

Quantum Recurrent Neural Networks with Encoder-Decoder for Time-Dependent Partial Differential Equations

Yuan Chen¹, Abdul Khalil^{1,2}, and Khaled M. Furati³

¹Computational and Data Science Program, Middle Tennessee State University, Murfreesboro, 37132, TN, USA

²Department of Mathematical Science, Middle Tennessee State University, Murfreesboro, 37132, TN, USA

³Department of Mathematics, King Fahd University of Petroleum & Minerals, Dhahran, 31261, Saudi Arabia

Nonlinear time-dependent partial differential equations are essential in modeling complex phenomena across diverse fields, yet they pose significant challenges due to their computational complexity, especially in higher dimensions. This study explores Quantum Recurrent Neural Networks within an encoder-decoder framework, integrating Variational Quantum Circuits into Gated Recurrent Units and Long Short-Term Memory networks. Using this architecture, the model efficiently compresses high-dimensional spatiotemporal data into a compact latent space, facilitating more efficient temporal evolution. We evaluate the algorithms on the Hamilton-Jacobi-Bellman equation, Burgers' equation, the Gray-Scott reaction-diffusion system, and the three dimensional Michaelis-Menten reaction-diffusion equation. The results demonstrate the superior performance of the quantum-based algorithms in capturing nonlinear dynamics, handling high-dimensional spaces, and providing stable solutions, highlighting their potential as an innovative tool in solving challenging and complex systems.

1 Introduction

Partial differential equations (PDEs) are fundamental mathematical tools for modeling diverse phenomena in many fields such as physics, biology, chemistry, and economics. However, for many complex and high-dimensional PDEs, analytical solutions are often unattainable due to

their intricate structures and non-linearities. To address this, numerical methods such as the finite-difference method (FDM) [1], finite-element method (FEM) [2], and finite-volume method (FVM) [3] have been developed to approximate solutions. These techniques have been effective in a variety of applications but face limitations in computational complexity, stability, and scalability, especially when applied to non-linear or high-dimensional problems.

In recent years, deep learning has emerged as a powerful data-driven approach for solving partial differential equations (PDEs), particularly when traditional numerical methods struggle with the complexities of real-world data. Techniques like physics-informed neural networks (PINNs) and their extensions [4, 5, 6] excel by learning complex nonlinear relationships and managing high-dimensional data. By embedding the underlying physical laws directly as constraints within the network architecture, these methods achieve remarkably accurate approximations of PDE solutions.

With the development of neural networks, Recurrent neural networks (RNNs) provided another promising approach in the context of PDEs, especially for problems involving temporal dynamics. RNNs were first introduced by Rumelhart et al. [7] as a class of neural networks designed for processing sequential data by incorporating feedback loops [8, 9, 10]. While standard RNNs demonstrated potential, they suffered from issues like vanishing gradients, which limited their ability to capture long-term dependencies. To address this, long short-term memory (LSTM) networks were introduced by Hochreiter et al. [11] as an extension of RNNs with gating mechanisms that enable effective learning of long-term dependencies. Later, gated recurrent

Yuan Chen: yc3y@mtmail.mtsu.edu

units (GRUs) were proposed by Cho et al.[12] as a simplified alternative to LSTMs, retaining similar capabilities but with fewer parameters.

The application of RNNs and their variants to PDEs has shown significant promise. These models can capture temporal dependencies and approximate complex solutions by learning from data-driven or hybrid approaches. For instance, a physics-incorporated convolutional recurrent neural network for identifying sources and forecasting dynamical systems was proposed by Saha et al [13]. Neural-PDE, an RNN-based framework specifically tailored for solving time-dependent PDEs introduced by Hu et al. [14]. These approaches demonstrate the versatility and potential of RNN-type algorithms in addressing the challenges posed by complex PDE systems. Meanwhile, another tool caught our eyes, an encoder-decoder architectures were proposed by Cho et al. [15], which could help solving PDEs by compressing high-dimensional spatial-temporal data into a lower-dimensional latent space, enabling the model to efficiently learn complex temporal dynamics and reconstruct accurate high-dimensional solutions. Other RNNs based PDE method can be found in [16, 17].

Parallel to the advancements in deep learning, quantum machine learning has emerged as a transformative technology that leverages quantum mechanics for computation. Recent studies have extended these quantum methods to tackle PDEs; Hunout et al. [18] introduced a variational quantum algorithm based on Lagrange polynomial encoding, Choi and Ryu [19] developed a method for multi-dimensional Poisson equations, and Childs et al. [20] demonstrated high-precision quantum algorithms for PDEs. Moreover, Variational Quantum Circuits (VQCs) have shown significant potential in enhancing classical neural network architectures by incorporating quantum entanglement and superposition [21, 22, 23]. This progress has paved the way for Quantum Recurrent Neural Networks (QRNNs), such as Quantum Long Short-Term Memory (QLSTM) [24] and Quantum Gated Recurrent Units (QGRU) [25], which are particularly adept at modeling complex temporal dependencies. Motivated by these advances, this study explores the application of QRNNs to solve challenging non-linear time-dependent PDEs.

This paper is organized as follows: Section 2

presents the methodology, including an overview of both classical and quantum RNN models. Section 3 covers four numerical experiments: the Burgers' equation, Gray-Scott reaction-diffusion system, Hamilton-Jacobi-Bellman (HJB) equation, and 3D Michaelis-Menten reaction-diffusion system. Finally, Section 4 provides the conclusion.

2 Methodology

In this work, we utilize a composite model architecture comprising three main components: an encoder, a quantum recurrent neural network (QGRU or QLSTM), and a decoder. The classical auto-encoder compresses higher-dimensional PDE snapshots into a lower-dimensional latent space. These latent representations are then used as inputs to an RNN (in this case, a quantum-classical RNN variant) that learns the temporal evolution of the PDEs. Finally, the RNN's predictions are decoded back to the original dimensional space, enabling full reconstruction and analysis of the time-dependent PDEs solution. To provide a more comprehensive understanding of our approach, the following sections first introduce the QLSTM and QGRU architectures, followed by a detailed explanation of the encoder-decoder structure.

2.1 Quantum Long Short-Term Memory (QLSTM)

VQCs form the core of quantum recurrent neural networks, comprising three primary parts: an encoding layer, a variational layer, and a quantum measurement layer. Figure 1 provides an example of a VQC.

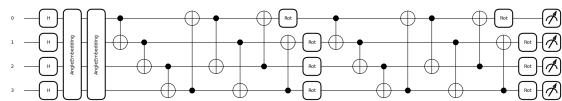


Figure 1: An example of VQC architecture with two layers of entanglements for QLSTM.

On the left of Figure 1, the encoding layer applies a Hadamard gate and two angle embeddings (treated as R_y and R_z rotations in quantum computing). The central portion is the variational (entanglement) layer. Note that both the number of qubits (4 in this instance) and the num-

ber of variational layers (2 here) are adjustable parameters that can be tuned to improve model performance.

The encoding layer is tasked with mapping classical data into quantum amplitudes. This process starts with initializing the quantum circuit in the ground state, then applying Hadamard gates to each qubit to create an unbiased initial state. The state of an N -qubit system can be expressed as:

$$|\psi\rangle = \sum_{(q_1, q_2, \dots, q_N) \in \{0,1\}^N} c_{q_1, q_2, \dots, q_N} |q_1\rangle \otimes |q_2\rangle \otimes \dots \otimes |q_N\rangle \quad (1)$$

where $c_{q_1, \dots, q_N} \in \mathbb{C}$ is the amplitude of each basis state and \otimes denotes the tensor product. By Born's rule, the probability of measuring any specific state is determined by the square of its amplitude's magnitude:

$$\sum_{(q_1, q_2, \dots, q_N) \in \{0,1\}^N} \|c_{q_1, \dots, q_N}\|^2 = 1. \quad (2)$$

Applying the Hadamard gate H on each qubit $|0\rangle$ transforms the circuit into a uniform superposition state. Here, i indexes through all possible N -qubit basis states. For each data input x_i , the two-angle encoding scheme assigns rotation angles $\arctan(x_i)$ and $\arctan(x_i^2)$, which correspond to R_y and R_z gates respectively.

$$(H|0\rangle)^{\otimes N} = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes N} = \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle. \quad (3)$$

In this work, the template `qml.templates.AngleEmbedding` is employed for these rotations, offering flexibility in selecting rotation axes (*e.g.*, R_x , R_y , or R_z).

The variational circuit forms the trainable portion of the VQC and includes parameterized unitary operations. This portion incorporates multiple CNOT gates to produce entanglement, followed by unitary rotations governed by learnable parameters α , β , and γ . In practice, one can repeat this variational block multiple times to enlarge the parameter space and increase expressive power.

Quantum measurement translates the quantum circuit's state back into classical information. Owing to quantum systems' inherent probabilistic nature, repeated measurements yield different

bit strings. The expectation value of a Hermitian operator \hat{O} with respect to $|\psi\rangle$ is defined as:

$$E[\hat{O}] = \langle \psi | \hat{O} | \psi \rangle. \quad (4)$$

These expectations may be calculated either analytically (in simulation) or derived via multiple shots on actual quantum hardware, incorporating specific noise models. Figure 2 depicts a single QLSTM with 6 VQCs.

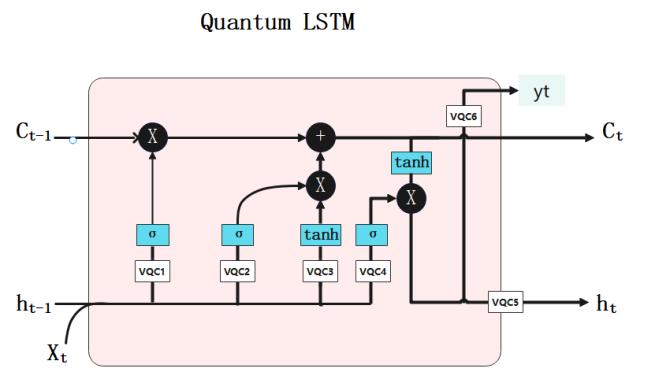


Figure 2: Structure of QLSTM.

Similar to classical LSTMs, QLSTMs maintain two primary memory elements: the hidden state h_t and the internal cell state c_t . The core operations in a QLSTM cell can be written as follows:

$$\begin{aligned} f_t &= \sigma(VQC1(v_t)), \\ i_t &= \sigma(VQC2(v_t)), \\ \tilde{c}_t &= \tanh(VQC3(v_t)), \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t, \\ o_t &= \sigma(VQC4(v_t)), \\ h_t &= VQC5(o_t * \tanh(c_t)), \\ \tilde{y}_t &= VQC6(o_t * \tanh(c_t)), \\ y_t &= \text{NN}(\tilde{y}_t), \end{aligned}$$

where $*$ denotes element-wise multiplication. At each timestep, the QLSTM cell takes v_t as input, which is the concatenation of h_{t-1} and x_t (the previous hidden state and current input, respectively).

2.2 Quantum Gated Recurrent Unit (QGRU)

Similar to QLSTMs, QGRUs enhance the classical GRU architecture by integrating VQCs. The example plot of a QGRU is shown in Figure 3.

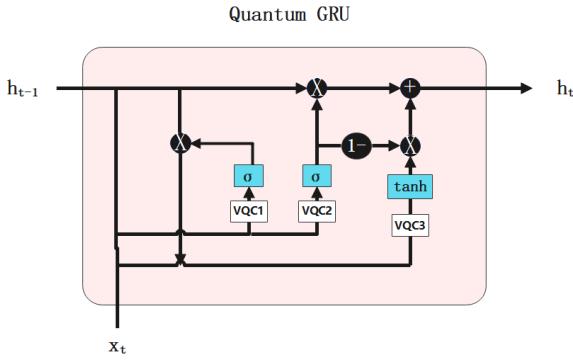


Figure 3: Structure of QGRU.

QGRU follows these equations:

$$\begin{aligned} r_t &= \sigma(\text{VQC1}(v_t)), \\ z_t &= \sigma(\text{VQC2}(v_t)), \\ o_t &= \text{cat}(x_t, r_t * H_{t-1}), \\ \tilde{H}_t &= \tanh(\text{VQC3}(o_t)), \\ H_t &= z_t * H_{t-1} + (1 - z_t) * \tilde{H}_t, \\ y_t &= \text{NN}(H_t). \end{aligned}$$

Here, r_t and z_t are the reset and update gates at time step t . The hidden state H_t evolves based on z_t and the candidate hidden state \tilde{H}_t , while v_t denotes the concatenation of the previous hidden state H_{t-1} with the current input x_t . Because QGRUs employ fewer quantum circuits than QL-STMs, they are typically more computationally efficient. Specifically, three VQCs are used to process quantum information within the gates and candidate hidden state computations. The final output y_t arises from a classical neural network acting on the hidden state H_t .

2.3 Encoder-Decoder Architecture with Quantum Recurrent Models

The encoder-decoder framework is central to efficiently modeling time-dependent PDEs. The encoder compresses high-dimensional spatial data into a lower-dimensional latent space, enabling the recurrent model to learn temporal patterns efficiently. The encoder maps input data $\mathbf{X} \in \mathbb{R}^{n_{\text{input}}}$ to a latent representation $\mathbf{Z} \in \mathbb{R}^{n_{\text{latent}}}$ using fully connected layers:

$$\mathbf{Z} = f_{\text{enc}}(\mathbf{X}; \theta_{\text{enc}}),$$

where θ_{enc} represents the encoder parameters.

In the latent space, the temporal dynamics are modeled using QRNNs as we introduced. These

quantum-enhanced recurrent models incorporate VQCs to capture complex temporal dependencies and non-linearities. Outputs from the recurrent model are transformed back into the high-dimensional space through the decoder, which reconstructs the original spatial resolution:

$$\hat{\mathbf{X}}_{t+1} = f_{\text{dec}}(\mathbf{Z}_{t+1}; \theta_{\text{dec}}).$$

This architecture reduces computational complexity, ensures accurate spatiotemporal predictions, and leverages quantum circuits to enhance expressiveness, making it highly effective for solving time-dependent PDEs.

In our study, we use an Multi-Layer Perceptrons (MLPs) structure for the both encoder and decoder, which is a fundamental neural network architectures consisting of stacked layers of neurons with learnable parameters and activation functions [7]. Fully connected layers are utilized where each neuron is connected to every neuron in the subsequent layer, allowing the network to capture complex hierarchical representations.

More precisely, our autoencoder structured as deep MLP networks. The encoder progressively reduces the input dimensionality, passing through layers of sizes 4096, 2048, 1024, and 512 before reaching a compressed latent space. Each layer employs the hyperbolic tangent (tanh) activation function. After the RNN models are done the prediction works in latent space. The decoder then reconstructs the input by mirroring the encoder structure, expanding the latent representation back to the original input size. A final sigmoid activation ensures the output values remain normalized between 0 and 1.

Here we show the pseudocode of proposed algorithm.

Algorithm 1 Encoder-decoder QRNNs for Solving Time-Dependent PDEs

- 1: **Input:** Discretized PDE data \mathbf{V} with dimensions (n_t, n_x, n_y) ; QRNN parameters θ_{qrnn} ; Encoder parameters θ_{enc} ; Decoder parameters θ_{dec} .
 - 2: **Output:** Predicted solutions of PDEs $\hat{\mathbf{V}}$.
 - 3: Normalize the input data \mathbf{V} to obtain \mathbf{V}_{norm} .
 - 4: Flatten the spatial dimensions of \mathbf{V}_{norm} .
 - 5: Encode \mathbf{V}_{norm} into latent space \mathbf{Z} using the encoder.
 - 6: Generate training sequences $\mathbf{X}_{\text{train}}$ and targets $\mathbf{Y}_{\text{train}}$.
 - 7: Randomly initialize QRNN parameters θ_{qrnn} .
 - 8: Train QRNN to approximate latent dynamics $\tilde{\mathbf{Z}}$.
 - 9: Decode $\tilde{\mathbf{Z}}$ into high-dimensional predictions $\hat{\mathbf{X}}$ using the decoder.
 - 10: Compute loss \mathcal{L} and minimize it using Adam optimizer.
 - 11: De-normalize predictions $\hat{\mathbf{X}}$ to match the original scale.
 - 12: **Return:** Final reconstructed predictions $\hat{\mathbf{V}}$.
-

3 Numerical Experiments

In this section, we investigate the performance of two quantum-enhanced encoder-decoder recurrent neural network architectures, QLSTM and QGRU in solving several nonlinear PDEs. We also evaluated the performance of a classical LSTM for comparison.

3.1 Experiment 1: Burgers' Equation

The 2D Burgers' equation [26] is a nonlinear partial differential equation widely used in fluid dynamics to model viscous flow and turbulence. It is represented as a coupled system:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad (5)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right), \quad (6)$$

where $u(x, y, t)$ and $v(x, y, t)$ represent the velocity fields in the x - and y -directions, ν is the kinematic viscosity, and t is the time variable. The spatial domain of the problem consists of a two-dimensional computational grid discretized

over $[0, 1] \times [0, 1]$ with 64×64 grid points in the x and y directions, respectively.

In experiment 1, we initialize the fields u and v with sinusoidal conditions to mimic complex dynamics:

$$u(x, y, 0) = \sin(2\pi x) \sin(2\pi y), \quad (7)$$

$$v(x, y, 0) = \sin(\pi x) \sin(\pi y). \quad (8)$$

The numerical solution to the Burgers' equation provides the training and test data for the models.

All the models were trained on time-series data generated from the simulation of the 2D Burgers' equation. Key hyperparameters were chosen as follows:

Parameter	Value
Hidden Size	4
Number of Layers	5
Number of Qubits	4
Number of Quantum Layers	4
Learning Rate	0.01
Epochs	50

Table 1: Hyperparameters used in the experiments.

All the models are evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to assess performance. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for each model are calculated as follows:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (9)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}, \quad (10)$$

where y_i are the true values, \hat{y}_i are the predictions, and n is the sample size. MAE focuses on the absolute deviations, while RMSE emphasizes larger errors, making it sensitive to larger deviations.

The following plots illustrate the evolution of the MAE and RMSE metrics for the training and testing phases across epochs.

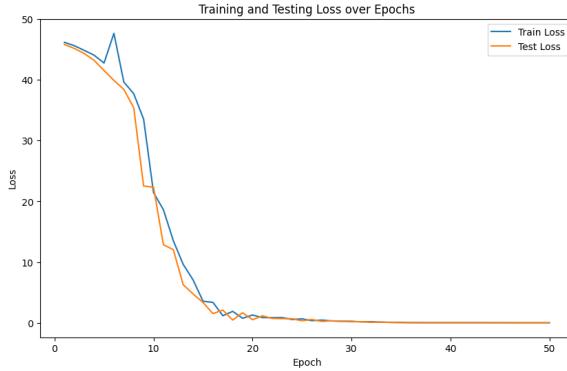


Figure 4: Training and test loss over epochs for LSTM model.

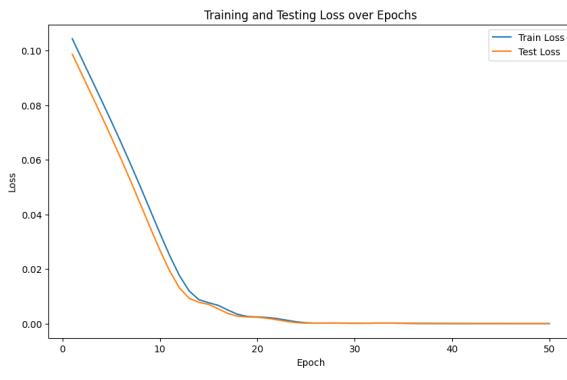


Figure 5: Training and test loss over epochs for QLSTM model.

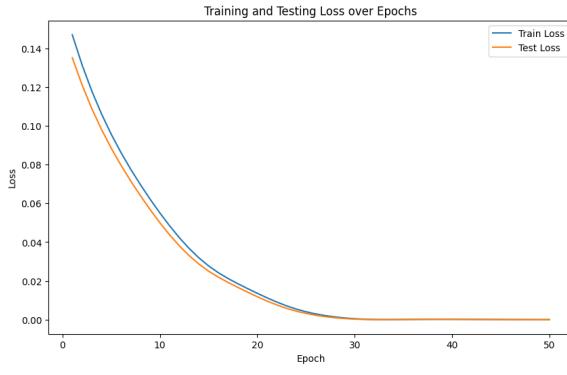


Figure 6: Training and test loss over epochs for QGRU model.

All models become stable at 30 epochs. After 50 epochs, the final MAE and RMSE for the test sets of all the approaches are summarized in Table 2.

Model	MAE	RMSE
LSTM	1.582×10^{-3}	2.137×10^{-3}
QLSTM	5.316×10^{-4}	8.110×10^{-4}
QGRU	2.981×10^{-4}	4.364×10^{-4}

Table 2: Experiment 1: Final MAE and RMSE values for the models.

The results in Table 2 clearly show that quantum-enhanced recurrent neural networks outperform the classical LSTM in both MAE and RMSE. This demonstrates the superior capability of QRNNs in capturing complex spatiotemporal patterns. Additionally, we present surface plots as shown in Figures 7 to 10.

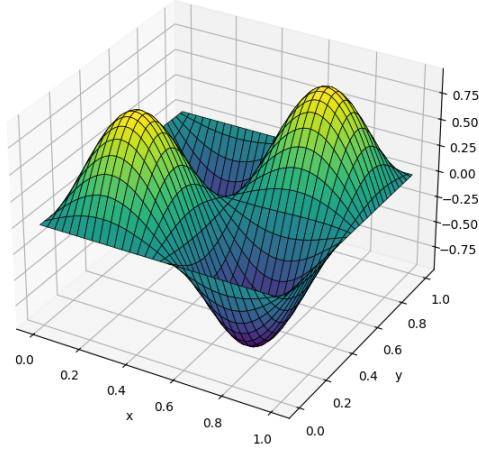


Figure 7: Surface plot of actual data for the 2D Burgers' equation.

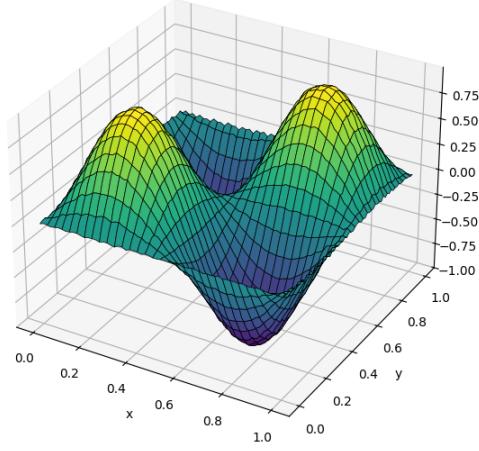


Figure 8: Surface plot of LSTM predictions for the 2D Burgers' equation.

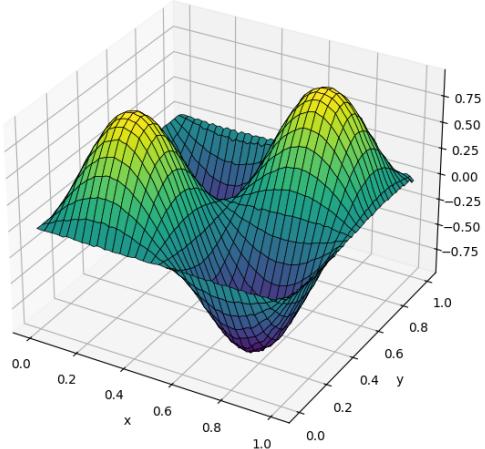


Figure 9: Surface plot of QLSTM predictions for the 2D Burgers' equation.

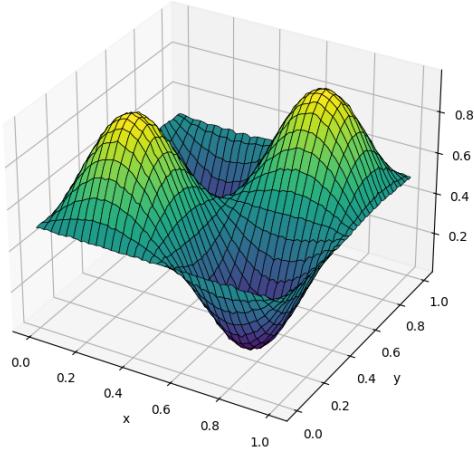


Figure 10: Surface plot of QGRU predictions for the 2D Burgers' equation.

The QGRU demonstrates the most accurate and smooth predictions among the models, whereas the QLSTM and LSTM exhibit more fragmented details near the edges. Nonetheless, all models successfully capture the overall trend.

3.2 Experiment 2: Gray-Scott Reaction-Diffusion System

The Gray-Scott model [27] is governed by the following partial differential equations, which describe the evolution of two chemical species, u and v , over time:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1 - u), \quad (11)$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F + k)v, \quad (12)$$

The variables u and v denote the concentrations of two interacting chemical species in the system. The diffusion coefficients, D_u and D_v , characterize the rate at which species u and v diffuse through the spatial domain, respectively. The parameter F represents the feed rate of species u , indicating the influx of this chemical into the system. Additionally, k denotes the removal rate of species v , determining how quickly it is depleted from the system. Together, these parameters govern the reaction-diffusion dynamics, shaping the spatial and temporal evolution of the species' concentrations.

The system begins with initial random perturbations and evolves into either stable or dynamic patterns based on the values of D_u , D_v , F , and k . The Gray-Scott model, known for its ability to generate self-organizing structures such as stripes, spots, and maze-like patterns, is simulated on a 64×64 spatial grid. In this experiment, the parameters are set to $D_u = 0.16$, $D_v = 0.08$, $F = 0.035$, $k = 0.060$, with spatial and temporal discretization steps of $dx = 1.0$ and $dt = 1.0$, respectively.

The simulation runs for 1000 time steps, and the concentrations u and v are recorded every 10 steps, generating a time series of spatial data. The QRNNs are trained to predict the future evolution of the concentration fields based on a sequence of previous time steps. The key hyperparameters for the models are the same as experiment 1.

After 100 epochs, the final MAE and RMSE for the predictions are summarized in Table 3 and Table 4.

Model	MAE	RMSE
LSTM	5.256×10^{-5}	8.669×10^{-5}
QLSTM	1.509×10^{-5}	2.130×10^{-5}
QGRU	5.747×10^{-5}	7.821×10^{-5}

Table 3: Experiment 2: Final MAE and RMSE values for Methods, Variable u .

Model	MAE	RMSE
LSTM	2.446×10^{-5}	4.060×10^{-5}
QLSTM	1.041×10^{-5}	1.347×10^{-5}
QGRU	5.386×10^{-5}	6.342×10^{-5}

Table 4: Experiment 3: Final MAE and RMSE values for Methods, Variable v .

QLSTM consistently achieves the lowest MAE and RMSE across both datasets, outperforming traditional LSTM and QGRU.

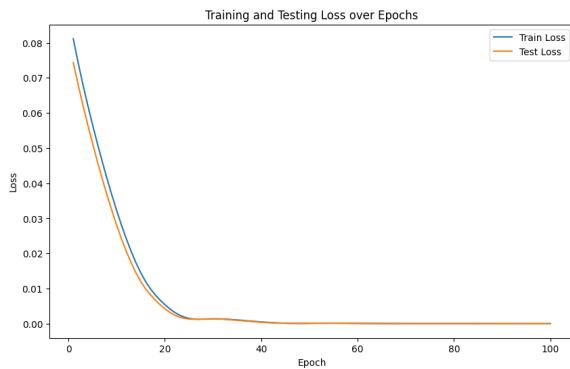


Figure 11: Training and test loss over epochs for LSTM.

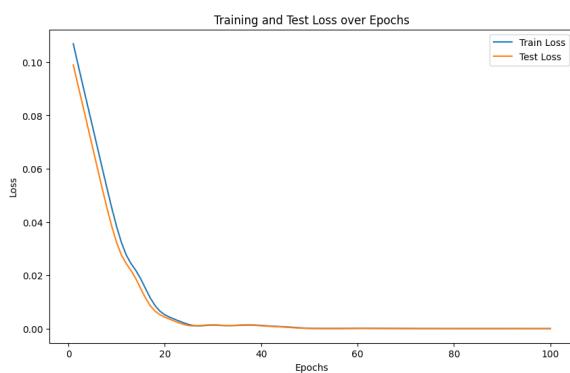


Figure 12: Training and test loss over epochs for QLSTM.

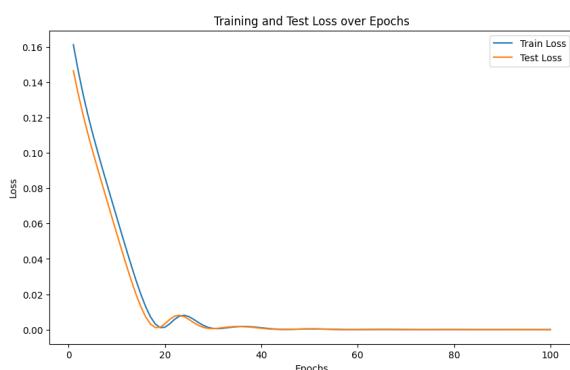


Figure 13: Training and test loss over epochs for QGRU.

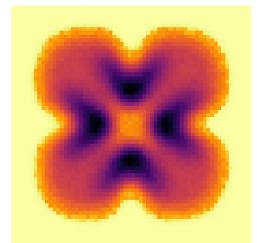
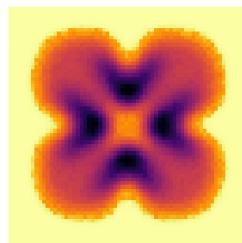


Figure 14: Actual Data vs LSTM Heatmaps for variable u .

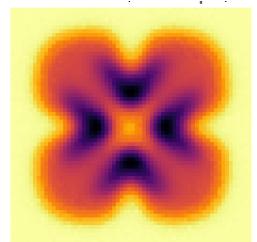
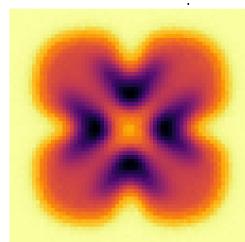


Figure 15: Actual Data vs QLSTM Heatmaps for variable u .

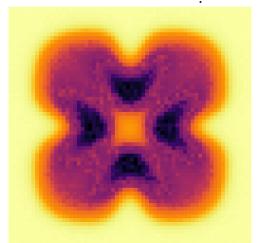
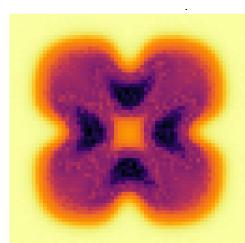


Figure 16: Actual Data vs QGRU Heatmaps for variable u .

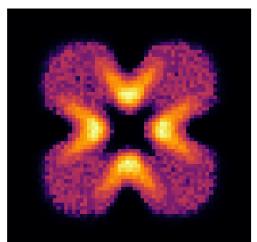
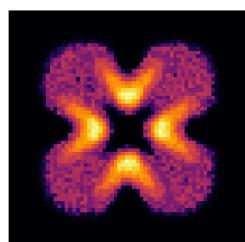


Figure 17: Actual Data vs LSTM Heatmaps for variable v .

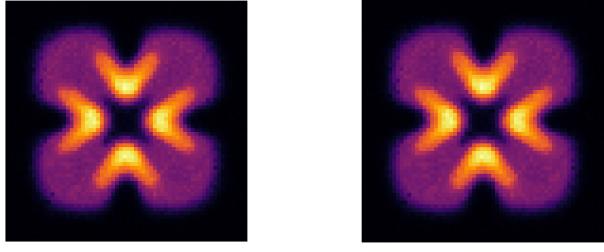


Figure 18: Actual Data vs QLSTM Heatmaps for variable v .

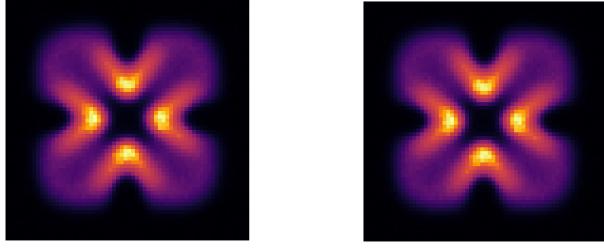


Figure 19: Actual Data vs QGRU Heatmaps for variable v .

Figures 11-13 show the loss curves for all the models over epochs, smooth and consistent decreases in loss for each model are observed. Additionally, Figures 14-19 present example heatmaps of the spatial-temporal evolution of the exact and predicted 2D Gray-Scott solutions for all models, on variables u and v . It is worth to mention, since the dataset was split randomly, the testing ground truth values may differ between models. To facilitate direct comparisons, separate heatmaps of the ground truth values are provided for each example alongside their respective predictions. Overall, all models demonstrate quite accurate predictions.

3.3 Experiment 3: Hamilton-Jacobi-Bellman Equation

For Experiment 3, we consider the 2D Hamilton-Jacobi-Bellman (HJB) equation [28], a cornerstone in optimal control theory. The HJB equation used is:

$$\frac{\partial V}{\partial t} - \frac{1}{2} \left[\left(\frac{\partial V}{\partial x} \right)^2 + \left(\frac{\partial V}{\partial y} \right)^2 \right] = 0, \quad (13)$$

where $V(x, y, t)$ represents the value function describing the optimal cost-to-go. This equation is derived by minimizing the Hamiltonian:

$$\mathcal{H}(x, y, u^*, \nabla V) = -\frac{1}{2} \left[\left(\frac{\partial V}{\partial x} \right)^2 + \left(\frac{\partial V}{\partial y} \right)^2 \right], \quad (14)$$

with the optimal control variables u_x^* and u_y^* expressed as:

$$u_x^* = -\frac{\partial V}{\partial x}, \quad (15)$$

$$u_y^* = -\frac{\partial V}{\partial y}. \quad (16)$$

This equation captures the spatiotemporal evolution of the value function and serves as a benchmark for evaluating our proposed modeling approach.

We simulate the 2D HJB equation using a finite difference method. The computational domain is discretized into a grid of 50×50 points over $x \in [0, 1]$ and $y \in [0, 1]$. The value function $V(x, y, t)$ is evolved over 100 time steps with a time step $dt = 0.0005$.

The spatial gradients $\frac{\partial V}{\partial x}$ and $\frac{\partial V}{\partial y}$ are computed using central differences with appropriate boundary conditions. The Hamiltonian is calculated as:

$$H = \frac{1}{2} \left[\left(\frac{\partial V}{\partial x} \right)^2 + \left(\frac{\partial V}{\partial y} \right)^2 \right]. \quad (17)$$

The value function is updated:

$$V^{n+1} = V^n - dt \cdot H, \quad (18)$$

where V^n is the value function at time step n . Boundary conditions are applied to ensure $V = 0$ at the domain boundaries.

The value function data generated by the simulation is preprocessed and normalized for training. An autoencoder is trained to reduce the dimensionality of the spatial data from $50 \times 50 = 2500$ to a latent size of 16.

As seen in the Table 5, QLSTM achieved the best results, with an MAE of 1.365×10^{-5} and an RMSE of 1.980×10^{-5} . The QGRU also demonstrated good performance but slightly underperformed compared to QLSTM, with an MAE of 3.183×10^{-5} and an RMSE of 4.548×10^{-5} .

Models	MAE	RMSE
LSTM	1.741×10^{-2}	2.585×10^{-2}
QLSTM	1.365×10^{-5}	1.980×10^{-5}
QGRU	3.183×10^{-5}	4.548×10^{-5}

Table 5: Experiment 3: Final MAE and RMSE values for the models.

The following figures show the training and test loss over epochs for the models.



Figure 20: Training and test loss over epochs for LSTM.

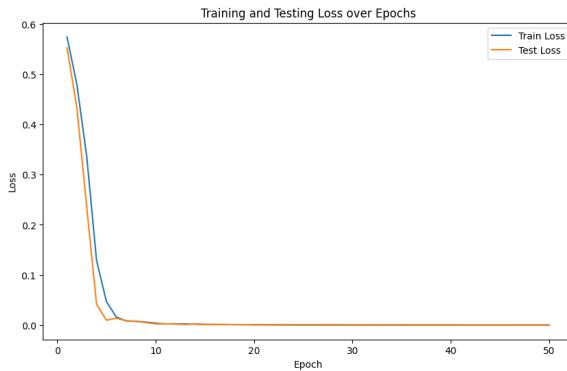


Figure 21: Training and test loss over epochs for QLSTM.

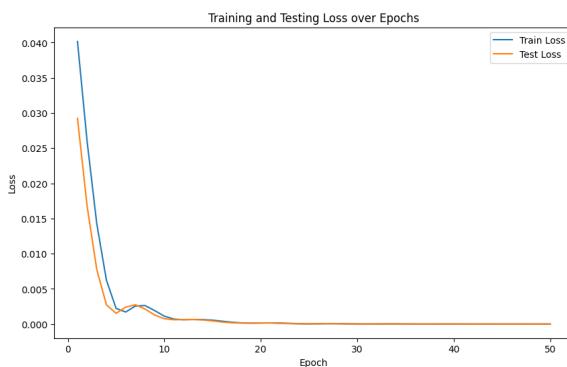


Figure 22: Training and test loss over epochs for QGRU.

In the comparison of losses, the LSTM model exhibits unstable behavior, with noticeable oscillations and spikes throughout the epochs. In contrast, the quantum-based models consistently demonstrate smooth and steadily decreasing loss curves. To further evaluate the performance of these models, the actual and predicted value functions are compared at specific time steps. The following figures display 2D heatmaps and surface plots of the actual and predicted value functions for these selected time steps.

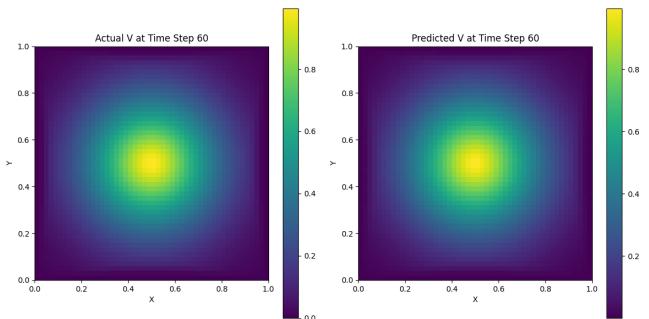


Figure 23: Heatmap: Actual vs LSTM.

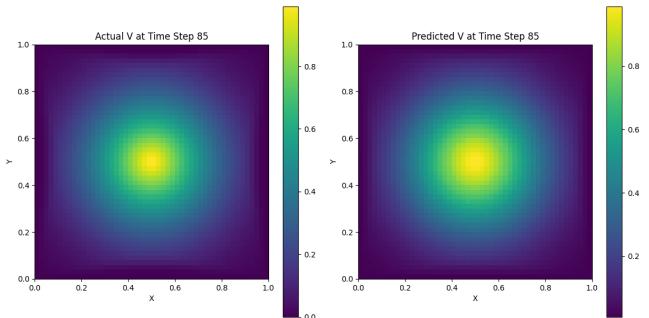


Figure 24: Heatmap: Actual vs QLSTM.

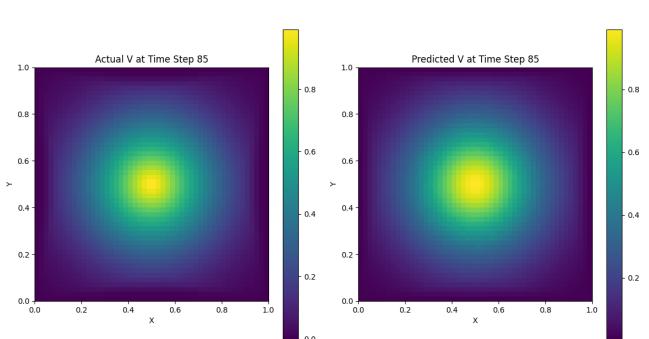


Figure 25: Heatmap: Actual vs QGRU.

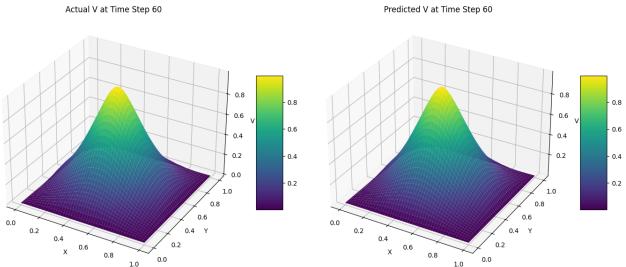


Figure 26: Surface plot: Actual vs LSTM.

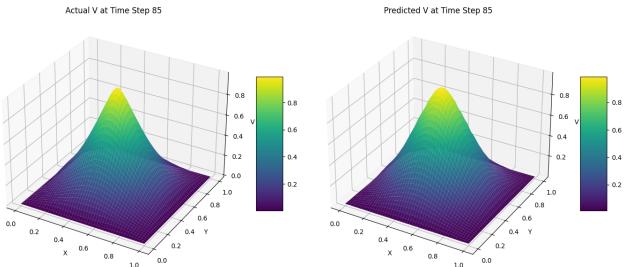


Figure 27: Surface plot: Actual vs QLSTM.

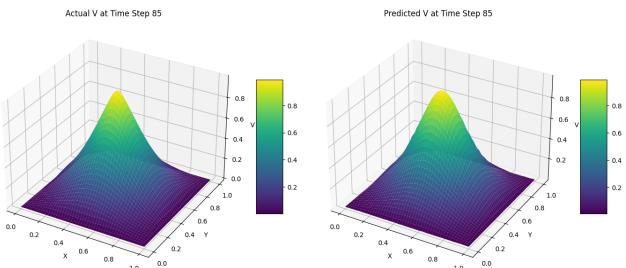


Figure 28: Surface plot: Actual vs QGRU.

From the selected timestep plots, it is evident that the models successfully capture the overall trends, maybe some finer details are not fully represented, the performance is still promising. We will go one more step and test the higher dimension problem in the next experiment.

3.4 Experiment 4: Michaelis–Menten Reaction-Diffusion System

For Experiment 4, we consider the Michaelis–Menten reaction-diffusion system [29], which models enzymatic reactions with diffusion. The governing equation is given by:

$$\frac{\partial u}{\partial t} = d \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) - \frac{u}{1+u}, \quad (19)$$

$$0 < x, y, z < 1, \quad t > 0, \quad (20)$$

where $u(x, y, z, t)$ represents the substrate concentration, $d = 0.2$ is the diffusion coefficient, and $\frac{u}{1+u}$ is the nonlinear reaction term. The computational domain is a unit cube, with homogeneous Dirichlet boundary conditions $u(x, y, z, t) = 0$ on the boundary, and the initial condition $u(x, y, z, 0) = 1$ in the interior.

The domain is discretized into a grid of $32 \times 32 \times 32$, with spatial steps $\Delta x = \Delta y = \Delta z = 1/31$. The system is evolved over time using a finite difference method with a time step $\Delta t = 10^{-4}$ up to $t = 1.0$, resulting in 10,000 time steps. Snapshots of the solution are recorded at 20 evenly spaced intervals for training and evaluation.

The resulting spatial data from each time step is flattened and normalized. An autoencoder is then employed to reduce the dimensionality of the data from $32^3 = 32,768$ to a latent space of size 16. The autoencoder architecture consists of five layers in both the encoder and decoder, utilizing hyperbolic tangent activations for the hidden layers and a sigmoid activation for the output layer. The autoencoder is trained for 100 epochs with a batch size of 32.

Models	MAE	RMSE
LSTM	5.812×10^{-3}	9.661×10^{-3}
QLSTM	4.041×10^{-3}	5.061×10^{-3}
QGRU	3.151×10^{-3}	3.418×10^{-3}

Table 6: Experiment 4: Final MAE and RMSE values for the models.

For this experiment, all the models are trained for 200 epochs, the following figures illustrate the training and test losses for the three models over the epochs.

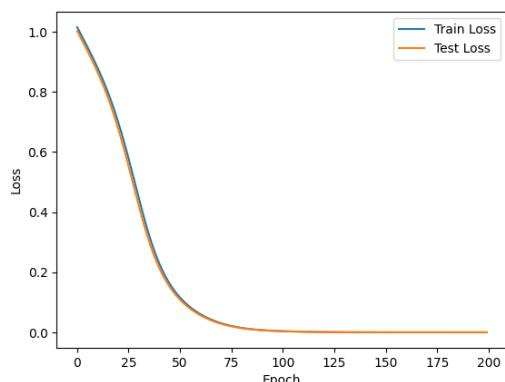


Figure 29: Training and test loss over epochs for LSTM.

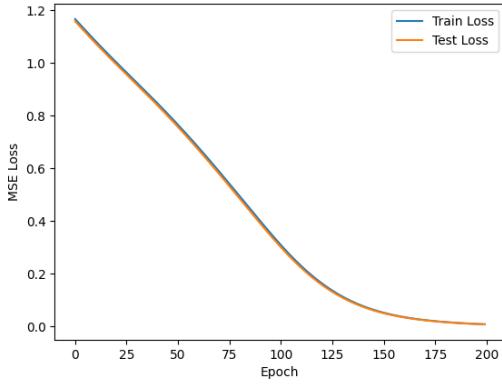


Figure 30: Training and test loss over epochs for QL-STM.

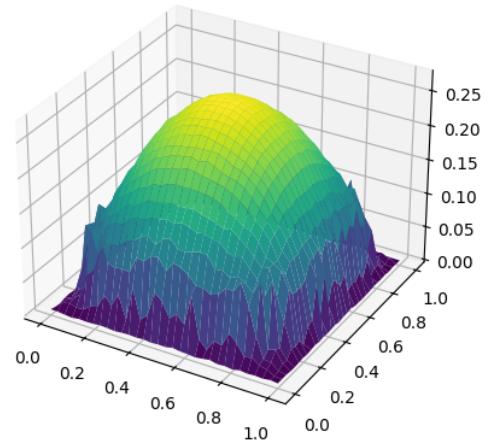


Figure 33: Predicted Slices LSTM.

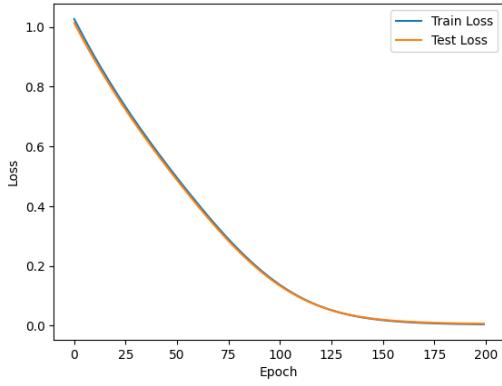


Figure 31: Training and test loss over epochs for QGRU.

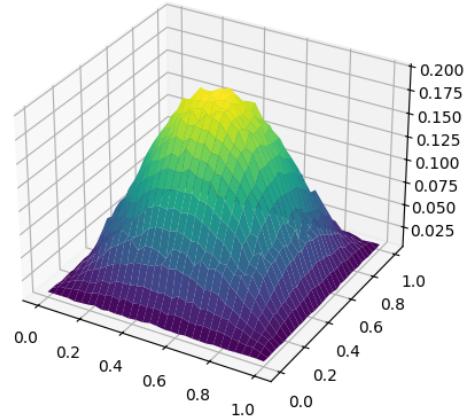


Figure 34: Predicted Slices QLSTM.

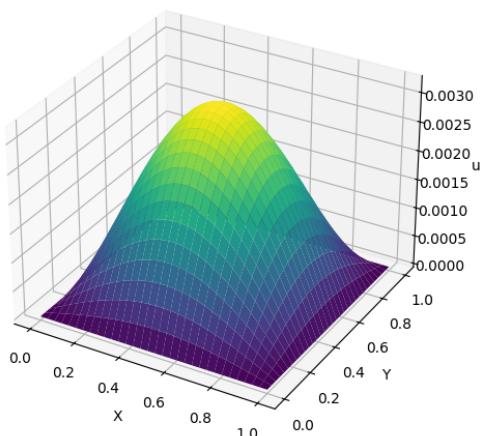


Figure 32: Predicted Slices Ground Truth.

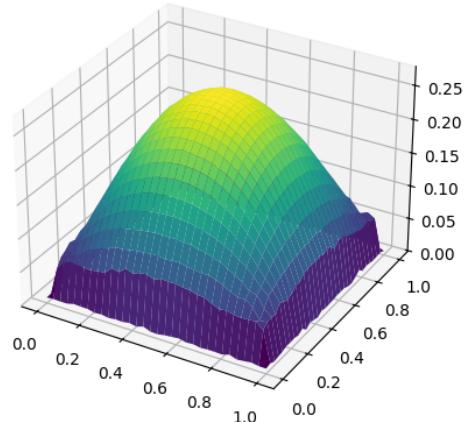


Figure 35: Predicted Slices QGRU.

From the surface plots, it is clear that the models face more challenges compared to the previous 2-D problems. The LSTM model produces predictions with noticeable artifacts, par-

ticularly at the edges, indicating difficulty in capturing the overall structure. Similarly, QGRU shows artifacts, but its predictions are significantly smoother than those of the LSTM, especially at the peak of the smaller hill. Among all the models, QLSTM demonstrates the best overall shape, although it still requires further refinement to capture finer details accurately.

It is important to note that since the higher-dimensional data are encoded using the autoencoder before performing time series predictions in the latent space, the reconstructed results are highly dependent on the performance of the encoder-decoder. The effectiveness of the predictions, therefore, relies on the autoencoder's ability to preserve key features of the original data during encoding and reconstruction for this algorithm.

4 Conclusion

The encoder-decoder Quantum Recurrent Neural Networks algorithms, specifically QLSTM and QGRU, provide a promising framework for solving nonlinear time-dependent PDEs by integrating Variational Quantum Circuits into recurrent architectures. Results from four numerical experiments demonstrate that QRNNs achieve competitive accuracy, with QLSTM consistently outperforming classical LSTM in stability and predictive precision.

As quantum hardware advances, QRNNs have the potential to become a useful tool in computational science, offering a novel approach for efficiently solving high-dimensional PDEs while improving accuracy, and stability.

References

- [1] Özişik, M. Necati, *Finite Difference Methods in Heat Transfer*. CRC Press, 2017.
- [2] Dhatt, Gouri; Lefrançois, Gilbert; and Touzot, Guy, *Finite Element Method*. Wiley, 2012.
- [3] Eymard, R.; Gallouët, T.; and Herbin, R., *The Finite Volume Method*. Springer, 2000.
- [4] Raissi, Maziar; Perdikaris, Paris; and Karniadakis, G. E., “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, 378: 686–707, 2019.
- [5] Liu, R. and Gerstoft, Peter, “SD-PINN: Physics-Informed Neural Networks for Spatially Dependent PDEs,” *ICASSP 2023 — 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, 2023, pp. 1–5.
- [6] Sharma, P.; Evans, L.; Tindall, M.; et al., “Stiff-PDEs and Physics-Informed Neural Networks,” *Archive of Computational Methods in Engineering*, 30: 2929–2958, 2023.
- [7] Rumelhart, David E.; Hinton, Geoffrey E.; and Williams, Ronald J., “Learning representations by back-propagating errors,” *Nature*, 323: 533–536, 1986.
- [8] Bahdanau, D.; Cho, K.; and Bengio, Y., “Neural Machine Translation by Jointly Learning to Align and Translate,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.
- [9] Graves, A., “Generating Sequences with Recurrent Neural Networks,” *arXiv preprint arXiv:1308.0850*, 2013.
- [10] Sutskever, I.; Vinyals, O.; and Le, Q. V., “Sequence to Sequence Learning with Neural Networks,” in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, pp. 3104–3112, 2014.
- [11] Hochreiter, Sepp; and Schmidhuber, Jürgen, “Long short-term memory,” *Neural Computation*, 9(8): 1735–1780, 1997.
- [12] Cho, Kyunghyun; Van Merriënboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; and Bengio, Yoshua, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.
- [13] Saha, Priyabrata; Dash, Saurabh; and Mukhopadhyay, Saibal, “Physics-incorporated convolutional recurrent neural networks for source identification and

- forecasting of dynamical systems," *Neural Networks*, 144: 359–371, 2021.
- [14] Hu, Y.; Zhao, T.; Xú, S.; Lin, L.; and Xu, Z., "Neural-PDE: a RNN based neural network for solving time dependent PDEs," *Communications in Information and Systems*, 22: 223–245, 2020.
- [15] Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y., "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," *arXiv preprint arXiv:1409.1259*, 2014.
- [16] Sirignano, J.; and Spiliopoulos, K., "DGM: A Deep Learning Algorithm for Solving Partial Differential Equations," *Journal of Computational Physics*, 375: 1339–1364, 2018.
- [17] Wu, B.; Hennigh, O.; Kautz, J.; Choudhry, S.; and Byeon, W., "Physics-Informed RNN-DCT Networks for Time-Dependent Partial Differential Equations," in: Groen, D.; de Mulatier, C.; Paszynski, M.; Krzhizhanovskaya, V. V.; Dongarra, J. J.; and Sloot, P. M. A. (eds), *Computational Science – ICCS 2022*, ICCS 2022. *Lecture Notes in Computer Science*, vol. 13351, Springer, Cham, 2022.
- [18] Hunout, Josephine, Sylvain Laizet, and Lorenzo Iannucci, *A New Variational Quantum Algorithm Based on Lagrange Polynomial Encoding to Solve Partial Differential Equations*, arXiv preprint [arXiv:2407.16363](#), 2024.
- [19] Choi, Minjin, and Hoon Ryu, *A Variational Quantum Algorithm for Tackling Multi-dimensional Poisson Equations with Inhomogeneous Boundary Conditions*, arXiv preprint [arXiv:2411.03009](#), 2024.
- [20] Childs, Andrew M., Jin-Peng Liu, and Aaron Ostrander, *High-Precision Quantum Algorithms for Partial Differential Equations*, *Quantum*, vol. 5, p. 574, 2021.
- [21] Mitarai, Kosuke; Negoro, Makoto; Kitagawa, Masahiro; and Fujii, Keisuke, "Quantum Circuit Learning," *Physical Review A*, 98: 032309, 2018.
- [22] Schuld, Maria; Bocharov, Alex; Svore, Krysta Marie; and Wiebe, Nathan, "Circuit-Centric Quantum Classifiers," *Physical Review A*, 2018.
- [23] Chen, S. Y.-C.; Yang, C.-H. H.; Qi, J.; Chen, P.-Y.; Ma, X.; and Goan, H.-S., "Variational Quantum Circuits for Deep Reinforcement Learning," *IEEE Access*, 8: 141007–141024, 2020.
- [24] Chen, Samuel Yen-Chi; Yoo, Shinjae; and Fang, Yao-Lung L., "Quantum Long Short-Term Memory," in *ICASSP 2022 — 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8622–8626, 2020.
- [25] Chen, Samuel Yen-Chi; Fry, Daniel; Deshmukh, Amol; Rastunkov, Vladimir; and Stefanski, Charlee, "Reservoir Computing via Quantum Recurrent Neural Networks," *arXiv preprint arXiv:2211.02612*, 2022.
- [26] Burgers, J. M., "A Mathematical Model Illustrating the Theory of Turbulence," *Advances in Applied Mechanics*, vol. 1, pp. 171–199, 1948.
- [27] Gray, P. and Scott, S. K., *Autocatalytic Reactions in the Isothermal, Continuous Stirred Tank Reactor: Oscillations and Instabilities in the System $A + 2B \rightarrow 3B$, $B \rightarrow C$* , Chemical Engineering Science, vol. 39, no. 6, pp. 1087–1097, 1984.
- [28] Bardi, M. and Capuzzo-Dolcetta, I., *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Birkhäuser, 1997.
- [29] Murray, J. D., *Mathematical Biology I: An Introduction*, 3rd ed., Springer, 2002.