Computational Physics Projects

Quantum Machine Learning

Sanidhya Katiyar

COMPUTATIONAL PHYSICS PROJECTS

Quantum Machine Learning

Sanidhya Katiyar[1]

[1]college name

[2]Indian Institute of Technology - Bombay

Abstract (Sample abstract)

Quantum Machine Learning (QML) is a promising field that combines quantum computing and classical machine learning to leverage quantum phenomena for enhanced computational power. The focus of this project is the development and application of Differentiable Quantum Circuits (DQCs) to solve supervised learning problems and approximate nonlinear functions. QML models, implemented using variational quantum circuits, utilize quantum feature maps to encode classical data into high-dimensional quantum states, enabling efficient representation of complex patterns.

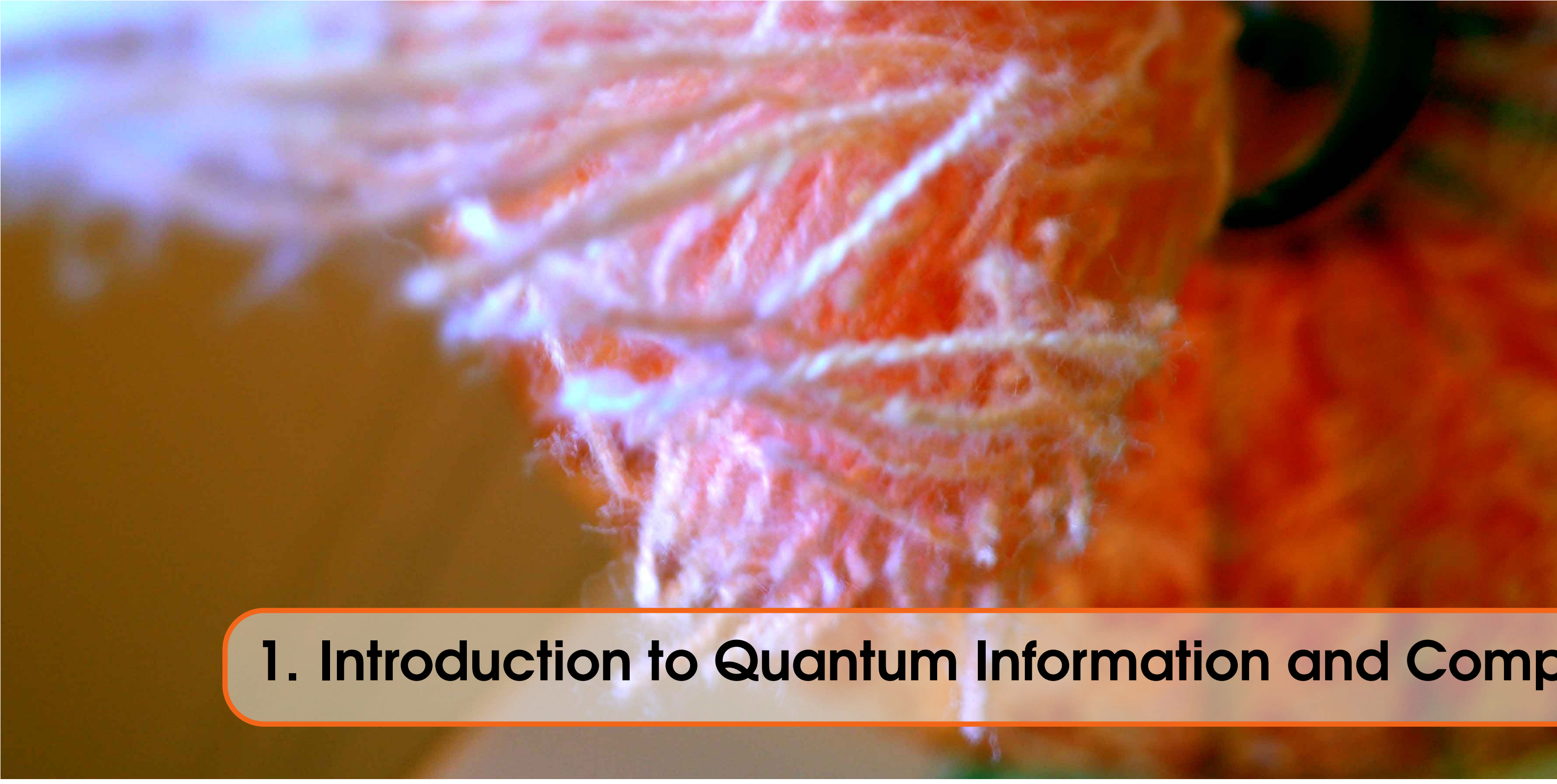
The project demonstrates the solution of the 2D Laplace equation as a use case for quantum machine learning, where the equation is solved with Dirichlet boundary conditions using a physics-informed loss function. Variational quantum circuits were optimized using hybrid quantum-classical techniques, with the loss function integrating contributions from the PDE residual and boundary conditions. The model's accuracy was validated against the analytical solution, achieving excellent agreement and demonstrating the potential of QML in solving partial differential equations (PDEs).

Additionally, quantum neural networks were applied to classification tasks, with the results showcasing the ability of QML models to capture non-linear relationships in data effectively. A comparative analysis of quantum and classical models highlighted the potential quantum advantage in specific scenarios, particularly in reducing feature space dimensionality and improving training efficiency.

The project establishes the utility of QML for both scientific computing and data-driven applications, emphasizing the scalability of quantum models for NISQ devices. The observed performance improvements in high-dimensional problems suggest that QML can play a transformative role in fields ranging from computational physics to artificial intelligence, paving the way for quantum advantage in real-world scenarios. .

Contents

Part One



1. Introduction to Quantum Information and Comp

1.1 Quantum Bits

In quantum computing, the **qubit** is the fundamental unit of quantum information, analogous to a classical bit. Unlike classical bits, which can be in one of two states, $|0\rangle$ or $|1\rangle$, a qubit can exist in a superposition of both states simultaneously. Mathematically, a qubit's state is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ (complex numbers) and satisfy the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Bloch Sphere Representation

The state of a qubit can be visualized on the Bloch sphere, where any pure state can be expressed as:

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

where θ and ϕ are spherical coordinates.

1.2 Multiple Qubit Systems

In quantum computing, systems involving multiple qubits exhibit unique properties such as **entanglement** and **tensor product spaces**, enabling quantum phenomena not present in classical systems.

Tensor Product Representation

The state of a system with n qubits is described by the tensor product of individual qubit states. For two qubits, the combined state is:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle,$$

where $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$. The combined state expands to:

$$|\psi\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle.$$

For n qubits, the state resides in a 2^n-dimensional Hilbert space.

Entanglement

Entanglement occurs when the state of a multi-qubit system cannot be expressed as a product of individual qubit states. For example, the Bell state:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

is an entangled state, as its components are inseparably linked. Measurement of one qubit instantaneously determines the state of the other, irrespective of distance.

1.3 Quantum Gates: Single and Multi-Qubit

Quantum gates are the building blocks of quantum circuits. These operations manipulate qubits and are represented by **unitary matrices** U, ensuring reversibility ($U^\dagger U = I$).

Single-Qubit Gates

Single-qubit gates act on a single qubit and are represented by 2×2 unitary matrices. Common gates include:

1. **Pauli Gates**:

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

2. **Hadamard Gate**:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

3. **Phase Gates**:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

4. **Rotation Gates**:

$$R_x(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}.$$

Multi-Qubit Gates

Multi-qubit gates operate on multiple qubits and are represented by $2^n \times 2^n$ matrices for n qubits.

1. **Controlled Gates**:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \text{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

2. **Swap Gate**:

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3. **Toffoli Gate (CCNOT)**:

$$\text{Toffoli} = \begin{bmatrix} I_6 & 0 \\ 0 & X \end{bmatrix},$$

where I_6 is a 6×6 identity matrix.

Applications

Quantum gates enable core operations in quantum algorithms, such as state preparation, entanglement, and error correction. Combined, they form circuits for tasks like Grover's search, Shor's factoring, and quantum machine learning.

1.4 Quantum Circuits

A quantum circuit is a model for quantum computation in which a sequence of quantum gates is applied to a set of qubits to manipulate their states. Quantum circuits are the quantum analog of classical logic circuits and are used to perform computations in quantum algorithms.

Structure of a Quantum Circuit

A quantum circuit typically consists of:

1. **Qubits**: Qubits are initialized in a specific state, usually $|0\rangle$ or $|1\rangle$.
2. **Quantum Gates**: Quantum gates are applied to the qubits to alter their state.
3. **Measurements**: After computation, qubits are measured to obtain classical results, collapsing the quantum state into one of the basis states.

Mathematical Representation

A quantum circuit with n qubits is represented as a unitary operator U acting on the 2^n-dimensional Hilbert space $\mathscr{H}$. If the initial state of the system is $|\psi_{\text{in}}\rangle$, the final state is:

$$|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle.$$

For example, applying a Hadamard gate H to all qubits creates a superposition:

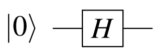
$$|\psi_{\text{out}}\rangle = H^{\otimes n}|0\rangle^{\otimes n}.$$

Quantum Circuit Example

Single-Qubit Circuit

A circuit applying a Hadamard gate:

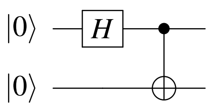
$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$



Two-Qubit Circuit

A Bell state $|\Phi^+\rangle$ is created by a Hadamard gate followed by a CNOT gate:

$$|00\rangle \xrightarrow{H\otimes I} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$



Measurement

Measurement collapses the state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \xrightarrow{\text{Measurement}} \quad \text{Outcome } \begin{cases} 0 & \text{with probability } |\alpha|^2, \\ 1 & \text{with probability } |\beta|^2. \end{cases}$$

1.5 Quantum Teleportation

Quantum teleportation is a protocol that allows the transfer of the quantum state of a qubit from one location to another, without physically moving the qubit itself. It exploits quantum entanglement and classical communication.

Theoretical Background

1. **Entanglement**: Alice and Bob share an entangled pair in the Bell state:

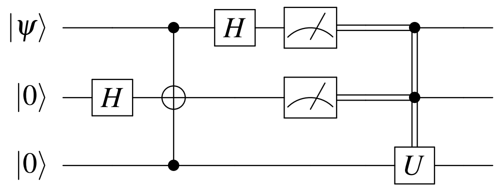
$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

2. **State to Be Teleported**: Alice holds a qubit in an unknown state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

   where $|\alpha|^2 + |\beta|^2 = 1$.
3. **Bell Measurement**: Alice performs a Bell-state measurement on her two qubits, collapsing the system into one of four Bell states.
4. **Classical Communication**: Alice communicates the result of her measurement to Bob using two classical bits.
5. **State Recovery**: Bob applies a unitary operation ($I, Z, X,$ or XZ) to recover $|\psi\rangle$.

Quantum Circuit for Teleportation



Applications

1. **Quantum Communication**: Secure key distribution and the basis of a quantum internet.
2. **Distributed Quantum Computing**: Efficient sharing of quantum information between remote processors.
3. **Fault Tolerance and Cryptography**: Used in error correction and secure quantum communication protocols.

1.6 Qiskit Overview

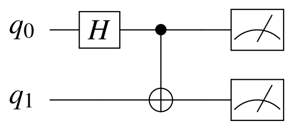
Qiskit is an open-source quantum computing framework that provides tools to create, simulate, and execute quantum circuits. It supports research, education, and development in quantum computing with access to IBM's quantum processors and simulators.

Creating Quantum Circuits

A quantum circuit is constructed by applying quantum gates to qubits. Below is an example of creating a quantum circuit in Qiskit:

```
from qiskit import QuantumCircuit

# Create a quantum circuit with 2 qubits and 2 classical bits
qc = QuantumCircuit(2, 2)

# Apply a Hadamard gate on qubit 0
qc.h(0)

# Apply a CNOT gate with control qubit 0 and target qubit 1
qc.cx(0, 1)

# Measure the qubits
qc.measure_all()

# Draw the circuit
print(qc.draw('text'))
```

The output of the above code is:



Aer Simulator

The Aer module provides high-performance simulators for testing quantum circuits before running them on real hardware. Common simulators include:

- **Statevector Simulator**: Simulates the full quantum state vector.
- **QASM Simulator**: Simulates measurement outcomes.
- **Unitary Simulator**: Computes the unitary matrix of a circuit.

Example:

```
from qiskit import Aer, transpile, execute

# Simulate the quantum circuit
backend = Aer.get_backend('qasm_simulator')
compiled_circuit = transpile(qc, backend)
job = execute(compiled_circuit, backend, shots=1024)

# Get results
results = job.result()
counts = results.get_counts(qc)
print("Measurement␣outcomes:", counts)
```

IBMQ

Qiskit provides access to IBM's quantum hardware through IBMQ. Users can run circuits on real quantum devices.

Steps:

1. Obtain an API token from IBM Quantum.
2. Save and load the account:

```
from qiskit import IBMQ
IBMQ.save_account('YOUR_API_TOKEN')
IBMQ.load_account()
```

3. Select a backend:

```
provider = IBMQ.get_provider(hub='ibm-q')
backend = provider.get_backend('ibmq_quito')
```

4. Execute on real hardware:

```
job = execute(qc, backend, shots=1024)
result = job.result()
print(result.get_counts(qc))
```

Visualization Tools

Qiskit provides several visualization tools:

- **Circuit Visualization:**

```
qc.draw('mpl')  # Render as an image
qc.draw('latex')  # Produce LaTeX code
```

- **Statevector Visualization:**

```
from qiskit.visualization import plot_bloch_multivector
from qiskit.quantum_info import Statevector
```

```
state = Statevector.from_instruction(qc)
plot_bloch_multivector(state.data)
```

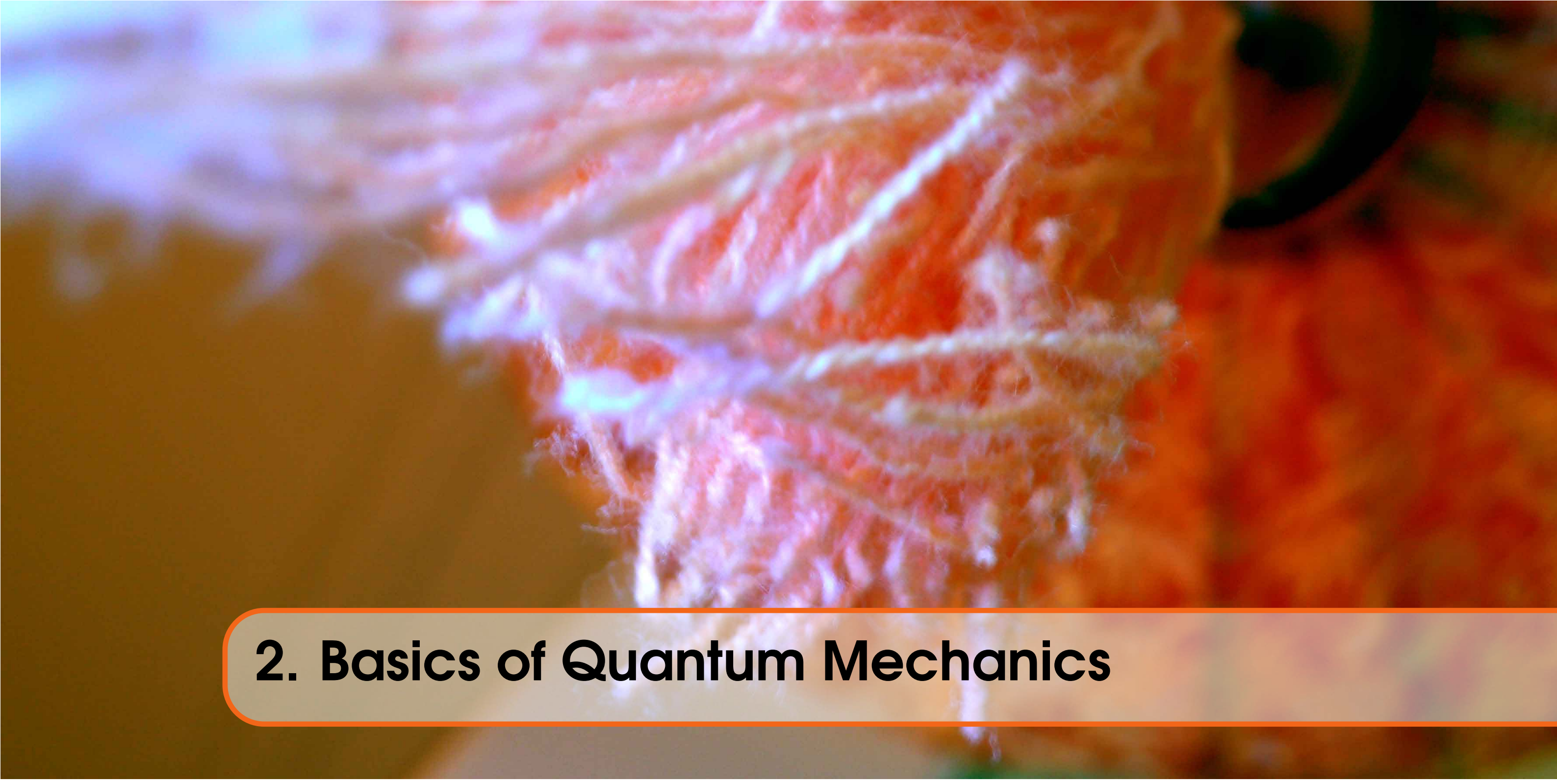
- **Measurement Outcomes:**

```
from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

Applications of Qiskit

1. **Quantum Algorithm Development:** Test algorithms like Grover's search and Shor's factoring.
2. **Quantum Machine Learning:** Hybrid quantum-classical models.

Qiskit bridges the gap between theoretical quantum computing and practical implementation, making it an indispensable tool for researchers and developers.



2. Basics of Quantum Mechanics

2.1 The Postulates of Quantum Mechanics

Quantum mechanics is built upon a set of fundamental postulates that describe the behavior of quantum systems. These postulates define the mathematical framework and provide tools for predicting the outcomes of quantum measurements.

Postulate 1: State Space

The state of a quantum system is represented by a vector $|\psi\rangle$ in a complex Hilbert space $\mathscr{H}$.

- The state must satisfy the normalization condition:

$$\langle\psi|\psi\rangle = 1.$$

- For mixed states, the system is described by a density operator ρ, satisfying:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \quad \text{where } p_i \geq 0 \text{ and } \sum_i p_i = 1.$$

Postulate 2: Observables and Measurement

Every measurable quantity (observable) is associated with a Hermitian operator $\hat{O}$ acting on $\mathscr{H}$.

- The possible measurement outcomes are the eigenvalues o_i of $\hat{O}$.
- Upon measurement, the system collapses into the eigenstate $|o_i\rangle$ corresponding to the observed eigenvalue o_i.
- The probability $P(o_i)$ of observing eigenvalue o_i is given by:

$$P(o_i) = |\langle o_i|\psi\rangle|^2.$$

Postulate 3: Evolution of Quantum States

The time evolution of a closed quantum system is governed by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle,$$

where $\hat{H}$ is the Hamiltonian operator representing the total energy of the system.

Postulate 4: Composite Systems

For a system composed of multiple subsystems, the total Hilbert space is the tensor product of the subsystems' Hilbert spaces:

$$\mathscr{H}_{\text{total}} = \mathscr{H}_1 \otimes \mathscr{H}_2 \otimes \cdots .$$

The state of the composite system is described as:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle.$$

Entanglement is a key feature of composite systems.

Postulate 5: Measurement Probability and Density Matrix

For a system in a mixed state described by ρ, the probability of obtaining a measurement outcome corresponding to a projection operator $\hat{P}_i$ is given by:

$$P(o_i) = \text{Tr}(\hat{P}_i\rho),$$

where Tr is the trace operation.

Postulate 6: Quantum Operators

Operators corresponding to observables satisfy:

- **Hermiticity:** Ensures real eigenvalues.
- **Commutativity:** For compatible observables $\hat{A}$ and $\hat{B}$:

$$[\hat{A},\hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A} = 0.$$

- **Unitarity:** Evolution operators $\hat{U}(t)$ satisfy:

$$\hat{U}(t)\hat{U}^\dagger(t) = I.$$

Applications and Implications

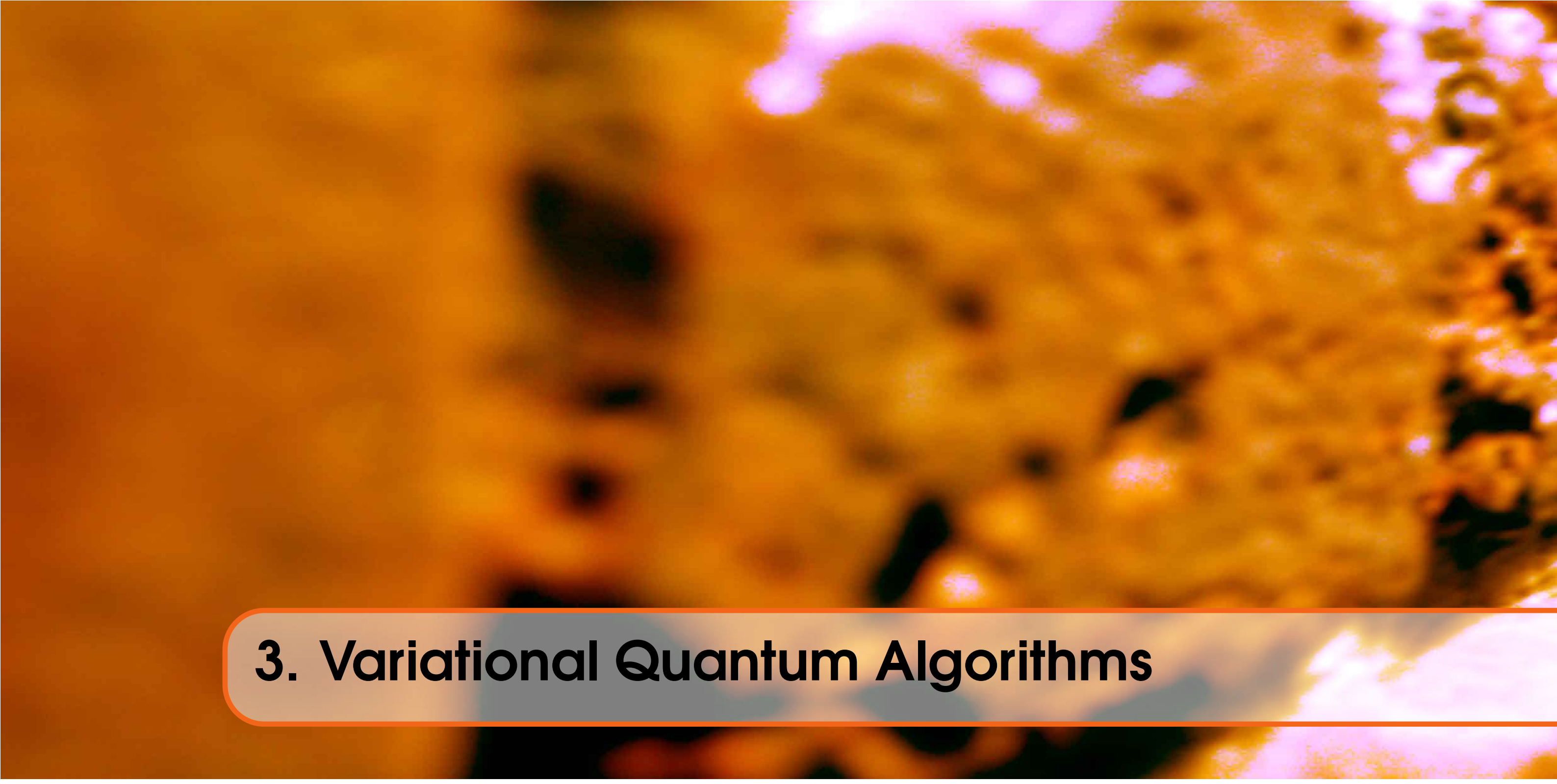
- **Wavefunction Collapse:** Explains the probabilistic nature of quantum measurements.
- **Uncertainty Principle:** Arises from non-commutative operators:

$$\Delta A \Delta B \geq \frac{1}{2}|\langle[\hat{A},\hat{B}]\rangle|.$$

- **Superposition and Entanglement:** Fundamental concepts for quantum computing and teleportation.

The postulates of quantum mechanics establish the foundation for understanding and applying quantum phenomena in both theoretical and practical domains.

Part Two



3. Variational Quantum Algorithms

3.1 Variational Quantum Algorithms (VQAs)

VQAs are hybrid quantum-classical algorithms designed to work on Noisy Intermediate-Scale Quantum (NISQ) devices. They utilize a parameterized quantum circuit (ansatz) optimized by a classical computer to solve a variety of tasks.

Cost Function

The cost function $C(\theta)$ encodes the problem into a landscape:

$$C(\theta) = f(\{\rho_k\}, \{O_k\}, U(\theta)),$$

where:

- ρ_k: Input states,
- O_k: Observables,
- $U(\theta)$: Parameterized quantum circuit.

For example:

$$C(\theta) = \sum_k f_k \left(\mathrm{Tr}[O_k U(\theta) \rho_k U^\dagger(\theta)] \right).$$

The cost function should be faithful, efficiently estimable, operationally meaningful, and trainable.

Ansatz

The ansatz is a parameterized quantum circuit:

$$U(\theta) = U_L(\theta_L) \cdots U_2(\theta_2) U_1(\theta_1),$$

with:

$$U_l(\theta_l) = \prod_m e^{-i\theta_m H_m} W_m,$$

where H_m are Hermitian operators and W_m are unparameterized unitaries. Common ansatz types include:

- **Hardware-efficient ansatz**: Tailored to hardware constraints.
- **Quantum Alternating Operator Ansatz (QAOA)**: Alternating applications of problem and mixer Hamiltonians.
- **Unitary Coupled Cluster Ansatz (UCC)**: Used in quantum chemistry.
- **Variable structure ansatz**: Adapts structure during training.

3.2 Quantum Machine Learning

VQAs are integral to Quantum Machine Learning (QML), allowing parameterized circuits to learn patterns in data.

Workflow

1. Input data x is embedded into a quantum state $V(x)|\psi_0\rangle$. 2. A parameterized unitary $U(\theta)$ acts on the state. 3. The output is measured, and the cost function:

$$C(\theta) = \sum_i \left[y^{(i)} - \langle \psi_0 | V^\dagger(x^{(i)}) U^\dagger(\theta) A U(\theta) V(x^{(i)}) | \psi_0 \rangle \right]^2,$$

is minimized using classical optimization.

Applications

- **Classification**: Embedding classical data into quantum states and optimizing classifiers.
- **Feature Maps**: Using quantum states to represent features non-linearly.
- **Kernel Methods**: Quantum kernels enhance model performance.

Advantages and Challenges

Advantages:

- Reduces circuit depth for NISQ devices.
- Applies to a wide range of problems: chemistry, optimization, and data science.

Challenges:

- Trainability: Barren plateaus in the cost landscape.
- Noise and limited qubits in NISQ devices.
- Choosing optimal ansatz and cost functions.

Conclusion

VQAs provide a promising framework for leveraging quantum computers in the NISQ era. Their hybrid nature allows for solving complex problems across quantum computing and machine learning, offering a pathway to near-term quantum advantage.

3.3 Summary of Variational Quantum Eigensolver (VQE) Implementation

This notebook implements a Variational Quantum Eigensolver (VQE), a hybrid quantum-classical algorithm, to approximate the eigenvalues of a given problem Hamiltonian. Below is a summary of the workflow.

1. Ansatz

The ansatz is a parameterized quantum circuit:

$$U(\vec{\theta}) = \prod_i e^{-i\theta_i H_i},$$

where H_i are Hermitian operators, and $\vec{\theta}$ represents the parameters to be optimized.

2. Observable

The problem Hamiltonian, represented as an observable, is decomposed into a weighted sum of Pauli operators:

$$H = \sum_i c_i P_i,$$

where P_i are tensor products of Pauli matrices, and c_i are coefficients.

3. Cost Function

The cost function is defined as the expected value of the observable:

$$C(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle,$$

where $|\psi(\vec{\theta})\rangle = U(\vec{\theta})|\psi_0\rangle$.
The implementation includes tracking cost function values across iterations for visualization and analysis.

4. Optimization

The classical optimizer used is COBYLA, with the following setup:

- **Initial Parameters (x_0)**: Randomly initialized values.
- **Arguments**: Ansatz, observable, and estimation function.
- **Options**: Maximum iterations set to 1000.

The optimization routine tracks the cost function values at each iteration.

5. Final Results

The optimization terminated successfully with the following results:

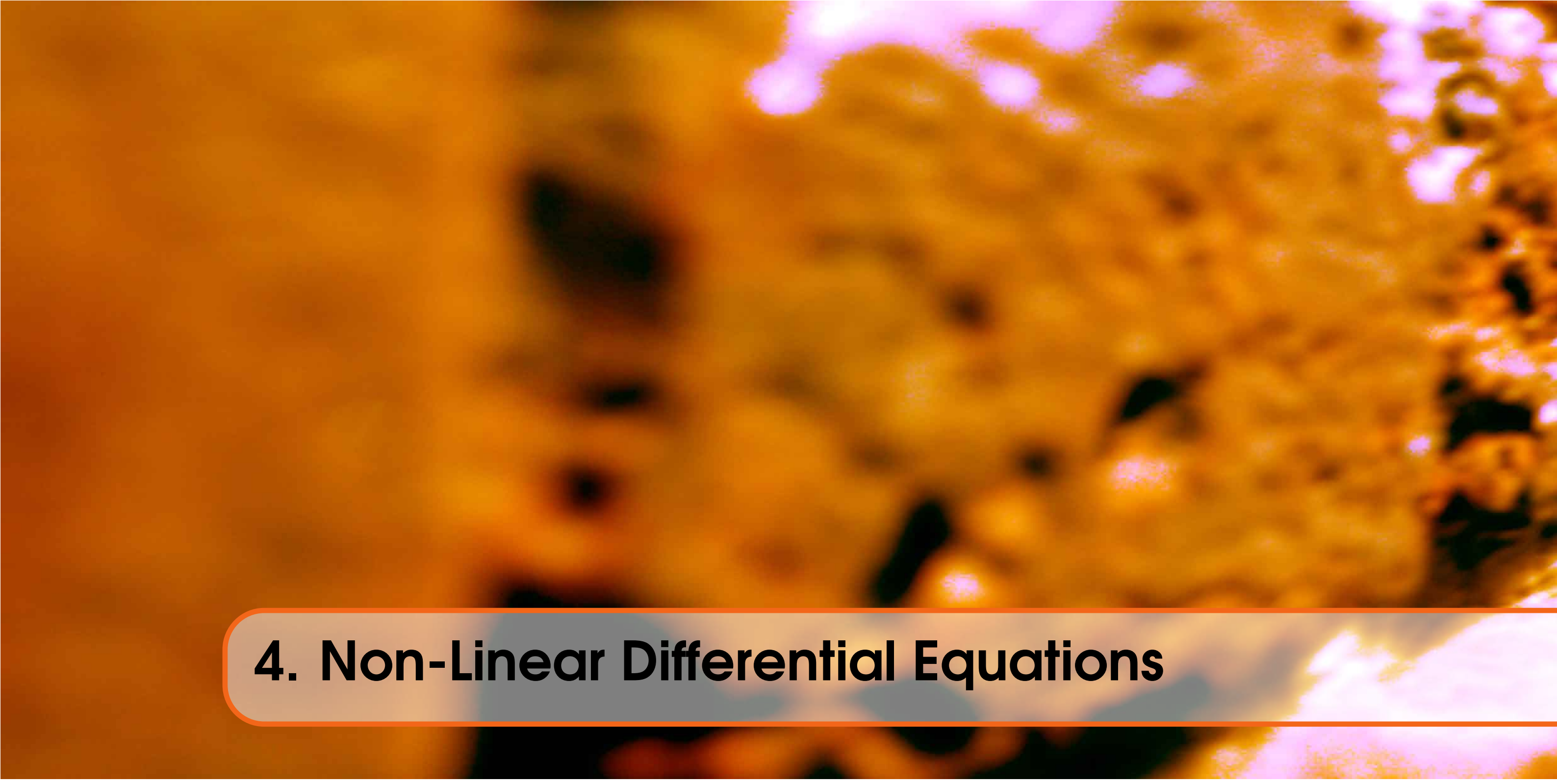
- **Minimum Cost Value**: $C_{\text{min}} = -2.236$
- **Optimal Parameters ($\vec{\theta}_{\text{opt}}$)**:

$$\vec{\theta}_{\text{opt}} = [1.571, 2.066, 0.002, 1.038, 1.570, 1.569, 1.300, 2.311]$$

- **Number of Iterations**: 926
- **Execution Time**: 2.05 seconds
- **Percent Error**: 1.45×10^{-7}

Conclusion

This VQE implementation demonstrates the utility of hybrid algorithms in approximating eigenvalues of quantum Hamiltonians efficiently, with results verified using eigenvalue solvers.



4. Non-Linear Differential Equations

4.1 Solving Nonlinear Differential Equations with Differentiable Quantum Circuits

Introduction

Differential equations are central to many fields, including physics, chemistry, ecology, and finance. Solving nonlinear differential equations (DEs) is challenging due to dimensionality and numerical instabilities. The proposed method uses Differentiable Quantum Circuits (DQCs) to encode solutions and their derivatives, leveraging quantum feature maps and variational optimization. This hybrid quantum-classical approach is designed for NISQ (Noisy Intermediate-Scale Quantum) devices.

Quantum Feature Maps

A quantum feature map $U_\phi(x)$ encodes a function $\phi(x)$ into the amplitudes of a quantum state:

$$U_\phi(x)|\emptyset\rangle = |\psi(x)\rangle.$$

Examples include:

- **Product Feature Map:** Encodes nonlinear functions using Pauli rotations, e.g.,

$$U_\phi(x) = \bigotimes_{j=1}^{N} R_y(\phi[x]).$$

- **Chebyshev Feature Map:** Uses Chebyshev polynomials $T_n(x)$ to provide a rich basis set for complex functions.
- **Evolution-Enhanced Map:** Combines product maps with Hamiltonian evolution $e^{-iH\tau}$, enabling access to entangled states.

Feature maps are crucial for representing solutions in high-dimensional latent spaces.

—

Variational Quantum Circuits

The variational quantum circuit U_θ acts on the encoded feature map to approximate solutions:

$$|\psi_\theta(x)\rangle = U_\theta U_\phi(x)|\emptyset\rangle.$$

Common ansatz types include:

- **Hardware-Efficient Ansatz (HEA):** Layers of single-qubit rotations and entangling gates.
- **Alternating Blocks Ansatz (ABA):** Localized entangling blocks for better trainability.

The choice of ansatz determines the expressivity and trainability of the quantum circuit.

—

Cost Function and Loss Optimization

The solution $f(x)$ is obtained as the expectation value of a Hermitian cost operator C:

$$f(x) = \langle\psi_\theta(x)|C|\psi_\theta(x)\rangle.$$

The optimization minimizes a loss function:

$$L_\theta = L_{\text{differential}} + L_{\text{boundary}},$$

where:

- $L_{\text{differential}}$ evaluates DE residuals over a grid of points.
- L_{boundary} enforces boundary conditions.

Loss functions such as Mean Squared Error (MSE) or Kullback-Leibler divergence are used, with gradients computed via automatic differentiation.

—

Boundary Handling and Regularization

Boundary conditions can be enforced using:

- **Pinned Boundaries:** Encode boundary values directly in the cost function.
- **Floating Boundaries:** Adjust solutions iteratively.
- **Optimized Boundaries:** Use classical parameters optimized alongside variational angles.

Regularization techniques guide solutions toward desired forms, with adaptive weights ensuring convergence.

—

Applications and Results

The DQC method was applied to solve nonlinear equations like the Navier-Stokes equations and a damped oscillator:

$$\frac{du}{dx} + \lambda u(\kappa + \tan(\lambda x)) = 0.$$

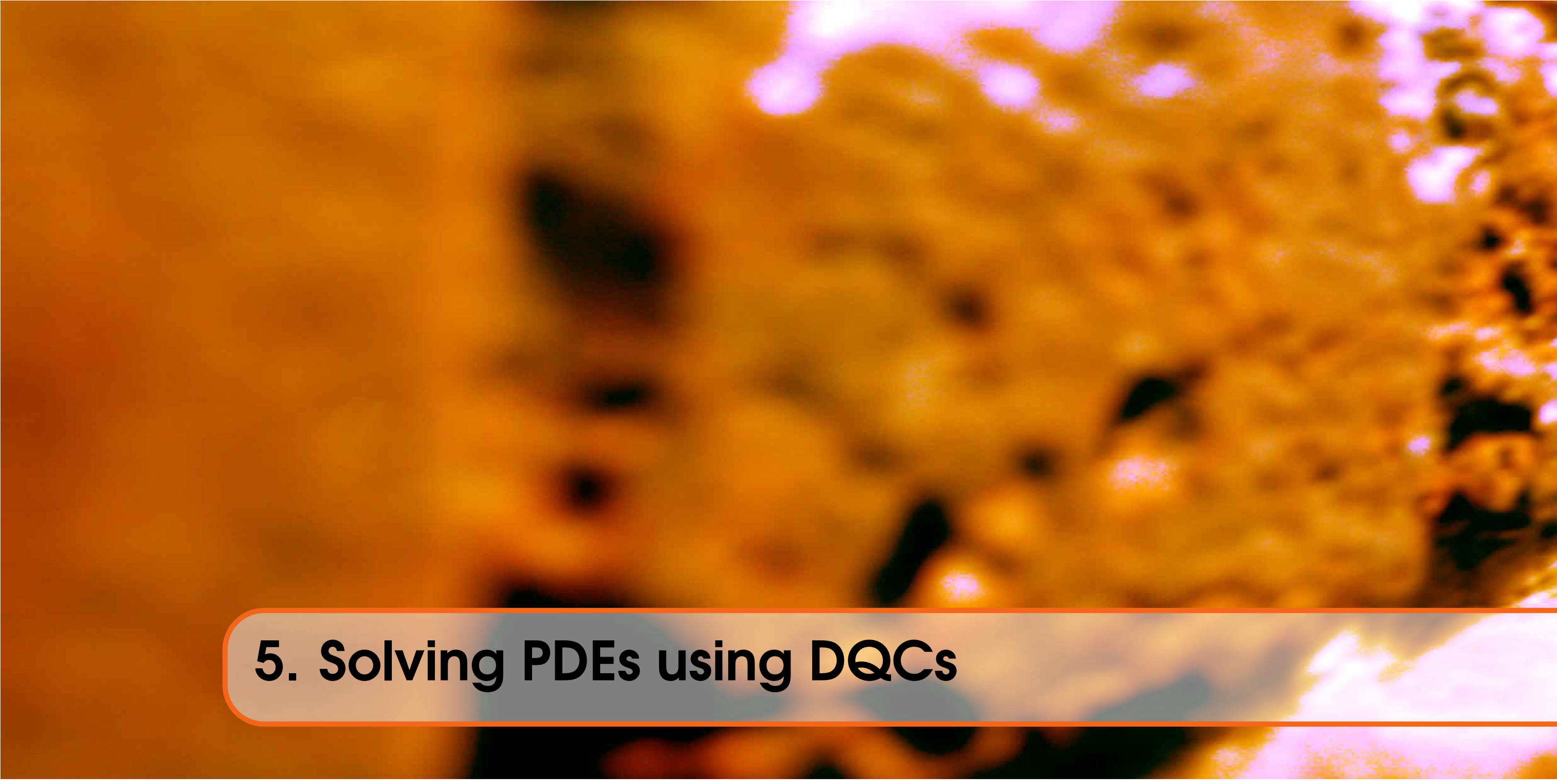
Results demonstrated that:

- Chebyshev feature maps provided accurate solutions for complex oscillatory functions.
- Tower Chebyshev maps achieved better expressivity for high-order nonlinearities.
- Loss metrics converged effectively using Adam optimization.

—

Conclusion

The DQC framework offers a powerful tool for solving nonlinear DEs, leveraging quantum expressivity and classical optimization. It provides scalable solutions for high-dimensional problems while overcoming limitations of classical methods such as numerical differentiation.



5. Solving PDEs using DQCs

Solving the 2D Laplace Equation with Differentiable Quantum Circuits (DQCs)

Introduction

The Laplace equation in two spatial dimensions is a widely studied partial differential equation (PDE) with applications in fields like physics and engineering. The equation is given by:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$

with the known exact solution:

$$u(x,y) = e^{-\pi x} \sin(\pi y),$$

when solved under the Dirichlet boundary conditions:

$$u(0,y) = \sin(\pi y), \quad u(x,0) = 0, \quad u(X,y) = e^{-\pi} \sin(\pi y), \quad u(x,Y) = 0,$$

where $x, y \in [0,1]$.
The goal is to approximate the solution using a quantum neural network (QNN) implemented through the Qadence framework.
—

Methodology

1. DQC Model

The DQC model uses a quantum neural network U_θ to approximate the solution $u(x,y)$. The ansatz is parameterized by θ, and the solution is represented as:

$$u(x,y) = \langle \psi(x,y) | C | \psi(x,y) \rangle,$$

where C is the cost function observable.

2. Problem Definition

The problem is decomposed into five terms:

- One PDE term corresponding to the Laplace equation.
- Four boundary condition terms for the domain edges.

Each term contributes to the total loss function as residuals, ensuring both the PDE and boundary conditions are satisfied.

3. Equation Domains

Each term is defined over a specific domain:

- **PDE Domain:** Interior points of the domain $x, y \in (0,1)$.
- **Boundary Domains:** Collocation points along the edges of the domain:

$$x = 0, x = 1, y = 0, y = 1.$$

Collocation points are sampled discretely within each domain.

4. DQC Architecture

The DQC model comprises:

- **Feature Map:** Encodes the input variables x and y into quantum states.
- **Variational Ansatz:** A parameterized quantum circuit optimized to approximate the solution.
- **Cost Observable:** Computes the expected value corresponding to $u(x,y)$.

5. Loss Function

The total loss combines the PDE residual and boundary losses:

$$L_\theta = L_{\text{PDE}} + \sum_{i=1}^{4} L_{\text{boundary},i}.$$

Each loss term is calculated as the mean squared error of the residuals.

6. Optimization

The model uses a classical deep learning optimizer (e.g., Adam) to minimize the total loss and update the circuit parameters θ.

—

Results

The left panel of Figure 5.1 shows the analytical solution $u(x,y) = e^{-\pi x}\sin(\pi y)$, while the right panel illustrates the DQC model's approximation.

—

Conclusion

This exercise demonstrates the use of DQCs for solving PDEs like the Laplace equation. By combining quantum feature maps, variational ansatz, and classical optimization, the DQC model efficiently approximates the solution while satisfying the boundary conditions.

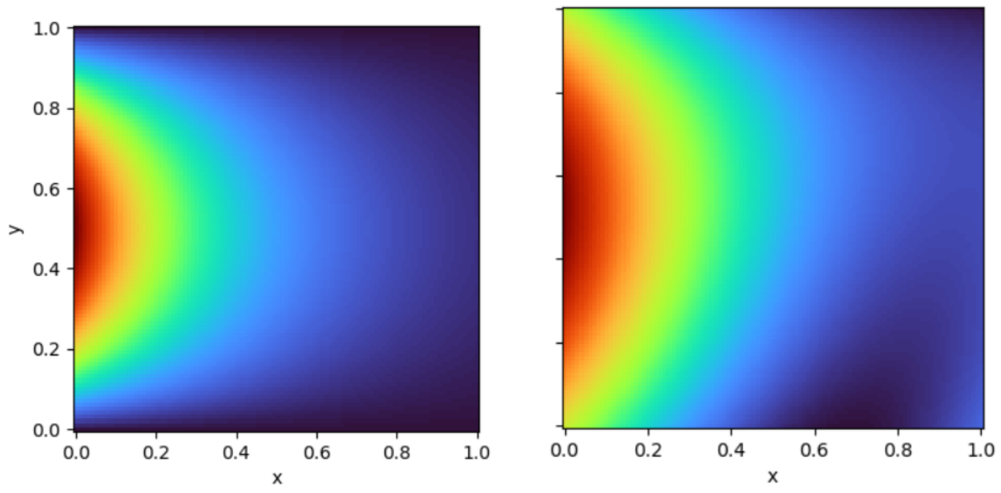


Figure 5.1: (Left) Analytical solution. (Right) DQC model approximation.