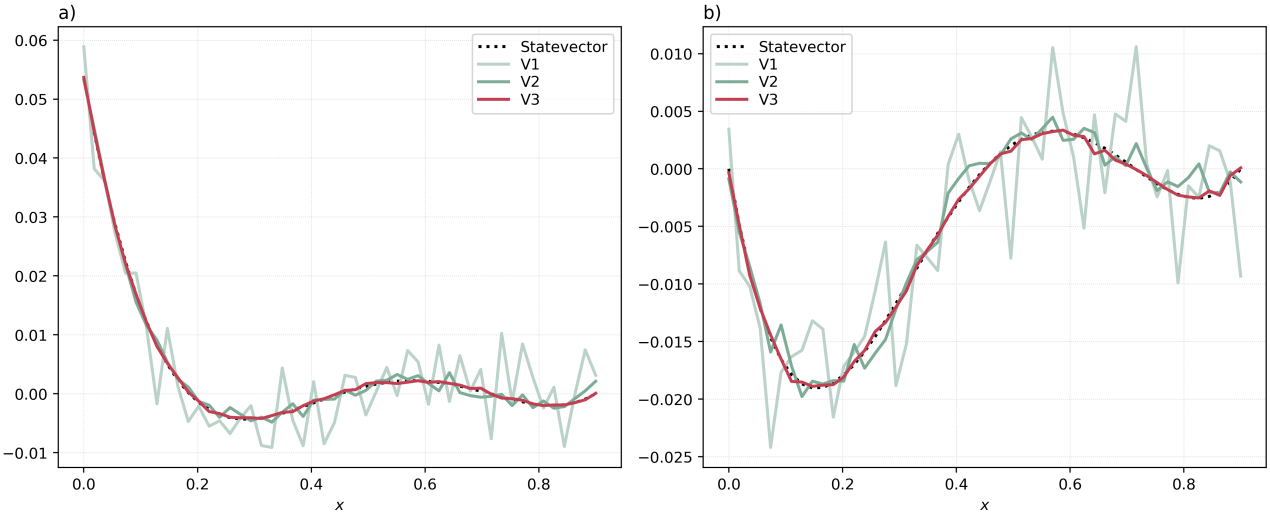


Figure 6.6: Comparison of the three versions (V1, V2, V3) of the Lagrange quantum feature map circuit simulated on the IBM Qiskit *AerSimulator* with 50 thousands shots and the statevector value. a) Z-measurement of the first qubit. b) Z-measurement of the second qubit.

Table 6.1: Comparison of three Lagrange encoding versions with scaled MSE for $\langle \hat{Z}_0 \rangle$ and $\langle \hat{Z}_1 \rangle$, and improvement factors over Version 1.

	Encoding function	MSE scaled $\langle \hat{Z}_0 \rangle$	MSE scaled $\langle \hat{Z}_1 \rangle$	Improvement relative to V1
Version 1	φ_k	2.5×10^{-5}	2.0×10^{-5}	–
Version 2	ψ_k	7.4×10^{-7}	1.7×10^{-6}	By factors of 33 and 11
Version 3	ψ_k and $\psi^*_{k,l}$	4.2×10^{-8}	7.7×10^{-8}	By factors of 592 and 265

Introduction of noise Finally, considering the best outcome obtained with the third version of the encoding, the same quantum feature map circuit was simulated using the *FakeNairobiV2* backend. This simulator implements a noise model derived from data collected up to December 2021 on the real quantum device named Nairobi, incorporating gate error rates for each type of basic gate, relaxation and coherence times, as well as readout errors. In this example, the simulation with noise exhibits larger oscillations and a small shift of the function across the entire interval, together with a reduced amplitude, particularly for the measurement of the second qubit. Nevertheless, the underlying trend remains discernible.

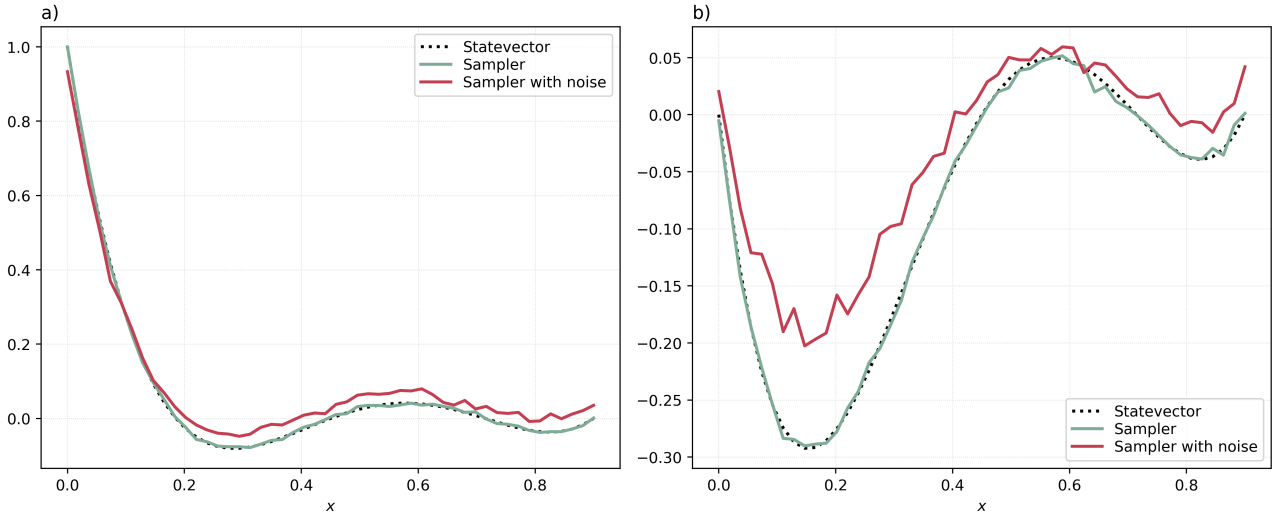


Figure 6.7: Comparison of the third version of the Lagrange quantum feature map circuit simulated on noise-free *AerSimulator* and noisy *FakeNairobiV2* simulator using 50 thousand shots, referenced to the statevector result. a) Z-measurement of the first qubit, and b) Z-measurement of the second qubit.

Simulation of a random function

Finally, to assess the impact of the variational ansatz and the complete cost function, the circuit was executed with random variational parameters θ, arranged either as a single $\hat{R}_X$ gate or as a two-gate sequence $\hat{R}_Z - \hat{R}_X$.

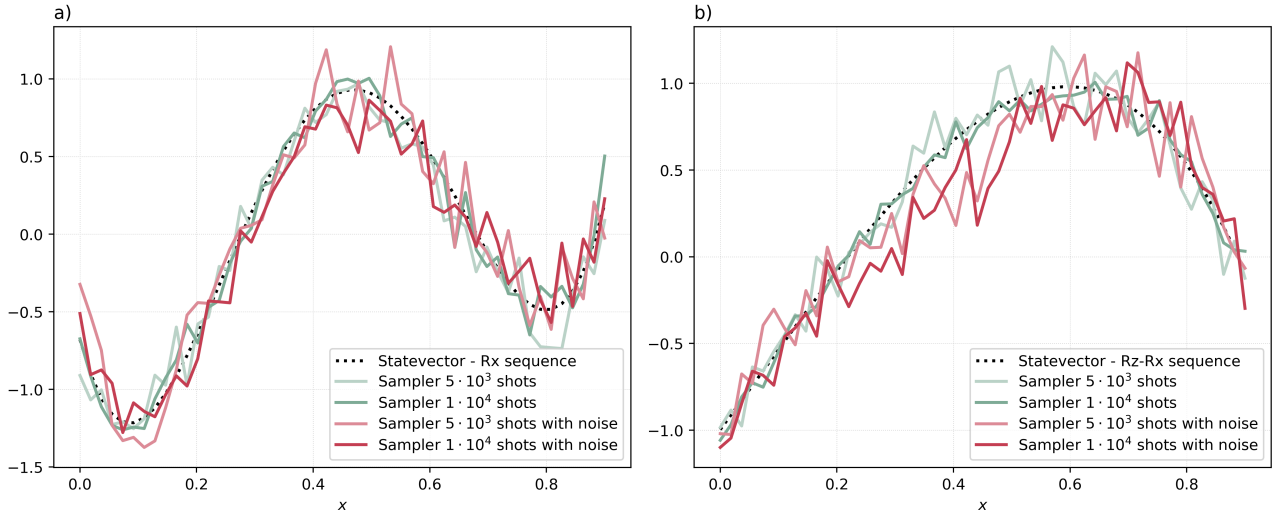


Figure 6.8: Impact of noise on the Lagrange encoded function circuit for two HEA, evaluated under the two measurement settings. Simulations were performed with 5 and 10 thousand shots on both the noise-free *AerSimulator* and the noisy *FakeNairobiV2* simulator, in comparison to the statevector value a) Single-gate sequence $\hat{R}_X$, and b) Two-gate sequence $\hat{R}_Z - \hat{R}_X$.

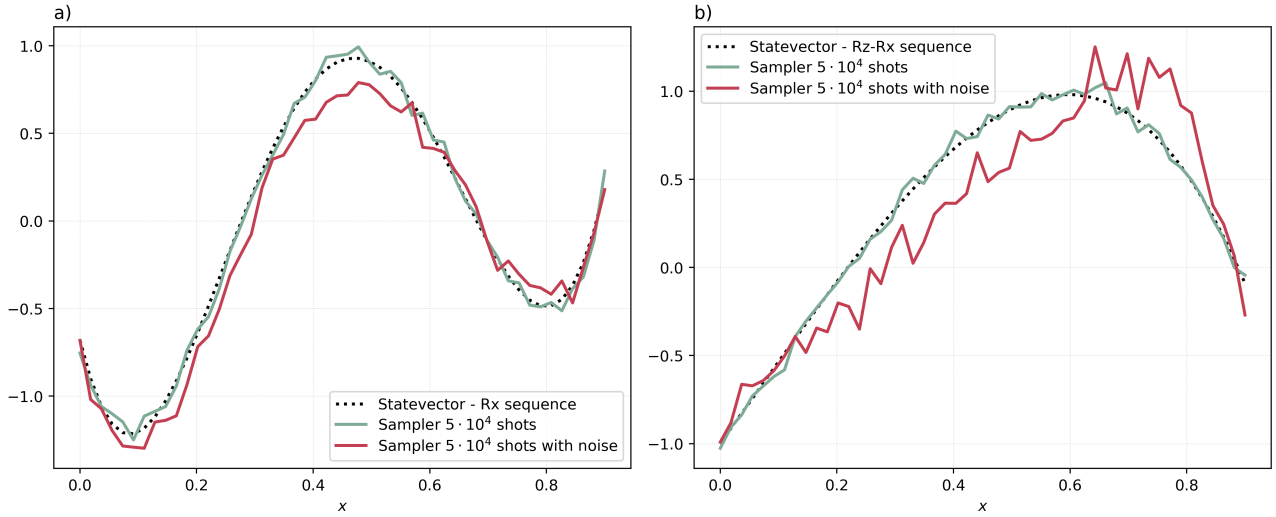


Figure 6.9: Impact of noise on the Lagrange encoded function circuit for two HEA, evaluated under the two measurement settings. Simulations were performed with 50 thousand shots on both the noise-free *AerSimulator* and the noisy *FakeNairobiV2* simulator, in comparison to the statevector value a) Single-gate sequence $\hat{R}_X$, and b) Two-gate sequence $\hat{R}_Z - \hat{R}_X$.

In both cases, the noise introduces oscillations and appears to reduce the amplitude of the output. The circuits were executed with five, ten, and fifty thousand shots. For the results with five and ten thousand shots shown in Figure 6.8, we observe in both ansätze that the simulations performed in the noisy environment display more pronounced oscillations than in the noise free environment, while still preserving a clear overall trend of the solution.

In Figure 6.9, we see that increasing the number of gates in the sequence of the variational ansatz decreases the accuracy, both in the noise free environment when using sampling measurements and even more in the noisy environment, whereas for the single gate sequence the noisy result remains relatively close to the noise-free one.

Simulation of a random function with Kyriienko et al. VQA

In a similar approach, we looked here at the measurement of a function circuit inspired from the work of Kyriienko et al. [4], with a structure over 6 qubits and a depth of 5, for a HEA sequence of only a single $\hat{R}_X$ gate and a three-gate sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, simulating a random function, first with the IBM Qiskit noise-free *AerSimulator* quantum simulator and then with the noisy simulator *FakeNairobiV2*.

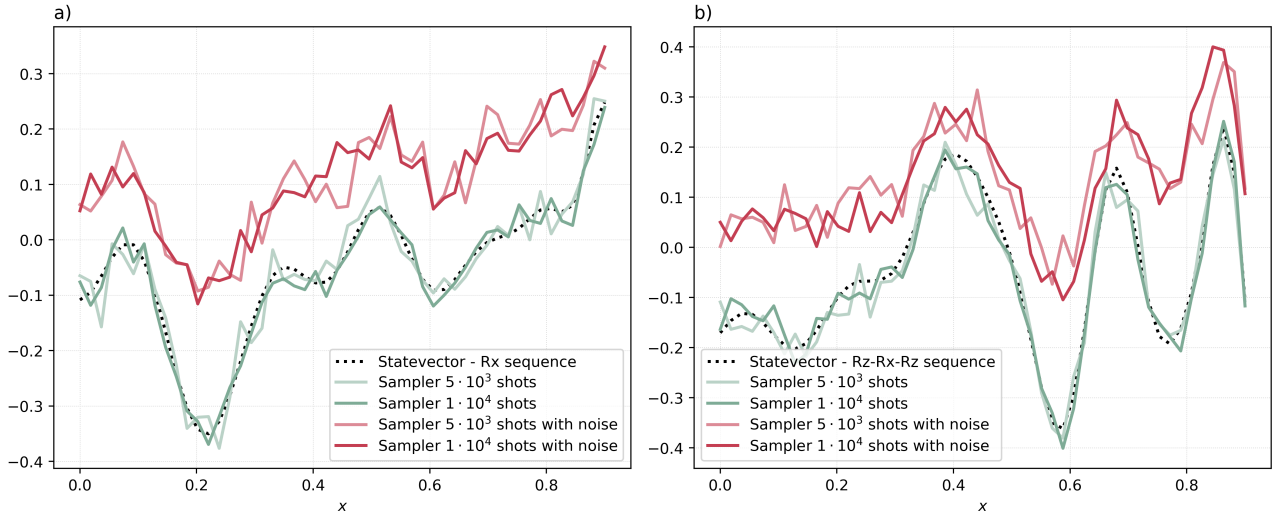


Figure 6.10: Comparison of a Chebyshev-encoded circuit implemented with two hardware-efficient ansätze, evaluated under two measurement settings. Simulations were performed on the IBM Qiskit noise free *AerSimulator* and noisy *FakeNairobiV2*, using 5 and 10 thousand shots, and compared to the Statevector value. a) Single-gate HEA sequence $\hat{R}_X$, b) Three-gate HEA sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$.

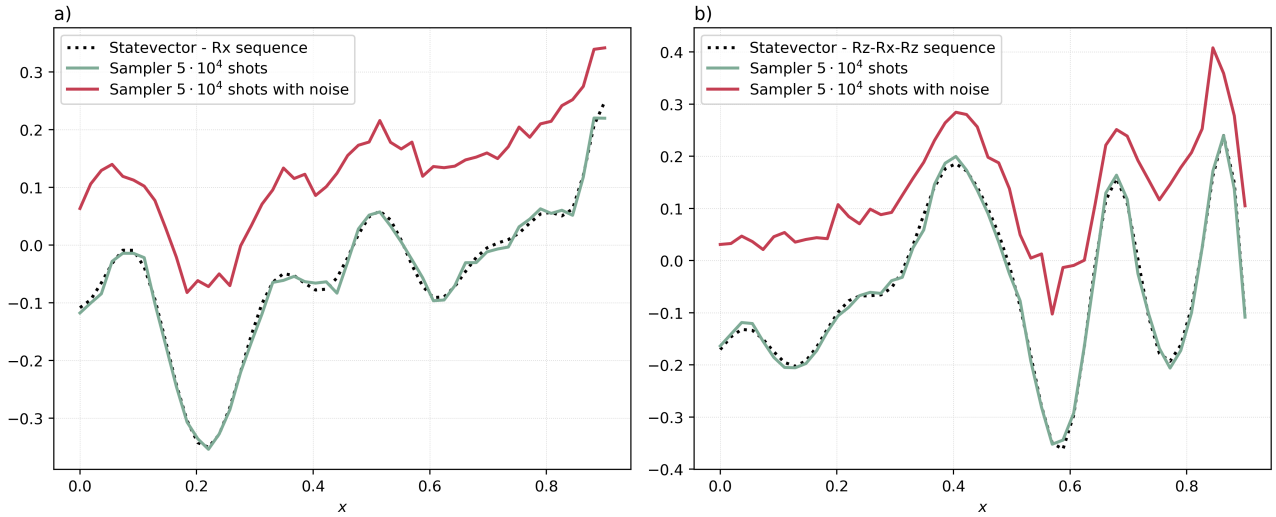


Figure 6.11: Comparison of a Chebyshev-encoded circuit implemented with two hardware-efficient ansätze, evaluated under two measurement settings. Simulations were performed on the IBM Qiskit noise free *AerSimulator* and noisy *FakeNairobiV2*, using 50 thousand shots, and compared to the Statevector value. a) Single-gate HEA sequence $\hat{R}_X$, b) Three-gate HEA sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$.

Noise-free simulation In Figures 6.10 and 6.11, the result obtained for the single-gate HEA circuit in a noise-free environment is similar than the one observed for the third version of the Lagrange encoding. The oscillations around the statevector value gradually decrease as the number of shots increases, up to fifty thousand.

Simulation with noise With the introduction of noise in the function circuit simulation, as illustrated in Figure 6.10 and 6.11, we again observe that the presence of noise amplifies the oscillations compared to the noise-free case, and introduces this time a more significant shift in the solution, along with a reduction in overall magnitude. Nevertheless, the general trend remains clearly identifiable.

6.1.1 Summary

This brief analysis helps to contextualize the algorithms discussed and evaluated in this work. It clearly demonstrates that the raw output from a noisy quantum device cannot be used directly and will require post-processing techniques for error mitigation. By adjusting certain fixed coefficients, we observed that the Lagrange encoding can be improved between the initial version proposed in our paper [1] and the third version presented in this section. Future work will therefore include a deeper investigation to improve the quantum feature map circuit measurement sampling output. While a higher number of parameterized gates in the HEA would be expected to lead to greater noise accumulation, this behaviour was not clearly observed when using the simulator *FakeNairobiV2* on the Chebysehv-encoded VQA, which actually contains a high number of variational gates. Finally, obtaining results that are both interesting and sufficiently accurate comes at a cost, as they appear to require a very high number of shots for each circuit evaluation. In the context of solving differential equations, this overhead propagates to the computation of the function circuit itself, its derivatives with respect to x, and the gradient with respect to θ. As shown in the previous chapter, the number of evaluations per iteration is already substantial. The situation becomes even more demanding with a three-gate HEA, as it introduces three times as many parameterized gates and, consequently, three times as many partial derivatives for gradient estimation. Therefore, when possible, it looks preferable to prioritise an HEA with fewer gates, reducing the number of circuits required to estimate the gradient.

6.2 Derivative Quantum Circuit Simulation

In the previous section, we observed that sampling measurements obtained from a simulator or quantum hardware differ from the exact statevector values. Within the context of reducing the overall complexity of the variational quantum algorithm, the goal is not only to decrease the number of gates and parameters, but also to minimize the number of circuits required in a single iteration. As discussed in the previous chapter, a key feature of VQAs lies in the ability to compute exact analytical derivatives of the circuit function. This is achieved using either the parameter-shift rule or the Hadamard test differentiation, combined with the chain rule [4], [106], [134], or differentiated encoding (Section 5.2.5) which strongly distinguished quantum approaches from classical finite element methods.

Computing the analytical quantum derivative requires between nM and $2nM$ derivatives quantum circuits, with n being the number of encoding gates and M the number of points considered, which are the training points in most cases. This raises the question of whether alternative methods, such as the forward difference approximation (FDA) or the central difference approximation (CDA), could be employed to reduce the number of required circuits by a factor of n or $2n$. Although FDA and CDA are closely related to finite element methods, they do not necessitate a fine mesh across the entire interval, which would not be appropriate in our setting, as the objective is to train the parameterized circuits on a small number of points, typically up to 20. It is important to note that VQAs based on polynomial interpolation are, by definition, spectral methods. For each training point x within the chosen interval, we can compute the parameterized circuit at the perturbed value $x + \epsilon$, where ϵ is assumed to be small, leading to the definition of a first and second order accurate methods:

$$\text{FDA: } \frac{df}{dx} = \frac{f(x+\epsilon) - f(x)}{\epsilon} + o(\epsilon) \tag{6.9}$$

$$\text{CDA: } \frac{df}{dx} = \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} + o(\epsilon^2). \tag{6.10}$$

In this case, applying the FDA requires only M additional circuit evaluations, on top of those already needed to compute the function f itself, while the CDA requires $2M$ additional circuits. Although these methods appear appealing in terms of reducing computational resources, we will examine in this section the cases in which they can be employed and how they differ from the exact derivatives obtained through dedicated derivative quantum circuits.

As discussed in the previous section, when simulating quantum circuits to study quantum algorithms, one can either work with the ideal measurement obtained directly from the statevector of the circuit, or employ a simulator that uses sampling-based measurements, aiming to mimic either a noise-free quantum system or a realistic noisy device. The final step typically involves testing on an actual quantum processor, depending on the stage of algorithm development. In this work, we primarily focus on the first approach, while examining the second with some curiosity in this chapter. As highlighted in Chapter 3, simulating VQAs can be computationally expensive. In this section, we will explore strategies to adapt and optimize the simulation process, transitioning from statevector-based simulations to sampling-based simulations.

6.2.1 Derivatives of Chebyshev-encoded VQCs

For a Chebyshev-encoded parameterized quantum circuit with $n = 4$ qubits and depth $d = 3$, constructed with random variational parameters θ, the application of the FDA on ideal statevector measurements becomes relevant for values of ϵ starting from $1 \cdot 10^{-2}$. The mean square error (MSE) between the derivatives obtained using the parameter-shift rule (PSR) and those computed via FDA decreases as the finite difference step ϵ becomes smaller, as illustrated in Figure 6.12.

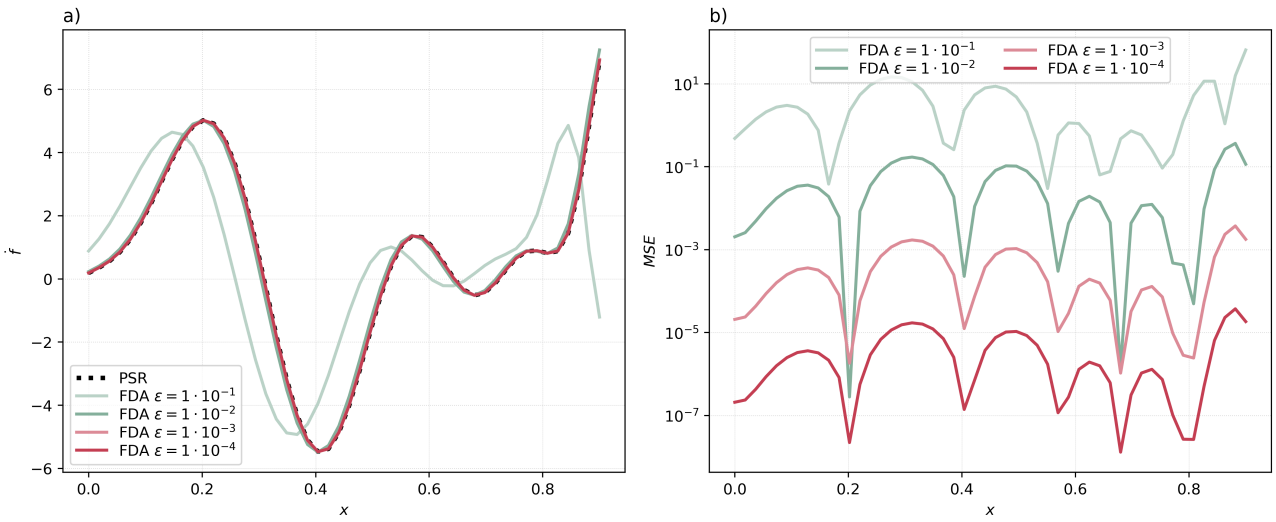


Figure 6.12: Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Forward Difference Approximation (FDA) from ideal statevector measurements, applied to a random function constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and FDA. b) Mean Square Error (MSE) between PSR and FDA derivatives.

When considering sampling-based measurements on the noise-free simulator AerSimulator [147] (using 50 thousands shots) from IBM Qiskit, the performance of the FDA drops significantly, suggesting a variance amplification due to finite differences. In Figure 6.13.a), the FDA result that appears most accurate corresponds to a relatively large value of ϵ. Larger step sizes were also tested, but they produced results so poor that including them would have obscured the oscillations and inaccuracies visible in the three cases presented.

Examining the MSE in Figure 6.13.b), the three FDA outcomes exhibit similar and poor accuracy overall. In contrast, the PSR result, shown in red, remains robust. Even under sampling-based measurements, the PSR derivative not only outperforms FDA but also nearly overlaps with the ideal statevector-based PSR result, highlighting its reliability.

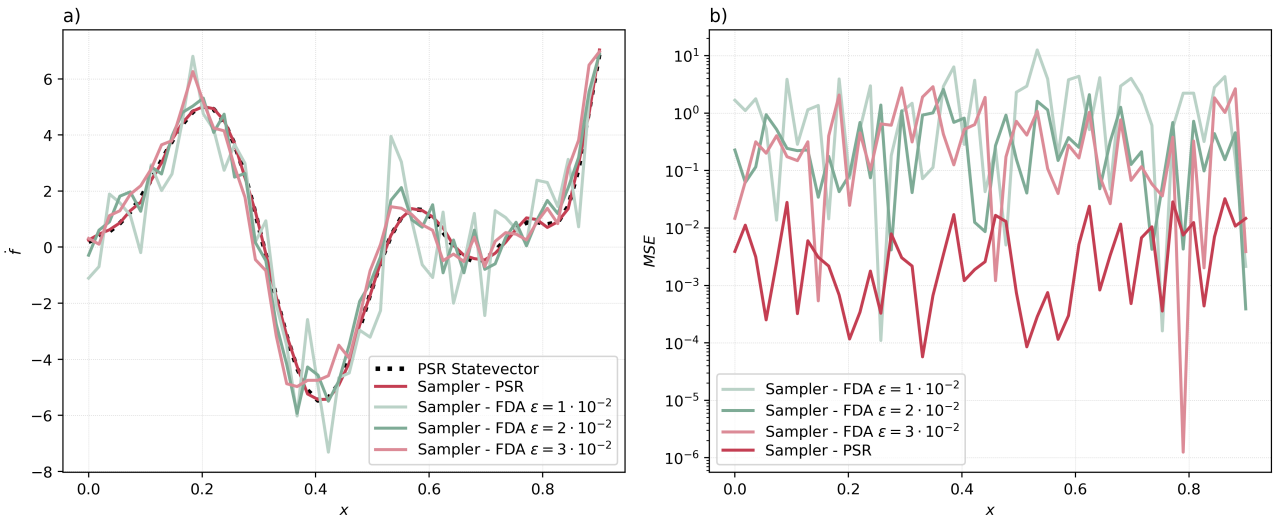


Figure 6.13: Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Forward Difference Approximation (FDA) from sampling-based measurements on a noise-free simulator, as well as PSR from the ideal statevector measurement. The test function is constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and FDA. b) Mean Square Error (MSE) between PSR and FDA derivatives.

Finally, when applying the Central Difference Approximation (CDA) with sampling-based measurements on the noise-free simulator, the most accurate result still corresponds to a relatively large step size ϵ. Although CDA demonstrates significantly higher accuracy compared to FDA for similar values of ϵ, its performance remains inferior to that of the Parameter-Shift Rule, which continues to provide the most reliable results, as shown in Figures 6.13.b) and 6.14.b).

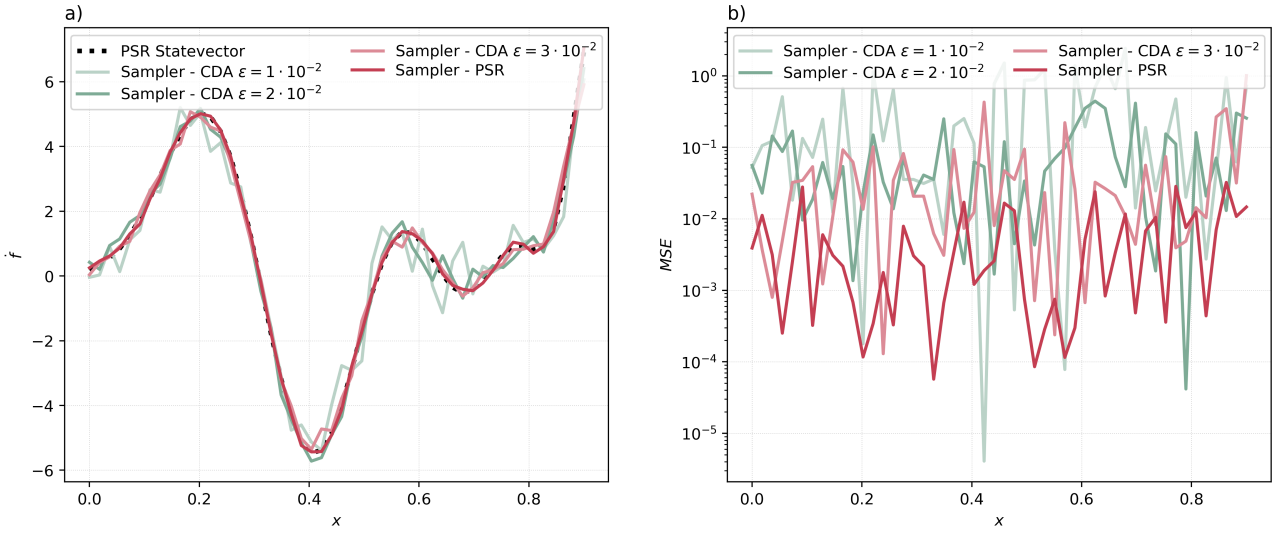


Figure 6.14: Comparison between derivatives obtained using the Parameter-Shift Rule (PSR) and the Central Difference Approximation (CDA) from sampling-based measurements on a noise-free simulator, as well as PSR from the ideal statevector measurement. The test function is constructed with a Chebyshev-encoded parameterized quantum circuit ($n = 4$, $d = 3$). a) Derivative with respect to x computed using PSR and CDA. b) Mean Square Error (MSE) between PSR and CDA derivatives.

6.2.2 Derivatives of Lagrange-encoded VQCs

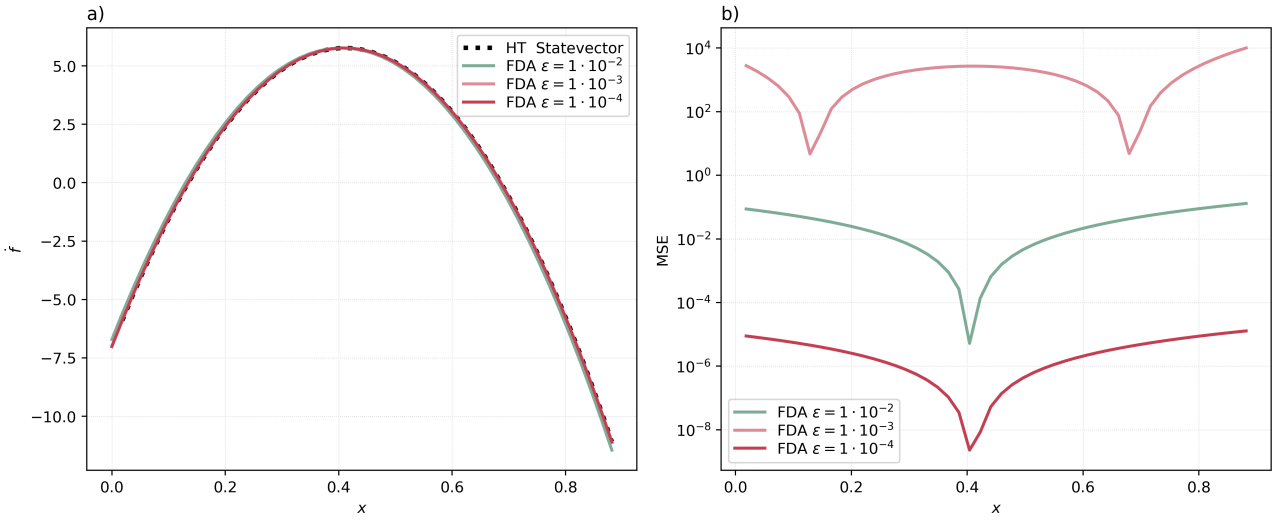


Figure 6.15: Comparison between derivatives obtained using the Hadamard test differentiation (HT) and the Forward Difference Approximation (FDA) from ideal statevector measurements, applied to a random function constructed with a Lagrange-encoded parameterized quantum circuit (with the second version of the encoding function) with $n = 4$ qubits. a) Derivative with respect to x computed using HT and FDA. b) Mean Square Error (MSE) between HT and FDA derivatives.

This time looking at a Lagrange-encoded parameterized quantum circuit (with the second version of the encoding function) with $n = 4$ qubits, constructed with random variational parameters θ, the application of the FDA on ideal statevector measurements becomes relevant for values of ϵ starting from $1 \cdot 10^{-2}$, similarly to the Chebyshev-encoded circuit. The mean square error (MSE) between the derivatives obtained using the Hadamard test differentiation (HT) and those computed via FDA does not systematically decrease as the finite difference step ϵ becomes smaller, as illustrated in Figure 6.15.

As observed for the Chebyshev encoding, the performance of the FDA deteriorates markedly under sampling-based measurements on the noise-free simulator *AerSimulator* [147], likely due to variance amplification inherent to finite differences, as illustrated in Figure 6.16.

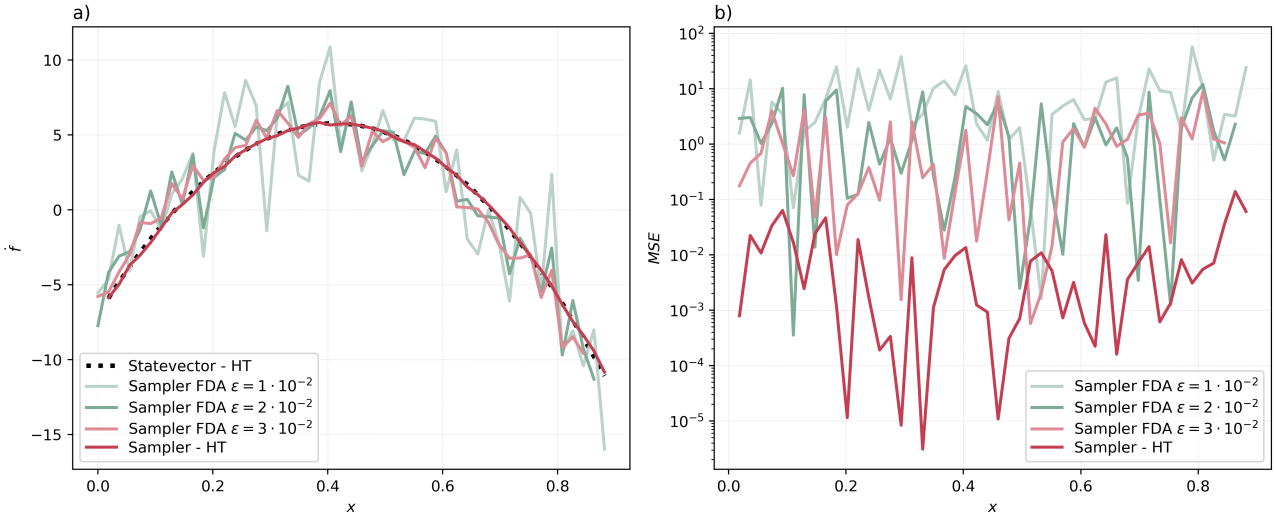


Figure 6.16: Comparison between derivatives obtained using the Hadamard test differentiation method (HT) and the Forward Difference Approximation (FDA) from sampling-based measurements on a noise-free simulator, as well as HT from the ideal statevector measurement. The test function is constructed with a Lagrange-encoded parameterized quantum circuit ($n = 4$ qubits, version 2). a) Derivative with respect to x computed using HT and FDA. b) Mean Square Error (MSE) between HT and FDA derivatives.

Remark As a side note, it is worth highlighting that when using the second version of the Lagrange encoding function introduced earlier, sampling-based measurements with *AerSimulator* remain highly accurate for the function itself, as shown in Figure 6.17(a). More interestingly, Figure 6.17b) reveals that the derivatives computed via derivative quantum circuits exhibit an almost perfect overlap between statevector-based ideal measurements and sampling-based measurements. This observation is

particularly noteworthy, as it suggests that the sum of all partial derivatives with respect to each encoding gate may effectively average out statistical fluctuations, thereby smoothing the result. This effect is visible not only with the Parameter-Shift Rule, as previously discussed, but also with the Hadamard test differentiation method (using automatic differentiation or differentiated encoding), which requires only n evaluations.

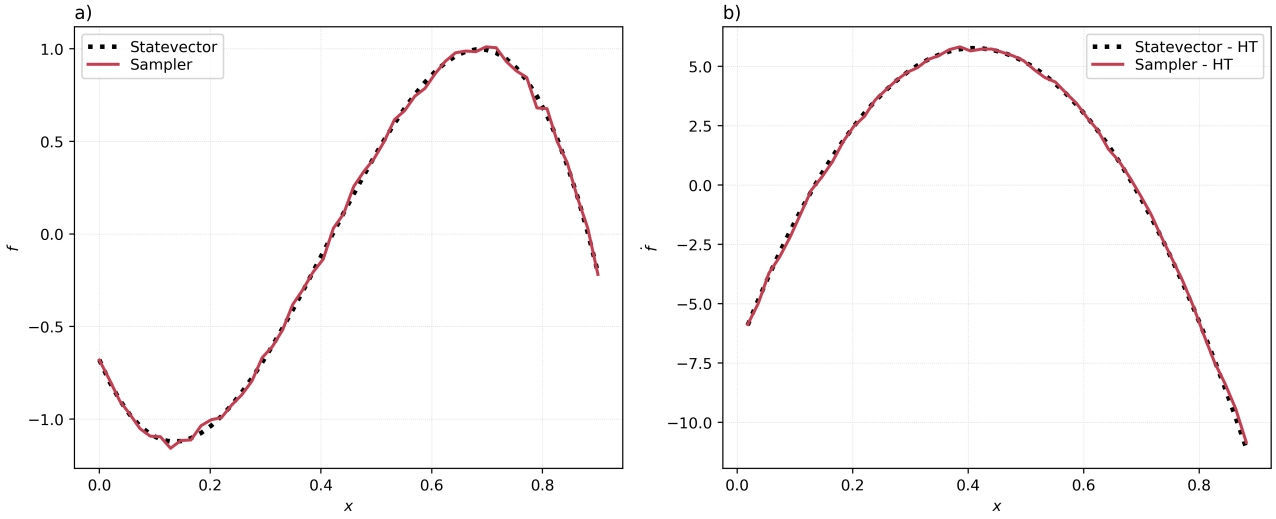


Figure 6.17: Comparison from ideal statevector measurements and the *AerSimulator* sampling measurements, for, a) the function circuit f, b) the derivative of the function circuit using the Hadamard test differentiation method

Finally, although one might have expected an improvement in accuracy when moving from FDA to CDA for the Lagrange-encoded VQC, Figure 6.18 demonstrates that this is not the case. Even the MSE, shown in Figure 6.18.b), does not indicate any notable improvement between the two methods. While for Chebyshev-encoded circuits, CDA could potentially be applied to simulated sampling measurements and eventually to real-device implementations, using a carefully chosen step size to accelerate the algorithm and reduce complexity without significantly compromising accuracy, the large oscillations observed here discourage the use of either CDA or FDA. In this context, derivatives computed via derivative quantum circuits clearly outperform finite-difference approximations.

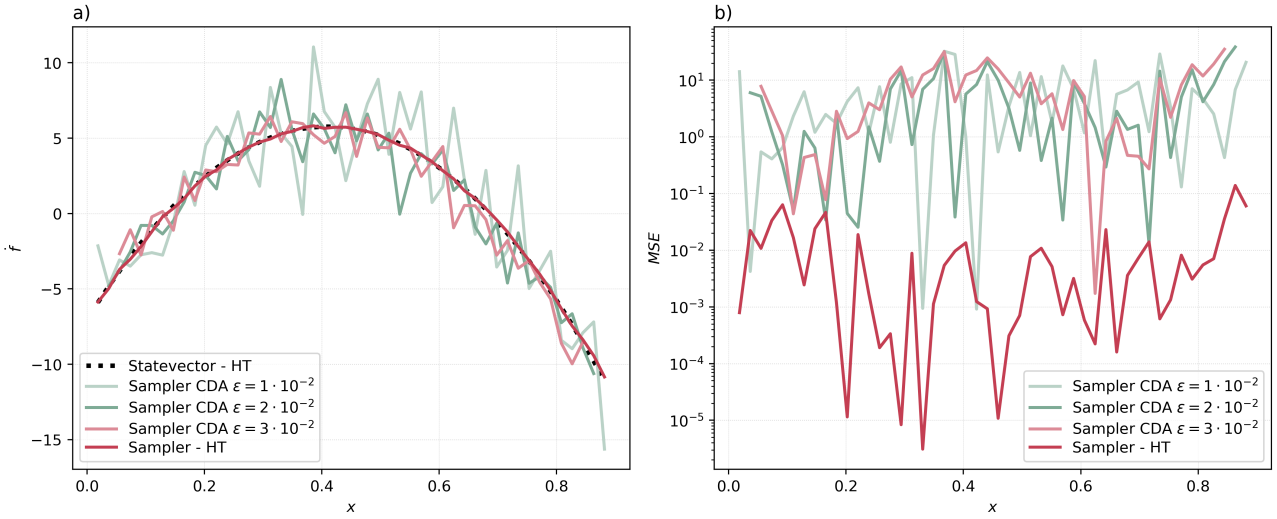


Figure 6.18: Comparison between derivatives obtained using the Hadamard test differentiation method (HT) and the Central Difference Approximation (CDA) from sampling-based measurements on a noise-free simulator, and HT from the ideal statevector measurement. The function is constructed with a Lagrange-encoded parameterized quantum circuit ($n = 4$ qubits, version 2). a) Derivative with respect to x computed using HT and CDA. b) Mean Square Error (MSE) between HT and CDA derivatives.

6.2.3 Summary

The quick analysis of the use of finite difference approaches to determine Chebyshev- and Lagrange-encoded circuit derivatives help to draw some conclusions. When simulating derivatives using ideal statevector measurements, FDA provides a computationally attractive approach, requiring only M additional circuit evaluations. This makes it appealing for early-stage algorithm development, as classical simulation of VQAs remains expensive in computational resources. However, when considering sampling-based measurements, even on a noise-free simulator, FDA suffers from variance amplification, resulting in poor accuracy. CDA improves over FDA in some cases, but it still underperforms compared to derivative quantum circuits, particularly for Lagrange-encoded VQCs where oscillations are pronounced.

Derivative quantum circuits consistently yield highly accurate results, both for Chebyshev- and Lagrange-encoded circuits, we noticed an almost perfect overlap between statevector and sampling-based simulations. This suggests that summing over partial derivatives can average out statistical fluctuations, highlighting the robustness of quantum differentiation techniques.

Overall, while FDA is useful for exploratory simulations in the early stage, derivative quantum circuits

provide the most reliable results and are essential when evaluating the true complexity of VQAs. These findings underscore the promise of quantum spectral methods, in which exact analytical derivatives are essential not only for computing derivatives with respect to x, but also for obtaining gradients with respect to θ, which drive the training of parameterized quantum circuits.

6.3 Variational Quantum Algorithm Runtime

6.3.1 Runtime of VQAs on quantum computers

Running quantum algorithms on real devices is fundamentally different from classical simulation. As discussed in Chapter 2, the runtime of a quantum circuit can be expressed as a sequence of steps, each corresponding to a set of gates executed in parallel. The duration of these steps depends on the type of operations applied. For simplicity, the runtime can be classified into three categories: single-qubit gates, two-qubit gates, and measurement. As shown in Section 2.4, execution times vary depending on the hardware considered, but a general hierarchy can be observed: measurement typically takes around one to five times longer than two-qubit gates, which themselves are about an order of magnitude slower than single-qubit gates.

Considering the two polynomial-encoded VQAs described so far, and assuming ideal connectivity and no error correction (since the latter would introduce additional CNOT gates and limit parallelisation), we can estimate their respective minimum runtimes as follows.

Lagrange-encoded VQA

For the Lagrange-encoded VQA, the runtime depends on the structure of the algorithm, as detailed in Chapter 4. For the extended version of the quantum feature map, the large CNOT network can be partially parallelised, resulting in approximately $(n-1)$ steps. Taking into account all components of the feature map, we obtain:

$$N_{qfm,Le} = 3T_1 + 2(n-1)T_2, \tag{6.11}$$

where T_1 is the execution time of a single-qubit gate, T_2 the execution time of a two-qubit gate, and n the number of encoding points. For the reduced version of the Lagrange-encoded VQA, which uses fewer qubits but requires more steps, the runtime is given by:

$$N_{qfm,Lr} = 2n(T_1 + T_2). \tag{6.12}$$

In both cases, the complete circuit includes additional steps for the variational ansatz (which in this work is reduced to one step, though more could be added) and a final measurement step, typically the longest in the execution process.

Chebyshev-encoded VQA

For the Chebyshev-encoded VQA proposed by Kyriienko *et al.*[4], the runtime is dominated by the hardware-efficient ansatz, while the encoding itself requires only a single step. This is the opposite of the Lagrange-encoded approach. The runtime can be approximated by:

$$N_{hea,C} = (dN_{\text{sequence}})T_1 + d(n-1)T_2, \tag{6.13}$$

where d is the depth of the circuit and N_{sequence} is the number of single-qubit gates in the sequence of the hardware-efficient ansatz.

Summary

The estimated runtimes for the different approaches are summarised in Table 6.2.

	Quantum Feature Map	Variational Ansatz	Measurement
Lagrange-encoded Extended	$3T_1 + 2(n-1)T_2$	$\sim T_1$	T_m
Lagrange-encoded Reduced	$2n(T_1 + T_2)$	$\sim T_1$	T_m
Chebyshev-encoded	T_1	$(dN_{\text{sequence}})T_1 + d(n-1)T_2$	T_m

Table 6.2: Summary of runtime components. Here d denotes the circuit depth, N_{sequence} the number of single-qubit gates per hardware-efficient layer, T_1 and T_2 the execution times of single- and two-qubit gates respectively, T_m the measurement time, and n the number of encoding points.

Example For a Lagrange structure with $n = 8$ and a Chebyshev structure with $n = 6$ and $d = 5$, the number of steps per execution-time category is as follows: for the reduced Lagrange-encoded VQC, $17T_1 + 16T_2$, and for the Chebyshev-encoded VQC, $16T_1 + 25T_2$. Assuming that two-qubit gates take up to ten times longer to execute, in this example, the Chebyshev-encoded VQA would require approximately 1.56 times more execution time than the Lagrange-encoded VQA. The two structures are chosen based on equivalent performance rather than size, as they exhibited comparable results in Chapter 7, ensuring a fair comparison.

6.3.2 Runtime of simulated VQAs

Simulating variational quantum algorithms can be computationally demanding, since their principle relies on many iterations and consequently a large number of circuit evaluations. As discussed in the previous section, when working with statevectors, the use of a forward difference approximation can slightly reduce the overall computational cost to obtain the derivatives and gradients. However, as the number of parameters increases, the number of required circuits grows accordingly, resulting in a substantial computational burden.

The time needed to obtain the statevector of a quantum circuit increases not only with the number of gates, and thus the circuit depth, but also with the number of qubits, even for an empty circuit. This strongly limits the size of circuits that can be simulated or the number of iterations that can be performed within a reasonable runtime.

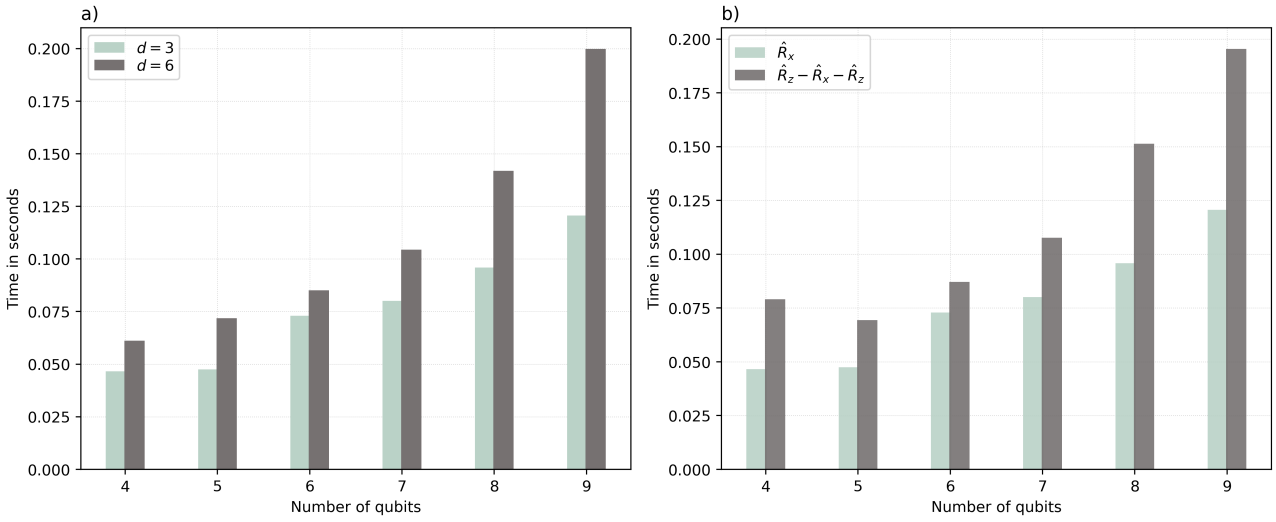


Figure 6.19: Runtime of the Chebyshev-encoded circuit measured with total Z measurement for 20 points with a varying number of qubits. a) Evolution of the runtime with a depth increase from 3 to 6, with 1 gate sequence $\hat{R}_X$, b) Evolution of the runtime with a HEA gate sequence increase from $\hat{R}_X$ to $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, with a depth of 3.

For example, Figure 6.19 shows the runtime of statevector simulation for a Chebyshev-encoded VQC evaluated over 20 points, for different numbers of qubits. In Figure 6.19.a, the circuit depth grows from 3 to 6, while in Figure 6.19.b, the HEA sequence increases from 1 gate sequence $\hat{R}_X$ to 3 gates sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$. As observed, classical simulation cannot reproduce the parallelism inherent to quantum hardware. Consequently, even with modern simulators, the need for real devices remains

essential to unlock the potential of quantum algorithms.

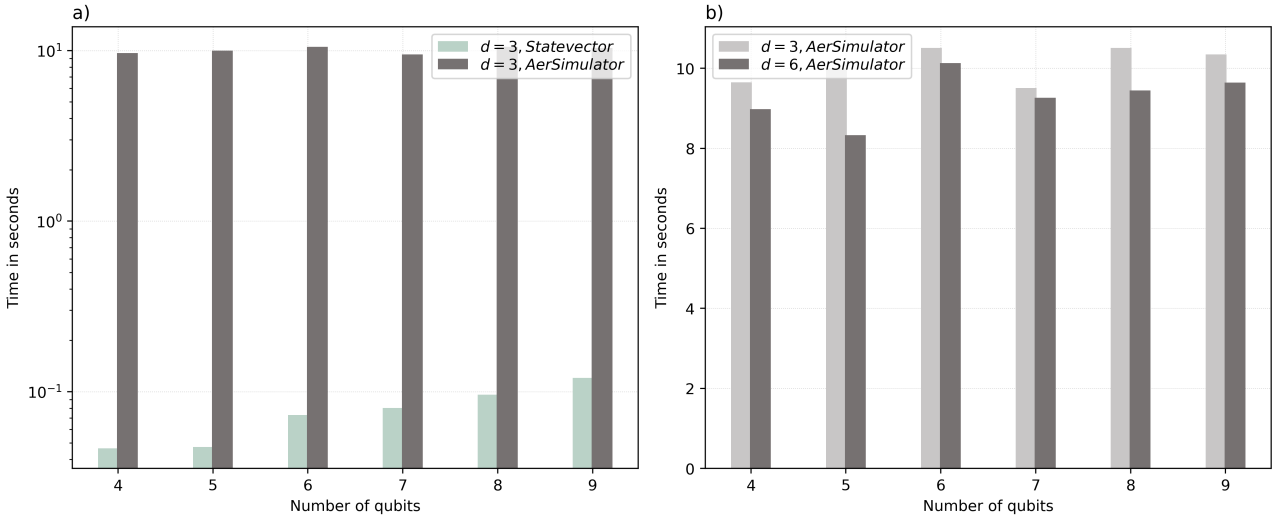


Figure 6.20: Runtime of the Chebyshev-encoded circuit measured with total Z measurement for 20 points with a varying number of qubits. a) Evolution of the runtime between statevector measurement and noise-free simulation using IBM AerSimulator, with a depth of 3 and HEA sequence of $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, b) Evolution of the runtime with a depth increase from 3 to 6, with 1 gate sequence $\hat{R}_X$, measured by the noise-free simulator IBM AerSimulator.

Working with statevectors in the early stages of quantum algorithm development offers multiple advantages. First, it allows to peak inside the quantum state and look at the statevector to better understand what the algorithm is actually doing. But it also provides significant efficiency: statevector simulations can evaluate a quantum circuit approximately 100 to 200 times faster than a quantum sampler simulator such as IBM Qiskit's AerSimulator, as shown in Figure 6.19. Nevertheless, samplers remain necessary to assess how a VQC would behave under measurement, and to identify cases where an encoding might only work theoretically, as discussed in the first section of this chapter.

Here we performed the statevector simulation and the noise-free sampler measurement using the IBM AerSimulator on a single CPU core, not to evaluate the absolute execution time but rather to compare the relative differences in runtime between various measurement methods and circuit sizes.

Classical simulation of a VQA also presents structural advantages. For a given iteration, all circuits can be simulated independently and therefore in parallel, for each training point and each partial derivative. This benefit becomes increasingly significant when higher-order derivatives are required, and consequently accelerate the early development of quantum algorithms, specially VQAs which requires many iterations.

6.4 Complexity Estimation - Gate Count

Finally, to understand and evaluate the performance of VQAs, a last interesting point is the gate count, which represents an interesting metric for a few reasons. First, reducing the number of gates of a circuit decreases the number of gate errors, thereby contributing to its robustness and precision, a is very important aspect in the NISQ era [148].

Additionally, as seen in the previous chapters, a high gate count is linked to difficult trainability of the VQC due to barren plateaus, as well as high runtime. In the realm of VQAs in the NISQ era, aiming for a shallow circuit, and consequently reducing the depth of the VQC, is the strategy that has been highlighted by the field so far [110], [148], [149].

Finally, while looking at all different metrics is obviously more precise and exhaustive, we also introduced the gate count of the overall VQA, for comparing methods, which takes into account the number of circuits per iteration (directly impacting the runtime), as well as the number of iterations (relating to the trainability of the VQC and also assesses the optimizer). It has to be noted that this metric should not be taken in isolation, but it gives a great overview.

6.4.1 Chebyshev-encoded VQA

Regarding the quantum solver inspired by the work of Kyriienko *et al.* [4], one iteration involves, as detailed in previous Chapters, circuits for the functions f_θ, its derivatives $\frac{df_\theta}{dx}$; $\frac{d^2 f_\theta}{dx^2}$, as well as their respective gradients $\nabla_\theta(f_\theta)$; $\nabla_\theta(\frac{df_\theta}{dx})$; $\nabla_\theta(\frac{d^2 f_\theta}{dx^2})$ for all controlled points and boundary conditions. In this algorithm, the gradients are obtained using the parameter shift rule method, which requires two circuits for a partial derivative. As a result, the number of circuits per iteration is defined as follows:

$$N_{\text{Circuits/iteration}} = N_{\text{Points}}(1 + 2N_{\text{Parameters}}) \left(N_{\text{Circuits}}(f_\theta) + N_{\text{Circuits}}(\frac{df_\theta}{dx}) + N_{\text{Circuits}}(\frac{d^2 f_\theta}{dx^2}) \right), \quad (6.14)$$

where N_{Circuit} is zero if the respective component does not appear in the differential equation. In the case of the damped-mass-spring-system problem solved in section 7.2, the differential equation is

composed of all the components mentioned above.

$$\begin{cases} N_{\text{Points}} = N_{\text{Controlled points}} + 1, \\ N_{\text{Parameters}} = N_{\text{qubits}}\ N_{\text{layers}}\ N_{\text{sequence}}, \\ N_{\text{Circuits}}(f_\theta) = 1, \\ N_{\text{Circuits}}(\frac{df_\theta}{dx}) = 2N_{\text{qubits}}\ , \\ N_{\text{Circuits}}(\frac{d^2 f_\theta}{dx^2}) = 4N_{\text{qubits}}^2\ . \end{cases} \tag{6.15}$$

Moreover, the Chebyshev-encoded VQA employs a quantum circuit structure based on basic quantum gates such as CNOT gates and single-qubits rotational gates $\hat{R}_{X,Y,Z}$. The count of gates per circuit can be expressed as follows:

$$N_{\text{Basic gates/circuit}} = N_{\text{qubits}}\ + N_{\text{Parameters}} + N_{\text{layers}}\ (N_{\text{qubits}}\ - 1). \tag{6.16}$$

6.4.2 Lagrange-encoded VQA

As outlined in Chapter 4 and depicted in Figure 4.1, each iteration of the Lagrange-based algorithm draws inspiration from the methodology proposed by Kyriienko *et al.* [4]. Consequently, the number of circuits per iteration can be defined by Equation (6.14). However, due to variations in encoding and structure, the derivative circuits with respect to x are obtained via the Hadamard test differentiation method, or differentiated encoding, necessitating two times fewer circuits per derivative.

$$\begin{cases} N_{\text{Circuits}}(\frac{df_\theta}{dx}) = N_{\text{Encoding points}}\ , \\ N_{\text{Circuits}}(\frac{d^2 f_\theta}{dx^2}) = N_{\text{Encoding points}}^2\ . \end{cases} \tag{6.17}$$

In both the extended and reduced structures, the derivative circuit necessitates an additional gate. Notably, the reduced structure exhibits a slightly higher count of gates overall, despite being composed of half the number of qubits.

For the extended structure:

$$N_{\text{Basic gates/circuit}}(f_\theta) = 2N_{\text{Encoding points}}\ (N_{\text{Encoding points}}\ + 1) + N_{\text{Parameters}} \tag{6.18}$$

For the reduced structure:

$$N_{\text{Basic gates/circuit}}(f_\theta) = 2\ (3N_{\text{Encoding points}}\ - 1) + N_{\text{Parameters}}. \tag{6.19}$$

And in both cases,

$$\begin{cases} N_{\text{Basic gates/circuit}}(\frac{df_\theta}{dx}) = N_{\text{Basic gates/circuit}}(f_\theta) + 1 \\ N_{\text{Basic gates/circuit}}(\frac{d^2 f_\theta}{dx^2}) = N_{\text{Basic gates/circuit}}(f_\theta) + 2. \end{cases} \tag{6.20}$$

Additionally, in the application of this algorithm, the number of qubits can evolve throughout the algorithm depending on the number of interpolation points. Consequently, the estimation of the number of circuits as well as the number of gates should be considered step by step.

6.4.3 Sato *et al.* VQA

The Sato *et al.* algorithm [5] was designed to solve the Poisson equation example described in Chapters 3, 7. This algorithm employs a fundamentally different approach and circuit structure. For each iteration, evaluating the loss function, defined as the total potential energy of the system requires 3, 4, or 5 circuits for periodic, Dirichlet, or Neumann boundary conditions, respectively. Additionally, the derivative of the loss function for each variational parameter must be evaluated. This is achieved using the Hadamard test differentiation method, which requires only one modified circuit per partial derivative. Thus, the number of circuits needed for one iteration can be expressed as follows:

$$N_{\text{Circuits/iteration}} = N_{\text{Observables}}\ (1 + N_{\text{Parameters}}), \tag{6.21}$$

where $N_{\text{Observables}} = 3, 4$ or 5 depending on the boundary condition, and $N_{\text{Parameters}} = N_{\text{Layers}} \times N_{\text{Parameters/layer}} + N_{\text{Encoding qubits}}$. For the Poisson equation solved in the Results section, 5 qubits are used for encoding, the number of layers of the ansatz is set to 5, and the number of parameters per layer is 8, resulting in a total of 45 parameters. To account for the gate complexity of the algorithm, the number of basic quantum gates (Pauli and Clifford gates) per iteration is determined as follows:

$$N_{\text{Basic gates/circuit}} = 3 + N_{\text{Basic gates}}(\hat{\mathcal{U}}_b) + N_{\text{Basic gates}}(\hat{\mathcal{U}}_\theta) + N_{\text{Shift Circuits}} N_{\text{Basic gates}}(\hat{\mathcal{U}}_{\text{Shift}}), \tag{6.22}$$

$$\text{with, } \begin{cases} N_{\text{Basic gates}}(\hat{\mathcal{U}}_b) = N_{\text{Encoding qubits}} + 1, \\ N_{\text{Basic gates}}\,(\hat{\mathcal{U}}_{\text{Shift}}) = 1 + N_{\text{Encoding qubits}}\, N_{\text{Basic gates}}(MCP), \\ N_{\text{Basic gates}}\,(\hat{\mathcal{U}}_\theta) = N_{\text{Parameters}}\, N_{\text{Basic gates}}\,(CRY) + N_{\text{Layers}}\, N_{\text{Basic gates}}\,(MCP). \end{cases} \tag{6.23}$$

The CRY quantum gate refers to the controlled rotational gate around the Y axis, which can be decomposed into four basic quantum gates. The MCP quantum gate, or the multi-controlled phase gate, is equivalent to a Toffoli quantum gate in the case of two controlled qubits, which can be decomposed into 18 basic quantum gates.

6.5 Summary

In this chapter, we explored the practical challenges of running VQAs on real quantum devices, highlighting how they differ fundamentally from classical simulations. We showed that factors such as gate durations, measurement times, and inherent noise can strongly affect the accuracy of results. In this context, the Lagrange-encoded VQA was improved to remain readable and reliable even when using sampling measurements, addressing a key limitation of earlier versions.

We observed that hardware-efficient ansätze with many parameterized gates, while theoretically powerful, tend to increase the complexity of the algorithm. This extra overhead propagates directly to the evaluation of derivatives and gradients, which are essential for training the VQA to solve differential equations.

Next, we investigated derivative computation methods. Finite-difference approaches, such as FDA and CDA, are computationally attractive in statevector simulations, but they suffer from amplified variance under sampling, limiting their reliability. In contrast, derivative quantum circuits consistently delivered highly accurate results for both Chebyshev- and Lagrange-encoded VQCs, even under realistic sampling conditions. This highlighted the robustness of quantum differentiation techniques and the crucial role of exact analytical derivatives in quantum spectral methods.

We also examined the role of circuit depth and gate count, which remain critical metrics in the NISQ era. Reducing the number of gates not only decreases error accumulation and improves precision, but also mitigates barren-plateau effects and shortens runtime. By combining metrics such as total gates per iteration and the number of iterations, we obtained a more complete picture of VQA performance, while recognizing that no single metric tells the whole story.

Overall, this chapter indicates that a performant polynomial-encoded VQAs relies on carefully structured, shallow circuits and the use of derivative quantum circuits for accurate gradients. These design choices together pave the way for accurate, efficient, and practical implementations of VQAs on real quantum hardware.

Chapter 7

Results

In this chapter, the behaviour of different variational quantum algorithms across various differential equations is investigated. We examine the Chebyshev-encoded VQA introduced by Kyriienko et al. [4], our extended version of this method adapted to second-order differential equations, the Lagrange-encoded approach developed in this work, and finally, a discretized VQA proposed by Sato et al. [5], specifically designed to solve the Poisson equation with a particular source term. All simulations presented in this chapter are performed using the statevector representation, an aspect that is important to consider when interpreting the results.

7.1 First-order Linear Ordinary Differential Equation

7.1.1 Problem definition

Beginning with a simple toy problem: a first-order linear ordinary differential equation given by:

$$\frac{df}{dx} + \lambda f = \cos(x), \tag{7.1}$$

with $\lambda = 1$, initial condition $f(x_0) = 1$ at $x_0 = 0$, and domain $x \in [0, 10]$.

The objective is to apply two polynomial-based variational quantum algorithms: the Chebyshev- and Lagrange-based VQAs, to this toy problem. This approach enables an examination of their behaviour,

an assessment of their trainability, and the identification of the hyperparameters to which they are most sensitive. To do so, we consider both the complete and homogeneous solutions of the differential equation, as they exhibit different characteristics, with one displaying oscillations while the other does not. Although the original interval is $[0, 10]$, the problem is rescaled to the range $[0, 0.9]$ to ensure compatibility with the quantum encoding domains of definition.

7.1.2 Analytical solution

The homogeneous solution of this differential equation f_h is defined as follows:

$$f_h(x) = Ce^{-\lambda x}, \tag{7.2}$$

where C is a constant. With the initial condition $f(0) = 1$ and $\lambda = 1$, we have $C = 1$.

For the particular solution, consider:

$$f_p(x) = A\cos(x) + B\sin(x), \tag{7.3}$$

with A and B constants determined by the differential equation. For the chosen parameters, this leads the complete solution:

$$f(x) = f_h(x) + f_p(x) = \frac{1}{2}e^{-x} + \frac{1}{2}(\cos(x) + \sin(x)). \tag{7.4}$$

7.1.3 Quantum resolution of the homogeneous ODE

In this section, we will investigate how the two polynomial-based VQAs presented before can be applied to solve the homogeneous version of the ordinary differential equation mentioned earlier:

$$\frac{df}{dx} + f = 0, \quad f(0) = 1 \Rightarrow f_{sol}(x) = e^{-x} \tag{7.5}$$

Lagrange-encoded VQA

With the primary objective of obtaining an accurate approximation of the solution, we first examine, for a system with six encoding points, which configuration of these points provides the best performance and how the number of training points affects the efficiency of the VQA. We consider the three sets of encoding points introduced in Chapter 5: equidistant points, or Chebyshev nodes (shifted, referred to as type 1, or truncated, referred to as type 2). Each configuration is trained on sets of 10, 20, and 30 equidistant training points. For robustness, the algorithm is executed five times per configuration to identify consistent trends and assess the influence of each hyperparameter.

The optimization is performed using the Adam optimizer [150] with its default learning rate $\alpha = 0.01$, starting from a random vector of variational parameters θ. The cost function combines contributions from the differential equation and the initial condition, with respective weights $\eta_{DE} = 1$ and $\eta_{Cs} = 10^3$. No regularization points are included in this configuration. The relatively large weight assigned to the initial condition ensures that this constraint is satisfied prior to reducing the residual of the differential equation. Two stopping criteria are introduced: 1) a target precision of $1 \cdot 10^{-4}$ on the loss (which was not reached), and 2) a convergence condition based on parameter updates, where optimization is terminated once the absolute variation of all parameters falls below $2 \cdot 10^{-3}$.

Performance is evaluated by computing the average loss over 50 equidistant points within the domain. The corresponding results are presented in Figure 7.1.

From Figure 7.1, the best results occur when training on 20 points. Although the overall trend suggests that the loss generally decreases with more training points, this does not guarantee the most accurate solution in every case. Nevertheless, accuracy tends to improve on average as the training set grows.

When comparing encoding strategies, for 20 and 30 training points, Chebyshev nodes of type 1 consistently achieve the best performance, followed by type 2, with equidistant points yielding the worst results. For 10 training points, however, no clear advantage emerges between the different strategies, and type 1 Chebyshev nodes do not outperform type 2.

As the same VQA circuit architecture is employed across all configurations, the number of iterations,

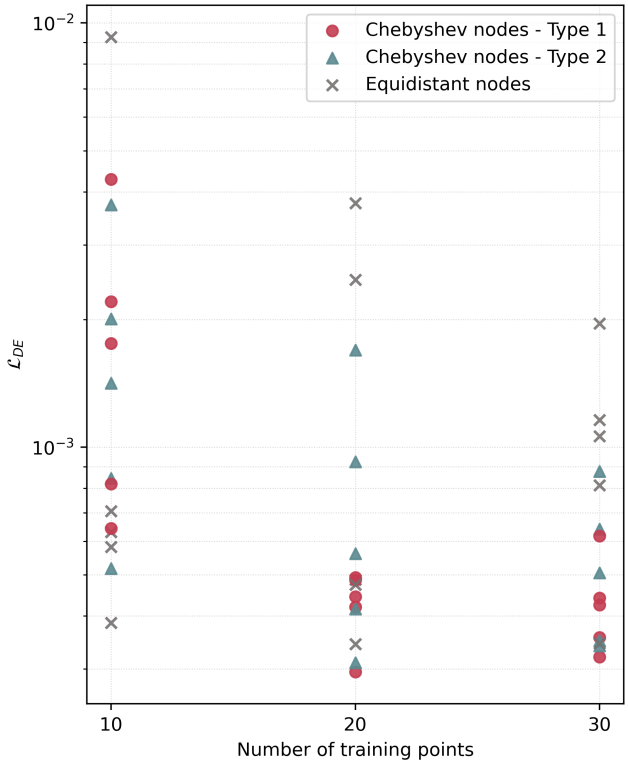


Figure 7.1: Overall loss from the differential equation term $\mathcal{L}_{DE}$ for each run of the algorithm (y-axis), shown on a logarithmic scale, as a function of the number of training points (x-axis) and the type of encoding points. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits.

steps, and gates scale proportionally according to:

$$N_{\text{gates}} = (5n + 2(n-1))N_{\text{circuits}}, \tag{7.6}$$

$$N_{\text{circuits}} = M(1 + 3n + 2n^2)N_{\text{iterations}}, \tag{7.7}$$

where n is the number of encoding points and M the number of training points. Figure 7.2 compares the average loss and the average number of gates over five runs.

As expected, the number of circuits increases with the size of the training set, which is intuitive. However, for type 1 Chebyshev nodes, the loss decreases sharply when moving from 10 to 20 training points, but additional points do not provide further improvement. In this case, complexity increases without meaningful gains. For equidistant points and type 2 Chebyshev nodes, more training points yield slight improvements, but these come at a significant computational cost, making the process less efficient overall.

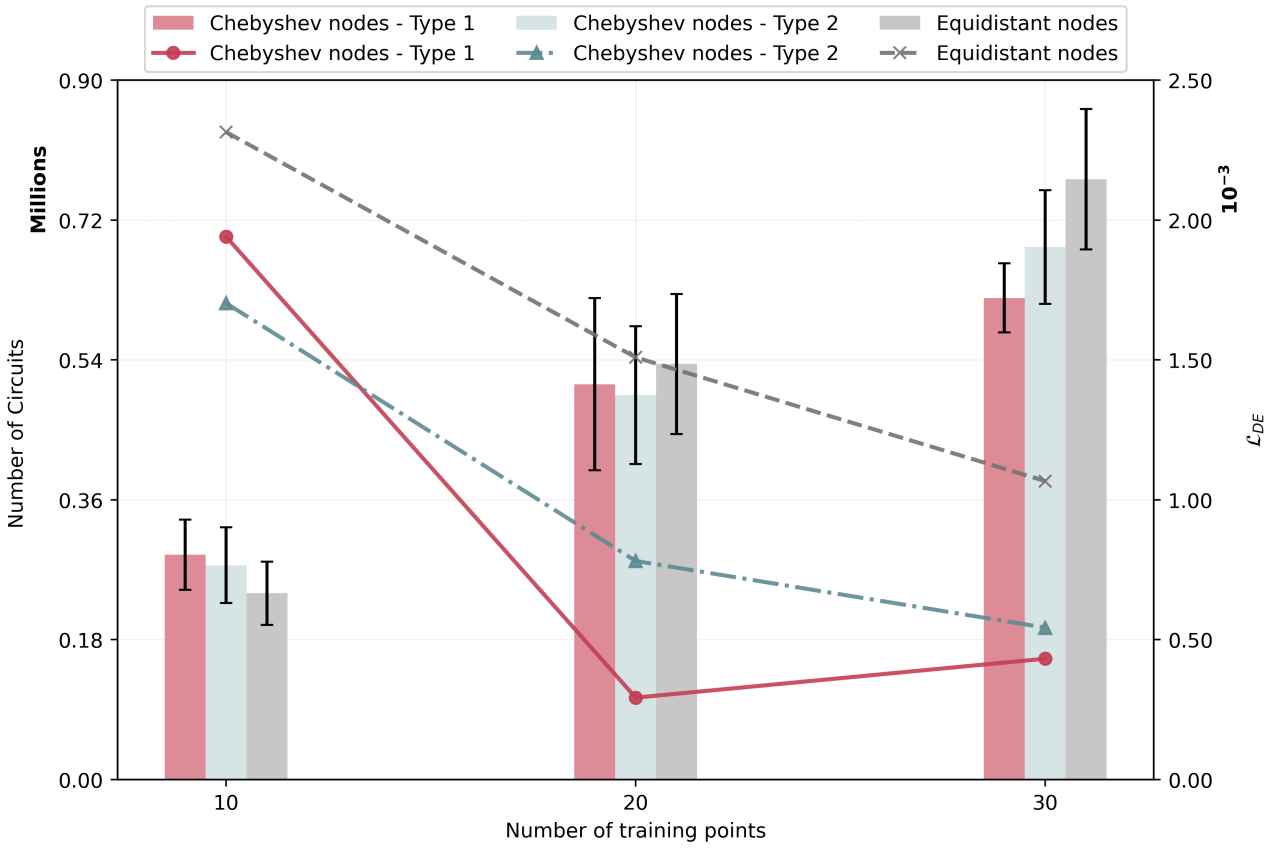


Figure 7.2: Average number of circuits (left axis) and average overall loss over five runs (right axis) for each configuration. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits.

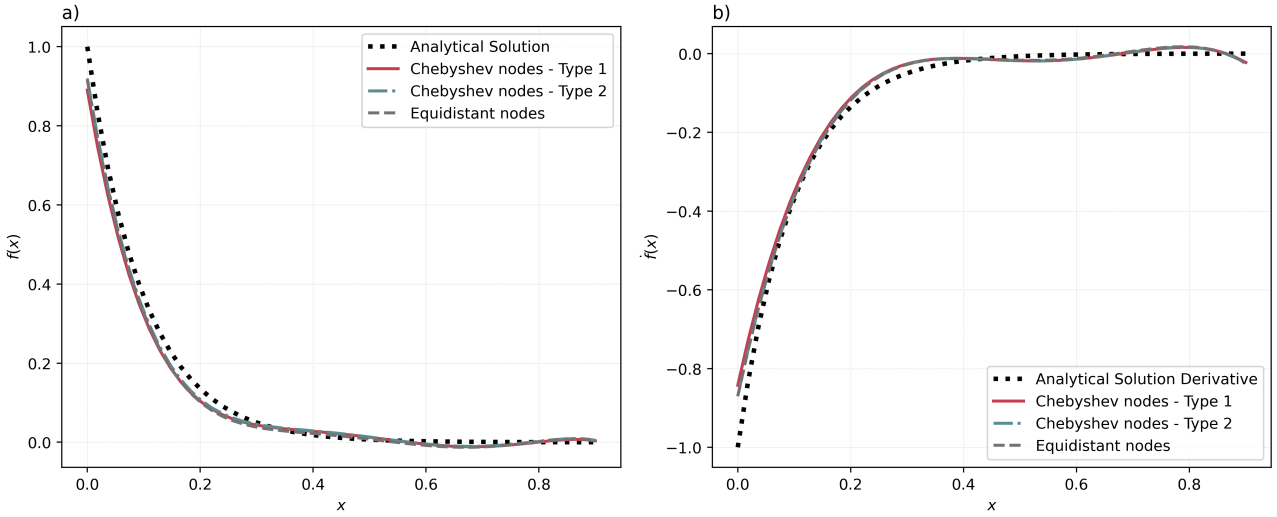


Figure 7.3: a) Circuit function and b) first-order derivative for the best result obtained in each configuration. Results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6$ encoding points and $N = n + 1$ qubits.

Finally, when comparing the best approximations from each encoding set, the losses remain close: $2.9 \cdot 10^{-4}$ for type 1 Chebyshev nodes, $3.1 \cdot 10^{-4}$ for type 2, and $3.4 \cdot 10^{-4}$ for equidistant points. These differences are negligible in practice, leading to approximations that can be considered equivalent. This similarity likely arises because, for small values of n, the three sets exhibit only minor differences.

Two-step optimization As observed in the previous section, the approximation obtained from a single run is close to the true solution but does not perfectly overlap with it. Since the stopping criterion has consistently been the convergence of the Adam optimizer, we first investigated the effect of modifying the optimization procedure. We performed an initial optimization over 10 training points with a learning rate of $\alpha = 0.02$, followed by a second round with $\alpha = 0.01$, 20 training points, and a reset of the first and second moment estimates. This approach accelerated convergence by reducing the number of steps required for optimization. However, the total number of iterations was not reset, which also influences convergence behaviour. We conducted five runs using $n = 6$ encoding points over the Chebyshev type 1 set.

Increasing the number of encoding points An alternative strategy to further improve accuracy was to increase the number of encoding points to $n = 7$, resulting in a system of eight qubits.

Comparison between the three approaches The best results for a single training with six encoding points, a two-step training with six encoding points using Chebyshev type 1 nodes, and a single-step training with seven encoding points using Chebyshev type 1 nodes are shown in the following figures. The results show that applying the optimizer in two steps slightly reduced the loss and, interestingly, also decreased the circuit count on average across the five runs. However, increasing the number of encoding points from $n = 6$ to $n = 7$, and consequently the number of qubits, improved the overall loss $\mathcal{L}_{DE}$ by an average factor of 6.5 while maintaining a circuit count within the same range than the two steps approach over six points. This highlights the importance of both the optimization strategy and the choice of the number of encoding points.

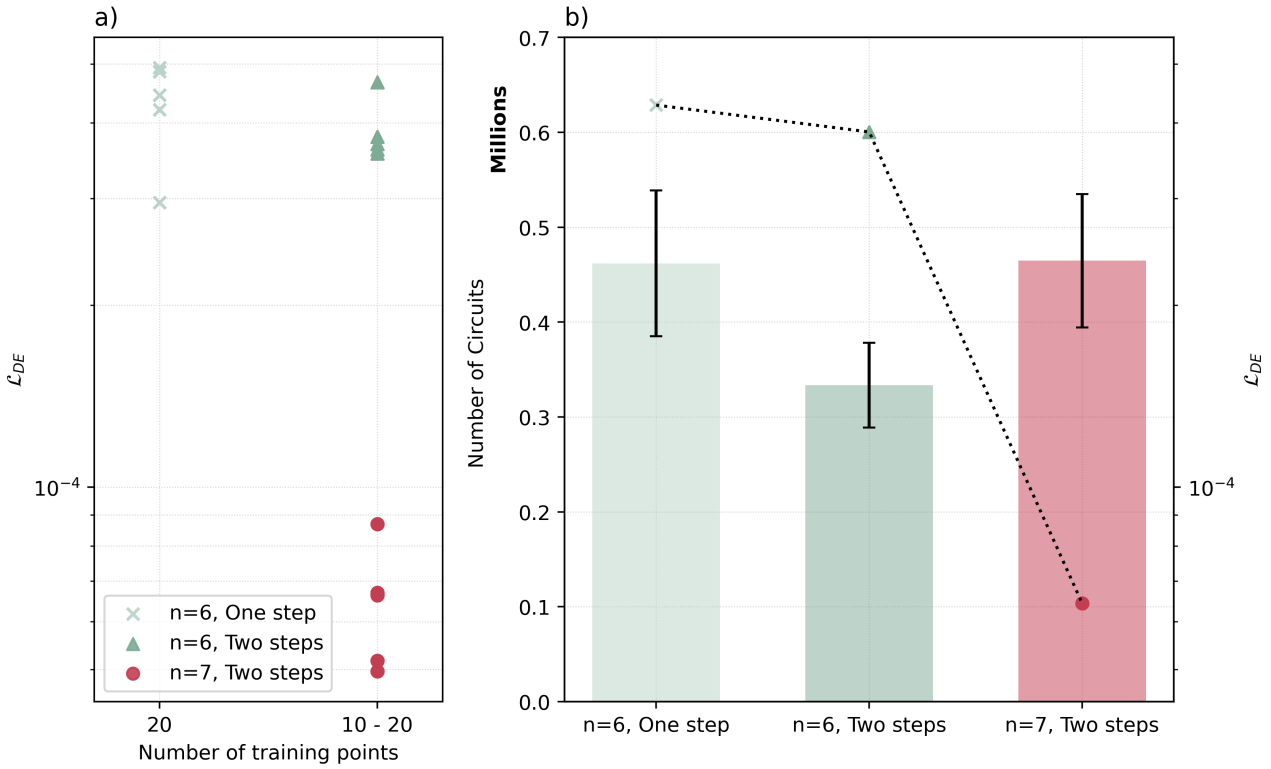


Figure 7.4: Comparison between the three approaches. a) Overall loss from the differential equation term $\mathcal{L}_{DE}$ for each run of the algorithm (y-axis), plotted on a logarithmic scale. b) Average overall loss over five runs (right axis) and average number of circuits (left axis) for each configuration. The results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6, 7$ encoding points and applying the optimizer in one step or two steps.

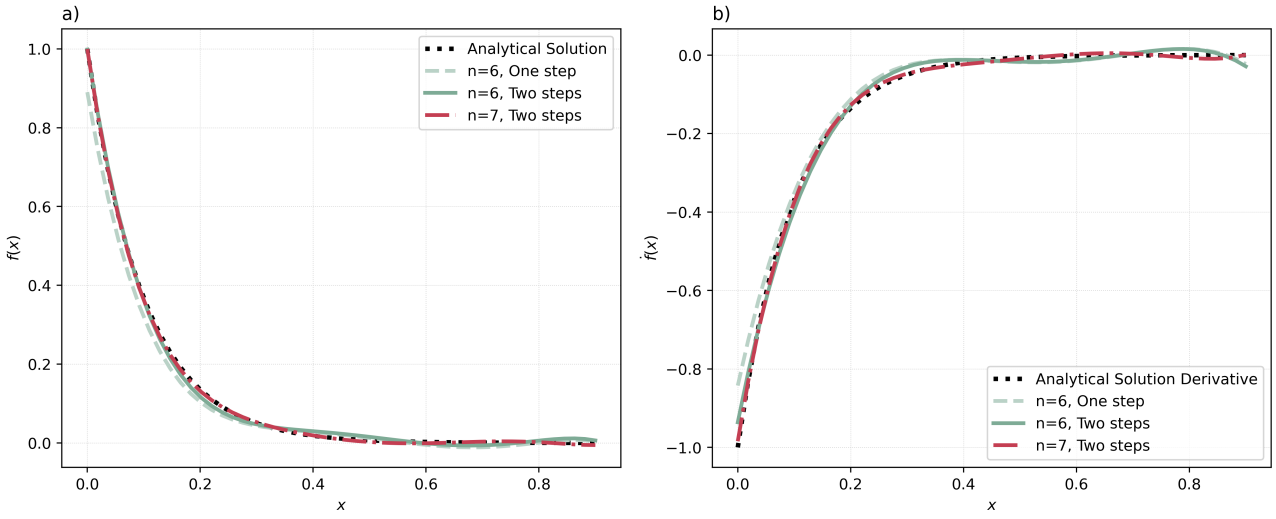


Figure 7.5: a) Circuit function, b) first-order derivative for the best result obtained in each configuration. The results correspond to the application of the Lagrange-encoded VQA to the homogeneous first-order linear ODE, using $n = 6, 7$ encoding points and applying the optimizer in one step or two steps.

Figure 7.4.a) shows that the two-step approach with six encoding points produces a much better approximation of the function, particularly near the boundary condition. Some oscillations are noticeable in the region where the function approaches its asymptote as $f \to 0$ when $x \to +\infty$ (here, up to $x = 0.9$), and similar behaviour appears for the first-order derivative, as shown in Figure 7.4.b). In contrast, the case with seven encoding points exhibits an almost perfect overlap with the true solution, accurately satisfying both the initial condition and the asymptotic behaviour for both the function and its derivative.

To conclude this analysis, by examining the loss function $\mathcal{L}_{DE}(x)$ across the interval for the three approaches shown in Figure 7.5, it becomes evident that the method using seven encoding points achieves a significantly smaller loss overall compared to both configurations with only six points. While the two-step optimization with six encoding points shows a slightly lower loss in certain regions compared to the single-step version, this difference remains minor when contrasted with the improvement obtained by increasing the number of encoding points to seven.

Chebyshev-encoded VQA

We now consider the same differential equation, but this time solved using the Chebyshev-encoded VQA. The algorithm follows the two-step optimization process introduced earlier. Starting from a random vector of variational parameters θ, the parameters are first optimized using the Adam optimizer over 10 equidistant training points, with a learning rate $\alpha = 0.02$. After convergence of this first phase, a second optimization step is performed with $\alpha = 0.01$, now using 20 equidistant training points. The initial condition is incorporated through a floating boundary condition handling rather than as a term in the cost function. To provide a robust assessment of the algorithm's behaviour, the VQA is executed five times, each starting from a different random initialization of the variational parameters.

The encoding is implemented as a tower Chebyshev encoding over six qubits. In this scheme, the degree of the Chebyshev polynomials encoded into the $\hat{R}_Y$ rotations increases with the qubit index. The variational ansatz is the HEA, built from a repeating sequence of three gates: $\hat{R}_Z$, $\hat{R}_X$, and $\hat{R}_Z$. We focus on a six-qubit system and compare circuits with depths of 3, 5, and 7. This choice aligns with the structure proposed by Kyriienko et al. [4], where the authors introduced this VQA for

first-order differential equations using six qubits and an ansatz of depth 5, consistent with the original formulation of the HEA with around hundred variational parameters.

Since the simulations rely on statevector measurements and involve 54, 90, or 126 parameters (for depths 3, 5, and 7, respectively), derivatives are computed using the forward difference approximation (FDA) introduced in the previous chapter. While this approach reduces simulation complexity, it is not suitable for real quantum hardware. For this reason, when estimating the number of required circuits, we assume the use of the parameter-shift rule, which is the standard choice for actual devices.

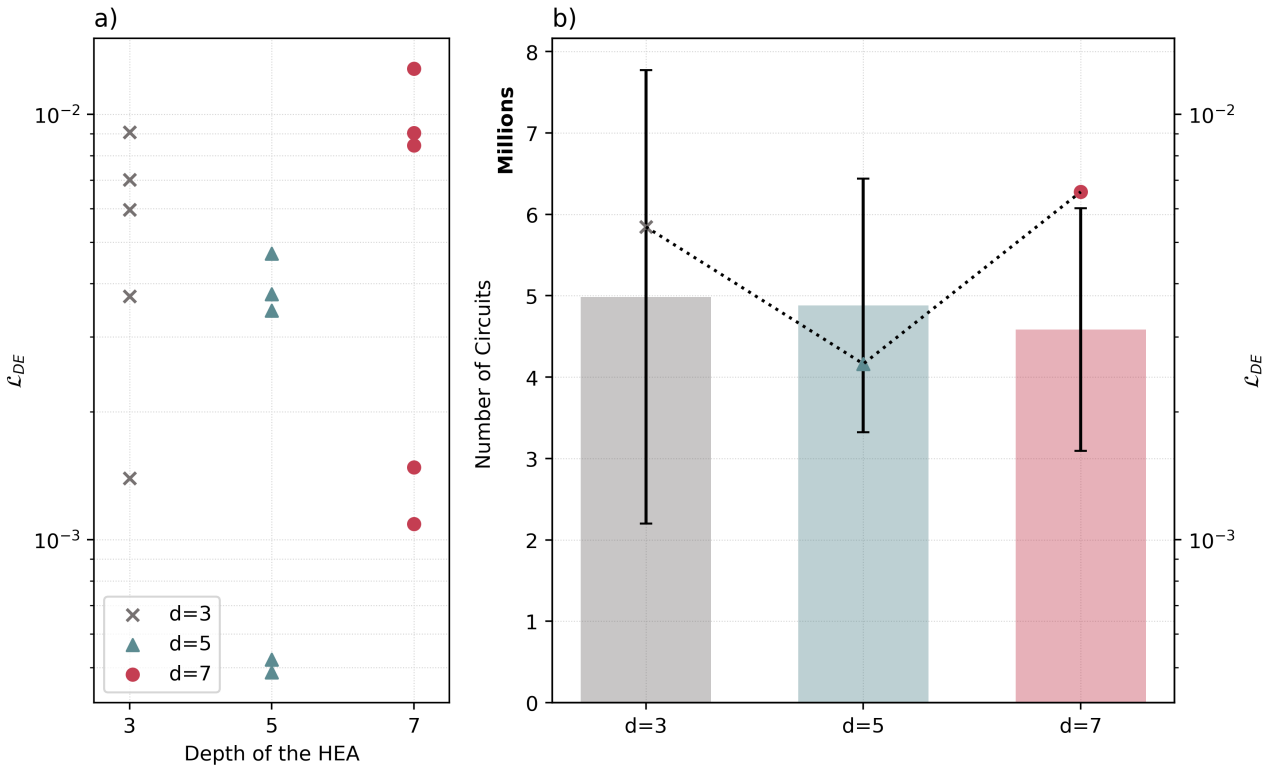


Figure 7.6: a) Overall loss $\mathcal{L}_{DE}$ for each run of the algorithm across different ansatz depths. b) Average overall loss and corresponding circuit count as a function of depth. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits.

While circuit counts provide useful insight for comparing architectures with different numbers of parameters, the corresponding number of gates must also be considered, given by:

$$N_{\text{gates}} = (n + d(4n - 1))\ N_{\text{circuits}}, \tag{7.8}$$

$$N_{\text{circuits}} = M(1 + 2n(d + 1) + 2dn^2)\ N_{\text{iterations}}, \tag{7.9}$$

where n is the number of qubits, M the number of training points, and d the circuit depth.

Figure 7.6 compares the loss and circuit requirements for different depths. Interestingly, the circuit count remains in a similar range across all depths, with only a slight decrease for deeper circuits. However, this trend provides limited insight, as the loss does not improve significantly with increasing depth. On average, across five runs, the best results are observed at a depth of 5. As shown in Figure 7.6.a), a depth of 5 yields the best outcome in only two out of five cases, while in the remaining cases its performance is comparable to that of a depth of 7.

Examining the intermediate and final results after the two optimization steps (Figure 7.7), we see that the first step provides a reasonable initial approximation, especially near the left boundary where the initial condition is enforced. Nevertheless, this first step alone is insufficient for high accuracy, highlighting the importance of the second optimization phase.

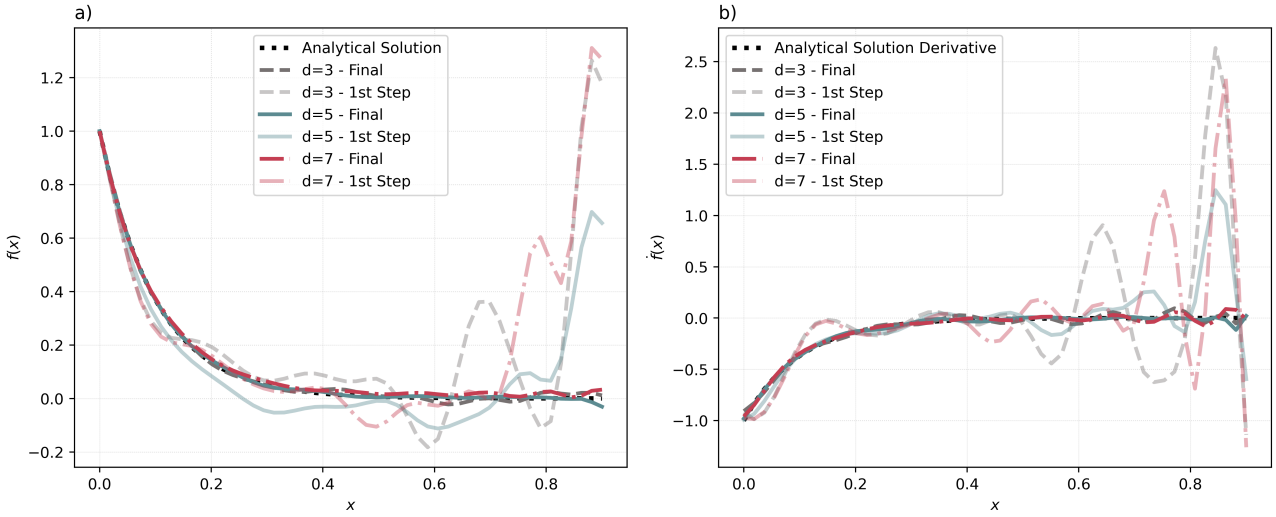


Figure 7.7: Solutions from the best run for each depth studied, showing both the intermediate result after the first optimization step (10 training points) and the final result after the second step (20 training points). a) Function f, b) First-order derivative $\hat{f}$. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits.

When focusing on the best final result for each depth (Figure 7.8), the depth-5 circuit performs best overall. However, oscillations appear near the right boundary, where the solution approaches zero asymptotically. These oscillations are particularly pronounced for depths 3 and 7, suggesting that both under-parameterization and over-parameterization can negatively impact stability. Finally, Figure 7.9 confirms that the depth-5 configuration achieves the best overall accuracy across the domain,

particularly near the boundaries.

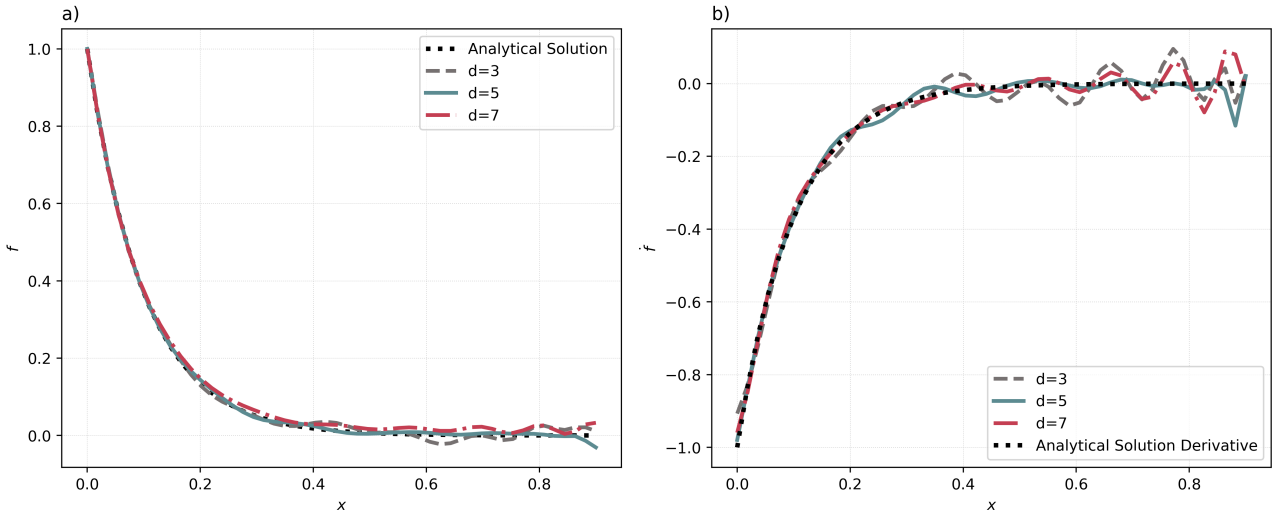


Figure 7.8: Final solutions from the best run for each depth studied, obtained after the second optimization step (20 training points). a) Function f, b) First-order derivative $\dot{f}$. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits.

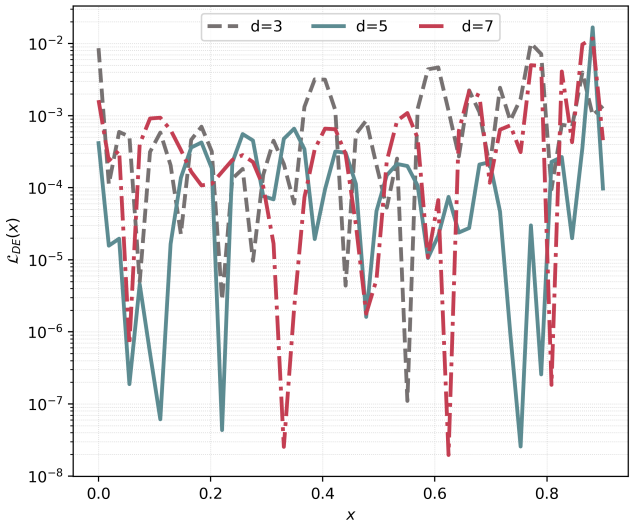


Figure 7.9: Loss evaluated across the interval for the best run corresponding to each depth, shown on a logarithmic scale. Results correspond to the homogeneous first-order linear ODE solved using a Chebyshev-encoded VQA with six qubits.

We also explored alternative configurations, including different numbers of qubits and a simplified hardware-efficient ansatz consisting of a single $\hat{R}_X$ gate per layer. These alternatives yielded significantly poorer results, even worse than the intermediate solutions after the first optimization step (10 training points), as illustrated in Figure 7.7.

Chebyshev- vs. Lagrange-encoded VQAs

To conclude the study of the homogeneous first-order linear differential equation, we compare the best results obtained from the two polynomial-encoded VQAs introduced previously.

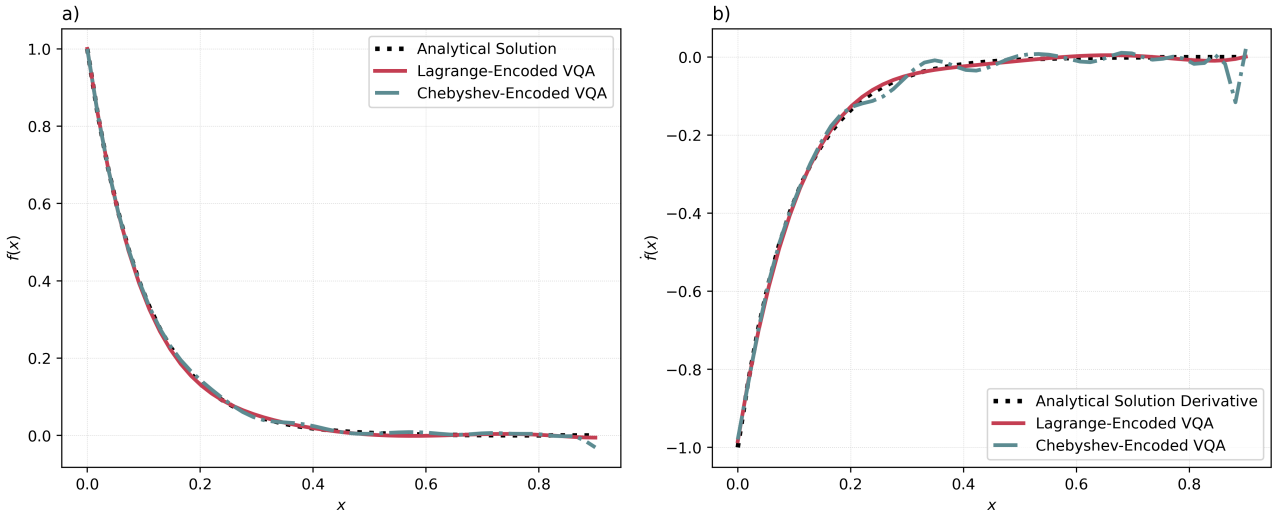


Figure 7.10: Comparison of the approximated solutions of the homogeneous first-order linear ODE obtained from Chebyshev- and Lagrange-encoded VQAs. a) Function f, b) First-order derivative $\dot{f}$. The Chebyshev encoding uses 6 qubits, a tower structure, and a hardware-efficient ansatz of depth 5 with a three-gate sequence, resulting in 90 variational parameters. The Lagrange encoding uses 7 interpolation points, corresponding to 8 qubits and 7 variational parameters.

From Figure 7.10, the Lagrange-encoded VQA outperforms its Chebyshev counterpart, particularly near the right boundary, where the latter exhibits oscillations as the function approaches zero.

The difference is even more striking when considering circuit counts (Figure 7.11). The Chebyshev-encoded VQA requires almost ten times more circuits on average than the Lagrange-based approach. This discrepancy stems mainly from the number of parameters (90 for Chebyshev versus 7 for Lagrange) and the gradient computation method: the Hadamard test used for Lagrange encoding requires roughly half the number of circuits compared to the parameter-shift rule employed for Chebyshev.

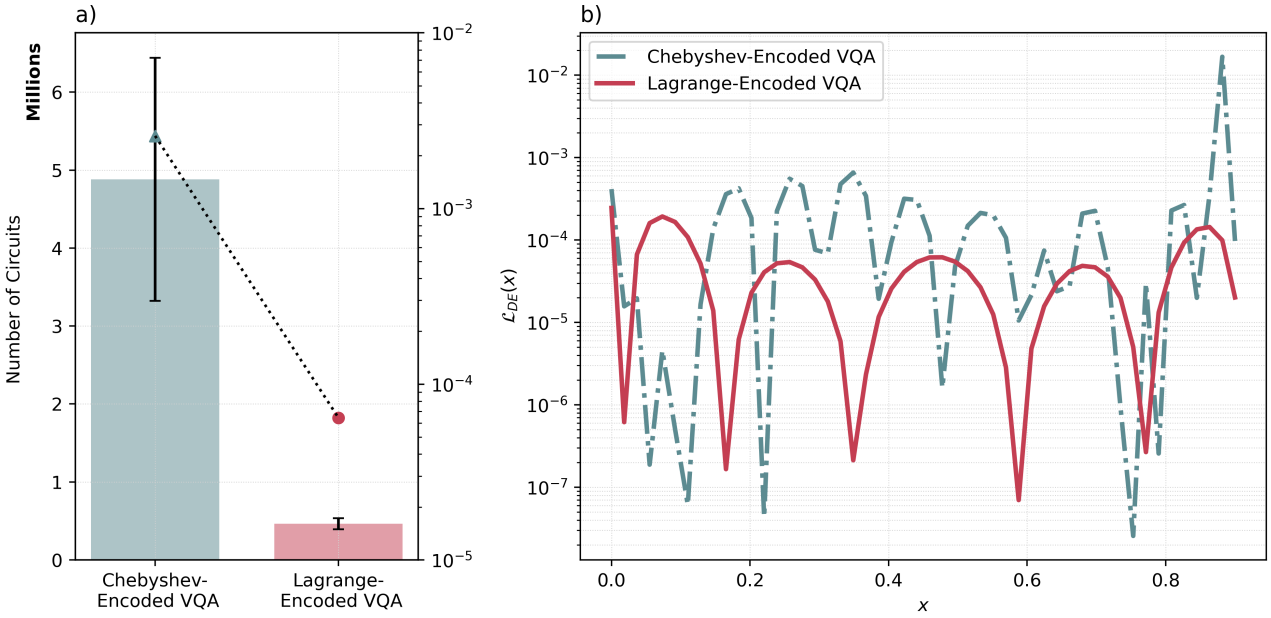


Figure 7.11: Comparison of Chebyshev-encoded and Lagrange-encoded VQAs for the homogeneous first-order linear ODE. a) Average circuit counts (left axis) and average overall loss (right axis), b) Loss function across the interval for the best-performing instance of each VQA. The Chebyshev encoding uses 6 qubits, tower encoding, and a hardware-efficient ansatz of depth 5 with a three-gate sequence, for 90 variational parameters. The Lagrange encoding uses 7 interpolation points, corresponding to 8 qubits and 7 variational parameters.

7.1.4 Quantum resolution of the complete ODE

In this section, we consider the complete ODE, including the $\cos(x)$ term. The interest here is to study the algorithm's behavior with an oscillatory function, which is naturally better represented by Chebyshev polynomials than by Lagrange interpolation, and consequently to assess the performance of both encodings across a broader range of functions.

Lagrange-encoded VQA

Solving the complete ODE with the Lagrange-encoded VQA, we first considered a parameterized circuit built over 6 encoding points, following the two-step optimization procedure described earlier. The Adam optimizer is applied first with a learning rate $\alpha = 0.02$, and then with $\alpha = 0.01$. The initial condition is included in the loss function with a weight $\eta_{Cs} = 1 \cdot 10^3$, while the differential equation contribution has a weight $\eta_{DE} = 1$.

As the results for $n = 6$ encoding points were not fully satisfactory, we extended the study to 7 and 8 encoding points. Figure 7.12.a shows that increasing the number of encoding points significantly reduces the overall loss. Additionally, the results over the five runs tend to cluster according to the number of encoding points, indicating good consistency. As expected, increasing the number of encoding points also increases both the circuit count and its variability, as illustrated in Figure 7.12.b.

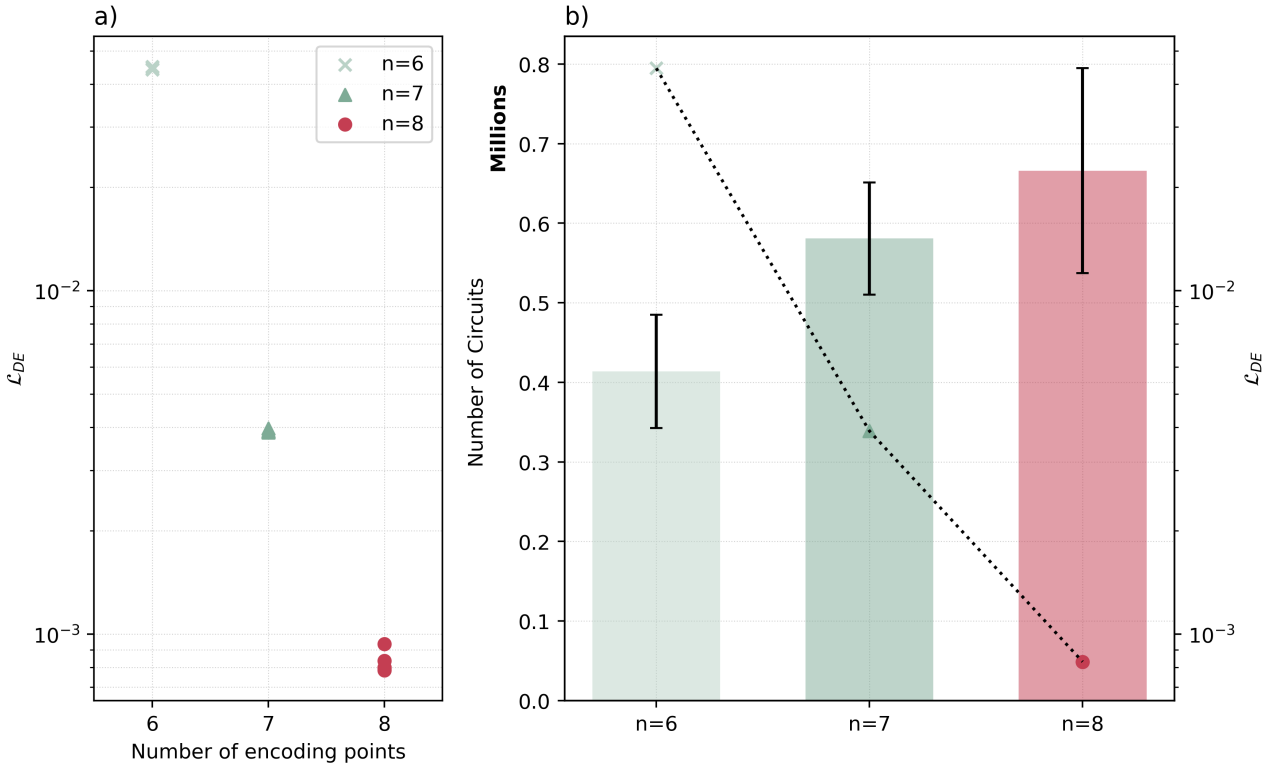


Figure 7.12: Comparison of Lagrange-encoded VQA performance for 6, 7, and 8 encoding points. a) Overall loss over five runs per configuration. b) Average circuit count (left axis) and corresponding average overall loss (right axis).

Figure 7.13 shows the best result obtained for each configuration, compared to the analytical solution. With 6 encoding points, the approximation captures the general shape of the solution but deviates noticeably from the exact function. The 7-point encoding provides a reasonable approximation, while the 8-point encoding achieves an almost perfect overlap with the analytical solution.

This difference is further confirmed by the loss function evaluated over the interval for the best outcome of each configuration, as shown in Figure 7.14.

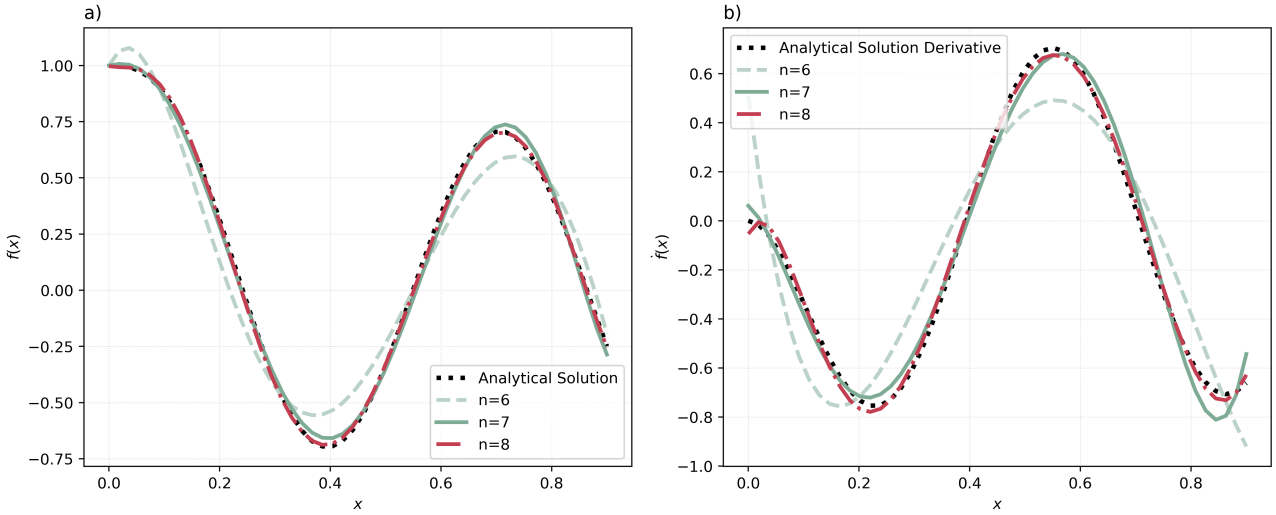


Figure 7.13: Best Lagrange-encoded VQA solutions for 6, 7, and 8 encoding points. a) Function $f(x)$, b) First-order derivative $\dot{f}(x)$.

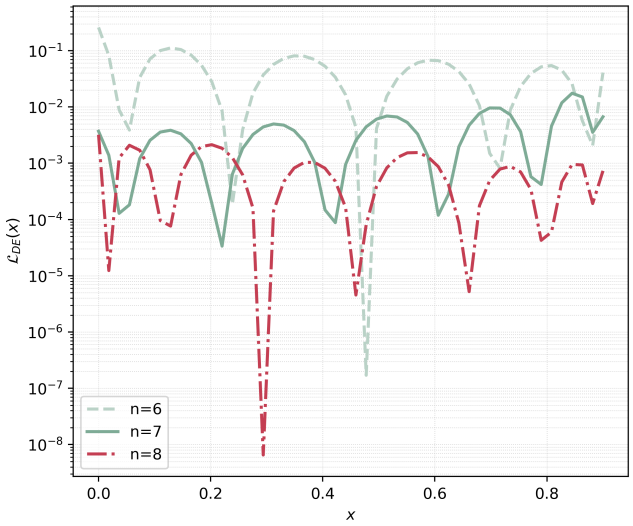


Figure 7.14: Loss function $\mathcal{L}_{DE}$ over the interval for the best result obtained with 6, 7, and 8 encoding points in the Lagrange-encoded VQA, compared with the analytical solution.

Chebyshev-encoded VQA

For the Chebyshev-encoded VQA, as observed for the homogeneous differential equation, using a HEA with only a single gate per layer fails to produce satisfactory results, regardless of circuit size or depth. In contrast, the configuration previously employed, 6 qubits, a HEA consisting of a three-gate sequence $\hat{R}_Z - \hat{R}_X - \hat{R}_Z$, and depth 5, performs very well.

Interestingly, among the five runs, only one achieved an almost perfect approximation, without notice-

able disturbances at the end of the interval, similar to what was observed in the homogeneous case. This behavior is illustrated in Figure 7.15.

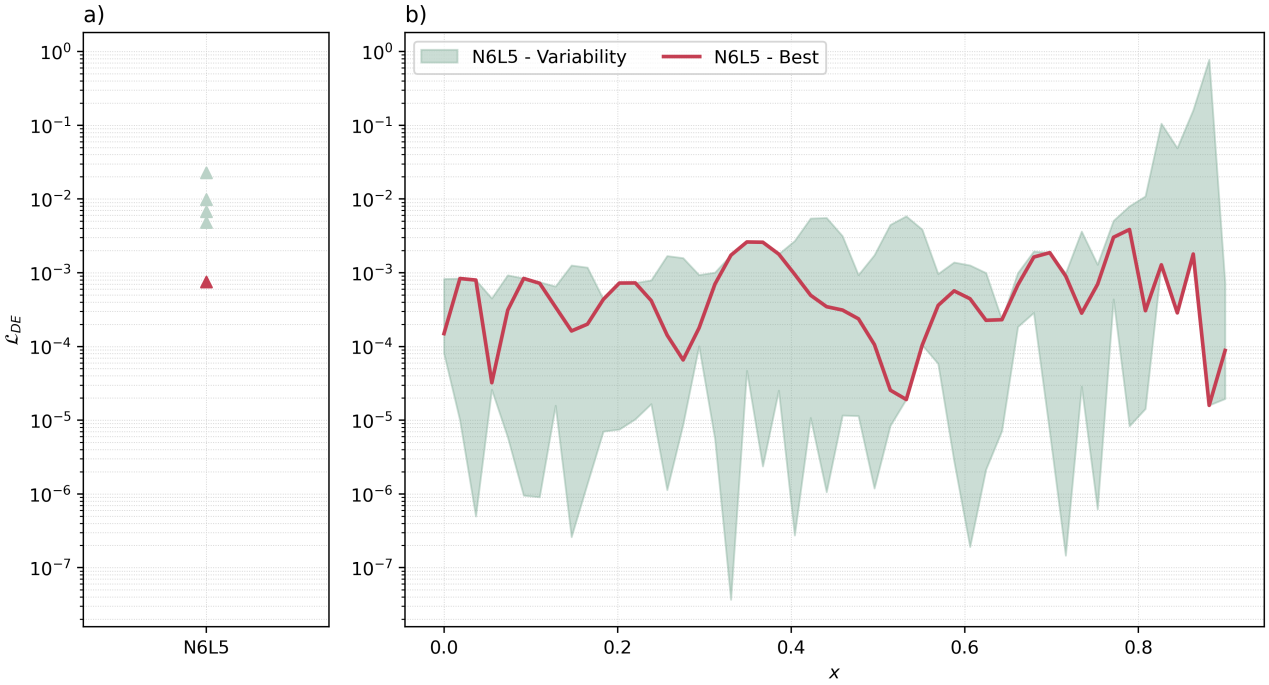


Figure 7.15: Chebyshev-encoded VQA applied to the complete ODE: comparison of five runs. a) Overall loss for each run, plotted on a logarithmic scale. b) Loss function over the interval, showing the best result in red and the variability of other runs in the green area.

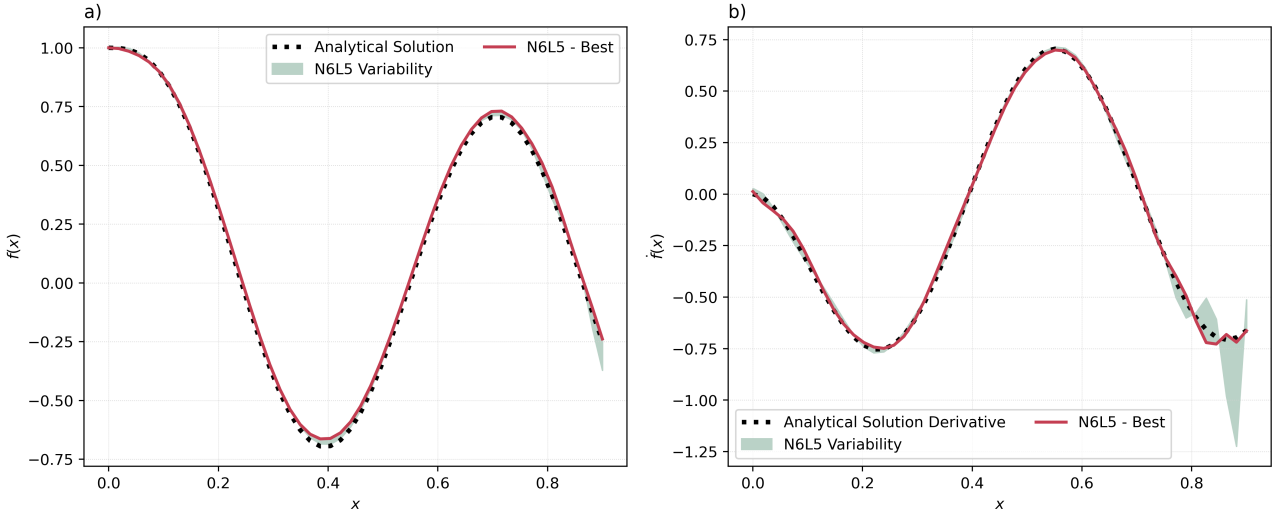


Figure 7.16: Chebyshev-encoded VQA applied to the complete ODE: comparison of five runs with the analytical solution. a) Function $f(x)$. b) First-order derivative $\dot{f}(x)$. The red curve corresponds to the best run, while the green area shows the variability of the remaining runs.

Examining the function and first-order derivative for the best outcome (Figure 7.16), we observe an

almost perfect overlap for the function and minor disturbances at the end of the interval for the derivative. The variability area, representing the range of other outcomes, shows oscillations near the right boundary. This explains the pronounced region above the red line in the loss function plotted in Figure 7.15.b.

Chebyshev- vs. Lagrange-encoded VQAs

Finally, we compare the best outcomes obtained with the two polynomial-encoded VQAs for the complete ODE. The results are illustrated in Figure 7.17.

As observed in Figures 7.17 and 7.18, the results of the two approaches have very similar quality. When examining the average loss, the Chebyshev-encoded VQA performs slightly worse. The main difference lies in the number of circuits required.

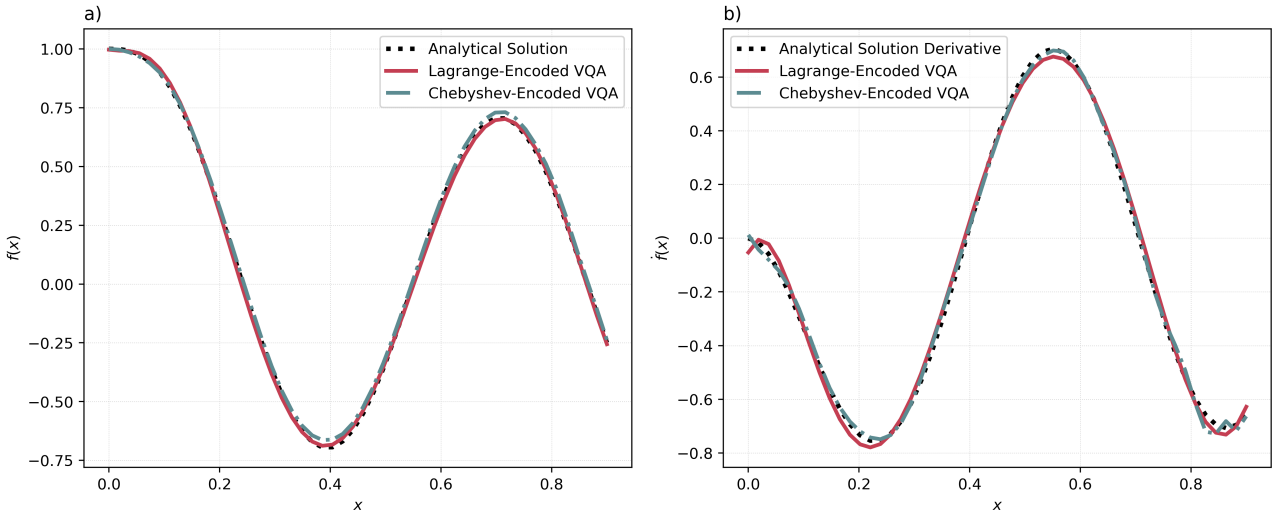


Figure 7.17: Comparison of the best outcomes of the Chebyshev and Lagrange encoded VQAs applied to the complete ODE, plotted against the analytical solution. a) Function $f(x)$. b) First-order derivative $\hat{f}(x)$. The Chebyshev encoding uses 6 qubits, a hardware-efficient ansatz with three gates and depth 5. The Lagrange encoding uses 8 encoding points, corresponding to 9 qubits. Both are trained using a two-step optimization procedure.

It is also important to highlight the high variability of the Chebyshev-encoded results, with a coefficient of variation of approximately 130 percent. This reflects an extreme sensitivity to the initial random choice of variational parameters: while four runs converged in fewer than 200 iterations, one run required over 1000 iterations. Consequently, this outlier was excluded from the performance plot

for visibility, but it underscores the pronounced sensitivity of the Chebyshev encoding to the initial parameter vector. Since the same hyperparameters and classical optimizer were used for both VQAs, this variability appears to stem primarily from the quantum circuit itself rather than the classical optimizer.

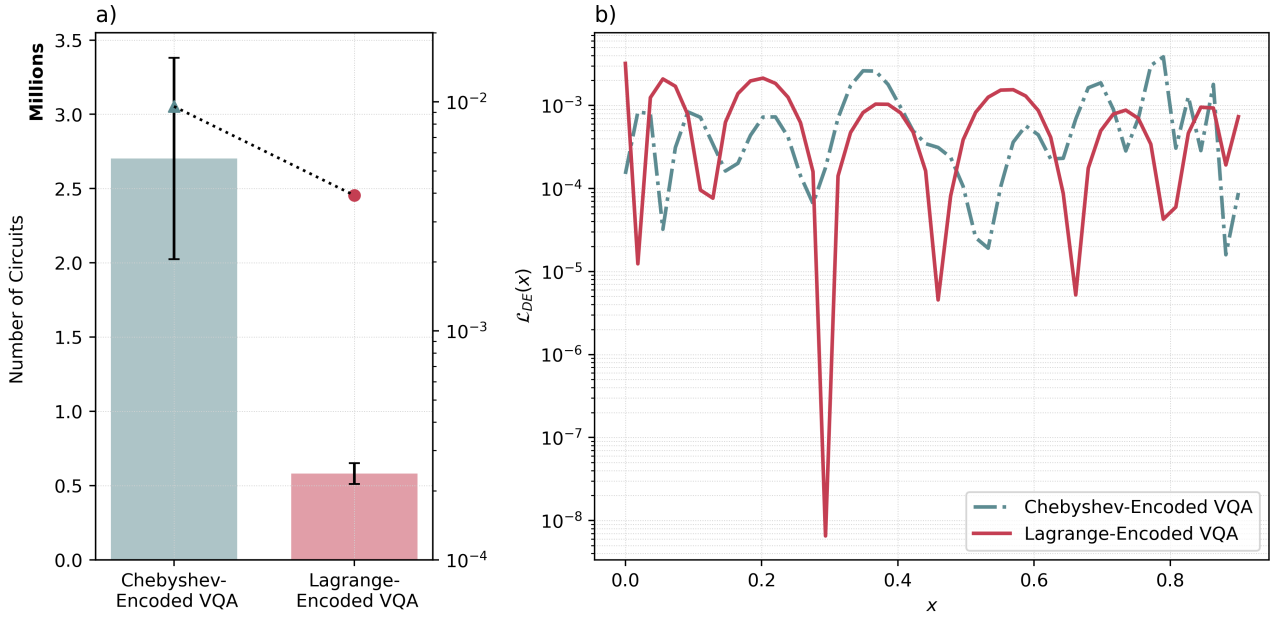


Figure 7.18: Comparison of the performance best outcomes of the Chebyshev- and Lagrange-encoded VQAs applied to the complete ODE solution. a) Average circuit count (left axis) and average overall loss (right axis). b) average loss over the interval, plotted on logarithmic scale. The Chebyshev encoding uses 6 qubits, a hardware-efficient ansatz with three gates and depth 5. The Lagrange encoding uses 8 encoding points, corresponding to 9 qubits. Both are trained using a two-step optimization procedure. The Chebyshev-encoded VQA performance plotted exclude the outlier

7.1.5 Summary

To summarize this example, we observed that, regarding the Lagrange-encoded VQA, the set of encoding points of the first type (shifted) generally led to better results than other sets, for the same number of encoding points. Additionally, increasing the number of points allowed us to reach better accuracy. However, we did not explore the effect of using more qubits than strictly necessary. Since the method relies on Lagrange interpolation, the quality of the approximation does not necessarily increase with the polynomial degree, but rather on the choice of encoding points. The number of encoding points, which determines the polynomial degree, must be large enough to capture the essential features of the function, but not too large to avoid introducing significant oscillations. For both the

homogeneous and complete equations, the Lagrange encoding provided results with high accuracy.

The Chebyshev-encoded VQA performed very well for the complete equation, particularly because the solution exhibited a wave-like behaviour. However, when interpolating a nearly constant or linear segment, the accuracy decreased and oscillations appeared. Consequently, this method may not be ideal for cases involving damping or asymptotic behaviour near the interval boundaries.

Regarding the circuit count, we observed that the Chebyshev-encoded VQA required significantly more circuits than the Lagrange-encoded one. This can be explained primarily by the number of parameters, which was more than ten times higher. The best-performing Chebyshev-encoded VQA used 6 qubits and a Hardware Efficient Ansatz with 5 layers of a 3-gates sequence, resulting in 90 variational parameters. In contrast, the Lagrange-based approach achieved good results with only 7 or 8 variational parameters. Consequently, computing the gradient of the objective function $\nabla_\theta f$ at each iteration required approximately 180 circuits for the Chebyshev encoding, compared to only 14 or 16 for the Lagrange encoding. Moreover, evaluating the gradient of the first-order derivative $\nabla_\theta \frac{df}{dx}$ introduces an additional overhead, as it necessitates even more circuits when using techniques such as the parameter-shift rule or the Hadamard test differentiation, thereby amplifying the computational cost gap between the two approaches.

As noted earlier, the Chebyshev-based algorithm was also tested with a single-gate sequence for various depths and qubit numbers, but these configurations failed to converge to the correct solution. We can therefore conclude that using a three-gate sequence appears more expressive, allowing to achieve accurate results with this VQA, though it comes at a considerable cost.

Finally, it is important to emphasize that these results and conclusions are valid for a noise-free simulated environment, for this specific differential equation, and using the classical optimizer Adam [150]. Future work could include an investigation of different behaviours when employing alternative optimizers or under noisy conditions.

7.2 Damped Mass-Spring System

This section is based on the results presented in our publication [1], providing additional details.

7.2.1 Problem definition

A Damped Mass-Spring System (DMSS) is a physical model used to describe the behaviour of a mass attached to a spring when it is subject to a damping force. This system is widely studied in mechanics and physics to understand oscillatory motion and damping effects. In this work, we focus on an initial value problem (IVP) of a DMSS solved over a time interval of $I = [0, 10]$ seconds, using the Chebyshev-encoded VQA expanded to second-order differential equations and the proposed Lagrange-encoded algorithm. The DE can be expressed as:

$$m\frac{d^2f}{dt^2} + b\frac{df}{dt} + kf = 0, \tag{7.10}$$

with $m = 1kg$; $b = 1kg.s^{-1}$; $k = 1N.m^{-1}$ and the initial conditions of the DE are $u_0 = 1m$ and $\dot{u}_0 = 0m.s^{-1}$.

In this study, the two polynomial-encoded VQAs are trained using the two types of Chebyshev nodes introduced in Chapter 5 as training points. The objective is to introduce an additional dimension to the set of training hyperparameters, in order to assess whether selecting Chebyshev nodes as training points can enhance efficiency.

The two approaches are compared in terms of their accuracy, evaluated through the loss function over the interval, and their computational cost, measured by the number of gates. However, as highlighted in the previous example, these results are highly sensitive to the initial random values of the parameter vector θ. Consequently, the reported figures should be interpreted as indicative estimates of the order of magnitude for the quantum circuits and gate counts required to solve the equation.

7.2.2 Analytical solution

The dynamics of a damped mass-spring system described by the second-order differential equation, referenced as Equation (7.10). Its corresponding characteristic equation is formulated as:

$$m\lambda^2 + b\lambda + k = 0. \tag{7.11}$$

For the scenario explored in this paper, the discriminant $\Delta = b^2 - 4mk$ is negative, indicating an underdamped case. Consequently, the solution takes the form:

$$f(x) = e^{\alpha x}(C_1 \cos(\beta x) + C_2 \sin(\beta x)), \tag{7.12}$$

where α and β are the real and imaginary parts of the complex conjugate roots $\lambda_{1,2} = \alpha \pm i\beta$. The coefficients C_1 and C_2 are determined by the initial conditions $f(x_0) = u_0$ and $f'(x_0) = \dot{u}_0$.

7.2.3 Quantum resolution

Chebyshev-encoded VQA

The application of the Chebyshev-encoded VQA to a second-order differential equation, specifically the DMSS problem, has been investigated using a range of VQCs with 3 to 5 qubits, a single-gate sequence for the HEA, and circuit depths varying from 2 to 7. The experiments were performed across 7, 9, and 12 training points for both sets of Chebyshev nodes.

The choice of a single-gate sequence for the HEA is primarily motivated by considerations of computational complexity. As with the first-order ODE studied previously, this configuration serves as an initial baseline. Transitioning to a three-gate HEA would only be justified if the results obtained with the single-gate HEA were significantly unsatisfactory, since increasing the number of gates introduces substantial additional computational overhead.

It is important to note that, in this section, the term Chebyshev-encoded does not refer to the original approach proposed by Kyriienko *et al.*, but rather to our adapted version for second-order differential equations, implemented with a single-gate HEA sequence.

Each VQC is initialized with a random vector of variational parameters θ. To ensure a fair comparison of the influence of the size and type of training set, the same initial vector θ is used for all configurations corresponding to a given size and depth of the VQC.

As discussed previously, one of the main limitations this work aims to address arises from the quantum feature map employed in the Chebyshev-encoded VQA. This feature map is constructed from high-degree Chebyshev polynomials, which tend to exhibit oscillatory behaviour near the boundaries of the interval. Such oscillations can significantly affect the accuracy of the solution, particularly at the extremities of the domain. As a result, this implementation of the Chebyshev-encoded approach initially produced poor results. To address this limitation, the training was carried out over an extended interval of $[0, 12]$ seconds, with the aim of displacing the oscillatory behaviour beyond the target interval of $[0, 10]$ seconds.

Additionally, it is important to note that the quantum feature map inherent to this algorithmic structure is independent of the chosen training set. Consequently, increasing the number of training points does not necessarily guarantee a more efficient optimization process. On the contrary, a larger set of points may require additional iterations to achieve convergence.

In this study, the VQCs were trained using all points from the selected set, employing the Adam classical optimizer [112] with a learning rate of $\alpha = 0.01$. Convergence was assessed according to the criteria established in Chapter 4, namely: (i) satisfying a gradient threshold $\epsilon_{\partial\mathcal{L}} = 10^{-4}$, as defined in section 4.3, or (ii) reaching the target accuracy corresponding to a loss function value of $\epsilon_{\mathcal{L}} = 10^{-4}$. The choice of the gradient threshold $\epsilon_{\partial\mathcal{L}} = 10^{-4}$ is based on prior empirical observations, which indicate that no significant improvement occurs once all gradient components fall below this value.

For this application, the initial condition is implemented with a floating boundary handling (4.1.3) for u_0, in combination with a contribution to the loss function (4.1.3) for $\dot{u}_0$. This ensures the satisfying of the first condition while avoiding to over-constraint the system. In this case, the loss function can be expressed by Equation (4.3), with $\eta_{DE} = 1$; $\eta_{Cs} = 1$.

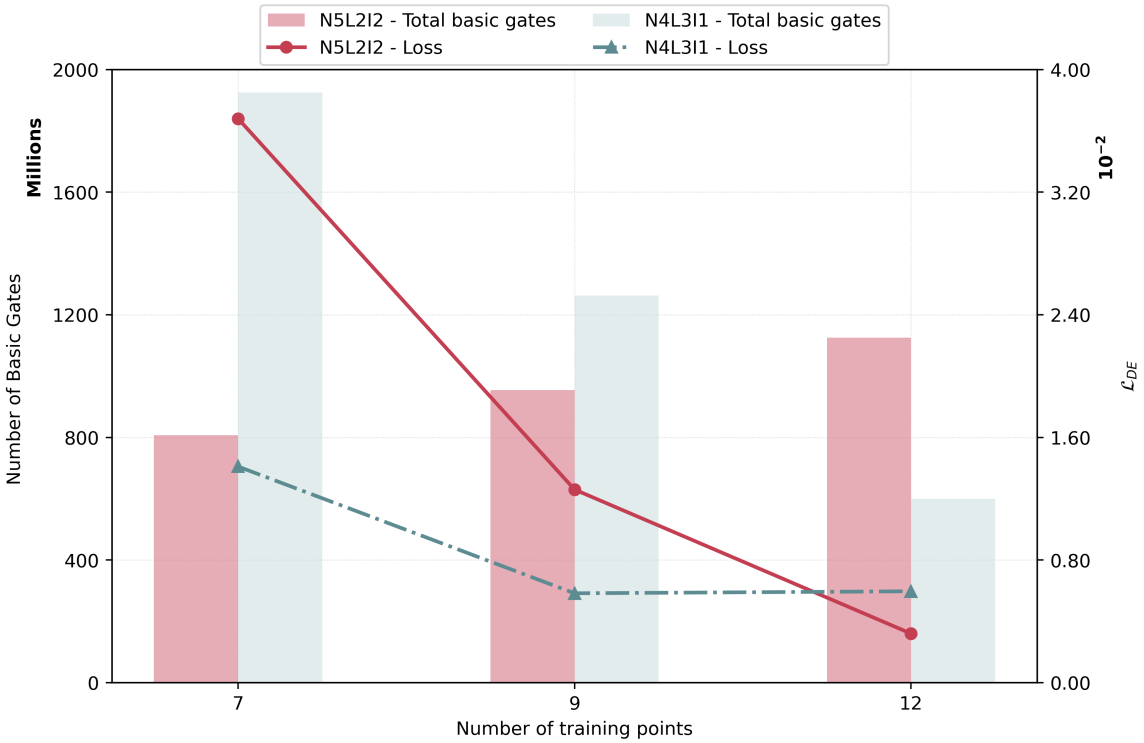


Figure 7.19: Average total loss and gate complexity for the Chebyshev-encoded VQA applied to the most efficient VQC structures with varying training set lengths over 5 tries. $N5L2I2$ stands for the structure of 5 qubits and 2 layers across a training set of the second type, while $N4L3I1$ stands for the structure of 4 qubits and 3 layers across a training set of the first type.

Among the different VQCs structures studied for the Chebyshev-encoded, the lowest average DE losses were observed with a VQC of 5 qubits and 2 layers over a set of 12 points of the second type (referred as $N5L2I2$) and with a VQC of 4 qubits and 3 layers over a set of 12 points of the first type (referred as $N4L3I1$). According to the results obtained from this algorithm, an increased number of points in the training set does not always correlate with a higher gate complexity or highly improved accuracy. However, the selection of the training points themselves and the structure of the VQC seems to have the most significant impact on both the quality of the solution and the gate complexity, as shown in the Figure 7.19.

Interestingly, when examining the evolution of the overall loss with respect to the number of training points, a trend similar to that observed for the first-order ODE with the Lagrange-encoded VQA emerges. Specifically, for the set of Chebyshev nodes of the first type, the loss tends to plateau, and increasing the number of points only slightly improves accuracy. However, in the present study, a larger set of hyperparameters was explored. Among the two configurations that performed best, the

one yielding efficient results on the first-type Chebyshev nodes employed fewer qubits, and its gate complexity decreased as the number of points increased. This behaviour suggests that each iteration was more effective, thereby requiring fewer iterations to converge toward the solution.

Nevertheless, this observation cannot be generalized, as the opposite trend was found for the best-performing configuration using the second-type Chebyshev nodes.

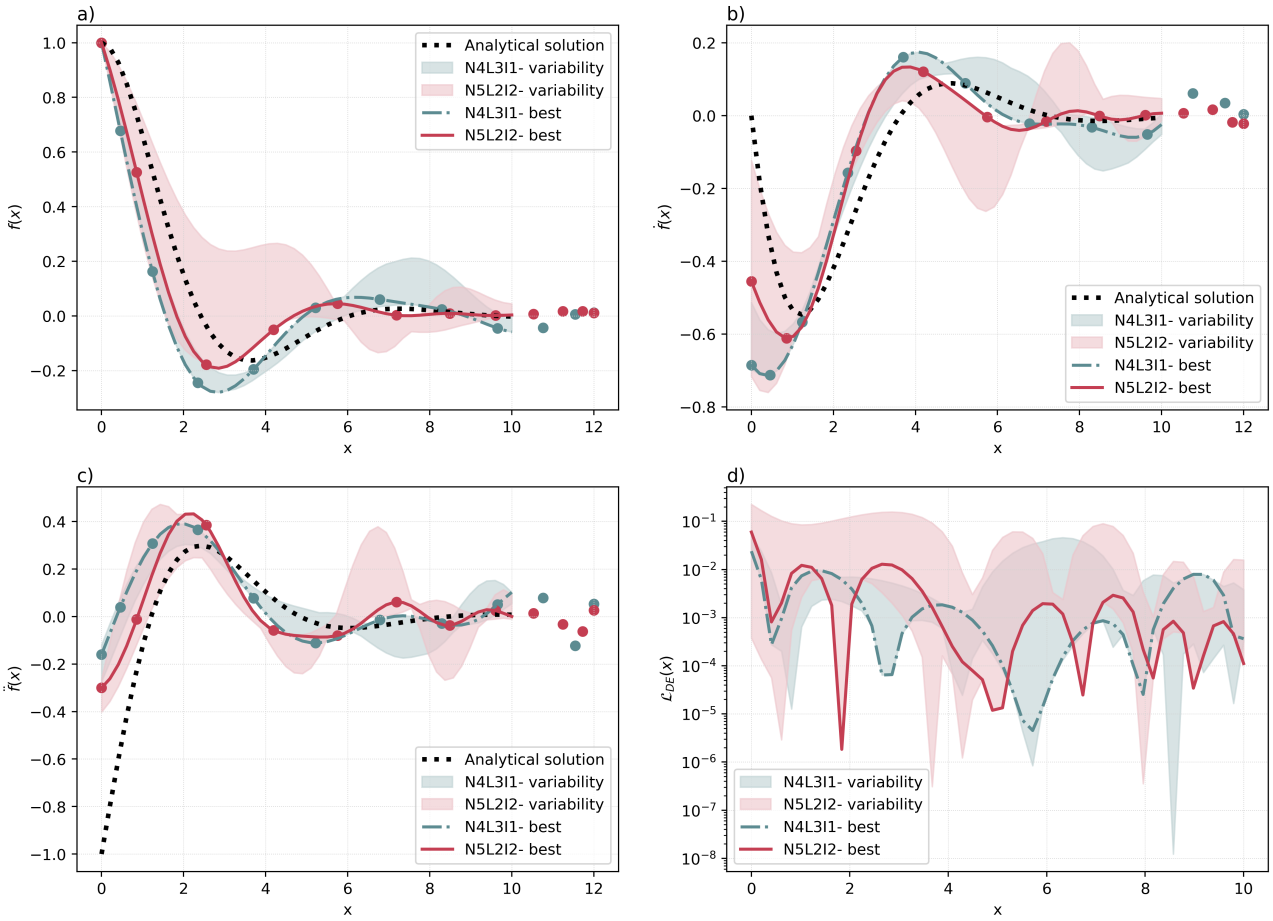


Figure 7.20: Results are presented for each of the most efficient VQC structures, illustrating variability and the best outcome achieved across five attempts for: a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale. $N5L2I2$ stands for the structure of 5 qubits and 2 layers across a training set of the second type, while $N4L3I1$ stands for the structure of 4 qubits and 3 layers across a training set of the first type.

Additionally, applying this algorithm with different random initial vectors of variational parameters θ revealed, again, a high sensitivity to these initial conditions. Across five independent attempts to solve the DMSS problem using the structures presented above, the $N5L2I2$ configuration exhibited a coefficient of variation of 20% in total loss and 57% in the number of iterations required for convergence.

Similarly, for the $N4L3I1$ structure, the coefficient of variation was 50% in total loss and 40% in iterations. Consequently, the accuracy and gate complexity shown in Figure 7.19 represent average outcomes, rather than the best or worst results achieved.

The variability in accuracy is further visualised in Figure 7.20, where the best solution achieved for each VQC structure is plotted alongside a shaded region, which represents where the other outputs would be, and consequently the worst-case outputs for each structure, effectively capturing the range of variability across the attempts.

For the two VQC structures that produced the best results, some results presented important oscillations toward the end of the interval, despite placing some training points outside the interval to mitigate this issue. Additionally, for both structures, the initial conditions for the first- and second-order derivatives were not satisfied. Finally, the corresponding total DE losses obtained for the best results highlighted in Figure 7.20 are, respectively, $2.50 \cdot 10^{-3}$ for $N5L2I2$ and $2.77 \cdot 10^{-3}$ for $N4L3I1$. However, the initial conditions for the first- and second-order derivatives are not fully met, resulting in a BC loss of 0.447 and 0.485 for each VQC structure, respectively, as illustrated in Figure 7.23.

Lagrange-encoded VQA

The algorithm proposed in this work introduces interpolation points in the encoding stage of the VQC. These points are chosen to be similar to the training set, with the objective of assessing whether satisfactory results can be achieved by training directly on the encoding points. The underlying motivation is that the computational cost of a VQA grows with the size of the training set. Consequently, this approach aims to reduce the overall circuit complexity and, therefore, the associated gate complexity.

The Lagrange-encoded VQA has been evaluated using sets of 5 to 8 points. Among these configurations, the optimal VQC size was found to correspond to 7 interpolation points, requiring 8 qubits, combined with the reduced circuit structure illustrated in Figure 4.3. As previously noted, the focus of this work lies in the theoretical aspects of variational quantum algorithms. For this reason, all quantum circuits considered here have been simulated in an ideal noise-free environment. Under these conditions, the use of a reduced structure is preferred because of its lower computational resource requirements.

When dealing with a second-order differential equation, whose expression involves the function itself as well as first- and second-order derivatives, the amount of encoded information is significantly higher. As a result, convergence for both the Chebyshev-encoded and Lagrange-encoded VQAs proved to be more challenging compared to the case of a first-order ODE. To address this, the algorithm allows dynamic adjustment of the number of training points and qubits during the optimization process. Furthermore, the optimization is split into two stages, relying on an evolving set of training points and additional regularization points to improve stability. The initial and boundary conditions are incorporated following the same methodology used in the Chebyshev-encoded VQA, but with modified weights: $\eta_{DE} = 1$, $\eta_{Cs} = 0.6$, and $\eta_R = 1$.

For the first stage of the optimisation, the set of training points initially consists of three Chebyshev nodes located on the left side of the domain, implemented using four qubits. Among these, the first node contributes to the boundary condition loss function $\mathcal{L}_{Cs}$, while the other two nodes contribute to the differential equation loss function $\mathcal{L}_{DE}$. Once the algorithm achieves convergence for these nodes, defined as all components of the loss function gradient falling below a threshold of 10^{-4}, an additional node is introduced. This new node contributes to $\mathcal{L}_{DE}$, while the second node is reassigned as a regularization point and begins contributing to the regularization loss function $\mathcal{L}_R$. This iterative process continues until all nodes are included and convergence is reached, which in this example corresponds to using eight qubits. The optimization is carried out using the Adam optimiser [112], with an adaptive learning rate that evolves from $\alpha = 0.04$ to $\alpha = 0.01$ depending on the total loss value $\mathcal{L}$.

The second stage of the optimization begins once the full set of nodes and qubits is established. At this point, the size of the VQC remains fixed. The nodes are incorporated in groups of three, each group contributing to $\mathcal{L}_{DE}$, while the remaining points act as regularization points. This process starts again from the left side of the domain. During this stage, the Adam optimiser is employed with a fixed learning rate of $\alpha = 0.01$.

The application of the Lagrange-encoded VQA, using this two-part optimisation, has been observed to be insensitive to the initial random vector of variational parameters θ. For each type of training set, the solutions obtained after the first part of the optimization over 5 trials with different initial random values of the vector of variational parameters θ, were extremely consistent, with a mean total loss of

$1.24 \cdot 10^{-2}$ and $2.76 \cdot 10^{-2}$ with a coefficient of variation of 1.0% and 1.5% for the first and second type of training set, respectively. Similarly, the average number of iterations to achieve the convergence of the first part is, respectively, 670 and 620 iterations with a coefficient of variation around 11% in both cases. The second part of the optimization process consistently starts from a similar state, resulting in comparable deviation in terms the solution accuracy and the number of iterations.

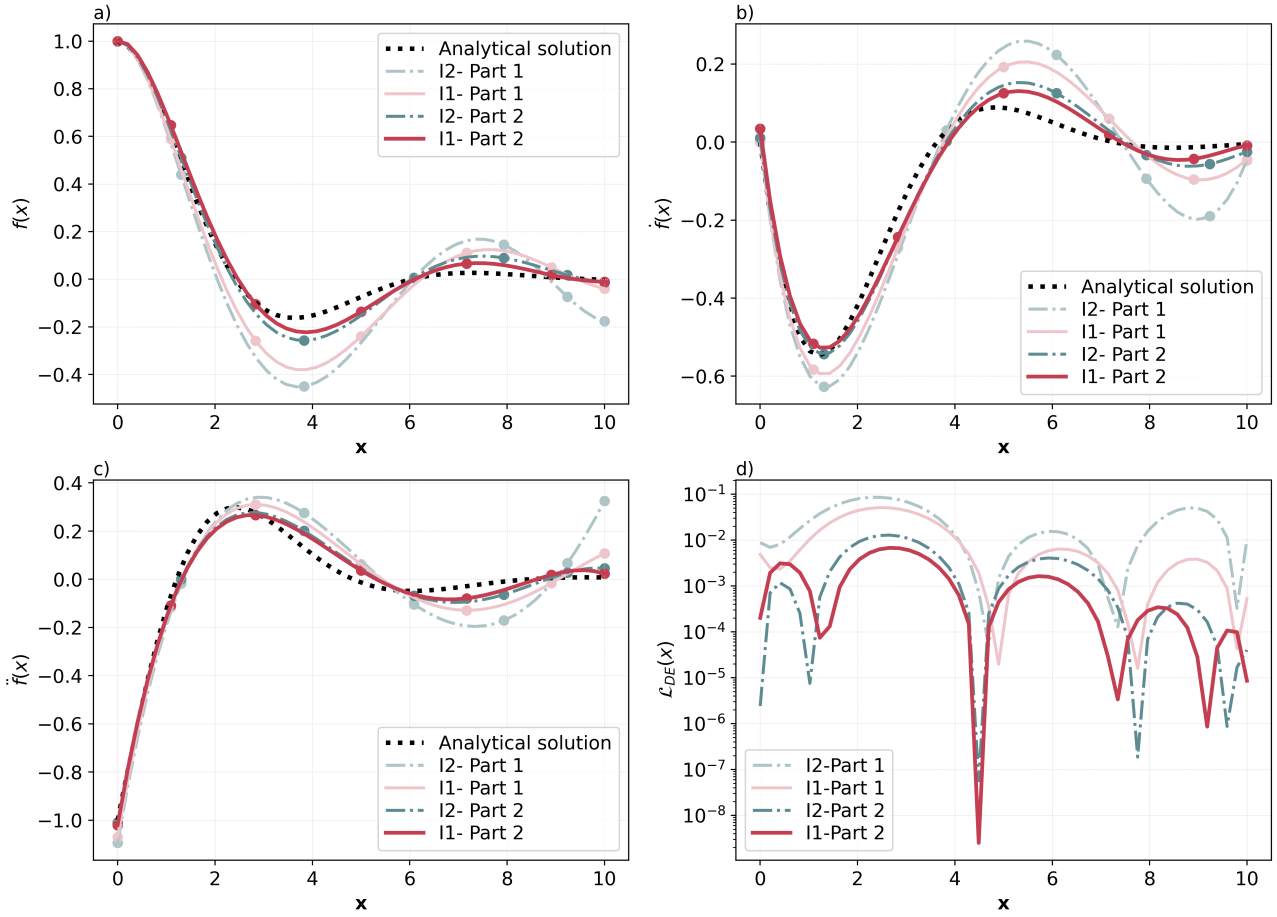


Figure 7.21: Results of the Lagrange-encoded VQA for both parts of the optimization process, achieved across five attempts. a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale. $I1$ stands for the first type of training set of points while $I2$ stands for the second.

As illustrated in Figure 7.21, the first part of the optimization yielded a close approximation of the solution, achieving slightly better accuracy for the set of type 1 training points. Similarly, the second part of the optimization process produced improved results for the set of type 1 training points, with a total DE loss of respectively $1.51 \cdot 10^{-3}$ and $2.87 \cdot 10^{-3}$. Additionally, the initial conditions are more effectively met this time, with a loss $\mathcal{L}_{Cs}$ of respectively $1.18 \cdot 10^{-3}$ and $1.07 \cdot 10^{-4}$, as depicted in the Figure 7.23.

Lagrange-encoded VQA vs. Chebyshev-encoded VQA

For both quantum algorithms, the number of qubits and Chebyshev nodes is different. The idea here was to select, by trial and error, the optimal set of training points and structures to obtain the most accurate solution for each approach.

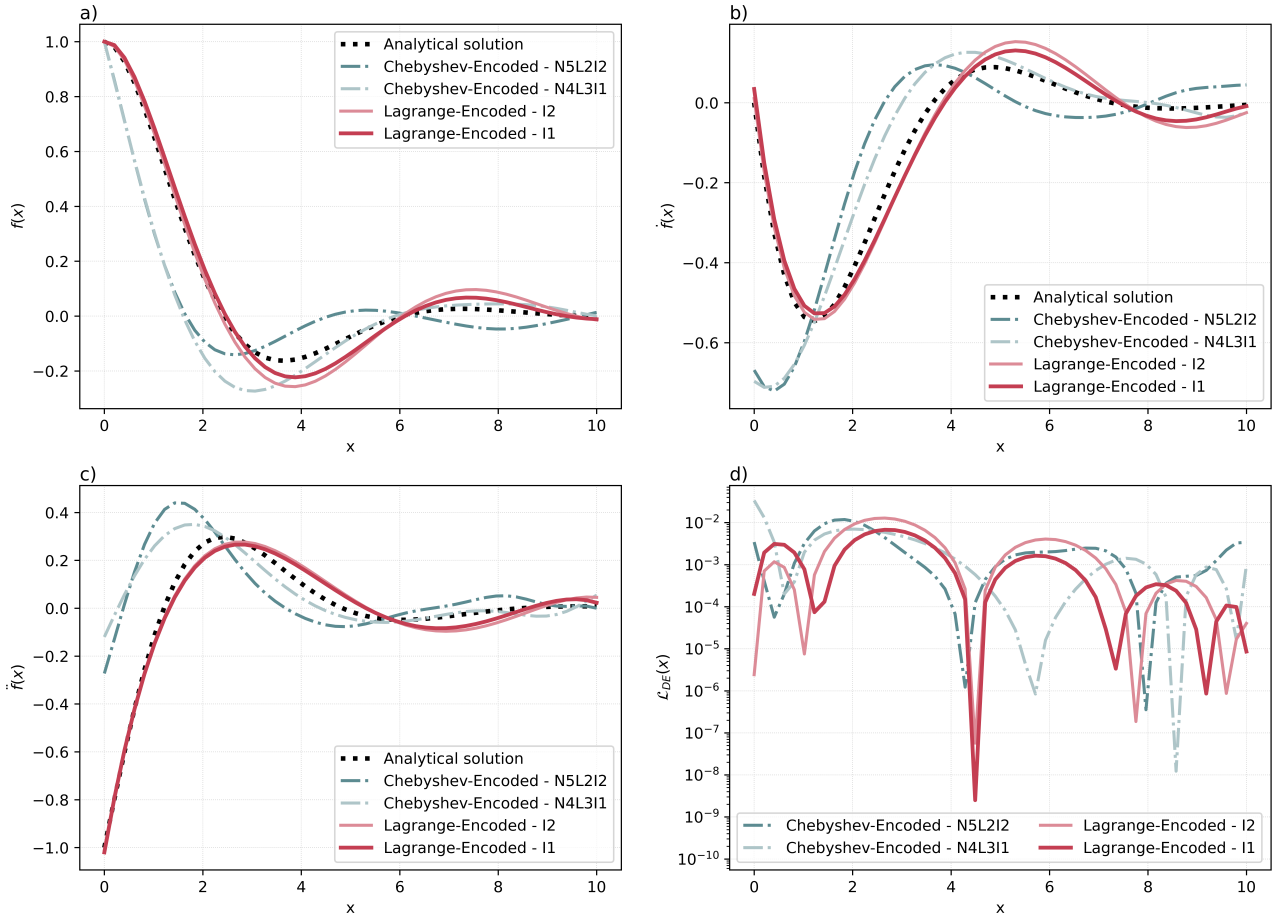


Figure 7.22: Results comparison between the Lagrange-encoded VQA and the best outcomes from this work version of the Chebyshev-encoded VQA for both types of training sets of points, for : a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$ and d) the DE loss contribution $\mathcal{L}_{DE}(x)$ across 50 equispaced points in the interval, plotted on a logarithmic scale.

As shown in Figure 7.22, both VQA provide a satisfactory approximation of the desired function $f_{\theta_{opt}}$ over the chosen interval, with better accuracy with the Lagrange-encoded VQA. But we have not a perfect overlap. Additionally, we can still see some oscillations in the derivatives with respect to x specially for this work's version of the Chebyshev-encoded VQA, despite training being performed over Chebyshev nodes. Training over equidistant points was excluded from the result because it showed poor results.

Additionally, for this algorithm, the initial conditions are not met for the derivatives, suggesting that the contribution of the loss function for the initial values may conflict with the overall loss function. It suggests that this quantum algorithm may lack flexibility to take into account all types of initial conditions, potentially due to constraints in the encoding or variational ansatz.

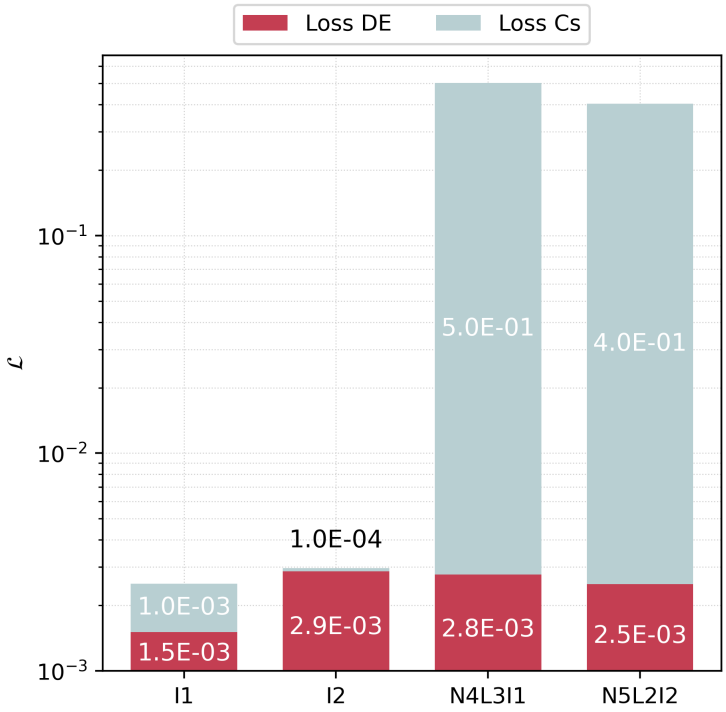


Figure 7.23: DE and BC loss comparison between the Lagrange-encoded VQA and the best outcomes of this work version of the Chebyshev-encoded VQA for both types of training sets of points, plotted on a logarithmic scale.

The Lagrange-encoded VQA reproduces Lagrange interpolation and enables the determination of the fitting polynomial with the smallest possible degree over the given set of points. As a result, the function and its derivatives are smoother and exhibit fewer oscillations compared to the version of the Chebyshev-encoded algorithm developed in this work.

Moreover, the Lagrange-based algorithm adheres more effectively to initial conditions. Overall, the solutions obtained with the Lagrange-encoded VQA appear to be of higher quality than those produced by the Chebyshev-based approach, particularly near the boundaries, and show good agreement with the analytical solution, as illustrated in Figure 7.22.

When analyzing the different loss contributions in Figure 7.23, both algorithms converge toward solutions with a comparable range of differential equation loss. However, the boundary-condition

term in the loss function significantly reduces the solution quality for the Chebyshev-encoded VQA in its current implementation.

It should be noted that the original Chebyshev-encoded algorithm proposed by Kyriienko *et al.* [4] was extensively tested on first-order differential equations with highly convincing results. It is likely that the implementation presented in this work would require further tuning, particularly in handling boundary and initial conditions, and possibly an increased number of variational parameters, such as introducing a three-gate sequence in the HEA, to improve its performance on higher-order equations.

Another observation is the strong influence of the initial random vector of variational parameters, not only on the number of iterations, affecting both the performance of the classical optimizer and the trainability of the VQC, but also on the quality of the solution. This suggests that the current system may lack sufficient expressivity.

Consequently, future work could include testing a three-gate sequence in the HEA to determine whether the system is underconstrained, leading to convergence toward local minima, or overconstrainning, making it difficult to reach the correct solution. It is important to note that in this study, while statevectors were used, gradients were computed via derivative quantum circuit methods, which introduced significant overhead in simulating both VQAs. As discussed in Chapter 6, techniques exist to reduce the computational burden of such simulations in early research stages when working with statevector simulations.

Finally, a direct comparison of the number of gates required by the Lagrange-encoded VQA and the best-performing Chebyshev-encoded VQA implemented in this work reveals a significant difference, as illustrated in Figure 7.24.

This comparison highlights the Lagrange-encoded algorithm developed in this work as both the most precise and the most efficient approach for solving, at least, this damped mass-spring system problem. While improvements to the Chebyshev-encoded VQA implementation could enhance its accuracy, such enhancements would not address its substantial computational cost. Achieving low computational overhead while working with Chebyshev polynomials therefore remains an open question.

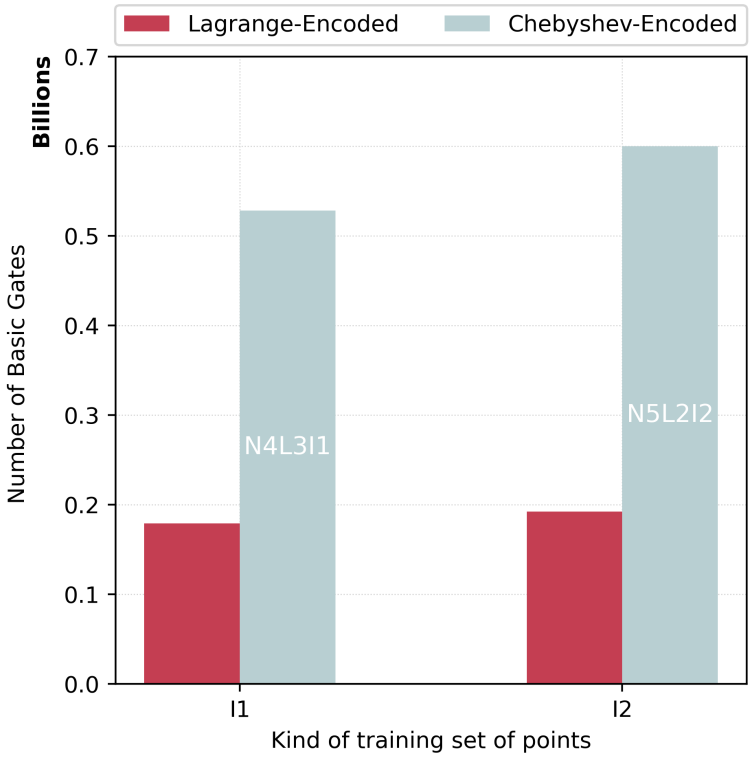


Figure 7.24: Gate complexity comparison between the Lagrange-encoded VQA and the best outcomes of this work version of the Chebyshev-encoded VQA for both types of training sets of points.

To conclude, when considering both solution quality and gate complexity, the optimal result is obtained using the Lagrange-encoded VQA trained on a set of seven Type-I training points.

7.2.4 Trainability and Expressivity

When comparing the results obtained with the Chebyshev-encoded VQA and the Lagrange-encoded VQA, we observe that a perfect overlap with the analytical solution was not achieved, despite exploring several circuit structures.

In this setting, which involves a second-order differential equation incorporating the function, its first derivative, and its second derivative, classical simulations of the VQAs were computationally demanding. Consequently, we did not explore configurations with a larger number of qubits, deeper circuits, or a three-gate sequence in the HEA for the Chebyshev encoding. Interestingly, we obtained reasonably accurate results with a one-gate sequence in the HEA for the Chebyshev case. This contrasts with the toy problem, which failed to converge toward the solution.

A natural first step is therefore to question the expressivity of the chosen VQC structures.

To assess this, we considered the best-performing VQCs and retrained them, not to solve the differential equation directly, but to interpolate its known solution. This resulted in the following loss function:

$$\mathcal{L}_{\text{interpolation}} = \frac{1}{M} \sum_{t=1}^{M} \big(f_\theta(x_t) - u(x_t)\big)^2, \tag{7.13}$$

where u denotes the exact solution of the differential equation, and $\{x_t\}_{t=1}^{M}$ is the set of M training points. The interpolation results over 20 equidistant points for the Chebyshev-encoded configuration *N5L2I2* and for the Lagrange-encoded configuration using a set of type 1 Chebyshev nodes are presented in Figure 7.25.

From these results, we see that interpolation of the function itself yields a good overlap with the exact solution. This suggests that both VQCs are expressive enough to represent the target function. However, when examining the first and second derivatives, the approximation deteriorates significantly. This observation is crucial in the context of polynomial-encoded VQAs: the VQC should not only represent the solution of the differential equation but also its derivatives with comparable fidelity.

Consequently, this indicates that the approximation obtained in the previous section represents the best compromise between the accuracy of the function and its derivatives, as dictated by the objective of minimizing the loss function, even if it does not correspond to an exact interpolation of the analytical solution.

On a side note, the Chebyshev polynomial reconstructed from the Chebyshev-encoded VQC after the interpolation training exhibits oscillations near the end of the interval in its first- and second-order derivatives. These oscillations increase in amplitude with the order of differentiation. As discussed in Chapter 5, derivatives with respect to x are obtained via automatic differentiation, using an encoding function that diverges at the endpoints of the interval. This divergence may amplify the observed oscillations. Interestingly, these oscillations were largely mitigated in the method employed for the quantum resolution of the differential equation, underscoring both the efficiency and the necessity of introducing training points outside the original domain.

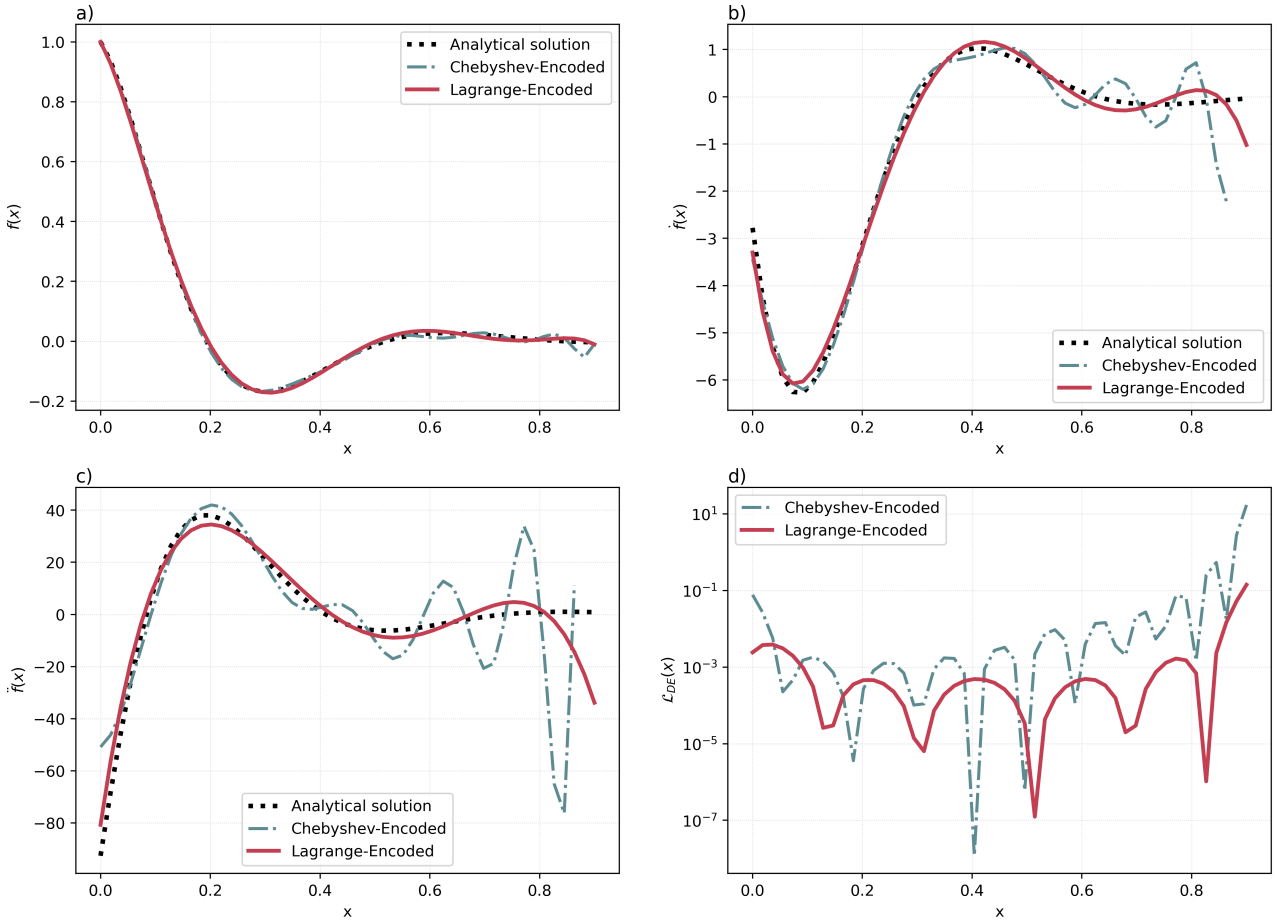


Figure 7.25: Interpolation results for the Chebyshev and Lagrange encodings applied to the damped mass-spring system. a) the function $f(x)$, b) the first derivative $\dot{f}(x)$, c) the second derivative $\ddot{f}(x)$, and d) the loss function of the differential equation $\mathcal{L}_{DE}(x)$ across the interval.

Beyond expressivity, we must also question the trainability of the chosen VQCs. The current loss function, which computes the mean squared error of all terms in the differential equation averaged over the interval, may not be optimal. Alternative formulations could potentially improve convergence. Finally, the choice of classical optimizer may play a significant role, it impacts the accuracy, convergence speed, or computational complexity. Future work should investigate these directions to further refine and enhance the performance of the two proposed methods.

7.3 Poisson Equation

This section is based on the results presented in our publication [1], providing additional details.

7.3.1 Problem definition

The Poisson equation is one of the most computationally intensive DEs to solve, requiring very expensive iterative methods. Previous work from Sato *et al.* [5] solved the Poisson equation using a quantum approach for different boundary conditions and a step function s from $1/2^{n/2}$ to $-1/2^{n/2}$ for $n = 5$, using a VQA based on the minimal potential energy, with $n + 1 = 6$ qubits over $2^n = 32$ points. In this work, we solved the same equation using the Lagrange-encoded VQA with a reduced structure on 4 qubits trained over 3 points. The Poisson equation can be expressed as

$$\frac{d^2 f}{dx^2} + s = 0, \tag{7.14}$$

where f is the state field and s the given step function as a source term, plotted in Figure 7.26. Different types of boundary conditions have been tested for this equation, the periodic, Neumann and Dirichlet boundary conditions. It is a great opportunity to assess the versatility of the Lagrange-encoded VQA. This Poisson equation can be analytically solved by splitting the interval in half. The solution over each half interval is a second degree polynomial where the coefficients are real numbers and depend on the source function and the boundary conditions.

7.3.2 Analytical solution

The Poisson equation is defined in Equation (7.14), with a given source term defined as follow:

$$s(x) = \begin{cases} +(1/2)^{n/2} & \text{if } x < \frac{a+b}{2}, \\ -(1/2)^{n/2} & \text{if } x > \frac{a+b}{2}, \end{cases} \tag{7.15}$$

with $n = 5$, over the interval $[a, b]$. In the work of Sato *et al.* [5], this Poisson equation is solved over 32 nodes, defining the interval as $[\![0, 31]\!]$. The analytical solution can be determined by splitting the

interval in half. The solution over each half is a second degree polynomial as expressed in the following equation:

$$f(x) = \begin{cases} Ax^2 + B_1 x + C_1 & \text{if } x < \frac{a+b}{2}, \\ Ax^2 + B_2 x + C_2 & \text{if } x > \frac{a+b}{2}, \end{cases} \tag{7.16}$$

with $A = -\frac{1}{2}^{\frac{n}{2}+1}$. B_i, C_i are real coefficients which depend on the boundary conditions. In each of all cases, the continuity of the middle point $x_m = \frac{a+b}{2}$ for the solution function and its first derivative gives the following definitions:

$$\lim_{x \to x_m^+} f(x) = \lim_{x \to x_m^-} f(x) = f(x_m) \tag{7.17}$$

$$\lim_{x \to x_m^+} \frac{df}{dx}(x) = \lim_{x \to x_m^-} \frac{df}{dx}(x) = \frac{df}{dx}(x_m) \tag{7.18}$$

Other conditions depend on the boundary. For the periodic boundary condition, the solution follows $f(x+T) = f(x)$, with the period T defined in Sato *et al.* [5] work as $T = b - a + 1 = 32$. The Dirichlet boundary condition consists in defining the values at the extremities of the interval. In Sato *et al.* [5] work, it is defined as $f(a-1) = f(b+1) = 0$. Finally, the Neumann boundary condition indicates the values of the first order derivative at the extremities, which is defined here as $f'(a) = f'(b) = 0$. All considered, the analytical solution for each boundary equation are:

- Periodic :

$$f_P(x) = \begin{cases} A\left(x^2 - \frac{1}{2}(3a + b - 1)x + \frac{1}{2}(a - \frac{1}{2})(a + b + 1)\right) & \text{if } x < x_m, \\ 0 & \text{if } x = x_m, \\ -A\left(x^2 - \frac{1}{2}(3b + a + 1)x + \frac{1}{2}(b + \frac{1}{2})(a + b + 1)\right) & \text{if } x < x_m, \end{cases} \tag{7.19}$$

- Dirichlet :

$$f_D(x) = \begin{cases} A\left(x^2 - \frac{1}{2}(3a + b - 2)x + \frac{1}{2}(a - 1)(a + b)\right) & \text{if } x < x_m, \\ 0 & \text{if } x = x_m, \\ -A\left(x^2 - \frac{1}{2}(a + 3b + 2)x + \frac{1}{2}(b + 1)(a + b)\right) & \text{if } x < x_m, \end{cases} \tag{7.20}$$

- Neumann :

$$f_D(x) = \begin{cases} A\left(x^2 - 2ax + \frac{1}{4}(3a^2 + 2ab - b^2)\right) & \text{if } x < x_m, \\ 0 & \text{if } x = x_m, \\ -A\left(x^2 - 2bx + \frac{1}{4}(3b^2 + 2ab - a^2)\right) & \text{if } x < x_m. \end{cases} \tag{7.21}$$

7.3.3 Quantum resolution

Lagrange-encoded algorithm vs Sato *et al.* algorithm

The two quantum approaches being compared here for the solution of the Poisson equation are quite different. The Sato *et al.* algorithm uses a discretised approach, where the output is the unitary vector of amplitudes representing the approximated solution function across the specified interval. In contrast, the algorithm using Lagrange polynomials encoding outputs the approximated solution function, which can be computed for any point/node within the chosen interval.

Given its discretised nature, the Sato *et al.* algorithm encodes both the source function s and the algorithm output ψ_θ into quantum states as vectors of amplitudes (Equation (3.18)). However, this implementation assumes the existence of an efficient unitary operator $\hat{\mathcal{U}}_s$ to prepare the quantum state $|s\rangle = \hat{\mathcal{U}}_s |0\rangle^{\otimes n}$, which may limit its application to other source functions or other DEs.

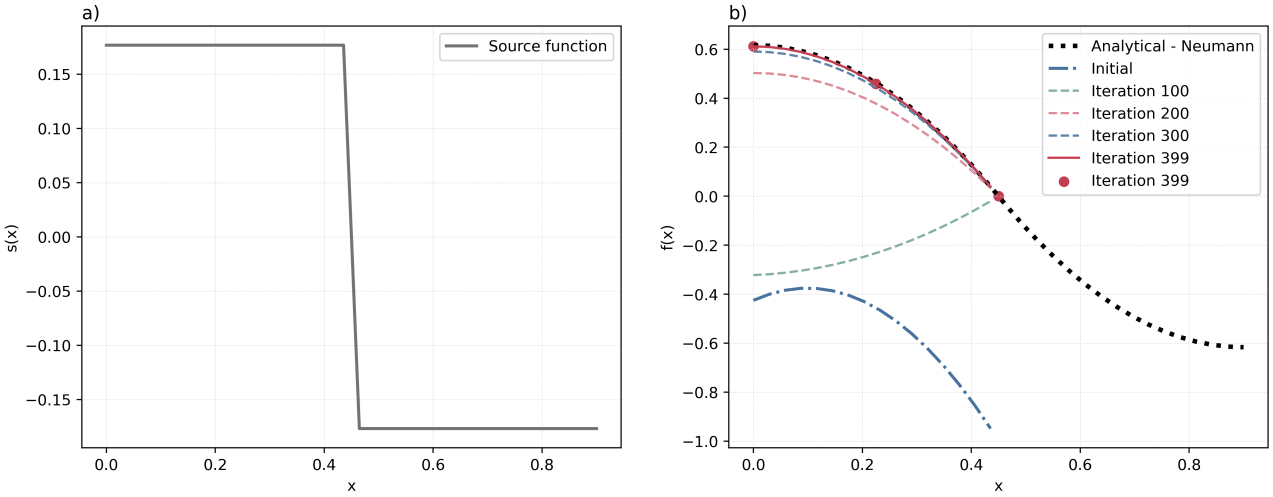


Figure 7.26: a) Source term s used for the Poisson equation application. b) Iterative evolution of the Lagrange-encoded solution for Neumann boundary conditions.

The Lagrange-encoded algorithm operates with continuous functions. In cases where the source function is discontinuous, as in this particular application, at least the second-order derivative will exhibit a discontinuity. To address this challenge, the algorithm can be applied to sets of sub-intervals. For example, the algorithm can be initially applied to the first half of the interval, and the solution over the second half can be determined by either symmetry or applying the algorithm again to the second half.

Here, the Lagrange-encoded algorithm was applied to the first half of the interval and iterated until it achieved a precision comparable to the Sato *et al.* solution, and further, as shown in Figure 7.26.b. Figure 7.26.a illustrates the shape of the source term, with a discontinuity. This figure illustrates how the Lagrange-encoded VQA is eventually converging to the analytical solution.

Using the Lagrange-encoded VQA, we ran the algorithm until convergence with the following hyper-parameters: $\eta_{DE} = 1, \quad \eta_{BCs} = 1,\ \varepsilon_{\alpha} = 1 \cdot 10^{-5},\ \varepsilon_{\partial\mathcal{L}} = 1 \cdot 10^{-6},\ \varepsilon_{\partial^2\mathcal{L}} = 1 \cdot 10^{-15}$.

These values were chosen to be very small in order to achieve high precision. We considered two stages of the solution. First, when the Lagrange-encoded VQA yields an approximation with an overall DE loss $\mathcal{L}_{DE}$ comparable to the result obtained using Sato *et al.*[5] code. Then, we pushed the optimization further to evaluate the maximum achievable accuracy. Note that the data obtained in this section are averaged over 5 tries to minimise the influence of the random initialisation.

Neumann boundary condition We begin by looking at the Neumann boundary condition, which corresponds to the case where the VQA proposed by Sato *et al.* exhibited the poorest performance, as discussed in the Chapter 3. Figure 7.27 illustrates the evolution of the loss function for one of the five runs conducted to assess the performance.

In Figure 7.27.a), we observe that the DE loss decreases with the number of iterations and reaches a value comparable to the DE loss of the Sato *et al.* solution after approximately 250 iterations. This trend can also be inferred from Figure 7.27.b), where the DE loss is plotted at different stages of the optimization. An important observation is that, for the Lagrange-encoded VQA, the contribution of the DE loss remains nearly constant across the domain, whereas this is not the case for the Sato *et al.* algorithm. For the latter, the loss is significantly larger near the boundaries of the computational domain.

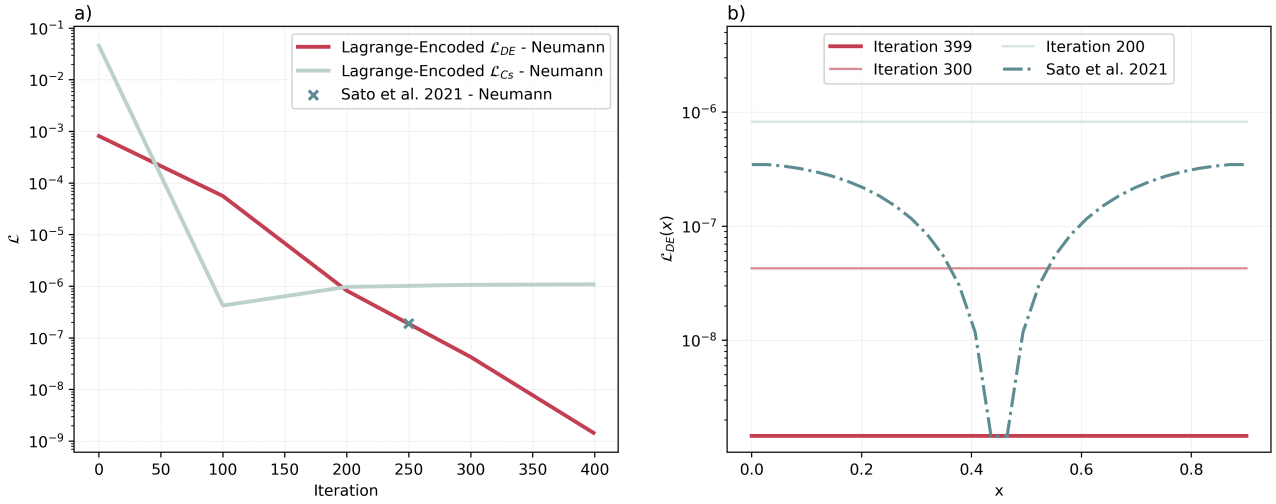


Figure 7.27: a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with Neumann boundary conditions, plotted on a logarithmic scale.

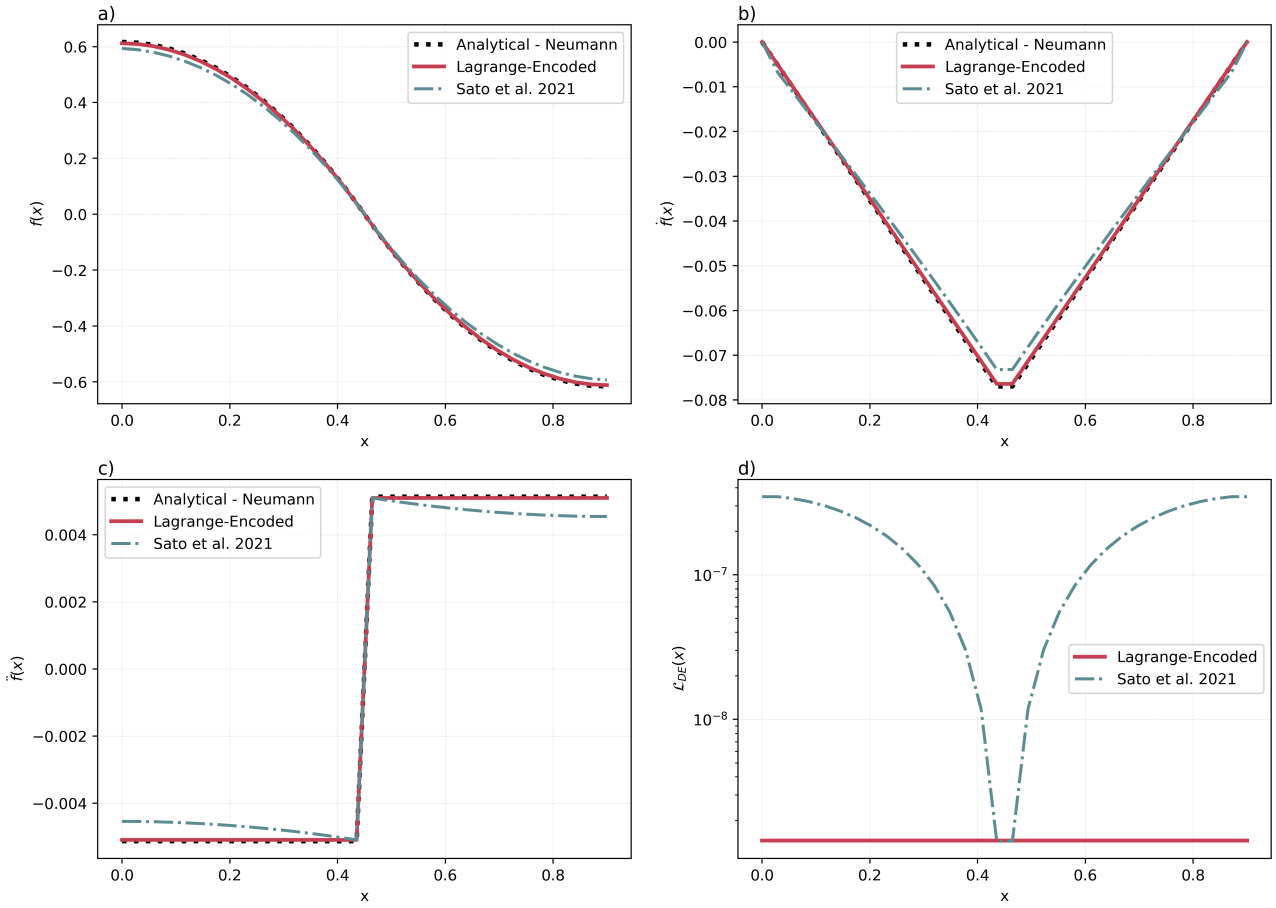


Figure 7.28: Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the Neumann boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$.

Additionally, when examining the condition loss, which enforces the boundary conditions, we observe that it is substantially larger than the DE loss contribution and tends to plateau during training. This plateau effectively limits the overall quality of the solution, as the DE loss becomes negligible compared to the boundary condition loss.

Figure 7.27.a) illustrates this behavior. The red line at iteration 399 marks the DE loss at convergence for the Lagrange-encoded VQA under the previously specified hyperparameters. At this point, the loss value coincides with the local minimum achieved by the Sato *et al.* VQA. The final output of the Lagrange-encoded VQA for the Neumann boundary condition, alongside the corresponding result from the Sato *et al.* VQA, is presented in Figure 7.28.

To further investigate the performance of both algorithms, it can be helpful to look at the first- and second-order derivatives of the solution of the Poisson equation. Focusing on the case with the Neumann boundary condition, it can be seen in Figure 7.28, that the Sato *et al.* solution deviates from the analytical solution at the boundary of the domain for the second order derivative and near the discontinuity for the first order derivative. It is important to note that the accuracy of the Sato *et al.* solution could be improved by increasing the spatial discretisation, but this would also increase the size of the VQC and consequently the complexity.

Periodic and Dirichlet boundary conditions Using the same methodology, the results for periodic boundary conditions are shown in Figures 7.29 and 7.30, while those for Dirichlet boundary conditions are presented in Figures 7.31 and 7.32.

Overall, both VQAs produced highly accurate results, exhibiting a good overlap with the analytical solution. Minor discrepancies appear in the derivatives obtained with the Sato *et al.* algorithm, which are most likely attributable to finite differencing. These differences could only be mitigated by increasing the number of qubits of the variational quantum circuit, rather than through parameter tuning.

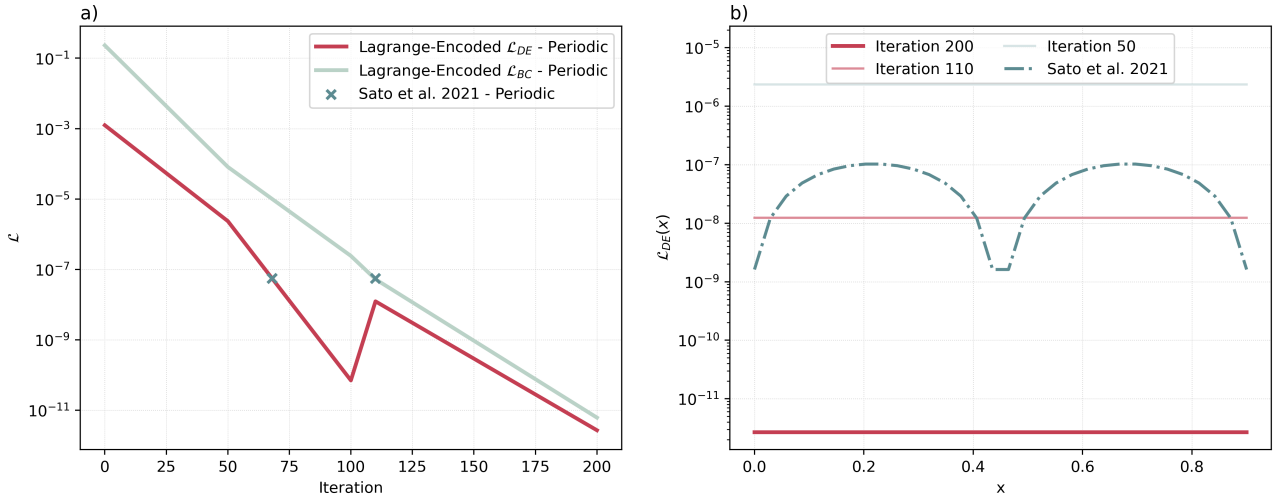


Figure 7.29: a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with periodic boundary conditions, plotted on a logarithmic scale.

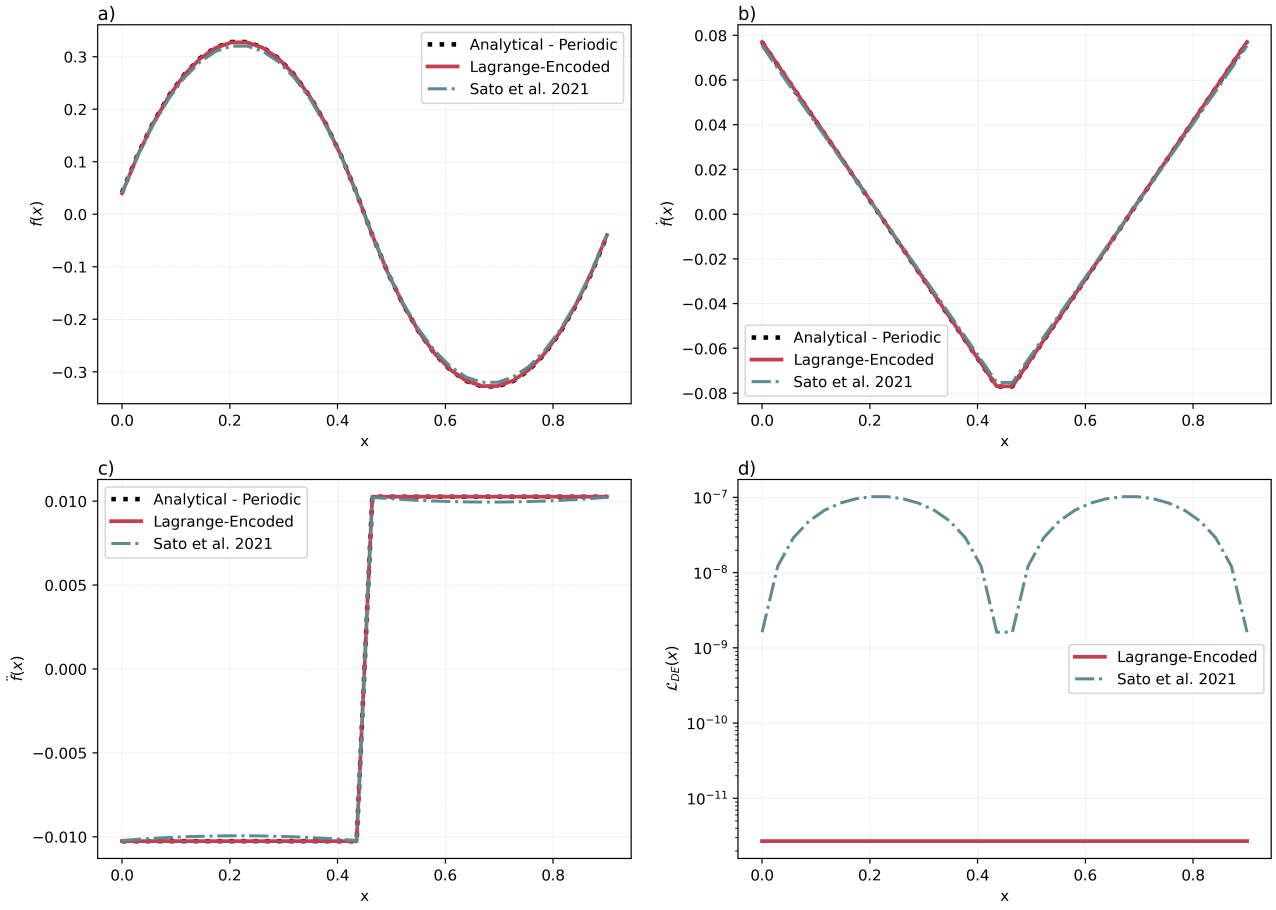


Figure 7.30: Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the periodic boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$.

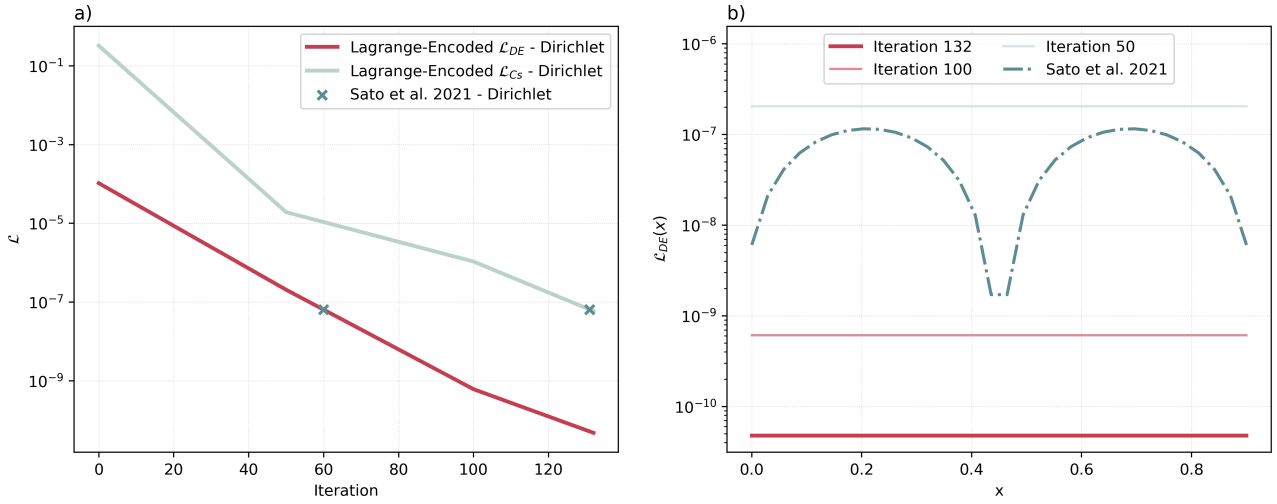


Figure 7.31: a) Iterative evolution of each terms of the loss functions $\mathcal{L}_{DE}, \mathcal{L}_{Cs}$ obtained with the Lagrange-encoded algorithm.. b) Detailed comparison of the DE loss contribution over the interval, $\mathcal{L}_{DE}(x)$. Both plots are obtained with Dirichlet boundary conditions, plotted on a logarithmic scale.

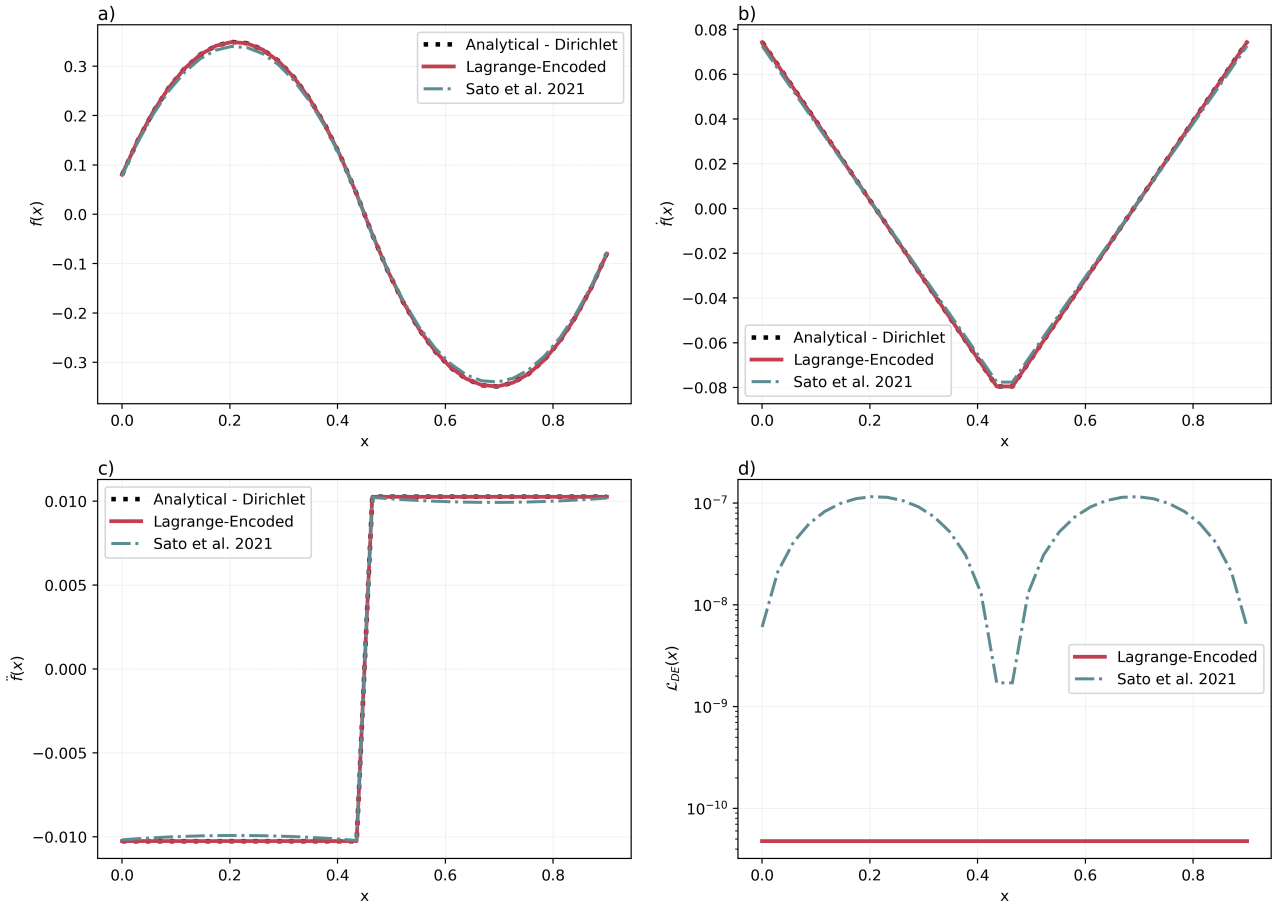


Figure 7.32: Comparison between the Lagrange-encoded solution and the solution obtained with the algorithm of Sato *et al.* [5] for the Dirichlet boundary condition, showing the a) function $f(x)$, b) first-order derivative $\dot{f}(x)$, c) the contribution of the differential equation term to the loss function over the domain $\mathcal{L}_{DE}(x)$.

Loss values The loss function of the Lagrange-encoded algorithm accounts for both the evaluation of the differential equation and the boundary conditions. Figure 7.33 illustrates the contribution in the total loss function from the differential equation and the constraints of the boundary conditions. It can be seen that the contribution from the boundary conditions constraint is different depending on the type of boundary conditions, with a significant contribution for Neumann boundary conditions and a relatively small contribution for periodic and Dirichlet boundary conditions. This behaviour arises from the nature of the Neumann condition, which imposes constraints on the derivative rather than on the function itself. Consequently, its influence on the training process of the VQA is indirect, making it more challenging to attain high accuracy.

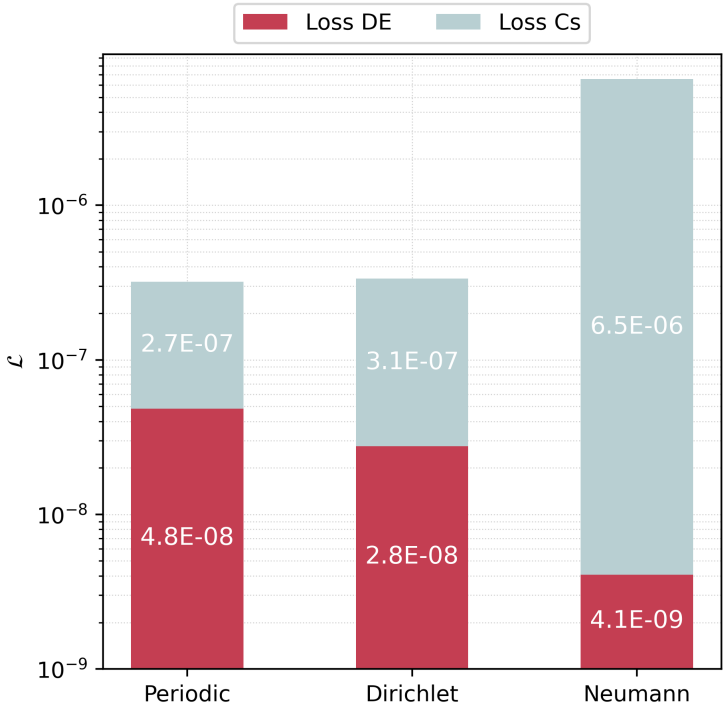


Figure 7.33: Overall loss value of the Lagrange-encoded algorithm applied to the different boundary conditions, plotted on a logarithmic scale.

For all boundary conditions (i.e., Periodic, Dirichlet, and Neumann), a direct comparison to the analytical solution revealed an overall good agreement for both VQA, even at the boundaries of the computational domain, as seen in Figures 7.28,7.30, 7.32. For each boundary condition, the mean absolute error of the solution obtained using the Sato *et al.* algorithm is 4, 3, and 6 times larger than the error obtained for the Lagrange-encoded VQA, for periodic, Dirichlet and Neumann boundary conditions, respectively.

Gate complexity estimation

Similarly to the previous application, the gate complexity of both VQAs has been estimated by the number of circuits as well as the number of basic gates required to complete the calculation. However, this estimation can only be regarded as an order of magnitude estimation due to the fundamentally different approaches of the two VQAs being compared. Moreover, for both algorithms, a different classical optimizer had been used, which significantly impacts the number of iterations and, consequently, the overall gate complexity. As previously mentioned in section 3.4.3, the BFGS optimizer used by Sato *et al.* [5] has shown better performances in training linear quantum solvers over few qubits [116] than the Adam optimizer used for the Lagrange-encoded algorithm.

For one iteration of the Sato *et al.* algorithm, the main circuit (Figure 3.8) is evaluated with 3, 4, or 5 observables for periodic, Dirichlet, and Neumann boundary conditions, respectively, to obtain the loss function and its gradient with respect to the variational vector θ, which contains 45 variational parameters. This requires numerous derivative circuits to compute the gradient of the main circuit with a large variational ansatz. Additionally, the main circuit (Figure 3.8) includes an encoding part and, in some cases, a shift part depending on the observable.

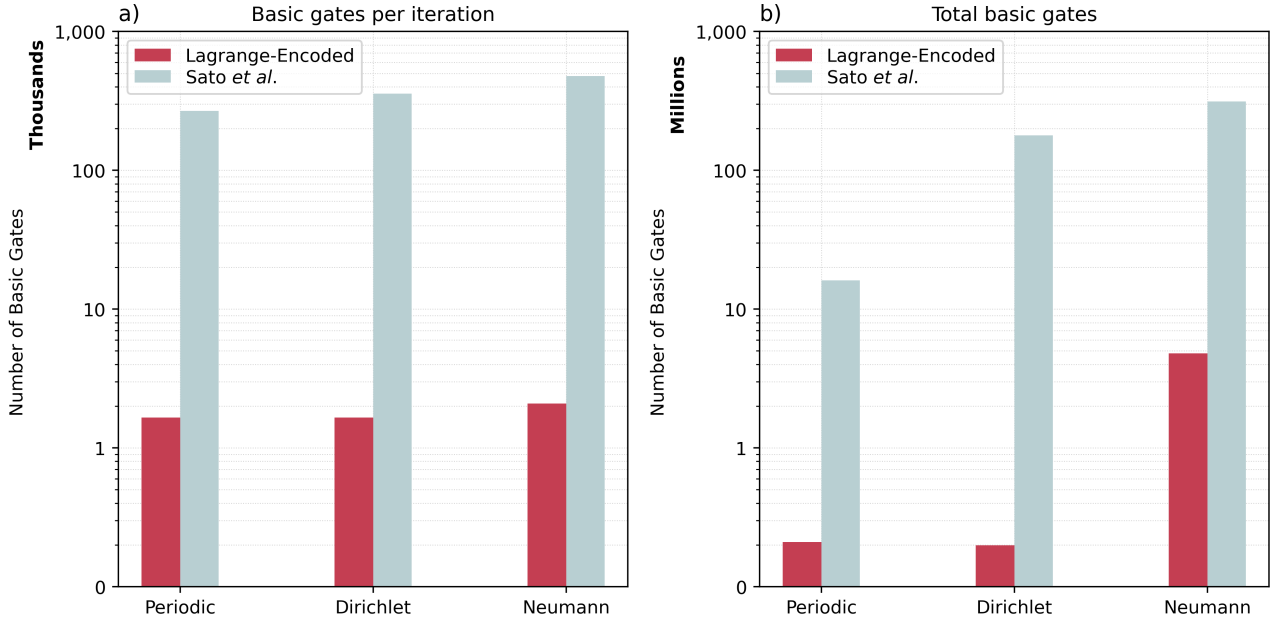


Figure 7.34: Gate complexity estimation of the Lagrange-encoded algorithm, averaged over five attempts per boundary conditions, compared with the Sato *et al.* algorithm [5], plotted on a logarithmic scale. a) Basic gates per iteration, b) Total basic gates.

In contrast, the proposed Lagrange-encoded algorithm requires circuits to evaluate the loss function (Equation (4.3)) and its gradient with respect to the variational vector θ for each iteration. This includes the differential equation loss over the control points $\{x_i\}$ and for the boundary conditions at x_0. Here, the variational vector contains only 3 parameters. This structural difference leads to a significant disparity in gate complexity. Although the Sato *et al.* algorithm [5] requires, on average, half as many iterations to converge to the solution of the Poisson equation, its circuit structure is much more complex, requiring 200 times more gates per iteration compared to the Lagrange-encoded algorithm, as illustrated in Figure 7.34.

Moreover, while the version of the HEA [135] employed in the algorithm of Sato *et al.* offers flexibility through its numerous variational parameters θ, it introduces significant complexities due to its multilayered structure and the inclusion of controlled rotational gates and Toffoli gates. Controlled rotational gates can be decomposed into two CNOT gates, which already implies two computational steps per such gate. In contrast, a single Toffoli gate, also known as a CCNOT or double-controlled NOT gate, requires six CNOT gates, in addition to multiple single-qubit rotations. Consequently, the circuit runtime grows rapidly. This large number of parameterized and fixed gates also exacerbates the challenges associated with classical optimization. Such complexity likely explains why the algorithm of Sato *et al.* exhibits a notably high gate count.

As an illustrative observation regarding simulation time, the code provided by Sato *et al.* required approximately 2.5 hours for the periodic case, 16 hours for Dirichlet, and an entire day for Neumann boundary conditions. These times correspond to an average of about 60, 500, and 700 iterations, respectively. By contrast, for the Lagrange-encoded VQA, using the same computational setup, the simulation time was drastically reduced to about 5 minutes, 3 minutes, and 2 hours for the periodic, Dirichlet, and Neumann cases. While simulation time does not directly reflect the actual runtime on quantum hardware, it remains a vital and fundamental aspect of algorithm development until real quantum devices become more widely available and it provides a strong indication of the circuit complexity and gate depth, which is consistent with the trends observed in Figures 7.34 and 7.35.

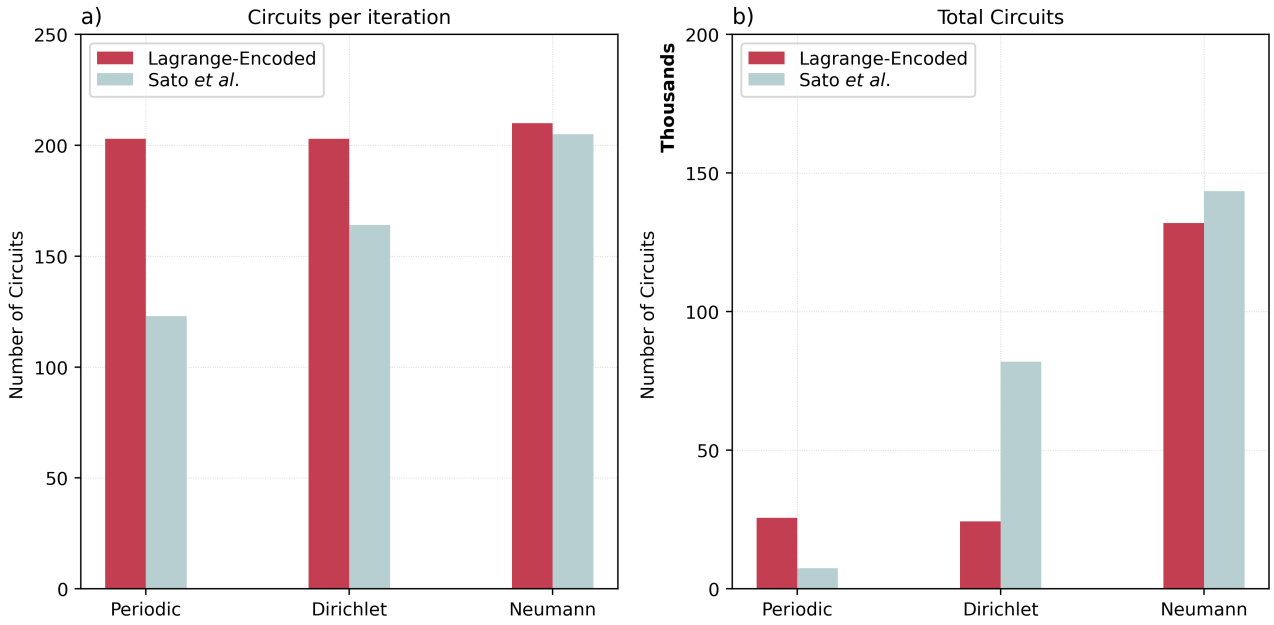


Figure 7.35: Circuit complexity estimation of the Lagrange-encoded algorithm, averaged over five attempts per boundary conditions, compared with the Sato *et al.* algorithm [5]. a) Circuits per iteration, b) Total Circuits.

Finally, considering the circuit count, we observe that the Lagrange-encoded approach tends to require a larger number of circuits per iteration. This is due to the need to evaluate the second-order derivatives of the variational quantum circuit (VQC), which are not required in the method of Sato *et al.*, as their approach relies on finite differencing. When focusing solely on the number of circuits per iteration, the two methods appear comparable. The main difference lies in the level of accuracy that can be achieved, which can be pushed significantly further with the Lagrange-encoded VQA, particularly in the case of the Neumann boundary condition.

Regarding the total number of circuits, similarly to the total number of gates, this quantity depends strongly on the number of iterations. It is shown here for completeness; however, it should be understood that, since different optimizers with different hyperparameters were employed, this metric reflects not only the characteristics of the VQCs themselves and their trainability, but also the behaviour of the full VQA, including the classical optimization.

7.4 First-order linear ODE with nonlinear coefficients

Lastly, in this last result section, we investigate the application of the Lagrange-encoded VQA to a first order differential equation with nonlinear coefficients and the eventual changes required to solve the DE.

7.4.1 Problem definition

The differential equation under consideration, introduced by Kyriienko et al. [4] and discussed in Chapter 3:

$$\frac{df}{dx} + \lambda f\ (\kappa + \tan(\lambda x)) = 0, \tag{7.22}$$

with $\lambda = 8$, $\kappa = 0.1$, and initial condition $f(0) = 1$. The corresponding analytical solution is

$$u(x) = e^{-0.8x}\cos(8x). \tag{7.23}$$

As this differential equation has nonlinear coefficients, it exhibits potentially unstable behaviour at certain points in the interval due to the $\tan(\lambda x)$ term. Specifically, $\tan(\lambda x)$ diverges when:

$$\lambda x = \frac{\pi}{2} + n\pi, \quad n \in \mathbb{Z}. \tag{7.24}$$

These points can therefore be formally identified as forbidden points of the differential equation:

$$x_f = \frac{\pi}{2\lambda} + \frac{n\pi}{\lambda}, \quad n \in \mathbb{Z}. \tag{7.25}$$

At these points, the term $f(x)\tan(\lambda x)$ would formally diverge unless the function $f(x)$ vanishes sufficiently rapidly. For the chosen analytical solution, $u(x)$, the zeros of the cosine factor coincide exactly with the singularities of $\tan(\lambda x)$, ensuring that the product $f(x)\tan(\lambda x)$ remains finite. Consequently, the solution is well-defined across the entire interval, despite the formal divergences in the differential equation.

This case illustrates a scenario where the Lagrange-encoded VQA is theoretically well-suited. The method requires the solution and its derivatives to be continuous, but it does not impose continuity on the expression of the differential equation itself, since training is performed only on a small set of points.

Furthermore, we can identify another category of notable points, namely those where the term $f(x)\tan(\lambda x)$ vanishes because $\tan(\lambda x)$ is zero. These occur at:

$$x_s = \frac{n\pi}{\lambda}, \quad n \in \mathbb{Z}. \tag{7.26}$$

At these points, the expression of the differential equation simplifies to:

$$\frac{df}{dx}(x_s) + \lambda\kappa f(x_s) = 0, \tag{7.27}$$

which admits as a solution: $v(x) = v_0 e^{-0.8x_s}$, with $v_0 = (-1)^n$, depending on the special point. However, the initial condition imposes $v_0 = 1$, leading to a contradiction.

Consequently, when selecting points to train the VQC, for a discontinuous differential equation such as this, capturing the full behaviour of the DE while keeping the training feasible requires careful prior analysis.

As discussed, three different types of points arise in solving this discontinuous DE: forbidden values, special values, and other points. For the interval $[0, 0.9]$, these are illustrated in Figure 7.36 as red, blue, and light green, respectively. The green points are not chosen randomly; they lie within the window $[\pi/6\lambda, \pi/3\lambda]$ modulo π, to remain sufficiently distant from forbidden and special points while still capturing as much of the DE behaviour as possible without being dominated by singularities.

7.4.2 Quantum resolution

The DE has been solved using a Lagrange-encoded VQA with six encoding points corresponding to the first type of Chebyshev nodes and a 7-qubit circuit. Training was conducted in three steps.

First, the VQC was trained using the Adam optimizer with a learning rate $\alpha = 0.02$, over the light green points shown in Figure 7.36, starting from a random parameter vector θ. Initial conditions

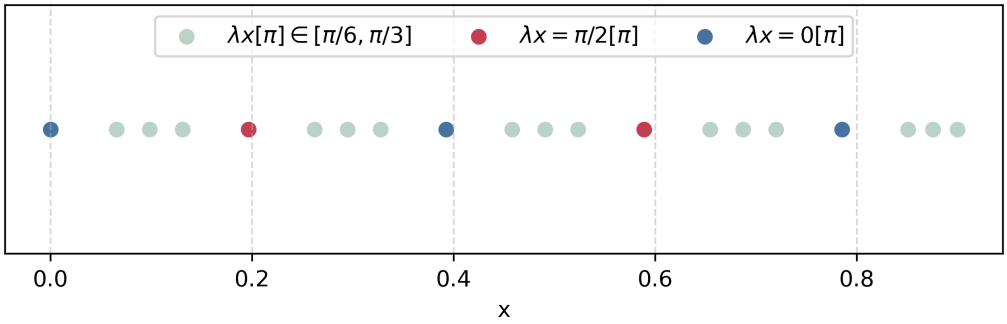


Figure 7.36: Illustration of the different types of points. Red indicates forbidden values x_f, where the DE expression diverges towards $\pm\infty$; blue indicates special values where the DE admits an different solution; green points represent training points in the window $[\pi/6\lambda, \pi/3\lambda]$ modulo π.

were incorporated into the loss function with a weight $\eta_{Cs} = 10^3$ to prioritize fitting them during early stages of training. As shown in Figure 7.37, this provides an initial good approximation, although some amplitudes where the derivative vanishes are not fully captured. These amplitudes are determined by the value of κ, which quickly becomes negligible compared to the $\tan(\lambda x)$ term.

Next, keeping the same learning rate, the VQC was trained further over the first half of the interval, using four points in the window $[\pi/6\lambda, \pi/3\lambda]$ rather than three. After convergence, these points and their corresponding function values were retained as regularization points with weight $\eta_R = 10^2$ for the subsequent step. Finally, the VQC was trained over the second half of the interval with four points in the same window and the previously defined regularization points. The evolution of the solution after each step is shown in Figure 7.37.

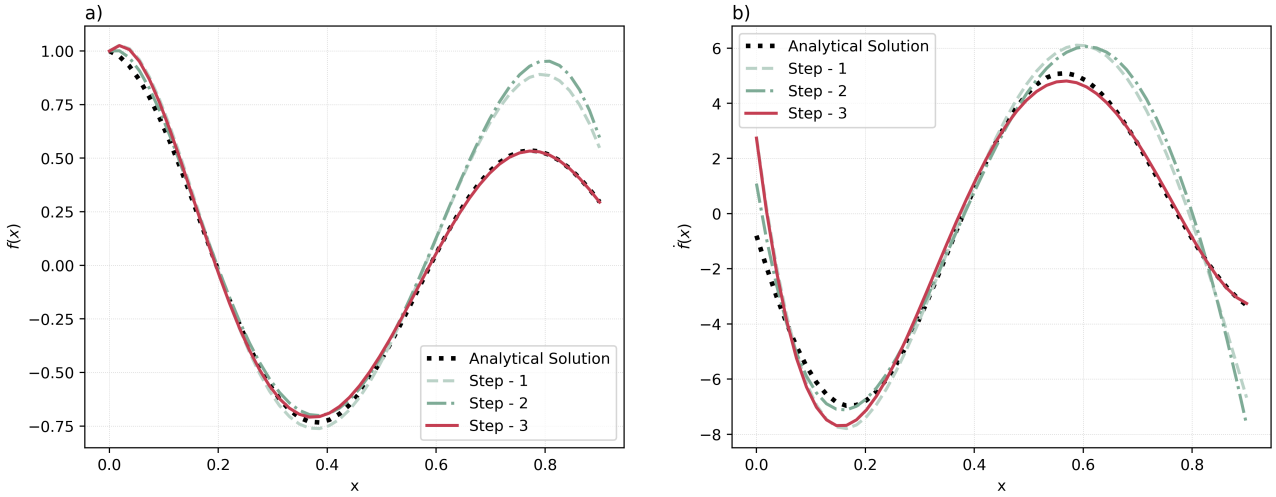


Figure 7.37: Evolution of the solution during training of the first-order discontinuous DE using a Lagrange-encoded VQA with six encoding points: a) function $f(x)$, b) first-order derivative $\dot{f}(x)$.

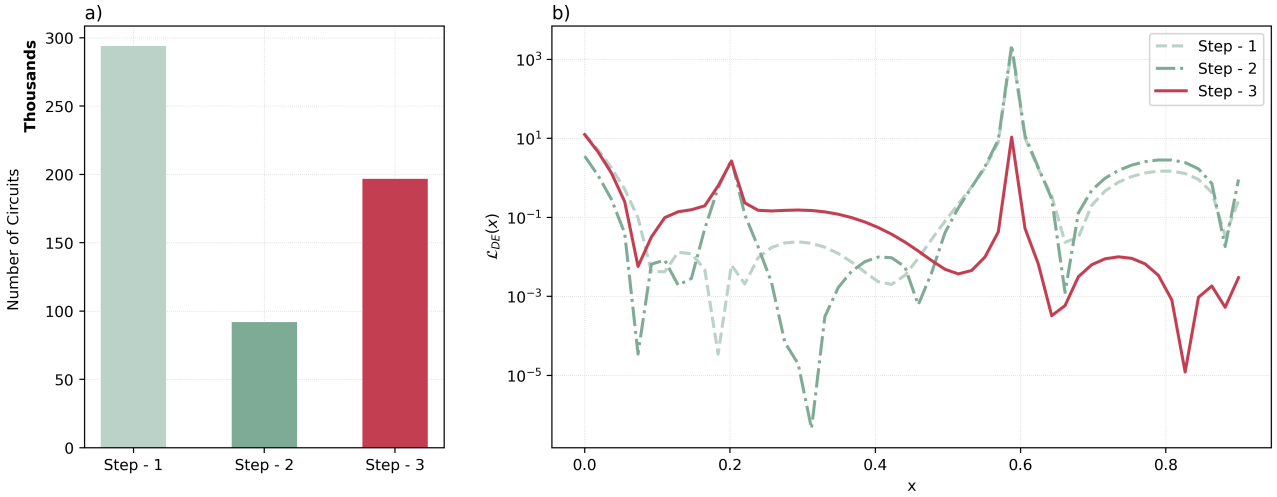


Figure 7.38: Evolution of training metrics over the three steps: a) circuit count, b) derivative loss $\mathcal{L}_{DE}$.

The choice of the training window was explored using different angles, from a single point at $\pi/4\lambda$ to a larger window starting at $\pi/12\lambda$. The best results were obtained with the current configuration. In all cases, the solution shape was captured, although the amplitude was not fully recovered due to the small influence of κ.

The three-step VQA training was repeated five times, yielding the results shown in Figure 7.39. One run performed slightly better, but overall the VQA results were not strongly sensitive to the initial random parameters.

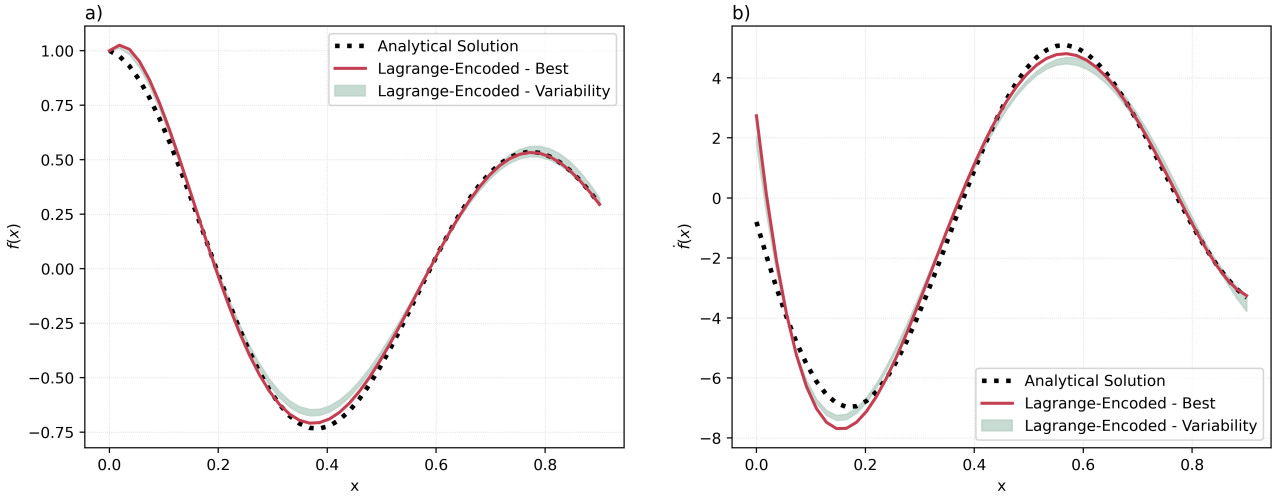


Figure 7.39: Approximation of the solution of the first-order discontinuous DE using a Lagrange-encoded VQA with six encoding points, over five runs: a) function $f(x)$, b) derivative $\dot{f}(x)$.

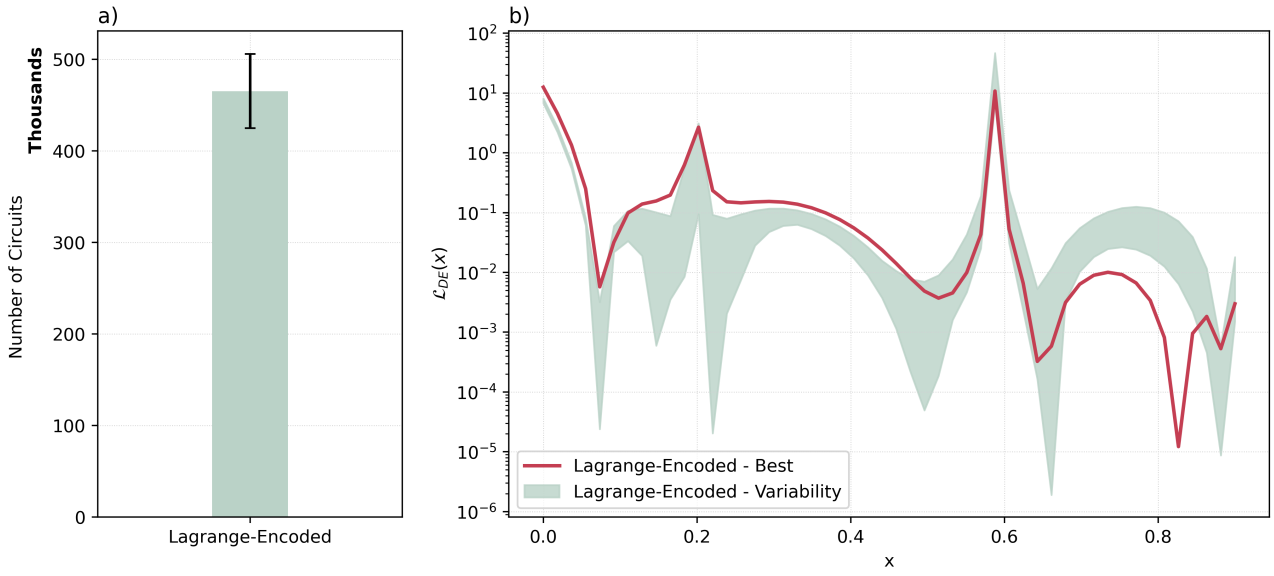


Figure 7.40: Circuit count and derivative loss for the five runs of the VQA solution: a) circuit count, b) derivative loss $\mathcal{L}_{DE}$.

Regarding the number of circuits, which here is proportional to the number of gates, we observe similar values across all runs, corresponding to similar loss values. This indicates a small influence of the initial parameters.

Overall, Figure 7.40 shows that the function approximation presents a good overlap but a small deviation at the beginning of the interval. The initial condition is not perfectly satisfied for the derivative, but the solution at the end of the interval aligns well with the analytical result. The loss plot shows decreasing error away from the initial condition, with two spikes around $x \approx 0.2$ and 0.6, corresponding to forbidden points. These spikes highlight the divergence of the DE at these points, confirming that they are unsuitable as training inputs for the VQC.

The application of the Lagrange-encoded VQA to a discontinuous differential equation, whose expression exhibits discontinuities when the function deviates from the exact solution, is highly promising. Future work will focus on evaluating the performance of different VQAs to enable a comparative assessment, as well as exploring larger system sizes to further improve accuracy. We have already observed that adding a single encoding point can drastically enhance the results, particularly when the current approximation of the function is already reasonably accurate. This additional study is not included here, as it exceeded the time allocated for this work.

7.5 Summary

In this chapter, we investigated the application of the Lagrange-encoded VQA to various differential equations: first-order, second-order, including a comparison between different VQA approaches, and first-order with nonlinear coefficients.

An interesting feature of the Lagrange-encoded VQA is its variability and its ability to provide satisfactory or even high-quality results for a range of differential equations, using very few gates and variational parameters. In contrast, other VQA approaches, while performing well, exhibited different limitations: large circuit depth and numerous variational parameters, resulting in a computationally intensive optimization process.

The Chebyshev-encoded VQA, in particular, is well-suited for oscillatory functions. We observed a tendency to oscillate, especially near the end of the interval, due to the influence of initial conditions at the beginning of the interval. Additionally, this method shows strong sensitivity to the initial random vector of variational parameters.

We also noted that the Sato *et al.* algorithm is computationally demanding and, as it was designed for a specific application, it is unclear how easily it could be applied to other differential equations, whether with completely different structures or source terms. However, in comparison to polynomial encoded VQAs, this discretized approach enables the approximation of non continuous solutions, which can be an interesting feature in engineering applications.

Finally, polynomial-encoded VQAs, and particularly the Lagrange-encoded VQA, are sensitive to the optimization process. The choice of training points and learning rate significantly influences performance. Consequently, it would be interesting to explore the behaviour of this VQA with alternative classical optimizers, as Adam may or may not be the most suitable for this approach.

Chapter 8

Conclusions and Future Work

8.1 Summary and Conclusions

Differential equations are fundamental in numerous scientific fields, including structural engineering, fluid dynamics, and financial modelling. However, their complexity often makes them challenging to solve with traditional computational methods. Recent advances in quantum computing have motivated researchers to explore quantum algorithms for solving differential equations.

This work focused on variational quantum algorithm approaches, particularly spectral methods for solving differential equations. The research was initially inspired by the work of Kyriienko *et al.* [4], and aimed to understand their approach, identify its limitations, and propose improvements. In doing so, we developed a new encoding that maintains the key elements of a polynomial-encoded VQA while offering a different paradigm and behaviour: The Lagrange-encoded VQA.

In most VQA methods proposed in the literature, the variational ansatz introduces entanglement between qubits while the encoding blocks remain shallow, as in the works of Kyriienko *et al.* [4] and Sato *et al.* [5]. The novel approach proposed in this work shifts this paradigm by incorporating entanglement directly within the encoding block, combined with a shallow variational ansatz. This design, inspired by the Hadamard test structure, enables an efficient representation of the complete set of Lagrange polynomial basis functions using a limited number of gates, and allows the use of the Hadamard test differentiation method, which reduces the computational load.

Two distinct architectures of the proposed Lagrange-encoded VQA were introduced, along with slight improvements to the encoding function to address the requirements and limitations imposed by sampling measurements in quantum computation.

VQA runtime and complexity metrics are proposed here, as no established benchmarks or widely accepted methods currently exist in the literature to quantify gate-induced noise or the runtime associated with circuit evaluations in VQAs. Challenges and limitations of VQA simulations were examined, along with strategies to reduce computational load when simulating VQC gradients, which were shown to be acceptable only in statevector simulations.

The Lagrange-encoded VQA was applied to several problems: two first-order linear ODEs alongside the original Chebyshev-encoded VQA, two second-order ODEs, the damped mass-spring system and the Poisson equation with various boundary conditions, and a first-order ODE with nonlinear coefficients. The results demonstrated that the Lagrange-encoded VQA reduces gate and circuit complexity while providing solutions of comparable or improved quality.

Both the Chebyshev-encoded and Lagrange-encoded VQAs employ meshless approaches and rely on polynomial fitting, which requires continuity in the problem definition. This contrasts with the discretized approach of Sato *et al.*, which can only solve linear or linearized DEs. Polynomial-fitting VQAs, however, do not require linearization and are thus well-suited for nonlinear DEs, or DEs with nonlinear coefficients.

These meshless approaches also distinguish between three types of points within the interval I: *training points* for gradient evaluation and parameter optimization, *evaluation points* for plotting the optimized function, and *encoding points* for the quantum feature map in the Lagrange-encoded algorithm. This independence allows flexibility in training, evolution of points, and inclusion of additional losses, such as boundary conditions.

Whereas the Chebyshev-encoded algorithm uses nonlinear polynomials, which can introduce oscillations at the interval boundaries, linear polynomial encoding with the Lagrange approach may be suboptimal for nonlinear problems. In both cases, the degree of the polynomial scales with circuit size, which will admit a limit on NISQ devices.

8.2 Future Work

8.2.1 VQA Encoding Feasibility

An important question remains: how does the algorithm perform on real quantum devices, with noise? As discussed in Chapter 6, noisy measurements introduce oscillations not observed in statevector simulations. Clever modifications, such as floating boundary handling, can mitigate these effects, but further work is needed to improve the feasibility and robustness of the VQA on real hardware.

If we target NISQ devices, the number of qubits and coherence times may be sufficient. As discussed in Chapter 6, simulations with modest noise indicate that results on a five-qubit register could be exploitable with limited error mitigation. However, a deeper analysis must address two questions. First, how does performance scale with larger circuits? Second, what is the computational cost of mitigating errors at each iteration for both the function and its derivative? As the simulated noise mimicked the behavior of a real device of rather poor quality compared to the latest hardware, it can therefore be seen as a worst-case scenario.

Studies on contemporary quantum devices would thus be invaluable for identifying the most suitable platforms for polynomial-based VQAs. If the target machines are fault-tolerant, this becomes a bottleneck. Technological progress will be required to fully benchmark the algorithm at larger scales, especially as classical simulation of quantum circuits rapidly becomes prohibitively expensive.

Additionally, tests on real quantum computers should investigate both the reduced and extended circuit structures. The extended structure uses more qubits but fewer gates, reducing gate-induced noise and decoherence, though it may be sensitive to qubit errors or cross-talk. Evaluating both structures on quantum hardware will provide insight into their practical performance.

8.2.2 Alternative Polynomials and Trainable Encoding

Chapter 5 highlighted that additional entanglement can allow the implementation of Hermite interpolation, which interpolates not only the function but also its derivatives. This approach has the potential to further increase the accuracy of the VQA solutions.

Moreover, as discussed in Chapter 4, the encoding points in the Lagrange-encoded VQA can be trained without increasing the quantum computational load. Further exploration of trainable encoding points could reduce oscillations and increase the expressivity of the VQC, enabling representation of a broader class of functions without enlarging the circuit.

8.2.3 Extended Applications

Based on the encouraging results obtained in this work, several directions for future application of polynomial-encoded VQAs are envisioned:

- **Higher-dimensional DEs:** This work focused on 1D DEs, including first and second-order engineering problems such as the damped mass-spring system and the Poisson equation. Many engineering applications involve 2D or 3D problems, so evaluating the performance of the VQA in higher dimensions and for more complex DEs (e.g., advection-diffusion, Burgers', Euler equations) is a natural next step. Additional variables could be handled with additional ancilla qubits or via coupled circuits.
- **Higher-order DEs:** Some engineering applications involve higher-order DEs, such as fourth-order equations in structural buckling analysis. Extending the VQA to solve such equations will be an important advancement.
- **Nonlinear DEs:** Nonlinear DEs are common in engineering and present a significant challenge for quantum algorithms due to the linearity of quantum evolution. Future work could investigate second-order nonlinear DEs, such as the pendulum equation with large initial angles.

Bibliography

[1] J. Hunout, S. Laizet, and L. Iannucci, "Variational quantum algorithm based on Lagrange polynomial encoding to solve differential equations," en, *Physical Review A*, vol. 111, no. 6, p. 062 404, Jun. 2025, ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.111.062404. Accessed: Jun. 3, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.111.062404.

[2] J. Hunout, S. Dylan Lovett, J. Wade, and I. Von Holstein, "Assessing the skills gap for the UK's quantum missions," en, *Nature Reviews Physics*, vol. 7, no. 4, pp. 171–173, Mar. 2025, ISSN: 2522-5820. DOI: 10.1038/s42254-025-00820-w. Accessed: Dec. 12, 2025. [Online]. Available: https://www.nature.com/articles/s42254-025-00820-w.

[3] J. Hunout, S. Dylan Lovett, J. Wade, and I. Von Holstein, "Quantum training in the UK needs better visibility," en, *Nature Reviews Physics*, Dec. 2025, ISSN: 2522-5820. DOI: 10.1038/s42254-025-00908-3. Accessed: Dec. 12, 2025. [Online]. Available: https://www.nature.com/articles/s42254-025-00908-3.

[4] O. Kyriienko, A. E. Paine, and V. E. Elfving, "Solving nonlinear differential equations with differentiable quantum circuits," *Physical Review A*, vol. 103, no. 5, pp. 1–22, May 2021, ISSN: 24699934. DOI: 10.1103/PhysRevA.103.052416. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.052416.

[5] Y. Sato et al., "Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation," *Physical Review A*, vol. 104, no. 5, pp. 1–4, Jun. 2021, ISSN: 24699934. DOI: 10.1103/PhysRevA.104.052409. [Online]. Available: http://arxiv.org/abs/2106.09333%20http://dx.doi.org/10.1103/PhysRevA.104.052409.

[6] J. Preskill, *Quantum computing and the entanglement frontier*, arXiv:1203.5813 [quant-ph], Nov. 2012. DOI: 10.48550/arXiv.1203.5813. Accessed: Jun. 1, 2025. [Online]. Available: http://arxiv.org/abs/1203.5813.

[7] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," en, *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1666-5. Accessed: Jun. 1, 2025. [Online]. Available: https://www.nature.com/articles/s41586-019-1666-5.

[8] E. Pednault, D. Maslov, J. Gunnels, and J. Gambetta, *On "quantum supremacy"*, ibm.com, Oct. 2019. Accessed: Jun. 1, 2025. [Online]. Available: https://www.ibm.com/quantum/blog/on-quantum-supremacy.

[9] C. Palacios-Berraquero, L. Mueck, and D. M. Persaud, "Instead of 'supremacy' use 'quantum advantage'," en, *Nature*, vol. 576, no. 7786, pp. 213–213, Dec. 2019, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/d41586-019-03781-0. Accessed: Jun. 1, 2025. [Online]. Available: https://www.nature.com/articles/d41586-019-03781-0.

[10] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," en, *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. Accessed: Jun. 2, 2025. [Online]. Available: http://epubs.siam.org/doi/10.1137/S0097539795293172.

[11] L. K. Grover, *A fast quantum mechanical algorithm for database search*, arXiv:quant-ph/9605043, Nov. 1996. DOI: 10.48550/arXiv.quant-ph/9605043. Accessed: Jun. 2, 2025. [Online]. Available: http://arxiv.org/abs/quant-ph/9605043.

[12] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum Algorithm for Linear Systems of Equations," en, *Physical Review Letters*, vol. 103, no. 15, p. 150 502, Oct. 2009, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.103.150502. Accessed: Jun. 2, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[13] S. Beauregard, *Circuit for Shor's algorithm using 2n+3 qubits*, arXiv:quant-ph/0205095, Feb. 2003. DOI: 10.48550/arXiv.quant-ph/0205095. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/quant-ph/0205095.

[14] T. Häner, M. Roetteler, and K. M. Svore, *Factoring using 2n+2 qubits with Toffoli based modular multiplication*, arXiv:1611.07995 [quant-ph], Jun. 2017. DOI: 10.48550/arXiv.1611.07995. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/1611.07995.

[15] C. Gidney, *Factoring with n+2 clean qubits and n-1 dirty qubits*, arXiv:1706.07884 [quant-ph], Jan. 2018. DOI: 10.48550/arXiv.1706.07884. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/1706.07884.

[16] C. Zalka, *Shor's algorithm with fewer (pure) qubits*, arXiv:quant-ph/0601097, Jan. 2006. DOI: 10.48550/arXiv.quant-ph/0601097. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/quant-ph/0601097.

[17] C. Chevignard, P.-A. Fouque, and A. Schrottenloher, "Reducing the Number of Qubits in Quantum Factoring," en, in *Advances in Cryptology – CRYPTO 2025*, Y. Tauman Kalai and S. F. Kamara, Eds., vol. 16001, Series Title: Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2025, pp. 384–415, ISBN: 978-3-032-01877-9 978-3-032-01878-6. DOI: 10.1007/978-3-032-01878-6_13. Accessed: Feb. 10, 2026. [Online]. Available: https://link.springer.com/10.1007/978-3-032-01878-6_13.

[18] C. Zalka, "Grover's quantum searching algorithm is optimal," *Physical Review A*, vol. 60, no. 4, pp. 2746–2751, Oct. 1999, arXiv:quant-ph/9711070, ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.60.2746. Accessed: Feb. 9, 2026. [Online]. Available: http://arxiv.org/abs/quant-ph/9711070.

[19] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and Weaknesses of Quantum Computing," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, Oct. 1997, arXiv:quant-ph/9701001, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539796300933. Accessed: Feb. 9, 2026. [Online]. Available: http://arxiv.org/abs/quant-ph/9701001.

[20] S. Aaronson, "Read the fine print," en, *Nature Physics*, vol. 11, no. 4, pp. 291–293, Apr. 2015, ISSN: 1745-2473, 1745-2481. DOI: 10.1038/nphys3272. Accessed: Jun. 14, 2025. [Online]. Available: https://www.nature.com/articles/nphys3272.

[21] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, eng, 10th anniversary edition. Cambridge: Cambridge university press, 2010, ISBN: 978-1-107-00217-3.

[22] E. Knill, "Quantum computing with realistically noisy devices," en, *Nature*, vol. 434, no. 7029, pp. 39–44, Mar. 2005, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature03350. Accessed: Feb. 10, 2026. [Online]. Available: https://www.nature.com/articles/nature03350.

[23] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High threshold universal quantum computation on the surface code," *Physical Review A*, vol. 80, no. 5, p. 052 312, Nov. 2009, arXiv:0803.0272 [quant-ph], ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.80.052312. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/0803.0272.

[24] Google Quantum AI and Collaborators et al., "Quantum error correction below the surface code threshold," en, *Nature*, vol. 638, no. 8052, pp. 920–926, Feb. 2025, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-024-08449-y. Accessed: Feb. 10, 2026. [Online]. Available: https://www.nature.com/articles/s41586-024-08449-y.

[25] B. M. Terhal, "Quantum error correction for quantum memories," *Reviews of Modern Physics*, vol. 87, no. 2, pp. 307–346, 2015.

[26] J. Roffe, "Quantum error correction: An introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019.

[27] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," en, *Applied Physics Reviews*, vol. 6, no. 2, p. 021 314, Jun. 2019, ISSN: 1931-9401. DOI: 10.1063/1.5088164. Accessed: Aug. 19, 2025. [Online]. Available: https://pubs.aip.org/apr/article/6/2/021314/570103/Trapped-ion-quantum-computing-Progress-and.

[28] QuEra, *Quantum Decoherence*. Accessed: Aug. 18, 2025. [Online]. Available: https://www.quera.com/glossary/quantum-decoherence.

[29] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032 324, 2012.

[30] J. Preskill, "Reliable quantum computers," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 385–410, 1998.

[31] G. Q. A. Collaboration, "Demonstration of fault-tolerant logical qubits with surface codes," *Nature*, vol. 614, pp. 676–681, 2023.

[32] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Physical Review Letters*, vol. 119, no. 18, p. 180 509, 2017.

[33] Y. Li and S. C. Benjamin, "Efficient Variational Quantum Simulator Incorporating Active Error Minimization," en, *Physical Review X*, vol. 7, no. 2, p. 021 050, Jun. 2017, ISSN: 2160-3308. DOI: 10.1103/PhysRevX.7.021050. Accessed: Aug. 18, 2025. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevX.7.021050.

[34] A. Kandala et al., "Error mitigation extends the computational reach of a noisy quantum processor," *Nature*, vol. 567, no. 7749, pp. 491–495, 2019.

[35] Z. Cai et al., "Quantum error mitigation," en, *Reviews of Modern Physics*, vol. 95, no. 4, p. 045 005, Dec. 2023, ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.95.045005. Accessed: Aug. 18, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.95.045005.

[36] I. L. Chuang et al., "Experimental realization of a quantum algorithm," *Nature*, vol. 393, no. 6681, pp. 143–146, May 1998, arXiv:quant-ph/9801037, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/30181. Accessed: Jun. 16, 2025. [Online]. Available: http://arxiv.org/abs/quant-ph/9801037.

[37] F. Tennie, S. Laizet, S. Lloyd, and L. Magri, "Quantum computing for nonlinear differential equations and turbulence," en, *Nature Reviews Physics*, vol. 7, no. 4, pp. 220–230, Jan. 2025, ISSN: 2522-5820. DOI: 10.1038/s42254-024-00799-w. Accessed: Aug. 18, 2025. [Online]. Available: https://www.nature.com/articles/s42254-024-00799-w.

[38] P. Krantz et al., "A quantum engineer's guide to superconducting qubits," en, *Applied Physics Reviews*, vol. 6, no. 2, p. 021 318, Jun. 2019, ISSN: 1931-9401. DOI: 10.1063/1.5089550. Accessed: Aug. 19, 2025. [Online]. Available: https://pubs.aip.org/apr/article/6/2/021318/570326/A-quantum-engineer-s-guide-to-superconducting.

[39] K. Wintersperger et al., "Neutral atom quantum computing hardware: Performance and end-user perspective," en, *EPJ Quantum Technology*, vol. 10, no. 1, p. 32, Dec. 2023, ISSN: 2662-4400, 2196-0763. DOI: 10.1140/epjqt/s40507-023-00190-1. Accessed: Aug. 19, 2025. [Online]. Available: https://epjquantumtechnology.springeropen.com/articles/10.1140/epjqt/s40507-023-00190-1.

[40] Google Quantum AI, *Meet Willow, our state-of-the-art quantum chip*. Accessed: Aug. 19, 2025. [Online]. Available: https://blog.google/technology/research/google-willow-quantum-chip/.

[41] Y.-Y. Jiang et al., "Advancements in superconducting quantum computing," en, *National Science Review*, vol. 12, no. 8, nwaf246, Jul. 2025, ISSN: 2095-5138, 2053-714X. DOI: 10.1093/nsr/nwaf246. Accessed: Aug. 19, 2025. [Online]. Available: https://academic.oup.com/nsr/article/doi/10.1093/nsr/nwaf246/8165689.

[42] IBM, *Quantum Roadmap 2033*. [Online]. Available: https://www.ibm.com/quantum/blog/quantum-roadmap-2033.

[43] D. Castelvecchi, "IBM releases first-ever 1,000-qubit quantum chip," *Nature*, vol. 624, no. 7991, p. 238, 2023, ISSN: 14764687. DOI: 10.1038/d41586-023-03854-1.

[44] Quantinuum, *System Model H2*. [Online]. Available: https://www.quantinuum.com/products-solutions/quantinuum-systems/system-model-h2.

[45] *Forte Entreprise*. [Online]. Available: https://ionq.com/quantum-systems/forte-enterprise.

[46] Pasqal, *Pasqal Roadmap*. [Online]. Available: https://www.pasqal.com/technology/roadmap/.

[47] Atom Computing, *Atom Computing AC1000*. [Online]. Available: https://atom-computing.com/ac1000/.

[48] I. Thorvaldson et al., "Grover's algorithm in a four-qubit silicon processor above the fault-tolerant threshold," *Nature Nanotechnology*, vol. 20, no. 4, pp. 472–477, Apr. 2025, ISSN: 1748-3395. DOI: 10.1038/s41565-024-01853-5. [Online]. Available: https://doi.org/10.1038/s41565-024-01853-5.

[49] Intel. [Online]. Available: https://www.intel.com/content/www/us/en/research/quantum-computing.html.

[50] H. Aghaee Rad et al., "Scaling and networking a modular photonic quantum computer," en, *Nature*, vol. 638, no. 8052, pp. 912–919, Feb. 2025, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-024-08406-9. Accessed: Feb. 10, 2026. [Online]. Available: https://www.nature.com/articles/s41586-024-08406-9.

[51] QuEra, *Photonic Qubits*. [Online]. Available: https://www.quera.com/glossary/photonic-qubits.

[52] M. Tremba, P. Hovland, and J. Liu, *Is Circuit Depth Accurate for Comparing Quantum Circuit Runtimes?* arXiv:2505.16908 [quant-ph], Jul. 2025. DOI: 10.48550/arXiv.2505.16908. Accessed: Aug. 19, 2025. [Online]. Available: http://arxiv.org/abs/2505.16908.

[53] A. Opremcak et al., "High-Fidelity Measurement of a Superconducting Qubit Using an On-Chip Microwave Photon Counter," en, *Physical Review X*, vol. 11, no. 1, p. 011 027, Feb. 2021, ISSN: 2160-3308. DOI: 10.1103/PhysRevX.11.011027. Accessed: Aug. 19, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.11.011027.

[54] S. L. Todaro et al., "State Readout of a Trapped Ion Qubit Using a Trap-Integrated Superconducting Photon Detector," en, *Physical Review Letters*, vol. 126, no. 1, p. 010 501, Jan. 2021, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.126.010501. Accessed: Aug. 19, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.126.010501.

[55] C. N. Mude et al., *Enabling Fast and Accurate Neutral Atom Readout through Image Denoising*, arXiv:2510.25982 [quant-ph], Dec. 2025. DOI: 10.48550/arXiv.2510.25982. Accessed: Feb. 10, 2026. [Online]. Available: http://arxiv.org/abs/2510.25982.

[56] K. Phalak, A. Chatterjee, and S. Ghosh, *Quantum Random Access Memory For Dummies*, arXiv:2305.01178 [quant-ph], May 2023. DOI: 10.48550/arXiv.2305.01178. Accessed: Aug. 19, 2025. [Online]. Available: http://arxiv.org/abs/2305.01178.

[57] O. D. Matteo, V. Gheorghiu, and M. Mosca, "Fault-Tolerant Resource Estimation of Quantum Random-Access Memories," *IEEE Transactions on Quantum Engineering*, vol. 1, no. 1, pp. 1–13, 2021, ISSN: 2689-1808. DOI: 10.1109/tqe.2020.2965803.

[58] P. Mukhopadhyay, "A quantum random access memory (QRAM) using a polynomial encoding of binary strings," en, *Scientific Reports*, vol. 15, no. 1, p. 11 002, Mar. 2025, ISSN: 2045-2322. DOI: 10.1038/s41598-025-95283-5. Accessed: Aug. 19, 2025. [Online]. Available: https://www.nature.com/articles/s41598-025-95283-5.

[59] Z. Wang, H. Qiao, A. N. Cleland, and L. Jiang, "Quantum Random Access Memory with Transmon-Controlled Phonon Routing," en, *Physical Review Letters*, vol. 134, no. 21, p. 210 601, May 2025, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.134.210601. Accessed:

Aug. 19, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.134.210601.

[60] D. K. Park, F. Petruccione, and J.-K. K. Rhee, "Circuit-Based Quantum Random Access Memory for Classical Data," en, *Scientific Reports*, vol. 9, no. 1, p. 3949, Mar. 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-40439-3. Accessed: Aug. 19, 2025. [Online]. Available: https://www.nature.com/articles/s41598-019-40439-3.

[61] V. Giovannetti, S. Lloyd, and L. MacCone, "Quantum random access memory," *Physical Review Letters*, vol. 100, no. 16, pp. 1–4, 2008, ISSN: 00319007. DOI: 10.1103/PhysRevLett.100.160501.

[62] K. Phalak, J. Li, and S. Ghosh, *Approximate Quantum Random Access Memory Architectures*, Version Number: 2, 2022. DOI: 10.48550/ARXIV.2210.14804. Accessed: Aug. 19, 2025. [Online]. Available: https://arxiv.org/abs/2210.14804.

[63] M. B. Hafeez and M. Krawczuk, "A Review: Applications of the Spectral Finite Element Method," en, *Archives of Computational Methods in Engineering*, vol. 30, no. 5, pp. 3453–3465, Jun. 2023, ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-023-09911-2. Accessed: Aug. 30, 2025. [Online]. Available: https://link.springer.com/10.1007/s11831-023-09911-2.

[64] A. T. Patera, "A spectral element method for fluid dynamics: Laminar flow in a channel expansion," en, *Journal of Computational Physics*, vol. 54, no. 3, pp. 468–488, Jun. 1984, ISSN: 00219991. DOI: 10.1016/0021-9991(84)90128-1. Accessed: Aug. 30, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0021999184901281.

[65] D. Siorikis, C. Rekatsinas, N. Chrysochoidis, and D. Saravanos, "A cubic spline layerwise spectral finite element for robust stress predictions in laminated composite and sandwich strips," en, *European Journal of Mechanics - A/Solids*, vol. 91, p. 104 362, Jan. 2022, ISSN: 09977538. DOI: 10.1016/j.euromechsol.2021.104362. Accessed: Aug. 30, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0997753821001285.

[66] D. Siorikis, C. Rekatsinas, N. Chrysochoidis, and D. Saravanos, "An extended layerwise spectral finite element framework for delamination growth simulation in laminated composite strips," en, *Composite Structures*, vol. 276, p. 114 452, Nov. 2021, ISSN: 02638223. DOI: 10.1016/j.compstruct.2021.114452. Accessed: Aug. 30, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0263822321009144.

[67] M. R. Machado, L. Khalij, and A. T. Fabro, "Dynamic Analysis of a Composite Structure under Random Excitation Based on the Spectral Element Method," en, *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 20, no. 2, pp. 179–190, Apr. 2019, ISSN: 2191-0294, 1565-1339. DOI: 10.1515/ijnsns-2018-0050. Accessed: Aug. 30, 2025. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/ijnsns-2018-0050/html.

[68] R. Khodayi-Mehr and M. M. Zavlanos, *VarNet: Variational Neural Networks for the Solution of Partial Differential Equations*, arXiv:1912.07443 [cs], Dec. 2019. DOI: 10.48550/arXiv.1912.07443. Accessed: Aug. 20, 2025. [Online]. Available: http://arxiv.org/abs/1912.07443.

[69] B. Meuris, S. Qadeer, and P. Stinis, "Machine-learning-based spectral methods for partial differential equations," en, *Scientific Reports*, vol. 13, no. 1, p. 1739, Jan. 2023, ISSN: 2045-2322. DOI: 10.1038/s41598-022-26602-3. Accessed: Aug. 20, 2025. [Online]. Available: https://www.nature.com/articles/s41598-022-26602-3.

[70] G. E. Karniadakis et al., "Physics-informed machine learning," en, *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, May 2021, ISSN: 2522-5820. DOI: 10.1038/s42254-021-00314-5. Accessed: Feb. 16, 2026. [Online]. Available: https://www.nature.com/articles/s42254-021-00314-5.

[71] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," en, *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 477–508, Jan. 2020, ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev-fluid-010719-060214. Accessed: Feb. 16, 2026. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-fluid-010719-060214.

[72] K. Fukagata and K. Fukami, "Compressing fluid flows with nonlinear machine learning: Mode decomposition, latent modeling, and flow control," *Fluid Dynamics Research*, vol. 57, no. 4, p. 041 401, Aug. 2025, ISSN: 0169-5983, 1873-7005. DOI: 10.1088/1873-7005/ade8a2. Accessed: Feb. 16, 2026. [Online]. Available: https://iopscience.iop.org/article/10.1088/1873-7005/ade8a2.

[73] H. Kim, J. Kim, S. Won, and C. Lee, "Unsupervised deep learning for super-resolution reconstruction of turbulence," en, *Journal of Fluid Mechanics*, vol. 910, A29, Mar. 2021, ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2020.1028. Accessed: Feb. 16, 2026. [Online]. Avail-

able: https://www.cambridge.org/core/product/identifier/S0022112020010289/type/journal_article.

[74] P. Gunnarson et al., "Learning efficient navigation in vortical flow fields," en, *Nature Communications*, vol. 12, no. 1, p. 7143, Dec. 2021, ISSN: 2041-1723. DOI: 10.1038/s41467-021-27015-y. Accessed: Feb. 16, 2026. [Online]. Available: https://www.nature.com/articles/s41467-021-27015-y.

[75] K. Taira, G. Rigas, and K. Fukami, "Machine learning in fluid dynamics: A critical assessment," en, *Physical Review Fluids*, vol. 10, no. 9, p. 090 701, Sep. 2025, ISSN: 2469-990X. DOI: 10.1103/8t52-mtb9. Accessed: Feb. 16, 2026. [Online]. Available: https://link.aps.org/doi/10.1103/8t52-mtb9.

[76] J. Biamonte et al., "Quantum machine learning," en, *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature23474. Accessed: Feb. 16, 2026. [Online]. Available: https://www.nature.com/articles/nature23474.

[77] D. Jaksch, P. Givi, A. J. Daley, and T. Rung, "Variational Quantum Algorithms for Computational Fluid Dynamics," *AIAA Journal*, vol. 61, no. 5, pp. 1885–1894, 2023, ISSN: 1533385X. DOI: 10.2514/1.J062426.

[78] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," in May 2002, pp. 53–74. DOI: 10.1090/conm/305/05215. [Online]. Available: http://arxiv.org/abs/quant-ph/0005055%20http://dx.doi.org/10.1090/conm/305/05215%20http://www.ams.org/conm/305/.

[79] B. Kacewicz, "Almost optimal solution of initial-value problems by randomized and quantum algorithms," *Journal of Complexity*, vol. 22, no. 5, pp. 676–690, Oct. 2006, ISSN: 10902708. DOI: 10.1016/j.jco.2006.03.001. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0885064X06000148.

[80] F. Gaitan, "Finding flows of a Navier–Stokes fluid through quantum computing," *npj Quantum Information*, vol. 6, no. 1, 2020, ISSN: 20566387.

[81] F. Oz, R. K. Vuppala, K. Kara, and F. Gaitan, "Solving Burgers' equation with quantum computing," *Quantum Information Processing*, vol. 21, no. 1, pp. 1–13, 2022, ISSN: 15731332.

DOI: 10.1007/s11128-021-03391-8. [Online]. Available: https://doi.org/10.1007/s11128-021-03391-8.

[82] F. Oz, O. San, and K. Kara, "An efficient quantum partial differential equation solver with chebyshev points," *Scientific Reports*, vol. 13, no. 1, p. 7767, 2023.

[83] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, pp. 1–4, 2009, ISSN: 00319007. DOI: 10.1103/PhysRevLett.103.150502.

[84] S. K. Leyton and T. J. Osborne, *A quantum algorithm to solve nonlinear differential equations*, Dec. 2008. [Online]. Available: http://arxiv.org/abs/0812.4423.

[85] D. W. Berry, "High-order quantum algorithm for solving linear differential equations," *Journal of Physics A: Mathematical and Theoretical*, vol. 47, no. 10, 2014, ISSN: 17518113.

[86] Y. Cao et al., "Quantum algorithm and circuit design solving the Poisson equation," *New Journal of Physics*, vol. 15, no. 1, 2013, 013021.

[87] P. C. Costa, S. Jordan, and A. Ostrander, "Quantum algorithm for simulating the wave equation," *Physical Review A*, vol. 99, no. 1, 2019, 012323.

[88] J.-W. Wang et al., "Integrated photonic quantum technologies," *Nature Photonics*, vol. 14, pp. 273–284, 2020.

[89] A. M. Childs, J.-P. Liu, and A. Ostrander, "High-precision quantum algorithms for partial differential equations," *Quantum*, vol. 5, 2021, 574.

[90] B. D. Clader, B. C. Jacobs, and C. R. Sprouse, "Preconditioned quantum linear system algorithm," *Physical Review Letters*, vol. 110, no. 25, 2013, 250504.

[91] A. Montanaro and S. Pallister, "Quantum algorithms and the finite element method," *Physical Review A*, vol. 93, no. 3, p. 032 324, Mar. 2016, ISSN: 24699934. DOI: 10.1103/PhysRevA.93.032324. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.93.032324.

[92] D. W. Berry, A. M. Childs, A. Ostrander, and G. Wang, "Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision," *Communications in Mathematical Physics*, vol. 356, no. 3, pp. 1057–1081, 2017, ISSN: 14320916. DOI: 10.1007/s00220-017-3002-y.

[93] D. W. Berry and P. C. S. Costa, "Quantum algorithm for time-dependent differential equations using Dyson series," en, *Quantum*, vol. 8, p. 1369, Jun. 2024, ISSN: 2521-327X. DOI: 10.22331/q-2024-06-13-1369. Accessed: Aug. 31, 2025. [Online]. Available: https://quantum-journal.org/papers/q-2024-06-13-1369/.

[94] A. M. Childs, R. Kothari, and R. D. Somma, "Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision," en, *SIAM Journal on Computing*, vol. 46, no. 6, pp. 1920–1950, Jan. 2017, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/16M1087072. Accessed: Aug. 31, 2025. [Online]. Available: https://epubs.siam.org/doi/10.1137/16M1087072.

[95] A. M. Childs and J. P. Liu, "Quantum Spectral Methods for Differential Equations," *Communications in Mathematical Physics*, vol. 375, no. 2, pp. 1427–1457, 2020, ISSN: 14320916. DOI: 10.1007/s00220-020-03699-z.

[96] M. Lubasch et al., "Variational quantum algorithms for nonlinear problems," *Phys. Rev. A*, vol. 101, p. 010 301, 1 2020.

[97] A. Sarma et al., "Quantum variational solving of nonlinear and multi-dimensional partial differential equations," *arXiv preprint arXiv:2311.01531*, 2023.

[98] H. Krovi, "Improved quantum algorithms for linear and nonlinear differential equations," en, *Quantum*, vol. 7, p. 913, Feb. 2023, ISSN: 2521-327X. DOI: 10.22331/q-2023-02-02-913. Accessed: May 28, 2025. [Online]. Available: https://quantum-journal.org/papers/q-2023-02-02-913/.

[99] H.-C. Wu, J. Wang, and X. Li, "Quantum Algorithms for Nonlinear Dynamics: Revisiting Carleman Linearization with No Dissipative Conditions," en, *SIAM Journal on Scientific Computing*, vol. 47, no. 2, A943–A970, Apr. 2025, ISSN: 1064-8275, 1095-7197. DOI: 10.1137/24M1665799. Accessed: May 28, 2025. [Online]. Available: https://epubs.siam.org/doi/10.1137/24M1665799.

[100] A. Surana, A. Gnanasekaran, and T. Sahai, "An efficient quantum algorithm for simulating polynomial dynamical systems," en, *Quantum Information Processing*, vol. 23, no. 3, p. 105, Mar. 2024, ISSN: 1573-1332. DOI: 10.1007/s11128-024-04311-2. Accessed: May 28, 2025. [Online]. Available: https://link.springer.com/10.1007/s11128-024-04311-2.

[101] A. Peruzzo et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. May, 2014, ISSN: 20411723. DOI: 10.1038/ncomms5213.

[102] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, 2016, ISSN: 13672630.

[103] J. Tilly et al., *The Variational Quantum Eigensolver: A review of methods and best practices.* 2022, vol. 986, pp. 1–128, ISBN: 0000000167. DOI: 10.1016/j.physrep.2022.08.003. [Online]. Available: http://arxiv.org/abs/2111.05176.

[104] C. Bravo-Prieto et al., "Variational Quantum Linear Solver," *Quantum*, vol. 7, 2023, ISSN: 2521327X.

[105] H.-L. Liu et al., "Variational quantum algorithm for the Poisson equation," en, *Physical Review A*, vol. 104, no. 2, p. 022 418, Aug. 2021, ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.104.022418. Accessed: Aug. 31, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.022418.

[106] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, pp. 1–3, 2018, ISSN: 24699934. DOI: 10.1103/PhysRevA.98.032309. [Online]. Available: doi:10.1103/PhysRevA.98.032309%20url:https://doi.org.

[107] A. J. Pool, A. D. Somoza, M. Lubasch, and B. Horstmann, "Solving Partial Differential Equations using a Quantum Computer," *Proceedings - 2022 IEEE International Conference on Quantum Computing and Engineering, QCE 2022*, pp. 864–866, 2022. DOI: 10.1109/QCE53715.2022.00146.

[108] M. Cerezo et al., "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature Communications*, vol. 12, no. 1, pp. 1–39, 2021, ISSN: 20411723.

[109] V. Heyraud et al., "Efficient Estimation of Trainability for Variational Quantum Circuits," *PRX Quantum*, vol. 4, no. 4, p. 1, 2023, ISSN: 26913399.

[110] M. Cerezo et al., "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021, ISSN: 25225820. DOI: 10.1038/s42254-021-00348-9.

[111] L. Bittel and M. Kliesch, "Training Variational Quantum Algorithms Is NP-Hard," *Physical Review Letters*, vol. 127, no. 12, 2021, ISSN: 10797114.

[112] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/1412.6980.

[113] D. Wierichs, C. Gogolin, and M. Kastoryano, "Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer," *Physical Review Research*, vol. 2, no. 4, p. 043 246, Nov. 2020, arXiv:2004.14666 [quant-ph], ISSN: 2643-1564. DOI: 10.1103/PhysRevResearch.2.043246. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2004.14666.

[114] O. Lockwood, *An Empirical Review of Optimization Techniques for Quantum Variational Circuits*, arXiv:2202.01389 [quant-ph], Feb. 2022. DOI: 10.48550/arXiv.2202.01389. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2202.01389.

[115] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992, ISSN: 00189286. DOI: 10.1109/9.119632. [Online]. Available: http://ieeexplore.ieee.org/document/119632/.

[116] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione, "A comparison of various classical optimizers for a variational quantum linear solver," *Quantum Information Processing*, vol. 20, no. 6, pp. 1–14, 2021, ISSN: 15731332. DOI: 10.1007/s11128-021-03140-x. [Online]. Available: https://doi.org/10.1007/s11128-021-03140-x.

[117] M. J. D. Powell, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," in *Advances in Optimization and Numerical Analysis*, Dordrecht: Springer Netherlands, 1994, pp. 51–67. DOI: 10.1007/978-94-015-8330-5{_}4. [Online]. Available: http://link.springer.com/10.1007/978-94-015-8330-5_4.

[118] M. Wilson et al., "Optimizing quantum heuristics with meta-learning," *Quantum Machine Intelligence*, vol. 3, no. 1, pp. 1–13, 2021, ISSN: 25244914. DOI: 10.1007/s42484-020-00022-w.

[119] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles, "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms," *Quantum*, vol. 4, p. 263, May 2020, ISSN: 2521-327X. DOI: 10.22331/q-2020-05-11-263. [Online]. Available: https://quantum-journal.org/papers/q-2020-05-11-263/.

[120] R. Sweke et al., "Stochastic gradient descent for hybrid quantum-classical optimization," *Quantum*, vol. 4, 2020, ISSN: 2521327X. DOI: 10.22331/q-2020-08-31-314. arXiv: 1910.01155.

[121] K. M. Nakanishi, K. Fujii, and S. Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms," *Physical Review Research*, vol. 2, no. 4, p. 043 158, Oct. 2020, ISSN: 2643-1564. DOI: 10.1103/PhysRevResearch.2.043158. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.043158.

[122] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, "Quantum Natural Gradient," *Quantum*, vol. 4, p. 269, May 2020, ISSN: 2521-327X. DOI: 10.22331/q-2020-05-25-269. [Online]. Available: https://quantum-journal.org/papers/q-2020-05-25-269/.

[123] B. Koczor and S. C. Benjamin, "Quantum natural gradient generalized to noisy and nonunitary circuits," *Physical Review A*, vol. 106, no. 6, p. 062 416, Dec. 2022, ISSN: 2469-9926. DOI: 10.1103/PhysRevA.106.062416. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.106.062416.

[124] X. Bonet-Monroig et al., "Performance comparison of optimization methods on variational quantum algorithms," en, *Physical Review A*, vol. 107, no. 3, p. 032 407, Mar. 2023, ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.107.032407. Accessed: Aug. 25, 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.107.032407.

[125] J. Larson, M. Menickelly, and J. Shi, "A Novel Noise-Aware Classical Optimizer for Variational Quantum Algorithms," *INFORMS Journal on Computing*, vol. 37, no. 1, pp. 63–85, Jan. 2025, arXiv:2401.10121 [quant-ph], ISSN: 1091-9856, 1526-5528. DOI: 10.1287/ijoc.2024.0578. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2401.10121.

[126] C. Moussa et al., "Resource frugal optimizer for quantum machine learning," *Quantum Science and Technology*, vol. 8, no. 4, p. 045 019, Oct. 2023, ISSN: 2058-9565. DOI: 10.1088/2058-9565/acef55. Accessed: Aug. 25, 2025. [Online]. Available: https://iopscience.iop.org/article/10.1088/2058-9565/acef55.

[127] D. Fitzek, R. S. Jonsson, W. Dobrautz, and C. Schäfer, "Optimizing Variational Quantum Algorithms with qBang: Efficiently Interweaving Metric and Momentum to Navigate Flat Energy Landscapes," *Quantum*, vol. 8, p. 1313, Apr. 2024, arXiv:2304.13882 [quant-ph], ISSN: 2521-327X. DOI: 10.22331/q-2024-04-09-1313. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2304.13882.

[128] J. R. McClean et al., "Barren plateaus in quantum neural network training landscapes," *Nature Communications*, vol. 9, no. 1, pp. 1–6, 2018, ISSN: 20411723.

[129] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," *Quantum*, vol. 3, p. 214, Dec. 2019, ISSN: 2521-327X. DOI: 10.22331/q-2019-12-09-214. [Online]. Available: https://quantum-journal.org/papers/q-2019-12-09-214/.

[130] H.-Y. Liu et al., "Mitigating barren plateaus with transfer-learning-inspired parameter initializations," *New Journal of Physics*, vol. 25, no. 1, p. 013 039, Jan. 2023, ISSN: 1367-2630. DOI: 10.1088/1367-2630/acb58e. [Online]. Available: https://quantum-journal.org/papers/q-2019-12-09-214/%20https://iopscience.iop.org/article/10.1088/1367-2630/acb58e.

[131] L. Friedrich and J. Maziero, "Avoiding barren plateaus with classical deep neural networks," *Physical Review A*, vol. 106, no. 4, p. 042 433, Oct. 2022, ISSN: 2469-9926. DOI: 10.1103/PhysRevA.106.042433. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.106.042433.

[132] C.-Y. Park and N. Killoran, "Hamiltonian variational ansatz without barren plateaus," *Quantum*, vol. 8, p. 1239, Feb. 2024, ISSN: 2521-327X. DOI: 10.22331/q-2024-02-01-1239. [Online]. Available: https://quantum-journal.org/papers/q-2024-02-01-1239/.

[133] J. Lee, W. J. Huggins, M. Head-Gordon, and K. B. Whaley, "Generalized Unitary Coupled Cluster Wave functions for Quantum Computation," *Journal of Chemical Theory and Computation*, vol. 15, no. 1, pp. 311–324, Jan. 2019, ISSN: 1549-9618. DOI: 10.1021/acs.jctc.8b01004. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jctc.8b01004.

[134] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043 001, Nov. 2019, ISSN: 20589565. DOI: 10.1088/2058-9565/ab4eb5. [Online]. Available: https://iopscience.iop.org/article/10.1088/2058-9565/ab4eb5.

[135] A. Kandala et al., "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," en, *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature23879. Accessed: Aug. 5, 2025. [Online]. Available: https://www.nature.com/articles/nature23879.

[136] L. Leone, S. F. E. Oliviero, L. Cincio, and M. Cerezo, "On the practical usefulness of the Hardware Efficient Ansatz," *Quantum*, vol. 8, p. 1395, Jul. 2024, arXiv:2211.01477 [quant-ph], ISSN: 2521-327X. DOI: 10.22331/q-2024-07-03-1395. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2211.01477.

[137] Quantinuum Ltd. (organization), *Hardware Efficient Ansatz*, V5.0.0. Accessed: Aug. 5, 2025. [Online]. Available: https://docs.quantinuum.com/inquanto/manual/ansatze/hea.html.

[138] M. Wang, P. Das, and P. J. Nair, "Qoncord: A Multi-Device Job Scheduling Framework for Variational Quantum Algorithms," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, arXiv:2409.12432 [quant-ph], Nov. 2024, pp. 735–749. DOI: 10.1109/MICRO61859.2024.00060. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2409.12432.

[139] IBM Quantum Platform, *Run jobs in a session*. [Online]. Available: https://quantum.cloud.ibm.com/docs/en/guides/run-jobs-session.

[140] q-ctrl, *Iterative and batch workloads*. [Online]. Available: https://docs.q-ctrl.com/fire-opal/execute/submit-jobs/iterative-and-batch-workloads.

[141] IonQ, *The value of Classical Quantum Simulators*. [Online]. Available: https://ionq.com/resources/the-value-of-classical-quantum-simulators.

[142] V. Kumar et al., *Towards secondary structure prediction of longer mRNA sequences using a quantum-centric optimization scheme*, arXiv:2505.05782 [quant-ph], May 2025. DOI: 10.48550/arXiv.2505.05782. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2505.05782.

[143] M. Morita, Y. Tomita, J. Koyama, and K. Kimura, "Simulator Demonstration of Large Scale Variational Quantum Algorithm on HPC Cluster," *IEEE Access*, vol. 12, pp. 85 219–85 230, 2024, arXiv:2402.11878 [quant-ph], ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3415168. Accessed: Aug. 25, 2025. [Online]. Available: http://arxiv.org/abs/2402.11878.

[144] A. Kandala et al., "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017, ISSN: 14764687. DOI: 10.1038/nature23879. [Online]. Available: http://www.nature.com/articles/nature23879.

[145] B. Jaderberg et al., "Let quantum neural networks choose their own frequencies," en, *Physical Review A*, vol. 109, no. 4, p. 042 421, Apr. 2024, ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.109.042421. Accessed: Feb. 26, 2026. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.109.042421.

[146] G. Dahlquist and A. Bjork, "Equidistant interpolation and the Runge phenomenon," in *Numerical Methods*, Englewood Cliffs, N.J., Prentice-Hall, 1974, pp. 101–103, ISBN: 0-13-627315-7.

[147] Qiskit, *AerSimulator*. [Online]. Available: https://qiskit.github.io/qiskit-aer/stubs/qiskit_aer.AerSimulator.html.

[148] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Science and Technology*, vol. 5, no. 4, p. 044 007, Oct. 2020, ISSN: 2058-9565. DOI: 10.1088/2058-9565/abae7d. Accessed: Aug. 26, 2025. [Online]. Available: https://iopscience.iop.org/article/10.1088/2058-9565/abae7d.

[149] K. Karuppasamy, V. Puram, S. Johnson, and J. P. Thomas, "A Comprehensive Review of Quantum Circuit Optimization: Current Trends and Future Directions," en, *Quantum Reports*, vol. 7, no. 1, p. 2, Jan. 2025, ISSN: 2624-960X. DOI: 10.3390/quantum7010002. Accessed: Aug. 26, 2025. [Online]. Available: https://www.mdpi.com/2624-960X/7/1/2.

[150] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.