SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Computational Science and Engineering

Model Predictive Control using Quantum Long Short-Term Memory in Reinforcement Learning Environments

Siva Karthikeya Mandarapu

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Computational Science and Engineering

Model Predictive Control using Quantum Long Short-Term Memory in Reinforcement Learning Environments

Author:	Siva Karthikeya Mandarapu
Supervisor(s):	Prof. Dr. Christian B. Mendl and PD. Dr. habil. Jeanette Miriam Lorenz
Advisor(s):	Dr. Daniel Hein and Dr. Steffen Udluft
Submission Date:	November 6th, 2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, November 6th, 2024 Siva Karthikeya Mandarapu

Acknowledgments

I would like to express my deepest gratitude to my advisors, Dr. Daniel Hein and, Dr. Steffen Udluft from Siemens for their invaluable guidance, support, and encouragement throughout my master's thesis. Their vast knowledge, constructive criticism, and prompt feedback helped shape my ideas and made this thesis possible.

I would like to thank my supervisors, Prof. Dr. Christian B. Mendl and PD. Dr. habil. Jeanette Miriam Lorenz, for supervising this work and giving me good ideas during our meetings.

I would also like to thank the faculty members of the Chair of Scientific Computing for their continuous support and for providing me with a challenging yet rewarding academic experience. Their passion and dedication to teaching and research have been a great source of inspiration to me.

I am indebted to my family and friends for their unwavering support, encouragement, and patience during this challenging journey. Their love and faith in me have been my constant motivation.

Abstract

Model Predictive Control (MPC) relies on precise models that capture the dynamics of the systems that should be controlled. Besides classical techniques like least squares, the effectiveness of differential methods like Long Short-Term Memory (LSTM) for sequence and temporal dependency data modeling has already been established. With the growing research interest in Quantum Machine Learning (QML), exploring the quantum variants of system identification methods that could be used in the MPC strategy for control-related tasks is particularly interesting. This thesis investigates system identification within MPC, leveraging Quantum Long Short-Term Memory (quantum LSTM) networks and comparing its performance against its classical counterpart. We demonstrate the potential of quantum LSTM on the standard Fully Observable Cart-Pole benchmark (FOCP benchmark) and the more challenging Partially Observable Cart-Pole benchmark (POCP benchmark), where we omit the velocity information from the cart-pole data. We implement Particle Swarm Optimization (PSO) for action sequence optimization in our MPC strategy. PSO performs an online optimization of an action sequence and outputs an action for a given system state. Through various experiments, we present the efficacy of the models in their ability to learn the system dynamics and evaluate the viability of using quantum LSTM as system identification models in an MPC framework using different metrics.

Contents

1 Introduction

1.1 Background and Motivation

Model Predictive Control (MPC) has gained considerable attention in control theory over the years, particularly for managing complex, multi-variable systems that require accurate real-time decisions. Accurate system models play a vital role in an MPC framework to predict the future behavior of systems and determine optimal control actions over a finite time horizon [1]. Machine learning models have become powerful tools for understanding complex systems. By acting as function approximators, they offer efficient methods for system identification, presenting a viable alternative to traditional first-principles models when it comes to capturing how a system behaves.

Long Short-Term Memory (LSTM) networks have gained recognition as a powerful tool for capturing sequence data and modeling temporal dependencies [2]. By addressing the vanishing gradient problem in traditional Recurrent Neural Networks (RNNs), LSTMs have shown remarkable performance in various prediction tasks, including those related to control systems with reduced modeling effort and without any physical knowledge [3]. The network's ability to retain long-term dependencies in time series data makes it a good candidate for system identification within the MPC framework.

Now that quantum computers are a reality, a growing field of research is focussing on integrating quantum computing techniques with classical machine learning methods. A key approach in this area is using Variational Quantum Circuits (VQCs) [4]. These are interesting models for Quantum Machine Learning (QML) because they can run on near-term quantum devices without good error correction.

This thesis focuses on exploring whether quantum computing techniques, specifically quantum LSTM networks built using VQCs, can compete with classical LSTM networks in system identification within the MPC framework. We aim to assess whether quantum-enhanced models can offer tangible improvements in system identification tasks relevant to MPC. Previous works on quantum LSTMs have shown that they learn more information than their classical counterparts in fewer epochs, and they are better at learning local features than LSTMs when there is a temporal structure involved in the data [5]. Quantum LSTMs have also been used as Q-function approximators to realize quantum deep recurrent Q-function learning in [6]. They have shown higher stability and average scores compared to their classical counterparts. However, to our knowledge, quantum LSTMs have not been used previously to model the dynamics of reinforcement learning (RL) environments and implement them in the MPC strategy, which we aim to do in this thesis.

As quantum computing transitions from theoretical interest to practical applications, there is a growing need to understand how quantum algorithms can be applied in real-world

control systems, especially within the limitations of Noisy Intermediate-Scale Quantum (NISQ) devices. Current quantum computers are prone to noise and lack full error correction. In this thesis, we demonstrate the potential of the quantum LSTMs on the Fully Observable Cart-Pole (FOCP) environment from the OpenAI Gym and also on the more challenging Partially Observable Cart-Pole (POCP) environment [7, 8].

In summary, this thesis explores a new approach to implementing MPC through QML techniques, specifically focusing on applying VQC-based quantum LSTMs for system identification. By leveraging the unique properties of quantum computing, this thesis aims to provide insights into the feasibility and effectiveness of QML models in real-time control systems.

1.2 Problem Statement

Traditionally, system identification for MPC has leveraged classical methods, such as least squares and neural networks, including RNNs and LSTM networks, to capture the system dynamics in sequential data [9][10]. However, with the advent of quantum computing, there is increasing interest in exploring quantum-enhanced machine learning models, specifically VQCs or VQC-based models, to understand whether they can improve prediction accuracy or computational efficiency in control-related tasks. VQCs were used in model-based approach for Quantum Reinforcement Learning (QRL) to learn the dynamics of the cart-pole environment from observational data and then use it as a surrogate model [11]. They were also used as function approximators within the discrete batch-constraint deep Q-learning (BCQ) algorithm and its performance was studied on the cart-pole benchmark and compared with classical neural network-based discrete BCQ [12].

A topic that has not been thoroughly investigated is the usage of quantum LSTM models to learn sequential or temporal data. Quantum LSTM leverages VQCs to replicate the functionality of classical LSTMs. The practical implementation of quantum LSTMs to model system dynamics for complex control systems remains largely unexplored, especially in standard benchmarks such as the FOCP problem or the more challenging POCP problem, a well-known test case in control theory and RL.

Several challenges arise when considering the use of quantum LSTMs within MPC frameworks. First, there is limited literature on quantum LSTMs applied to real-world control benchmarks. Most existing work focuses on simple function approximations, which do not fully demonstrate the capabilities of quantum models in dynamic, real-time control scenarios. Furthermore, implementing quantum LSTMs on simulators like PennyLane's lightning-qubit backend [13] involves significant computational overhead, particularly in training and inference stages. Hyperparameter tuning to achieve an accurate quantum LSTM model is time-consuming, and running the model in an MPC context, where it must predict future states for optimization algorithms such as PSO, increases the computational challenges.

In this thesis, we seek to address the gap in the current research by investigating the feasibility of using quantum LSTMs for system identification in MPC. Specifically, we aim to compare the performance of quantum LSTMs with classical LSTM networks and Multi-Layer

Perceptrons (MLP), in the context of the FOCP and POCP benchmark. Our evaluation focuses on key performance metrics, such as model accuracy measured through the loss values and prediction quality in rollouts, to determine the effectiveness of quantum LSTMs as potential system identification models in MPC.

This thesis aims to contribute to the field by systematically comparing quantum and classical models in MPC, addressing whether quantum LSTMs can serve as viable alternatives for system identification and control in dynamic, real-world settings.

1.3 Objectives of the Thesis

The main objective of this thesis is to explore the potential of quantum LSTM networks for system identification in MPC. We aim to determine whether quantum models can provide tangible benefits in terms of accuracy, or prediction quality in comparison with their classical counterparts.

To achieve this goal, the specific objectives of the thesis are as follows:

1. **Implement a quantum LSTM using VQCs:** Implement a hybrid quantum LSTM model used in literature by placing VQCs in the classical LSTM's gate structure and train this model on the cart-pole benchmark data generated using a random policy.

2. **Compare quantum LSTMs with classical LSTMs and other models:** Perform a comparative study between quantum LSTMs, classical LSTMs, and MLPs on the FOCP and the POCP benchmark. Use metrics such as model loss, prediction accuracy in rollouts, and overall performance in MPC.

3. **Evaluate the feasibility of quantum LSTMs in MPC:** Assess the viability of using quantum LSTMs as system identification models in an MPC framework. Evaluate their effectiveness in predicting future system states during the optimization phase, particularly in the context of PSO.

4. **Explore the applicability of quantum LSTMs for complex control problems:** Provide insights into whether quantum LSTMs can be effectively applied to more complex control problems like the POCP benchmark beyond simple benchmarks which will show its true potential.

By achieving these objectives, this thesis aims to provide a comprehensive evaluation of quantum LSTMs in MPC and contribute to the broader understanding of how QML can be applied to real-world control systems.

1.4 Structure of the Thesis

This thesis is organized into five chapters. In Chapter 1, we introduce the topic, outlining the work's background, motivation, problem statement, and objectives. In Chapter 2, we

present the theoretical background, covering key concepts such as MPC, RL, PSO, quantum computing, and VQCs, while also comparing classical and quantum approaches. In Chapter 3, we describe the methodology, including the experimental setup, implementation of models, MPC framework used in the study, and evaluation criteria. The results of the experiments, along with a discussion of their significance, are presented in Chapter 4. Finally, Chapter 5 concludes the thesis by summarizing the findings, highlighting the contributions, and proposing directions for future research.

2 Theoretical Background

This chapter discusses the theoretical background of various concepts used in the thesis. We begin by elaborating on MPC, which is the central focus of this thesis in Section 2.1. We then briefly discuss the fundamental concepts of RL in control systems in Section 2.2. In Section 2.3, we introduce the fundamental concepts involved in quantum computing and VQCs. Finally, we discuss the classical and quantum variants of LSTM networks extensively used in this thesis in Section 2.4.

2.1 Model Predictive Control

This section introduces the fundamental concepts of MPC as discussed in [14, 1].

2.1.1 Model Predictive Control Framework

Model Predictive Control is an advanced control strategy that optimizes control actions by solving a finite-horizon optimization problem at each control step. Unlike traditional control methods that rely on pre-defined control rules, MPC actively adjusts its control inputs by predicting future system behavior based on a dynamic model. This ability makes MPC particularly effective for managing complex, multivariable systems with constraints. A neat illustration of MPC is shown in Figure 2.1.

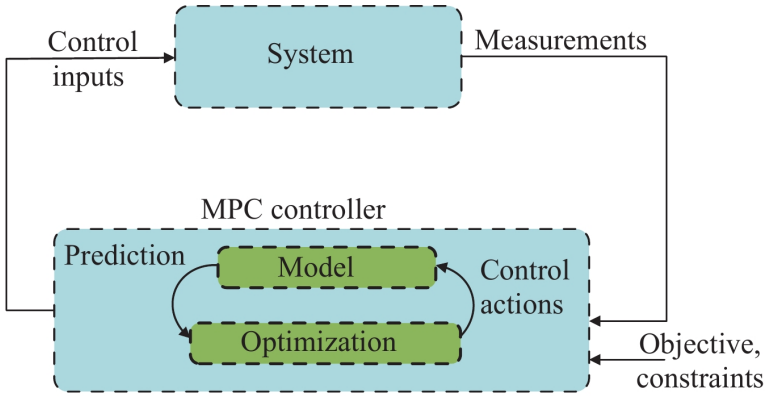

Figure 2.1: Illustration of model predictive control [15]

The central concept behind MPC is to use a model of the system to predict its future evolution over a finite prediction horizon. At each time step t, an optimal sequence of control inputs $\mathbf{u}_{t:t+N-1}$ is computed by solving an optimization problem over the following

N time steps, where N is the length of the prediction horizon. The first control action from this sequence u_t is then applied to the system, and the process repeats at the next step, incorporating updated state measurements.

System Model

MPC relies on a discrete-time mathematical model of describing the dynamics of the system simply because of the convenience it provides for digital computer implementation. This model can take various forms, including linear, nonlinear, and hybrid models. One common way to express the system's dynamics in MPC is through the following discrete-time state-space representation:

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k), \\ \mathbf{y}_k &= g(\mathbf{x}_k) \end{aligned} \tag{2.1.1}$$

where:

- $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector at time step k, representing the internal states of the system,
- $\mathbf{u}_k \in \mathbb{R}^m$ is the control input vector at time step k,
- $\mathbf{y}_k \in \mathbb{R}^p$ is the output vector (system outputs being controlled),
- $f(\cdot)$ represents the system's dynamics model, which can be linear or nonlinear, and
- $g(\cdot)$ represents the measurement function, mapping states to outputs.

The function $f(\mathbf{x}_k, \mathbf{u}_k)$ represents the system's dynamics, capturing how the current state $\mathbf{x}_k$ evolves in response to the control input $\mathbf{u}_k$, and $g(\mathbf{x}_k)$ relates the state to observable outputs. For more complex cases, this nonlinear model helps in better prediction of system behavior. To model the nonlinear dynamics of the cart-pole system in this thesis, we aim to use a quantum LSTM and compare it with other classical LSTM and MLP. We will learn more about the architecture of the models used in this thesis in Section 2.4.

Optimization Problem

At each control step, MPC solves an optimization problem to determine the optimal control sequence $\mathbf{u}_{k:k+N-1}$. The optimization objective is to minimize a cost function J over the prediction horizon N, which can be generally expressed as:

$$J = \phi(y_{k+N|k}) + \sum_{j=0}^{N-1} L(y_{k+j|k}, u_{k+j|k}, \Delta u_{k+j|k}) \tag{2.1.2}$$

where:

- $\phi(y_{k+N|k})$ represents the terminal cost based on the predicted output at the end of the prediction horizon,

- $L(\cdot)$ is the stage cost function, which evaluates the cost at each time step,
- $\Delta u_{k+j|k} = u_{k+j|k} - u_{k+j-1|k}$ denotes the change in control inputs between consecutive time steps,
- N is the prediction horizon length.

The objective function J balances tracking performance (minimizing deviations from a desired reference trajectory) with control effort (minimizing changes in the control inputs), making it suitable for optimizing system behavior over a finite horizon. We use PSO in this thesis to solve the optimization problem.

Control Horizon and Receding Horizon Approach

A key feature of MPC is its receding horizon approach [1, 16]. Although the optimal control sequence $\mathbf{u}_{k:k+N-1}$ is computed over the entire prediction horizon N, only the first control input u_k is applied to the system. At the next time step $k+1$, the optimization problem is solved again with updated state measurements $\mathbf{x}_{k+1}$, resulting in a new control sequence $\mathbf{u}_{k+1:k+N}$.

Constraints Handling

One of the major advantages of MPC over traditional control methods is its ability to handle constraints explicitly. Constraints are typically incorporated into the optimization problem as inequality constraints on both state and control variables:

$$\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}, \quad \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}. \tag{2.1.3}$$

These constraints ensure that the control inputs and states remain within allowable ranges, which is crucial for maintaining system safety and performance within operational limits.

Solving the Optimization Problem

The optimization problem in MPC can vary in complexity depending on the nature of the system dynamics (linear or nonlinear). For linear systems, the problem typically becomes a convex quadratic programming (QP) problem, which can be solved efficiently using standard solvers. If a system exhibits nonlinear system dynamics, which is usually the case with complex industry systems and plants, numerical online optimization of sequences of control actions is the general solution [17]. This results in non-convex optimization problems. PSO and evolutionary algorithms are some of the established heuristics for solving non-convex optimization problems [18].

2.1.2 System Identification for Model Predictive Control

System identification is the process of constructing mathematical models of dynamic systems based on observed data. In control theory, accurate system identification is essential for designing effective control strategies, such as MPC. The goal is to capture the system's behavior in a form that can be used for prediction and control, usually represented by differential equations, state-space models, or transfer functions.

System identification methods can be broadly categorized into *parametric*, *non-parametric*, and *data-driven* approaches. Each offers unique advantages depending on the complexity and nature of the system being modeled.

Parametric System Identification

Parametric methods assume a specific structure for the model and estimate its parameters using data. These methods include:

- **Least Squares Estimation:** A widely used method that minimizes the squared difference between observed and predicted outputs:

$$\min_{\theta} \sum_{t=1}^{N} (y_t - \hat{y}_t(\theta))^2 \tag{2.1.4}$$

  where y_t is the observed output, $\hat{y}_t(\theta)$ is the predicted output based on parameters θ, and N is the number of observations. Least square estimation works efficiently for linear systems but may face limitations in the presence of noise or nonlinearity.

- **State-Space Identification:** Models the system using state variables, leading to a state-space representation:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t, \quad \mathbf{y}_t = C\mathbf{x}_t + D\mathbf{u}_t + \mathbf{v}_t \tag{2.1.5}$$

  where $\mathbf{x}_t$, $\mathbf{u}_t$, and $\mathbf{y}_t$ are the state, input, and output vectors, respectively, while $\mathbf{w}_t$ and $\mathbf{v}_t$ represent process and measurement noise.

- **ARX/ARMAX Models:** ARX models relate the output y_t to input u_t using a difference equation:

$$A(q)y_t = B(q)u_t + e_t \tag{2.1.6}$$

  where $A(q)$ and $B(q)$ are polynomials in the time-shift operator q, and e_t is noise. ARMAX extends ARX with an additional noise model, improving flexibility.

Non-Parametric System Identification

Non-parametric methods estimate system behavior directly from data without assuming a specific model structure. These include:

- **Impulse and Frequency Response Identification:** The system's impulse response characterizes its time-domain behavior, while frequency response analysis measures how the system reacts to sinusoidal inputs at various frequencies.
- **Correlation-Based Methods:** These methods use cross-correlation between input and output signals to infer system dynamics, effectively filtering noise in the data.

Machine Learning-Based System Identification

Machine learning techniques offer powerful tools for system identification, especially when dealing with nonlinear dynamics or large datasets. Models such as LSTM networks have proven particularly effective:

- **LSTM Networks:** LSTMs are a type of RNNs that can capture long-term dependencies and nonlinear dynamics. For system identification, LSTMs are trained on time-series data, where the inputs u_t and outputs y_t are recorded over time. The network learns relationships without requiring explicit knowledge of the system's physics, making it a good choice for complex systems.
- **Other Machine Learning Approaches:** Besides LSTMs, other models such as Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), and Support Vector Machines (SVMs) are used in system identification. FNNs are suitable for static relationships, while CNNs are effective in identifying spatial patterns. SVMs are often applied in cases with noisy data or when the system can be treated as a regression problem.

Summary of MPC's Strengths and Limitations

MPC provides a comprehensive control framework that excels at managing multivariable systems with constraints. Its ability to optimize control actions make it an attractive choice for advanced applications. However, its computational demands, model dependency, and tuning requirements can be challenging in practical implementations.

2.2 Reinforcement Learning in Control Systems

RL is a branch of machine learning in which an agent learns to make sequential decisions by interacting with its environment [19]. Unlike supervised learning, RL doesn't rely on pre-labeled data. Instead, the agent explores actions and refines its policy based on rewards or penalties, making it ideal for dynamic and uncertain environments such as control systems and robotics.

The core of RL lies in the concept of a *Markov Decision Process (MDP)*, which provides a structured approach for decision-making in scenarios where outcomes depend on both the agent's actions and stochastic elements in the environment. An MDP is characterized by the tuple (S, A, P, R, γ), where:

- S is the state space, representing all possible states $s \in S$ the agent might experience.
- A is the action space, representing the set of all actions $a \in A$ the agent can take.
- $P(s'|s, a)$ denotes the transition probability, indicating the likelihood of transitioning from state s to state s' after taking action a.
- $R(s, a)$ is the reward function, providing a numerical reward r when the agent takes action a in state s.
- γ is the discount factor, $0 \leq \gamma \leq 1$, which determines the importance of future rewards compared to immediate rewards.

The objective of the agent is to learn an optimal *policy* $\pi(a|s)$, which prescribes the best action in each state to maximize the expected cumulative reward, or *return*, over time:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}). \tag{2.2.1}$$

Balancing short-term and long-term rewards is crucial in control applications where actions often have delayed and compounding effects.

2.2.1 Key Concepts in Reinforcement Learning

Several foundational concepts play a significant role in the effectiveness of RL-based control systems:

Exploration vs. Exploitation: RL agents must navigate the trade-off between exploring new actions to gather more information (exploration) and leveraging known actions that yield the highest reward (exploitation). Effective control systems must carefully balance this trade-off to achieve optimal long-term behavior.

Value Functions: Value functions quantify the expected cumulative reward, providing a measure of the desirability of states or actions:

- *State-Value Function* $V(s)$: The expected return starting from state s under policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s]. \tag{2.2.2}$$

- *Action-Value Function* $Q(s, a)$: The expected return of taking action a in state s and following policy π thereafter:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a]. \tag{2.2.3}$$

Bellman Equation: The Bellman equation expresses the recursive nature of value functions, relating the value of a state to the immediate reward and the value of subsequent states:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R(s,a) + \gamma V^{\pi}(s')]. \tag{2.2.4}$$

This recursive definition underpins many RL algorithms and enables iterative policy improvements.

Policy-Based and Value-Based Methods: RL methods can be categorized into:

- *Value-Based Methods:* These algorithms, like Q-learning, focus on estimating the action-value function $Q(s,a)$ and deriving a policy by selecting actions that maximize Q.
- *Policy-Based Methods:* These methods, such as REINFORCE, directly optimize the policy π by maximizing expected returns, allowing them to handle continuous action spaces more effectively.

2.2.2 Challenges of Reinforcement Learning in Control Systems

Applying RL in control systems comes with distinct challenges, particularly in real-time, safety-critical applications [20]:

- **Sample Efficiency:** Gathering training data in real-world control environments is often costly and time-consuming. Improving the sample efficiency of RL algorithms is crucial for practical control applications.
- **Delayed Rewards:** Many control tasks involve delayed rewards, where the outcomes of actions become apparent only after several timesteps. RL algorithms must be able to associate actions with delayed consequences effectively.
- **Continuous State and Action Spaces:** Control problems often operate in continuous spaces (e.g., force or torque control), requiring specialized algorithms like policy gradients or actor-critic methods that can handle these continuous domains.

2.2.3 Model-Free vs. Model-Based Reinforcement Learning in Control Systems

RL methods can be broadly divided into:

Model-Free RL: In model-free RL, the agent directly learns policies or value functions based on its interactions with the environment. Algorithms like Q-learning and Proximal Policy Optimization (PPO) fall under this category. Although these methods are flexible, they often require a large amount of interaction data, making them less sample-efficient.

Model-Based RL: In model-based RL, the agent builds an internal model of the system's dynamics to simulate future states and rewards. This approach improves sample efficiency but requires accurate model estimates to avoid suboptimal control strategies. Inaccurate models can lead to instability in control applications.

2.2.4 Applications of Reinforcement Learning in Control Systems

RL has demonstrated significant potential in control applications, including:

- **Robotics:** RL has been used in robotic manipulation, autonomous navigation, and locomotion tasks, enabling robots to adapt to dynamic and unpredictable environments [21].
- **Energy Management:** RL-based strategies optimize energy consumption in smart grids and buildings by responding dynamically to fluctuations in supply and demand.

2.2.5 Particle Swarm Optimization

PSO is a population-based, stochastic optimization technique often employed to solve non-convex optimization problems [18]. In the context of **MPC**, PSO is utilized to optimize sequences of control actions over a finite prediction horizon. This section describes how PSO is integrated into the MPC framework, inspired by an RL approach [9].

Formulating RL as an Optimization Problem: In an RL problem, the agent interacts with a system observed in discrete time steps $t \in \mathbb{Z}$. The system at each time step t is described by a state vector $s_t \in S$ from the state space S. The agent selects an action vector a_t, which consists of I different control parameters, $a_t \in A \subseteq \mathbb{R}^I$. The agent's goal is to find an optimal sequence of actions that maximizes the expected return G.

Given a deterministic system described by a state transition function $m : S \times A \to S \times \mathbb{R}$, the evolution of the system can be described as:

$$(s_{t+1}, R(s_t, a_t)) = m(s_t, a_t) \tag{2.2.5}$$

where s_{t+1} is the next state and $R(s_t, a_t)$ is the immediate reward received for applying action a_t at state s_t.

The problem of finding an optimal sequence $x = (a_t, a_{t+1}, \ldots, a_{t+T-1})$, which maximizes the expected return G over a finite horizon T, can be expressed as:

$$G(s_t, x) = \sum_{k=0}^{T-1} \gamma^k R(s_{t+k}, a_{t+k}) \tag{2.2.6}$$

where $\gamma \in [0, 1]$ is a discount factor that weights future rewards. The optimal sequence x^* is determined by solving:

$$x^* = \underset{x \in [x_{\min}, x_{\max}]}{\arg\max} \; G(s_t, x) \tag{2.2.7}$$

where $[x_{\min}, x_{\max}]$ represents the feasible range for control actions, defined as:

$$x_{\min,j} \leq a_{j,k} \leq x_{\max,j}, \quad \forall j \in \{1, 2, \ldots, I\}, \quad \forall k \in \{0, 1, \ldots, T-1\}. \tag{2.2.8}$$

PSO Framework: The PSO algorithm searches for optimal action sequences x over a finite horizon T using a population of particles. Each particle represents a candidate action sequence and updates its position iteratively based on its own experience and the experience of its neighbors. The framework can be described as follows:

1. **Initialize the Particles:** Each particle i is initialized with a random position x_i within the feasible control action range:

$$x_i = (a_{i,0}, a_{i,1}, \ldots, a_{i,T-1}), \tag{2.2.9}$$

and a corresponding velocity v_i.

2. **Update the Velocity:** The velocity v_{ij} of each particle in the j-th dimension is updated based on three components:

$$v_{ij}^{(p+1)} = w \cdot v_{ij}^{(p)} + c_1 \cdot \eta_1 \cdot \left(y_{ij}^{\text{best}} - x_{ij}^{(p)}\right) + c_2 \cdot \eta_2 \cdot \left(\hat{y}_{ij} - x_{ij}^{(p)}\right) \tag{2.2.10}$$

where:

- w is the inertia weight,
- c_1 and c_2 are acceleration coefficients for cognitive and social components,
- η_1 and η_2 are random numbers drawn from a uniform distribution in [0, 1],
- y_{ij}^{best} is the particle's own best-known position, and $\hat{y}_{ij}$ is the best-known position in the particle's neighborhood.

3. **Update the Position:** The position of each particle is updated using:

$$x_{ij}^{(p+1)} = x_{ij}^{(p)} + v_{ij}^{(p+1)}. \tag{2.2.11}$$

4. **Evaluate the Fitness:** The fitness function for each particle is evaluated based on the expected return G over the prediction horizon T:

$$f(x_i) = G(s_t, x_i) = \sum_{k=0}^{T-1} \gamma^k R(s_{t+k}, a_{t+k}) \tag{2.2.12}$$

where the return G is predicted by simulating the system's response to the action sequence using the system model.

5. **Constraint Handling:** To ensure feasible control actions, the updated positions are bounded within predefined limits:

$$x_{\min} \leq x_{ij}^{(p+1)} \leq x_{\max}. \tag{2.2.13}$$

6. **Receding Horizon Control:** Although a sequence of T actions is optimized, only the first action a_0 is applied to the system at each timestep. The process is then repeated for the subsequent state s_{t+1}, following the receding horizon principle of MPC.

Advantages of PSO in MPC: The PSO-based approach in MPC offers several advantages:

- **Global Search Capability:** PSO explores the search space globally, reducing the likelihood of getting trapped in local optima.
- **Flexibility:** The algorithm does not require derivative information, making it suitable for non-differentiable or complex cost functions.
- **Adaptability:** PSO can accommodate various constraints and objectives, making it versatile for different control problems.

Challenges of PSO in MPC: Despite its advantages, PSO presents certain challenges, such as:

- **Computational Cost:** Evaluating multiple candidate solutions in each iteration can be computationally intensive, especially for high-dimensional problems.
- **Parameter Tuning:** The performance of PSO is sensitive to its parameters, such as inertia weight and acceleration coefficients. Proper tuning is essential for achieving a good balance between exploration and exploitation.

In conclusion, PSO provides flexible framework for solving optimization problems in nonlinear MPC. By leveraging global search and not relying on gradient information, it effectively addresses non-convexity and complexity in the control problem.

2.3 Quantum Computing and Variational Quantum Circuits

This section introduces the fundamentals of quantum computing and VQCs. The material presented here is based on [22, 23].

2.3.1 Fundamentals of Quantum Computing

Quantum computing is based on the principles of quantum mechanics, and it offers a different way of handling computations compared to traditional computers. While classical computers use bits to represent information as either 0 or 1, quantum computers use quantum bits or *qubits*. What is interesting about qubits is that they can be in a state of both 0 and 1 simultaneously due to superposition.

This unique ability allows quantum computers to tackle specific problems much faster than classical computers can. They harness superposition and other phenomena like entanglement and quantum interference to enhance their computational power.

Qubits and Quantum States

A *qubit* is the basic unit of quantum information, analogous to a classical bit. However, unlike a classical bit that can only be in one of two definite states (0 or 1), a qubit can exist in a

superposition of both states simultaneously. The state of a qubit is described as a vector in a two-dimensional complex vector space. Mathematically, a qubit $|\psi\rangle$ is expressed as a linear combination of the two computational basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.3.1}$$

where α and β are complex numbers known as probability amplitudes, and they satisfy the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.3.2}$$

Here, $|\alpha|^2$ is the probability of the qubit being measured in state $|0\rangle$, and $|\beta|^2$ is the probability of the qubit being measured in state $|1\rangle$.

The Bloch Sphere Representation

The state of a single qubit can also be represented geometrically using the *Bloch sphere*, a unit sphere in three-dimensional space as shown in Figure 2.2.

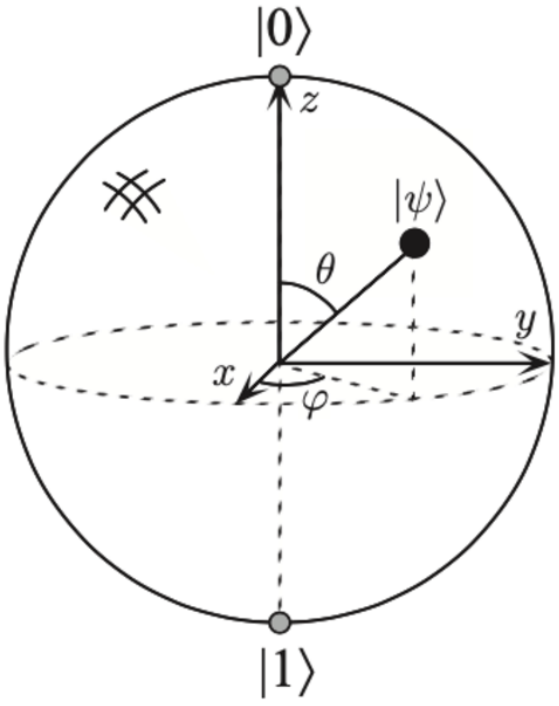

Figure 2.2: Bloch sphere [22]

The general state of a qubit can be written as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.3.3}$$

where θ and ϕ are real numbers that describe the position of the qubit on the Bloch sphere. The angles θ and ϕ are called the polar and azimuthal angles, respectively. The Bloch sphere provides an intuitive visualization of a qubit's state, with the north pole representing $|0\rangle$, the south pole representing $|1\rangle$, and any superposition of these states being a point on the surface of the sphere.

The Bloch sphere is particularly useful for visualizing single-qubit operations (quantum gates) as rotations of the qubit's state around the sphere.

Quantum Superposition and Entanglement

Quantum *superposition* is a fundamental principle that allows qubits to exist in multiple states simultaneously. A qubit in superposition is described as a linear combination of $|0\rangle$ and $|1\rangle$. When measured, however, the qubit collapses to one of the basis states with probabilities determined by the square of the probability amplitudes $|\alpha|^2$ and $|\beta|^2$.

Entanglement is a uniquely quantum phenomenon where two or more qubits become correlated in such a way that the state of one qubit depends on the state of another, even if they are spatially separated. For instance, in an entangled state, measuring one qubit immediately determines the state of the other qubit, irrespective of distance. An example of a maximally entangled state is the Bell state:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \tag{2.3.4}$$

Entanglement plays a crucial role in quantum algorithms and protocols, enabling phenomena such as quantum teleportation and superdense coding, and is essential for the power of quantum computation.

Single-Qubit Gates and Rotations

Quantum gates manipulate qubit states in a quantum circuit. A quantum gate is represented by a unitary matrix U, where $U^\dagger U = I$. Single-qubit gates correspond to rotations on the Bloch sphere, altering the state of the qubit.

Some common single-qubit gates include:

- **Pauli Gates (X, Y, Z):** These gates correspond to 180-degree rotations around the Bloch sphere's x, y, and z axes, respectively.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.3.5}$$

  The Pauli-X gate (also called the quantum NOT gate) flips the qubit's state: $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. The Pauli-Z gate flips the phase of the $|1\rangle$ state: $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$.

- **Hadamard Gate (H):** The Hadamard gate creates an equal superposition of $|0\rangle$ and $|1\rangle$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.3.6}$$

  Applying H to $|0\rangle$ results in:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{2.3.7}$$

  This gate is critical for initializing qubits into superposition states in many quantum algorithms.

- **Phase Shift Gates (RZ, S, T):** These gates introduce a phase shift in the qubit's state. The general phase shift gate is defined as:

$$R_z(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}. \tag{2.3.8}$$

  The S gate is a phase shift of $\lambda = \frac{\pi}{2}$, and the T gate is a phase shift of $\lambda = \frac{\pi}{4}$.

These gates are fundamental building blocks for more complex quantum circuits, and can be combined to perform arbitrary unitary transformations on a single qubit.

Multiple-Qubit Gates and Entanglement

While single-qubit gates perform operations on individual qubits, multi-qubit gates operate on two or more qubits and are crucial for generating entanglement. The most common multi-qubit gate is the *Controlled-NOT (CNOT)* gate. The CNOT gate flips the state of the target qubit if the control qubit is in state $|1\rangle$:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.3.9}$$

The CNOT gate is fundamental for constructing entanglement in quantum circuits. For example, applying a Hadamard gate to the control qubit followed by a CNOT gate entangles the two qubits, producing a Bell state:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \tag{2.3.10}$$

Other multi-qubit gates include the Toffoli (controlled-controlled-NOT) gate and the SWAP gate, which swaps the states of two qubits.

Quantum Circuits

Quantum circuits are the quantum analog of classical logic circuits. They consist of qubits and quantum gates, where the gates manipulate the qubits' states. A quantum circuit can be described as a series of unitary operations applied to an initial state, followed by a measurement operation.

A quantum algorithm is implemented by a quantum circuit where the sequence of quantum gates corresponds to the algorithm's steps. The circuit begins by initializing the qubits to a known state (often $|0\rangle^{\otimes n}$ for an n-qubit system), applies a series of quantum gates to perform computations, and finally measures the qubits, collapsing them into classical states to extract the final result.

Quantum circuits can perform complex computations in parallel, thanks to superposition and entanglement. This parallelism underpins the power of quantum algorithms like Shor's algorithm (for factoring integers) and Grover's algorithm (for unstructured search), which achieve exponential and quadratic speedups, respectively, compared to their classical counterparts.

Quantum Measurement and Decoherence

Measurement in quantum computing collapses the qubit's quantum state into one of the basis states $|0\rangle$ or $|1\rangle$, destroying the superposition. The outcome of a measurement is probabilistic, with probabilities determined by the square of the amplitude of the qubit's state.

Quantum systems are also susceptible to *decoherence*, which arises from interactions between the quantum system and its environment. Decoherence causes the qubit to lose its quantum properties, such as superposition and entanglement, and is a major challenge in building scalable, fault-tolerant quantum computers. Quantum error correction techniques and decoherence-free subspaces are being explored to mitigate this issue.

Summary

In summary, quantum computing is built on the principles of superposition, entanglement, and quantum gates that perform unitary transformations on qubits. Quantum circuits leverage these operations to implement algorithms that can outperform classical algorithms for specific tasks. While quantum computing holds the potential for significant computational advantages, several challenges, such as decoherence and the need for error correction, must be overcome to realize the full potential of quantum computing in practice.

2.3.2 Variational Quantum Circuits

VQCs are a class of quantum algorithms that leverage parameterized quantum circuits and classical optimization techniques to solve complex computational tasks. They are commonly used in QML and optimization problems due to their flexibility in combining quantum computation with classical optimization methods. A VQC consists of a quantum circuit with adjustable parameters that are optimized to minimize or maximize an objective function

evaluated using classical algorithms. The hybrid quantum-classical training algorithm for variational circuits comprises of utilizing a quantum processor to evaluate a specific output, after which a classical computer adjusts the circuit parameters to optimize alignment with the target output. This iterative process continues until the desired solution is achieved as shown in Figure 2.3.

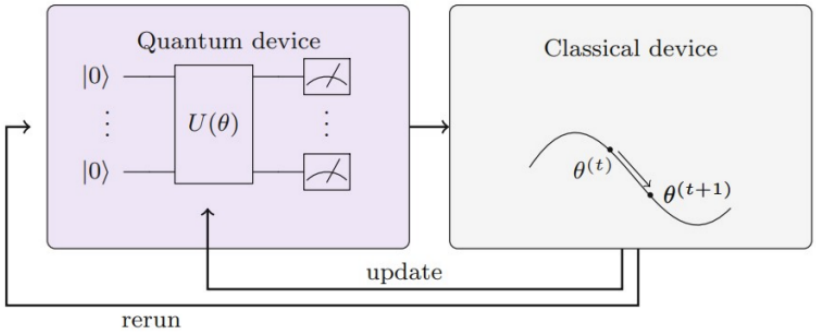

Figure 2.3: Hybrid quantum-clssical training algorithm [23]

Fundamental Structure of VQCs

A typical VQC includes the following key components:

- **Parameterized Quantum Gates:** The core of a VQC comprises quantum gates with adjustable parameters. These gates introduce trainable variables, allowing continuous variation of the quantum state through unitary operations.
- **Initial State Preparation:** The quantum circuit begins by preparing an initial state, such as $|0\rangle^{\otimes n}$ or a more complex entangled state, depending on the problem requirements.
- **Entangling Layers:** VQCs incorporate multi-qubit gates, such as CNOT gates, to create entanglement between qubits. Entanglement is essential for capturing complex correlations between components of the system.
- **Measurement and Cost Function:** After applying the parameterized gates, the quantum state is measured to obtain classical data. The measurement results are used to compute a cost function, which serves as the objective for the optimization.
- **Classical Optimization Loop:** A classical optimization algorithm, such as gradient descent or evolutionary methods, iteratively updates the parameters of the quantum gates until the cost function reaches an optimal value.

The structure of a VQC can be expressed as:

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle^{\otimes n} \tag{2.3.11}$$

where $U(\boldsymbol{\theta})$ represents the parameterized circuit and $\boldsymbol{\theta}$ is the vector of trainable parameters.

Parameterized Quantum Gates

The quantum gates in VQCs are parameterized to enable continuous adjustments of the quantum state. Common examples include:

Rotation Gates (RX, RY, RZ): These gates perform rotations around the x, y, and z axes of the Bloch sphere, with parameters controlling the rotation angle:

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{2.3.12}$$

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{2.3.13}$$

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}. \tag{2.3.14}$$

Phase Shift Gates: The $R_z(\theta)$ gate introduces a relative phase between states $|0\rangle$ and $|1\rangle$, which is essential for controlling phase relationships in the circuit.

Cost Function and Optimization in VQCs

The cost function $C(\boldsymbol{\theta})$ in a VQC is a scalar function that measures the performance of the circuit given parameters $\boldsymbol{\theta}$. The form of $C(\boldsymbol{\theta})$ depends on the specific problem:

Expectation Value of an Observable: For problems in quantum chemistry and physics, the cost function is often the expectation value of an observable O, such as the Hamiltonian H:

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle. \tag{2.3.15}$$

Classical Cost Functions: In QML, classical cost functions like cross-entropy or mean-squared error are common. The objective is to train the quantum circuit to approximate a desired function.

The optimization process updates $\boldsymbol{\theta}$ using classical techniques like gradient descent if the cost function is differentiable. For noisy or non-differentiable cost functions, gradient-free methods like the Nelder-Mead algorithm or evolutionary algorithms can be used.

Quantum Circuits as Universal Function Approximators

VQCs are being studied for their potential universal function approximation properties, and their capability of learning complex functions, much like classical neural networks. By training parameterized circuits, VQCs can be applied to classification, regression, and generative tasks. Inputs are encoded using methods such as amplitude encoding or angle encoding, and outputs are interpreted through measurements in the computational basis.

Expressibility and Barren Plateaus in VQCs

Expressibility in VQCs refers to their ability to represent a wide range of quantum states through variations in parameters. However, excessive expressibility can lead to *barren plateaus*, regions in the parameter space where the cost function's gradient is nearly zero. This makes optimization challenging due to vanishing gradients, particularly in deep or highly expressive circuits. To mitigate this, researchers focus on shallow circuit designs, structured initialization, and efficient gradient estimation techniques [24].

Applications of VQCs in QML

VQCs have several prominent applications in QML:

- **Variational Quantum Eigensolver (VQE):** Used for finding ground state energies of molecular systems by minimizing the energy expectation value.
- **Quantum Approximate Optimization Algorithm (QAOA):** Solves combinatorial optimization problems by parameterizing circuits that prepare approximate optimal states.
- **Quantum Neural Networks (QNN):** VQCs form the core of quantum neural networks, enabling them to perform classification and regression tasks by leveraging parameterized quantum gates.

Quantum LSTM Built on VQCs

The quantum LSTM network is a quantum-inspired version of the classical LSTM, leveraging the principles of VQCs. In a quantum LSTM, the key components of a classical LSTM—such as the input gate, forget gate, output gate, and cell state update—are implemented using parameterized quantum circuits. These circuits, designed as VQCs, are trained through the optimization of their parameters.

By building on VQCs, quantum LSTMs aim to enhance scalability and expressiveness, especially in high-dimensional time-series analysis or problems that are quantum-native. This makes them particularly promising for applications where classical LSTMs face limitations.

Summary

VQCs serve as a powerful framework for QML and optimization tasks, enabling the parameterization of quantum gates and optimization of quantum states for complex problems. VQCs are foundational to hybrid quantum-classical algorithms and advanced quantum neural networks, such as quantum LSTMs. VQCs are an active area of research due to their broad applicability in optimization and machine learning tasks.

2.4 Long Short-Term Memory Networks

2.4.1 Classical LSTM for Sequence Data

LSTM networks are a specialized type of RNNs designed to address the limitations of traditional RNNs in capturing long-term dependencies. By introducing memory cells and gating mechanisms, LSTMs effectively mitigate the vanishing and exploding gradient problems, making them highly suitable for tasks like time-series forecasting, natural language processing, and sequence-to-sequence modeling.

LSTM Architecture and Gates

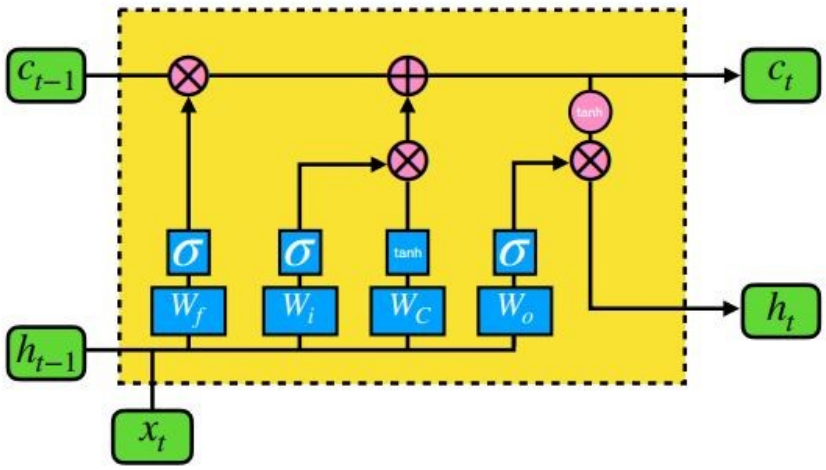

Figure 2.4: Long short-term memory [5]

The LSTM architecture is built around three key gates: the input gate, forget gate, and output gate, along with a memory cell that stores information over time. These gates regulate the flow of information at each time step, deciding what to remember, update, and discard. Figure 2.4 depicts the structure of LSTM in detail. The operations within an LSTM cell are described as follows:

$$\begin{aligned}
\textbf{Input Gate:} \quad & \mathbf{i}_t = \sigma\left(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right), \\
\textbf{Forget Gate:} \quad & \mathbf{f}_t = \sigma\left(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right), \\
\textbf{Output Gate:} \quad & \mathbf{o}_t = \sigma\left(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right), \\
\textbf{Cell State Update:} \quad & \tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right), \\
\textbf{Cell State:} \quad & \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\
\textbf{Hidden State:} \quad & \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned} \tag{2.4.1}$$

Here, $\mathbf{x}_t$ is the input vector at time t, $\mathbf{h}_{t-1}$ is the hidden state from the previous step, and $\mathbf{c}_{t-1}$ is the previous cell state. The weight matrices $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c$ and biases $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$ are

trainable parameters, while $\sigma(\cdot)$ represents the sigmoid activation function, and $\odot$ denotes the element-wise product.

Each gate plays a specific role:

- **Forget Gate $\mathbf{f}_t$**: Determines which parts of the previous cell state $\mathbf{c}_{t-1}$ should be retained or forgotten.
- **Input Gate $\mathbf{i}_t$**: Decides which new information should be written into the cell state.
- **Cell State Update $\tilde{\mathbf{c}}_t$**: Represents the candidate cell state that is combined with the input gate to update the current cell state $\mathbf{c}_t$.
- **Output Gate $\mathbf{o}_t$**: Regulates the amount of information from the cell state used to compute the hidden state $\mathbf{h}_t$.

The gating mechanism enables LSTMs to effectively capture both short-term and long-term dependencies, overcoming the limitations of traditional RNNs.

Training and Backpropagation Through Time (BPTT)

LSTMs are trained using the Backpropagation Through Time (BPTT) algorithm, where errors are propagated backward through each time step. The training objective is to minimize a loss function (e.g., cross-entropy or mean-squared error) by updating the weight matrices $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c$ and biases.

BPTT involves:

- **Forward Pass:** Computing the outputs $\mathbf{h}_t$ and cell states $\mathbf{c}_t$ for each time step.
- **Backward Pass:** Calculating gradients of the loss with respect to the parameters by unrolling the network and backpropagating errors.
- **Parameter Update:** Adjusting the weight matrices and biases using gradient descent or other optimizers like Adam [25].

The gating mechanisms in LSTMs help mitigate the vanishing gradient problem, making them more stable for training on long sequences.

Applications of LSTMs in Sequence Data

LSTMs excel in tasks where temporal dependencies are crucial. Some applications include:

- **Time-Series Forecasting:** Widely used for predicting future values in fields such as finance, weather, and energy management.
- **Natural Language Processing (NLP):** Essential for tasks like machine translation, speech recognition, and sentiment analysis, where sequential context is vital.
- **Sequence-to-Sequence Models:** Used in applications like language translation and automatic summarization, where LSTMs map an input sequence to an output sequence.

Summary

LSTMs are a versatile extension of RNNs designed to capture long-term dependencies in sequential data. By incorporating gating mechanisms, LSTMs effectively mitigate gradient issues and support stable training on long sequences. Their success in tasks like time-series forecasting, natural language processing, and anomaly detection demonstrates their robustness and adaptability in handling complex temporal patterns.

2.4.2 Quantum LSTM for Sequence Data

Quantum LSTM networks extend the classical LSTM by integrating VQCs into their gating mechanisms [26]. By leveraging quantum phenomena like superposition and entanglement, quantum LSTMs are aimed at modeling long-term dependencies in sequential data while exploiting quantum computational advantages. This hybrid quantum-classical approach is particularly suited for NISQ devices.

Quantum LSTM Architecture Overview

In quantum LSTM, the classical LSTM gates (input, forget, and output), and cell state update are replaced with parameterized quantum circuits. These circuits take in the hidden state $\mathbf{h}_{t-1}$ and input $\mathbf{x}_t$, encoding them as quantum states to modulate information flow. The architecture aims to allow quantum LSTMs to efficiently capture both short-term and long-term dependencies. The architecture of the quantum LSTM used in this thesis is depicted in Figure 2.5. It is similar to the quantum LSTM originally outlined in [26] except we omit optional post processing step and the VQC before the hidden state output to stick to the standard LSTM like process [27].

Mathematically, the quantum LSTM gates are defined as:

$$\begin{aligned}
\textbf{Forget Gate:} \quad & f_t = \sigma\left(\mathbf{VQC}_1(\mathbf{v}_t)\right), \\
\textbf{Input Gate:} \quad & i_t = \sigma\left(\mathbf{VQC}_2(\mathbf{v}_t)\right), \\
\textbf{Cell State Update:} \quad & \tilde{C}_t = \tanh\left(\mathbf{VQC}_3(\mathbf{v}_t)\right), \\
\textbf{Cell State:} \quad & C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \\
\textbf{Output Gate:} \quad & o_t = \sigma\left(\mathbf{VQC}_4(\mathbf{v}_t)\right), \\
\textbf{Hidden State:} \quad & h_t = o_t \odot \tanh(C_t).
\end{aligned} \tag{2.4.2}$$

Here, $\mathbf{v}_t = [\mathbf{h}_{t-1}, \mathbf{x}_t]$ is the concatenated input and hidden state vector.

Variational Quantum Circuits in Quantum LSTM

The VQC used in the quantum LSTM architecture is shown in Figure 2.6. Each gate in the quantum LSTM is modeled using a distinct VQC with the following components:

1. **Data Encoding Layer:** The circuit initializes each qubit in an unbiased superposition state using a series of Hadamard gates. The input data values are then encoded into

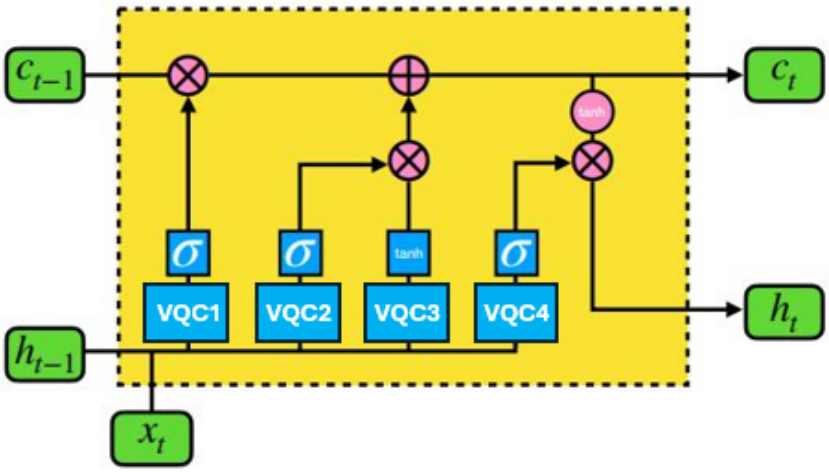

Figure 2.5: Quantum long short-term memory (Quantum LSTM). The design of the network is based on the works of [26, 27]

.

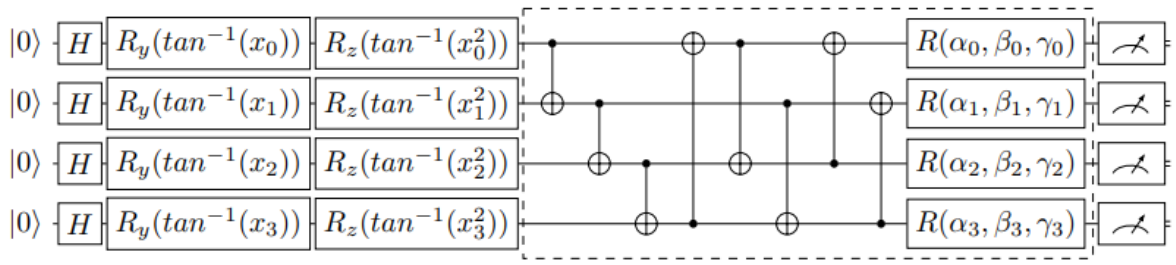

Figure 2.6: VQC used in quantum LSTM [26]

the qubits by transforming each input feature x_i using the arctangent function, i.e., $\tan^{-1}(x_i)$. This transformation is followed by applying the parameterized single-qubit rotational gates $R_y(\tan^{-1}(x_i))$ and $R_z(\tan^{-1}(x_i^2))$, which encode the classical input into quantum states. Squaring the features before applying the arctangent function and the R_y gate might be an attempt to improve the expressivity of the model by extending the function classes that the model can learn even further [28][29]. Due to the longer training times of the quantum LSTM, the effect of preprocessing in the encoding strategies is not explored in this thesis. However, it is an exciting avenue for future work.

2. **Variational Layer:** The central portion of the circuit, enclosed within the dashed box in Figure 2.6, consists of a set of CNOT gates arranged in a layered ring structure to generate multi-qubit entanglement. After establishing entanglement, a set of parameterized single-qubit rotation gates $R(\alpha_i, \beta_i, \gamma_i)$ is applied to each qubit. These gates introduce tunable parameters α_i, β_i, and γ_i which are adjusted during training. The number of such variational layers can be increased to control the circuit depth, but only a single layer is employed in this work.

3. **Measurement Layer:** The final stage of the VQC involves measuring each qubit in the computational basis. The output is obtained by evaluating the expectation values of the Pauli-Z operator for each qubit, yielding the predictions required for the quantum LSTM's computations.

Training Process of Quantum LSTM

The quantum LSTM is trained using a hybrid quantum-classical approach:

1. **Forward Pass:** Inputs are processed through quantum circuits representing the gates to generate outputs.

2. **Cost Function Evaluation:** A loss function, like mean-squared error (MSE), is computed based on model predictions.

3. **Quantum Gradient Calculation:** Quantum gradients are calculated using the adjoint differentiation method from Pennylane [30].

4. **Parameter Update:** Classical optimizers such as Adagrad [31] are used to adjust quantum gate parameters $\boldsymbol{\theta}$.

3 Methodology

3.1 Overview of Model Predictive Control using Quantum LSTM

MPC is a widely-used technique for controlling dynamic systems by predicting future states and computing optimal control actions. Traditionally, MPC relies on classical system identification models, such as neural networks or mathematical representations, to predict how a system will respond to various inputs.

In this approach, a quantum LSTM network is integrated into the MPC framework to serve as the system identification model. Built with VQCs, the quantum LSTM replaces conventional models to leverage the potential benefits of quantum computing, particularly in handling complex dynamics and time-series data. In this study, the quantum LSTM was trained on the cart-pole environment from OpenAI Gym, and the POCP environment, a more challenging version of the benchmark cart-pole environment.

Despite the shift to a quantum-enhanced model, the core MPC framework remains unchanged. The prediction of future states, computation of control actions, and optimization over a prediction horizon are consistent with classical MPC. In the cart-pole system, the quantum LSTM forecasts the cart and pole states, while an optimization algorithm, such as PSO, determines the optimal actions to maintain stability. At each time step, control actions are updated based on the quantum LSTM's predictions, similar to classical approaches.

By incorporating QML within MPC, this study explores the feasibility and performance of the quantum LSTM compared to its classical counterpart. The focus is on evaluating prediction accuracy and computational efficiency in controlling the cart-pole system, as well as examining the potential benefits of using quantum-enhanced models.

3.2 Problem Setup and Cart-Pole Benchmark

The cart-pole problem is a classic benchmark in control theory and RL, used to evaluate the effectiveness of control algorithms in managing dynamic and nonlinear systems. It consists of a cart that moves horizontally on a track with a pole attached, and the goal is to apply forces to keep the pole balanced upright while keeping the cart within the track's boundaries as depicted in Figure 3.1.

The system's state at time t is described by a four-dimensional vector $\mathbf{s}_t = [x_t, \dot{x}_t, \theta_t, \dot{\theta}_t]$, where:

- x_t is the cart's position,
- $\dot{x}_t$ is the cart's velocity,

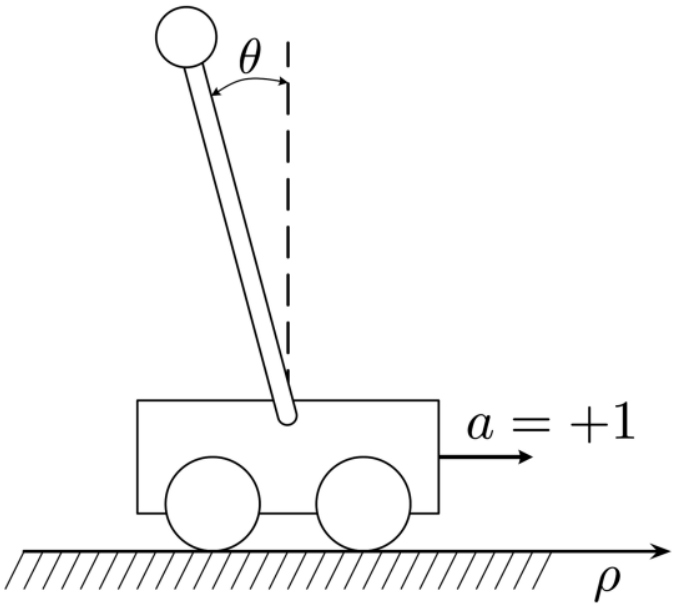

Figure 3.1: Cart-pole system [9]

- θ_t is the pole's angle from the vertical, and
- $\dot{\theta}_t$ is the pole's angular velocity.

The system is inherently unstable—any deviation from $\theta = 0$ causes the pole to fall unless corrective forces $u_t \in \{-F, +F\}$ are applied to push the cart left or right. The goal is to balance the pole for a preferably long sequence of time steps without moving out of the limits mentioned in Table 3.1.

Table 3.1: Bounds for the cart-pole environment parameters.

Parameter	**Lower Bound**	**Upper Bound**
Cart Position	-2.4	2.4
Cart Velocity	$-\infty$	∞
Pole Angle	$-\frac{\pi}{15} \approx -0.2095$	$\frac{\pi}{15} \approx 0.2095$
Pole Angular Velocity	$-\infty$	∞

The dynamics of the cart-pole are governed by these nonlinear differential equations:

$$\ddot{x} = \frac{F + m_p \sin(\theta)\left(l\dot{\theta}^2 + g\cos(\theta)\right)}{m_c + m_p \sin^2(\theta)}, \tag{3.2.1}$$

$$\ddot{\theta} = \frac{-F\cos(\theta) - m_p l\dot{\theta}^2 \sin(\theta)\cos(\theta) - (m_c + m_p)g\sin(\theta)}{l\left(m_c + m_p \sin^2(\theta)\right)} \tag{3.2.2}$$

where:

- m_c and m_p are the masses of the cart and pole,
- l is the pole length,
- g is the gravitational constant, and
- F is the applied control force.

For this study, the cart-pole environment is simulated using the OpenAI Gym framework, which provides a standardized implementation of the system. A random policy is used to collect training data by applying random forces and observing state transitions. This data is utilized to train classical and quantum models, which predicts the system's dynamics within the MPC framework.

Due to its nonlinear nature, the cart-pole problem serves as an ideal benchmark for evaluating the effectiveness of advanced control strategies and predictive models like quantum LSTMs. The cart-pole benchmark is a good starting point to test the trained models integrated into the MPC framework.

However, it is important to consider testing the models on a more challenging environment to observe their true potential. Hence, we also test the models on the POCP environment, where we omit the two velocities $[\dot{x}_t, \dot{\theta}_t]$ [8].

3.3 System Identification

In this work, quantum LSTM, classical LSTM, and an MLP are used for system identification within the MPC strategy. The goal is to learn the system dynamics of the cart-pole environment by predicting changes in the system's state based on the current state and corresponding action.

Data Generation and Preprocessing: We generated 10,000 data points using a random policy applied to the cart-pole system, which consists of four state variables: $\mathbf{s}_t = [x_t, \dot{x}_t, \theta_t, \dot{\theta}_t]$. The data, organized as shown in Table 3.2, includes the states, actions, and delta state variables. The dataset was scaled using the mean and standard deviation of the training data to maintain numerical stability during training.

Table 3.2: Sample dataset for model training

Time Step	Position	Velocity	Angle	Angular Velocity	Action	Episode	deltaCP	deltaCV	deltaA	deltaAV
1	-0.010442	0.026518	-0.045658	-0.048751	1	0	0.000530	0.195746	-0.000975	-0.306732
2	-0.009912	0.222264	-0.046633	-0.355483	0	0	0.004445	-0.194429	-0.007110	0.277621
3	-0.005467	0.027835	-0.053743	-0.077862	0	0	0.000557	-0.194312	-0.001557	0.275254
4	-0.004910	-0.166477	-0.055300	0.197393	1	0	-0.003330	0.195868	0.003948	-0.309602

Model Architecture: The classical LSTM network consists of 5 hidden units, MLP model consists of 10 hidden units, and the quantum LSTM network is designed with 16 hidden units and 4 qubits. The hyperparameters were chosen in such a manner to maintain a similar number of model parameters for all the models. More detials about the hyperparameters used for the experiment are tabulated for each experiment in Chapter 4. In FOCP benchmark, the models take the current state and the action as input and predict the delta state as output. In the POCP benchmark, the current partial state and the previous partial state with their corresponding actions are the input with a delta partial state as the output. The key difference between the models lies in the gate mechanism, where the quantum LSTM employs parameterized quantum circuits to replace the classical LSTM gates.

Training and Testing: Both models were trained using the MSE loss function, which measures the average squared difference between the predicted and actual delta states. The classical LSTM used the Adam optimizer, while the quantum LSTM leveraged the Adagrad optimizer. In addition, the quantum LSTM applied the *adjoint* differentiation method [30] because of its low memory usage and the compatibility with *lightning.qubit* plugin from Pennylane [13]. The dataset was divided into 8,000 training and 2,000 testing datapoints, and the models were trained with the goal of minimizing the MSE loss.

Rollout Evaluation: To assess the long-term prediction capabilities, we performed **rollouts** for both models and compared them with a base cart-pole data . During rollouts, the initial state and action were given as input to the model, which predicted the next state by outputting the delta state. The delta state is then added to the current to get the next state. This next state was used as input for the subsequent timestep. Rollouts were conducted over 50 steps, and the predicted states were compared against actual values to evaluate the accuracy of the classical LSTM, quantum LSTM, and MLP.

3.4 Model Predictive Control Framework

The trained models were integrated into the MPC framework as predictive models for action space optimization. Using state predictions from these models, the **PSO** algorithm optimized control actions. The performance of the MPC in controlling the cart-pole system directly relied on the accuracy of the state predictions. The detailed MPC framewrok with PSO optimizations implemented in this thesis in shown in Algorithm 1.

Algorithm 1 : MPC framework using PSO with quantum LSTM, classical LSTM, and MLP models

Notation:
s_0: Initial state, $g(s, a)$: Real-world system, I: Action dimensionality, T: Planning horizon.
Trained Models: Quantum LSTM, classical LSTM, and MLP models trained on the cart-pole environment.

Begin:
Initialize current state $s \leftarrow s_0$.

Repeat:
1. Perform PSO to find the best action sequence $\hat{\mathbf{x}}$ using the trained model (*PSO Procedure*):
PSO Procedure:
a) Initialize a population of particles with random positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$.
b) Evaluate the fitness of each particle based on model predictions $f(\mathbf{x}_i)$ using either of the trained models.
c) Update the best positions of each particle $\mathbf{y}_i$ and best neighborhood positions $\mathbf{y}_g$.
d) Adjust the velocities $\mathbf{v}_i$ and update the positions $\mathbf{x}_i$ according to PSO rules.
e) Repeat the steps until convergence or reaching the maximum number of PSO iterations.
f) Select the best overall action sequence $\hat{\mathbf{x}}$.
2. Extract the first action $\hat{a}$ from the sequence $\hat{\mathbf{x}}$.
3. Apply the action $\hat{a}$ to the real-world system $(s, r) \leftarrow g(s, \hat{a})$.
4. Update the current state s based on the response from the system.

Until: Termination conditions are achieved (e.g., reaching the planning horizon T or task success).

3.5 Reward Function used in Particle Swarm Optimization

The reward function aims to penalize the cart-pole's deviation from the upright and center positions of the track. A heavy termination penalty is included if the states are out of the allowed bounds of the cart-pole mentioned in Table 3.1.

The reward function takes a batch of states as input. The function calculates a reward for each state based on three components:

- **Upright Reward:** A negative squared penalty based on the deviation of the pole angle from the upright position.
- **Center Reward:** A negative squared penalty based on the cart's position deviation from the track's center.

- **Termination Penalty:** A significant penalty for states where the cart or pole is out of the defined bounds.

Th reward function used in the thesis is shown in Equation 3.5.1

$$\mathbf{R} = \left\{ r_i = -\left(\frac{\theta_i}{0.2095}\right)^2 - 10\left(\frac{x_i}{2.4}\right)^2 - 1000 \cdot \mathbb{1}_{\{x_i \notin [-2.4,2.4] \text{ or } \theta_i \notin [-0.2095,0.2095]\}} \right\}_{i=1}^{N} \tag{3.5.1}$$

where,

- $\mathbf{R}$: The reward vector, containing individual rewards r_i for each state i in a batch of size N.
- r_i: The reward associated with the i-th state, which combines upright, center, and termination penalties.
- θ_i: The pole angle of the cart-pole system in the i-th state. It represents the angular deviation of the pole from the upright (vertical) position, measured in radians. The target (upright) position is $\theta = 0$.
- x_i: The cart's position on the track in the i-th state. It represents the horizontal displacement of the cart, where $x = 0$ indicates the center of the track.
- $-\left(\frac{\theta_i}{0.2095}\right)^2$: The upright penalty, which penalizes the deviation of the pole angle θ_i from the upright position.
- $-10\left(\frac{x_i}{2.4}\right)^2$: The center penalty, which penalizes the deviation of the cart position x_i from the center of the track.
- $\mathbb{1}_{\{x_i \notin [-2.4,2.4] \text{ or } \theta_i \notin [-0.2095,0.2095]\}}$: The indicator function, which equals 1 if the conditions inside the braces are met (out-of-bounds) and 0 otherwise. It applies a penalty if x_i or θ_i fall outside their defined bounds.
- $-1000 \cdot \mathbb{1}_{\{x_i \notin [-2.4,2.4] \text{ or } \theta_i \notin [-0.2095,0.2095]\}}$: The termination penalty, which imposes a significant penalty of -1000 if the cart or pole goes out of bounds.

We first receive a batch of states with a size equal to the total number of particles we have initialized. From each state, we extract the position and the pole angle to calculate the upright reward, and the center reward. If the cart-pole goes out of bounds, a hefty termination penalty is imposed to discourage the system from reaching such states. The reward function returns the total reward, the summation of all three components calculated.

4 Results and Discussion

This chapter compares various results obtained using the quantum LSTM with the classical models. We first begin by using the quantum LSTM for simple function approximation tasks as described in [26]. We then proceed with the cart-pole benchmarks and discuss the training and rollout results obtained and the models' accuracy in capturing the environment's system dynamics. Further, we use these trained models in MPC and discuss their potential in balancing the cart-pole in FOCP and POCP environments.

4.1 Prediction of Sine Function

The sine function is a fundamental trigonometric function given by:

$$y = \sin(x), \tag{4.1.1}$$

where x represents the independent variable. The function is periodic with a period of 2π. This study sampled the sine function over the range $x \in [0, 50]$ with 501 points to create the dataset. We set $\sin(x)$ as our target variable and use the previous five $\sin(x)$ values as our model inputs. This simple oscillatory function is used to evaluate the quantum LSTM model's ability to approximate smooth periodic signals and is compared with the classical LSTM.

Figure 4.1 shows the quantum and classical LSTM's ability to approximate the sine function. The experiment has been repeated five times, and the average testing and training loss values achieved by both quantum and classical LSTM are presented in Table 4.1. The low loss values indicate that the models have accurately approximated the sine function without much difficulty. Figure 4.2 shows the scatter plot with data on the x-axis and model predictions on the y-axis. Accumulation of points on the bisector indicates that the model is making good predictions of the sine function.

Table 4.1: Average training and testing loss comparison between quantum LSTM and classical LSTM models for sine function approximation from five training repetitions.

	Training Loss	**Testing Loss**
Quantum LSTM	4.56×10^{-5}	3.45×10^{-5}
Classical LSTM	4.98×10^{-5}	3.12×10^{-5}

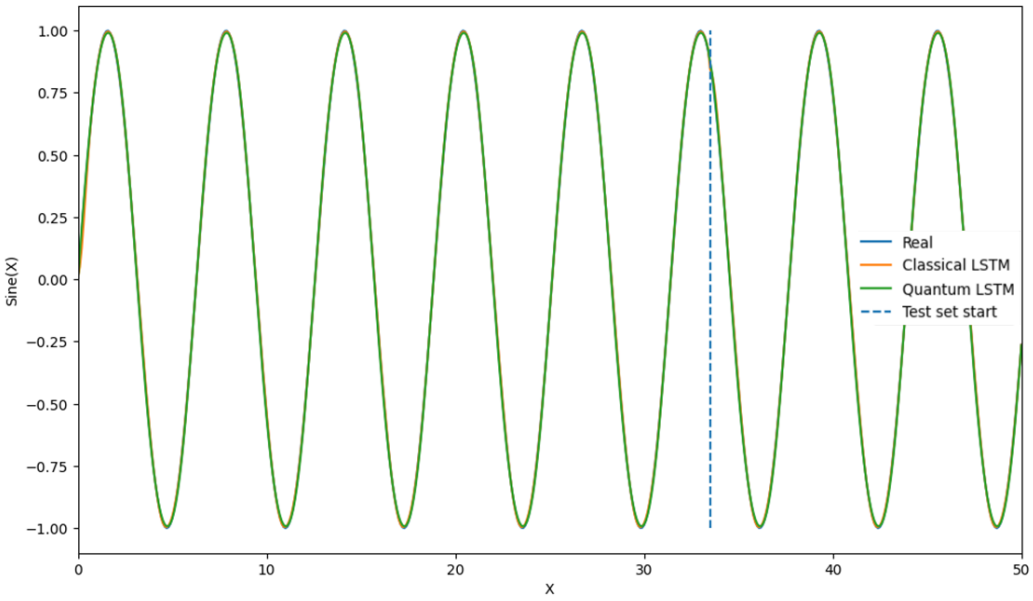

Figure 4.1: Sine function approximation

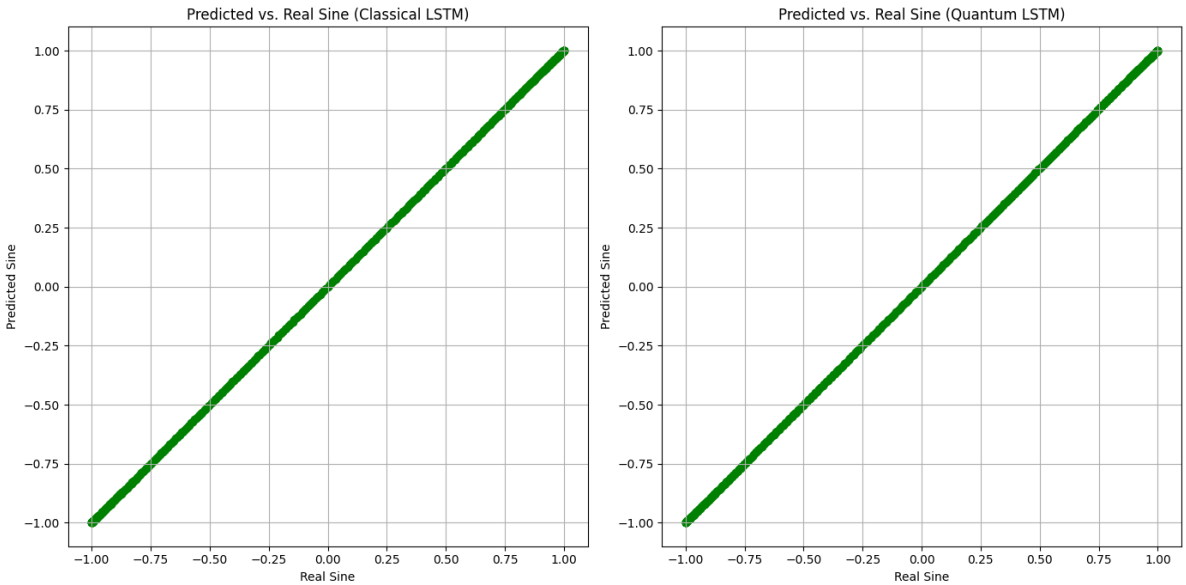

Figure 4.2: Scatter plot with the data on the x-axis and the model predictions on the y-axis for sine function.

4.2 Prediction of Damped Harmonic Oscillator

The Damped Harmonic Oscillator (DHO) is a classical physics model that describes the motion of a mass-spring-damper system. A second-order differential equation of the form governs it:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = 0, \tag{4.2.1}$$

where m is the mass, c is the damping coefficient, and k is the spring constant. The solution to this equation depends on the damping ratio ζ, which is defined as:

$$\zeta = \frac{c}{2\sqrt{km}}. \tag{4.2.2}$$

In the underdamped case, where $\zeta < 1$, the system exhibits oscillatory behavior with a decaying amplitude. The displacement $x(t)$ of the mass over time is given by:

$$x(t) = Ae^{-\zeta\omega_0 t}\cos(\omega_d t) + \left(\frac{B + \zeta\omega_0 A}{\omega_d}\right) e^{-\zeta\omega_0 t}\sin(\omega_d t), \tag{4.2.3}$$

where A and B represent the initial displacement and velocity, respectively, ω_0 is the undamped natural frequency defined as $\omega_0 = \sqrt{\frac{k}{m}}$, and ω_d is the damped natural frequency defined as:

$$\omega_d = \omega_0\sqrt{1 - \zeta^2}. \tag{4.2.4}$$

This work utilized the DHO model to generate synthetic data to train the quantum LSTM model and approximate the system's dynamics. The dataset comprises the displacement $x(t)$ as the target variable and the previous five displacement values as our model inputs. This provides a challenging but insightful testbed for validating the effectiveness of quantum LSTM in modeling oscillatory and decaying dynamical systems. Figure 4.3 shows the quantum and classical LSTM's ability to approximate the DHO function. The experiment has been repeated five times, and the average testing and training loss values achieved by quantum and classical LSTM are presented in Table 4.2. Figure 4.4 shows the scatter plot with data on the x-axis and model predictions on the y-axis. Again, we observe that the points are accumulated on the bisector, indicating good model predictions.

Table 4.2: Average training and testing loss comparison between quantum LSTM and classical LSTM models for DHO approximation from five training repetitions.

	Training Loss	**Testing Loss**
Quantum LSTM	5.70×10^{-5}	5.48×10^{-6}
Classical LSTM	3.31×10^{-5}	2.81×10^{-6}

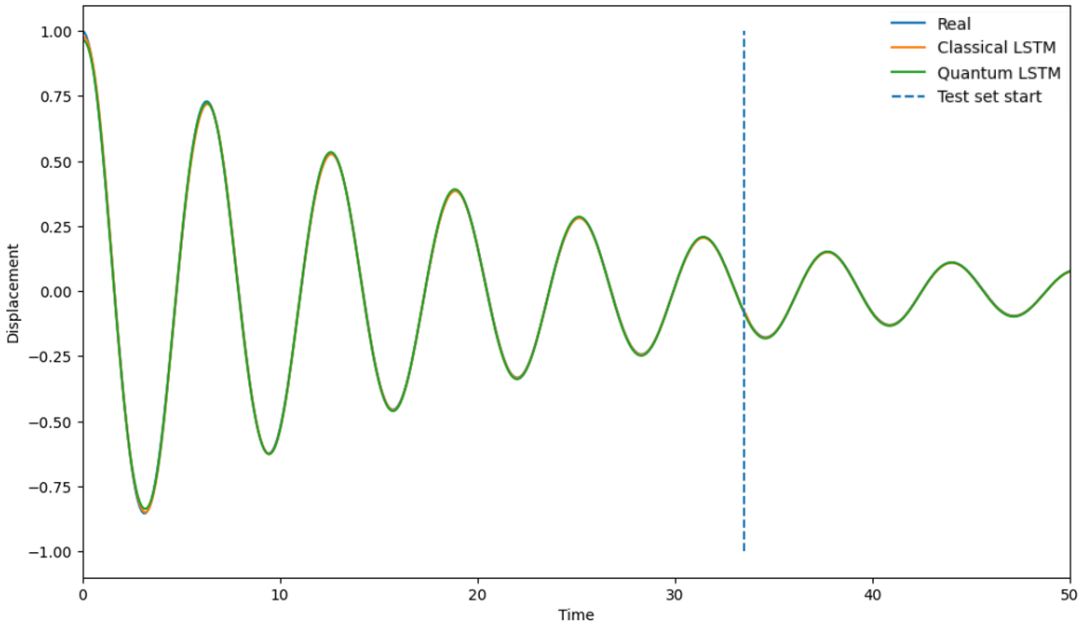

Figure 4.3: DHO approximation

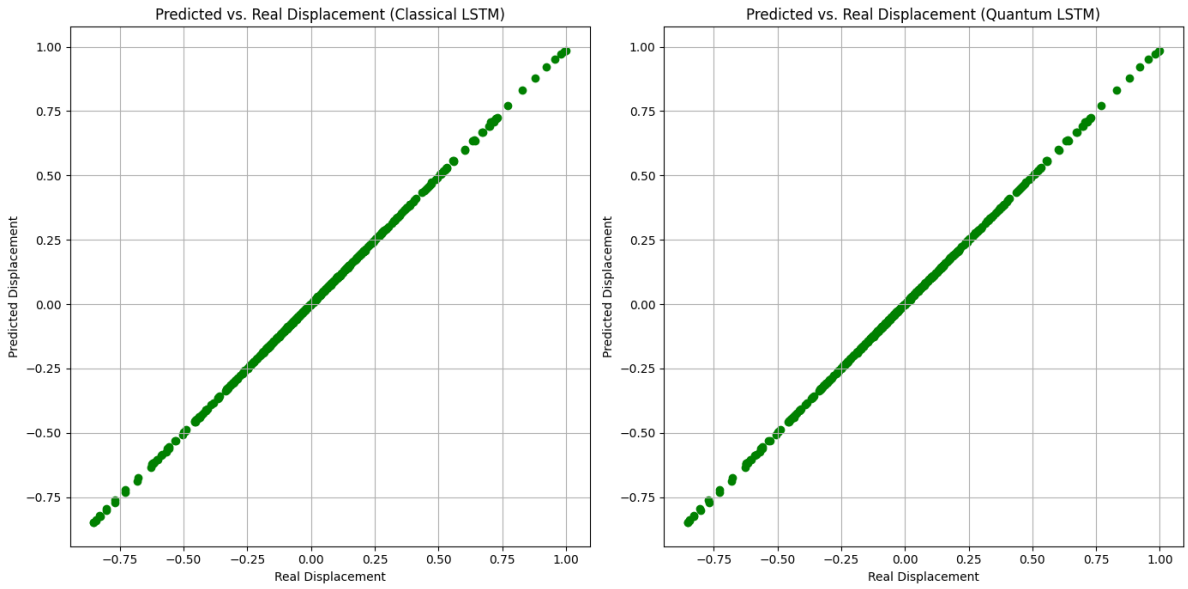

Figure 4.4: Scatter plot with the data on the x-axis and the model predictions on the y-axis for DHO.

4.3 Prediction of Bessel Functions

Bessel functions are a family of solutions to Bessel's differential equation, frequently arising in problems exhibiting cylindrical or spherical symmetry. The Bessel function of the first kind, denoted as $J_n(x)$, of order n is defined as the solution to:

$$x^2 \frac{d^2 J_n(x)}{dx^2} + x \frac{dJ_n(x)}{dx} + \left(x^2 - n^2\right) J_n(x) = 0. \tag{4.3.1}$$

For a given order n, the Bessel function $J_n(x)$ exhibits oscillatory behavior that decays with increasing x. In this study, the Bessel function of the first kind of order $n = 2$, denoted as $J_2(x)$, was used to generate synthetic data.

The values of $J_2(x)$ were computed over a range of x from 0 to 200, resulting in a dataset with 2001 points. The dataset was constructed as a time-series-like structure, where $J_2(x)$ value is the target variable and we use the previous five $J_2(x)$ values as our input to the model.

This dataset presents an additional challenge for the quantum LSTM model, allowing it to demonstrate its ability to approximate nonlinear and oscillatory functions accurately. The approximated function provides insight into how well the quantum LSTM can capture and generalize complex mathematical functions beyond simple harmonic motion.

Figure 4.5 shows the quantum and classical LSTM's ability to approximate the Bessel functions. The experiment has been repeated five times, and the average testing and training loss values achieved by quantum and classical LSTM are presented in Table 4.3. The loss values indicate that the models accurately approximated the second-order Bessel functions without difficulty. Figure 4.6 shows the scatter plot with data on the x-axis and model predictions on the y-axis. We observe that the points are accumulated on the bisector, indicating good predictions for both quantum and classical LSTM. In the following sections, we will look into the performance of the models on the FOCP and POCP benchmarks and if we can use them for MPC.

Table 4.3: Average training and testing loss comparison between quantum LSTM and classical LSTM models for Bessel function approximation five training repetitions..

	Training Loss	**Testing Loss**
Quantum LSTM	5.51×10^{-5}	1.00×10^{-5}
Classical LSTM	1.10×10^{-6}	1.27×10^{-6}

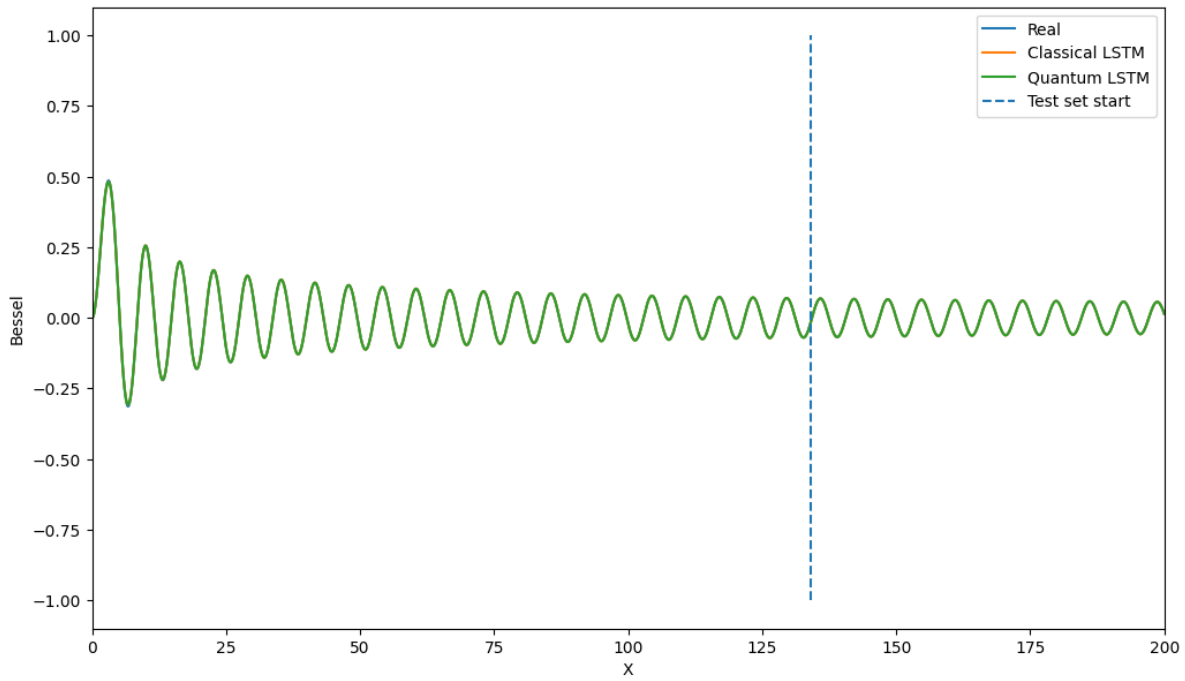

Figure 4.5: Bessel function approximation

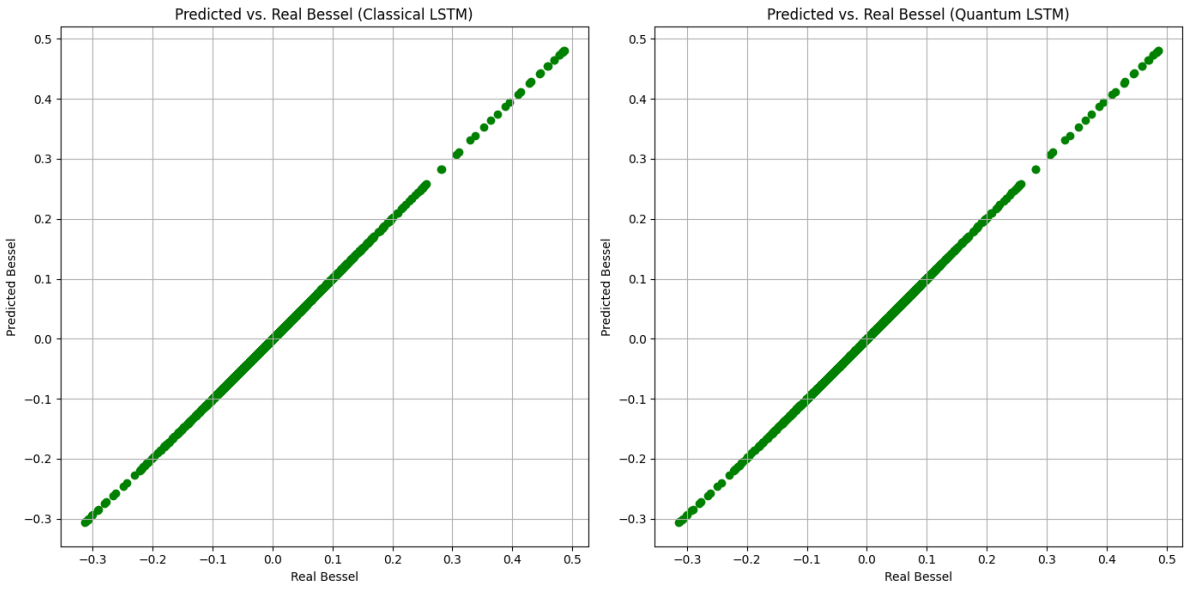

Figure 4.6: Scatter plot with the data on the x-axis and the model predictions on the y-axis for Bessel functions.

4.4 Prediciton of Fully Observable Cart-Pole

In this section, we evaluate the performance of our quantum LSTM and classical LSTM models that are trained on the FOCP dataset in terms of their ability to predict future states of the cart-pole. We are more interested in the single-step rollouts generated by the models and compare them with the original cart-pole data to see how accurate are the predictions before diverging from the original data. This will help us in deciding if our models are good enough to be used for MPC.

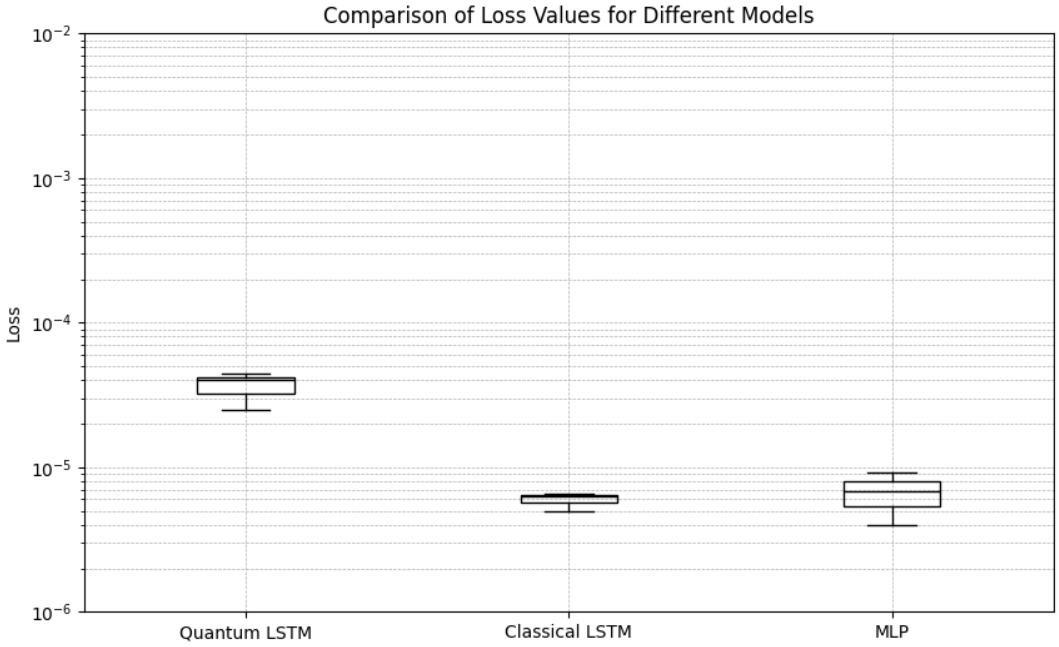

Figure 4.7: Loss comparison for quantum LSTM, classical LSTM, and MLP for FOCP benchmark

Cart-pole data generation and pre-processing information have already been discussed in Chapter 3. No detailed hyperparameter tuning of the models was performed in this study due to the prohibitively longer training times of the quantum LSTMs. Details regarding the hyperparameters are tabulated in Table 4.4. Figure 4.7 shows the test loss values obtained for each of the models for three repetitive trainings.

Table 4.4: Model configuration details

Model	Qubits	Hidden Units	Batch Size	Learning Rate
Quantum LSTM	4	16	8	0.02
Classical LSTM	-	5	8	0.00005
MLP	-	10	8	0.001

We observe that classical LSTM and MLP give us the best results out of all the models followed by quantum LSTM. The range of loss values obtained for classical and quantum LSTM is small enough to be considered for rollouts. To check if the models have learned the dynamics of the cart-pole environment correctly, the scatter plots of all four state variables of the cart-pole are plotted as shown in Figure 4.8, Figure 4.10, and Figure 4.12. In the scatter plots, the data is on the x-axis, and the model predictions are on the y-axis. We observe that the points lie close to the bisector, indicating that the model has learned the dynamics correctly and is making accurate predictions. The errors in the predictions of the models are plotted in Figure 4.9, Figure 4.11, and Figure 4.13. We see that the error is minimal for all three models, which conveys that the models are good enough to be used for rollouts.

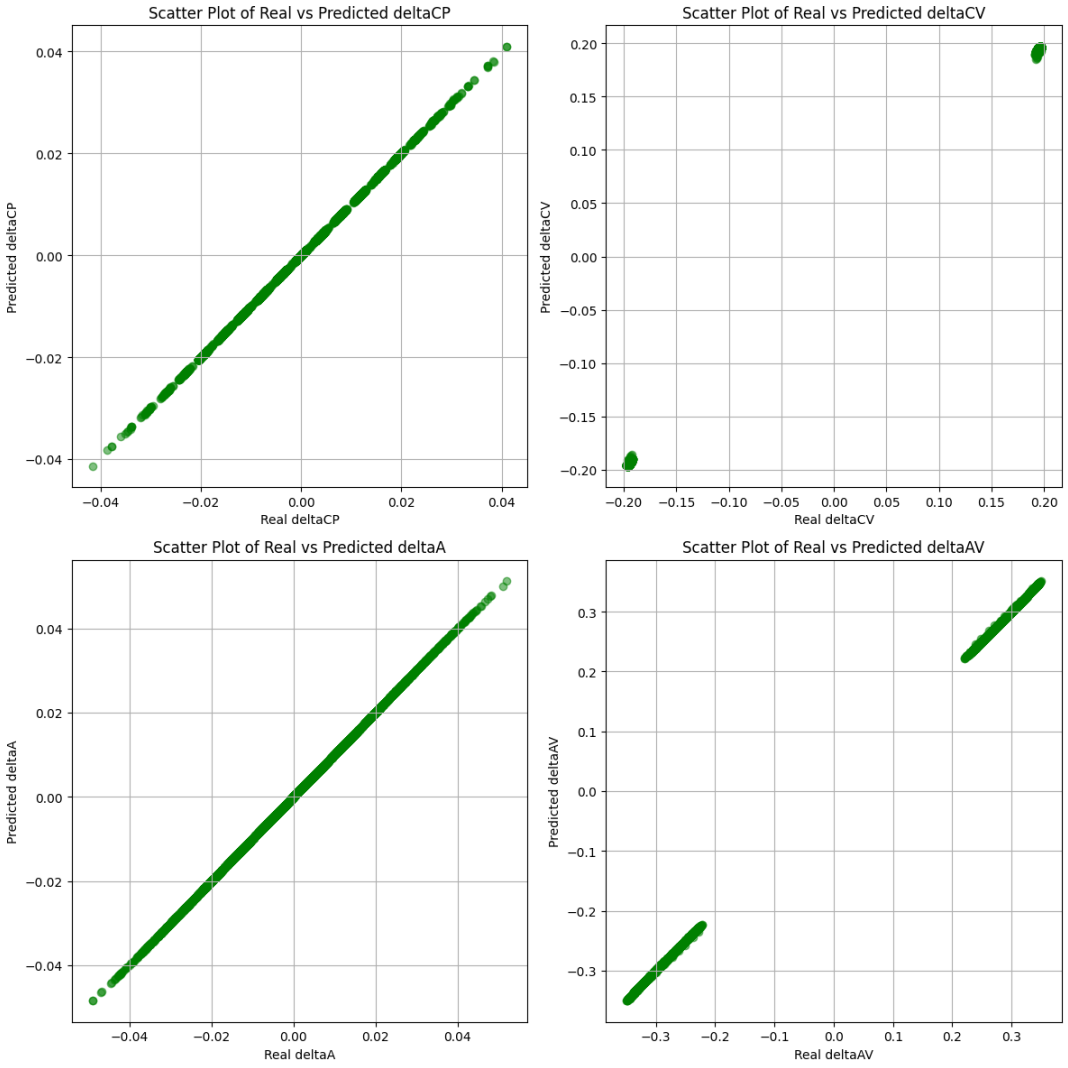

Figure 4.8: Scatter plot with the data on the x-axis and the classical LSTM's prediction on the y-axis.

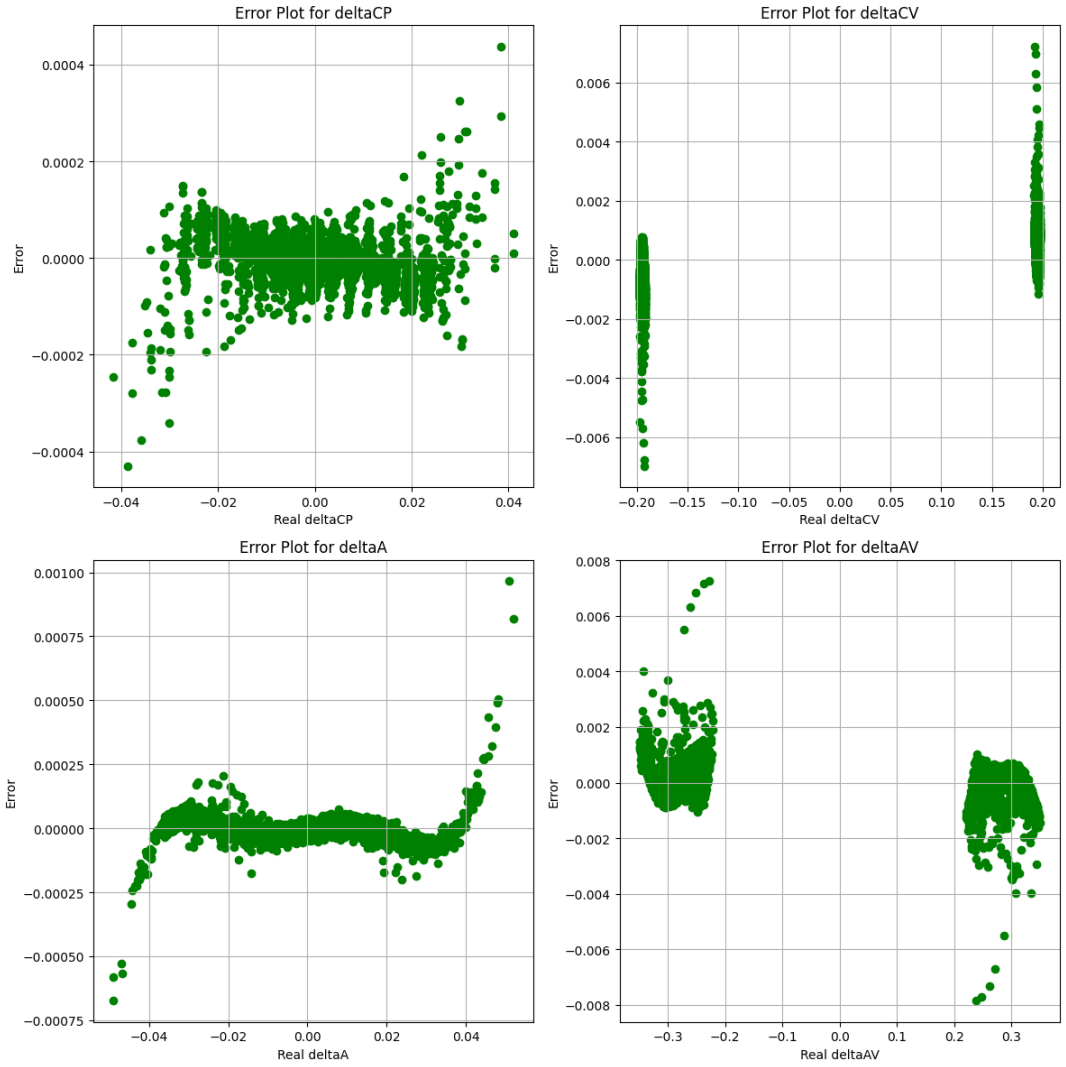

Figure 4.9: Scatter plot with the data on the x-axis and the error in classical LSTM's prediction on the y-axis.

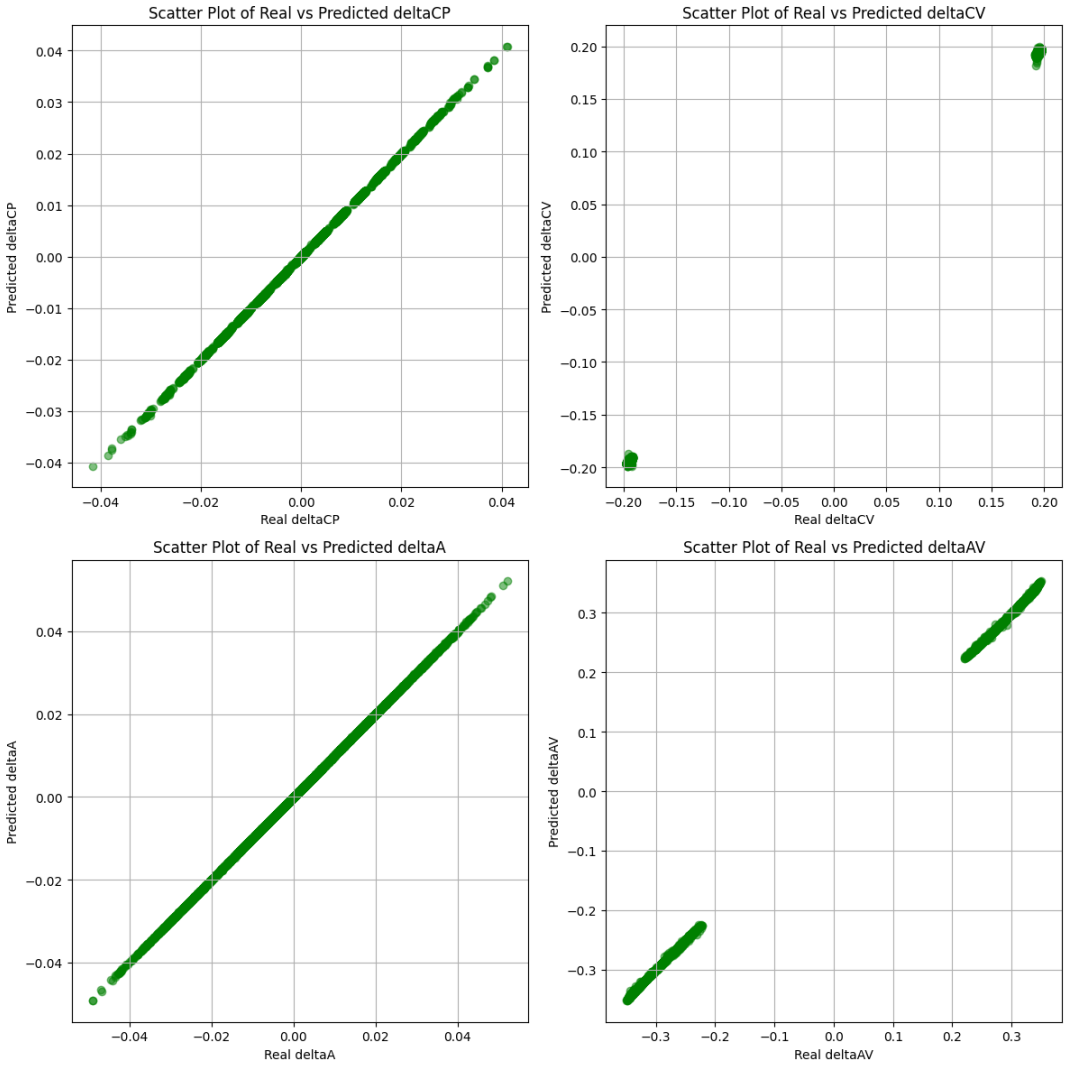

Figure 4.10: Scatter plot with the data on the x-axis and the quantum LSTM's prediction on the y-axis.

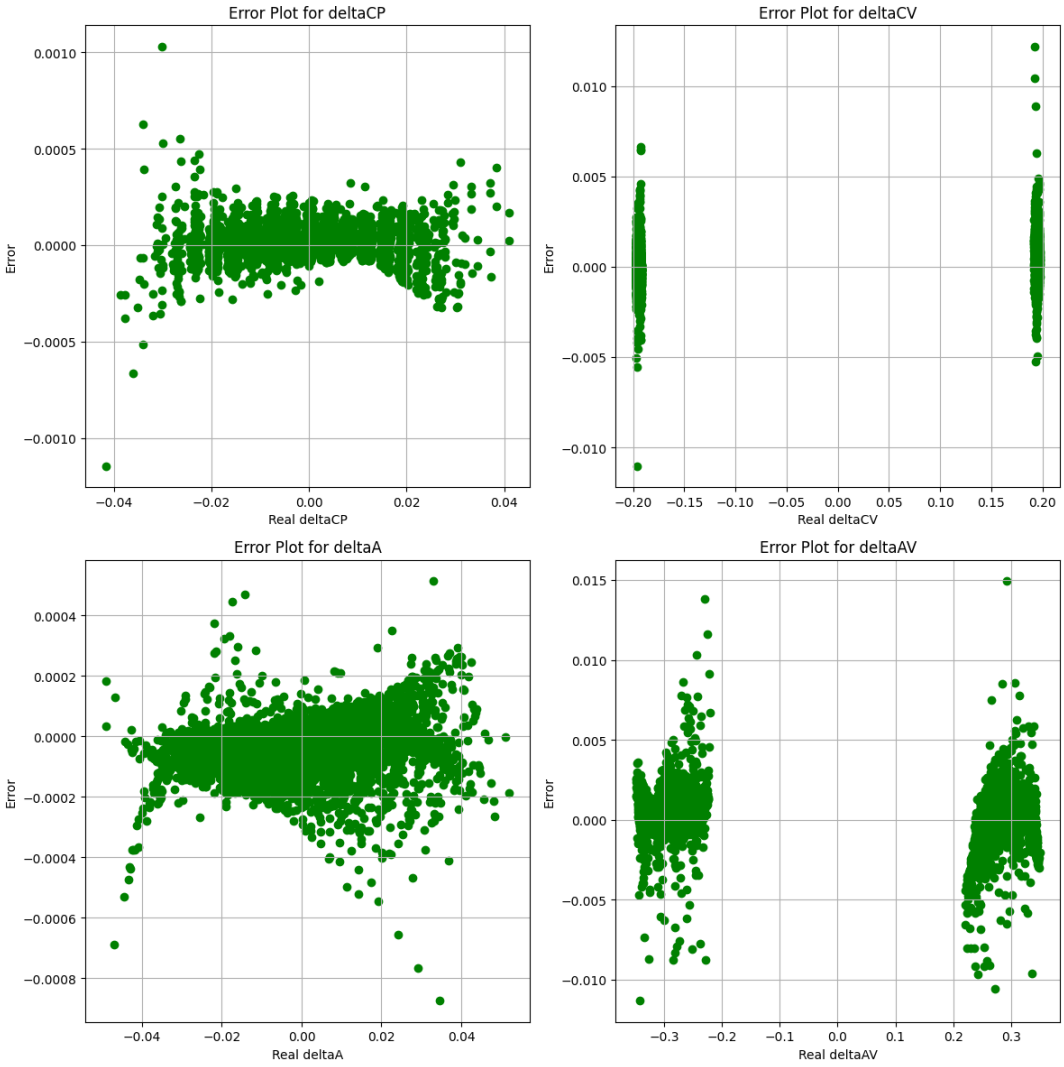

Figure 4.11: Scatter plot with the data on the x-axis and the error in quantum LSTM's prediction on the y-axis.

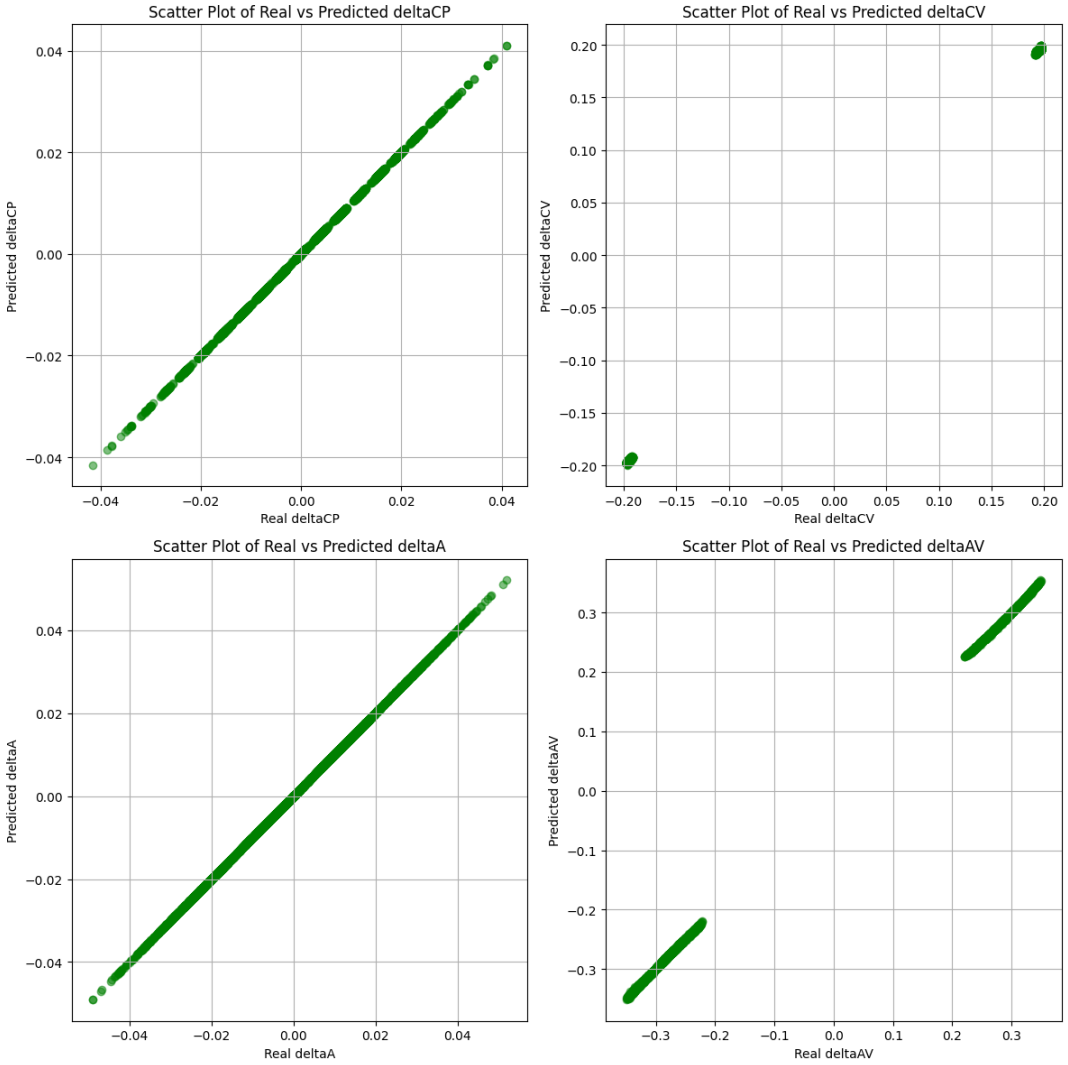

Figure 4.12: Scatter plot with the data on the x-axis and the MLP's prediction on the y-axis.

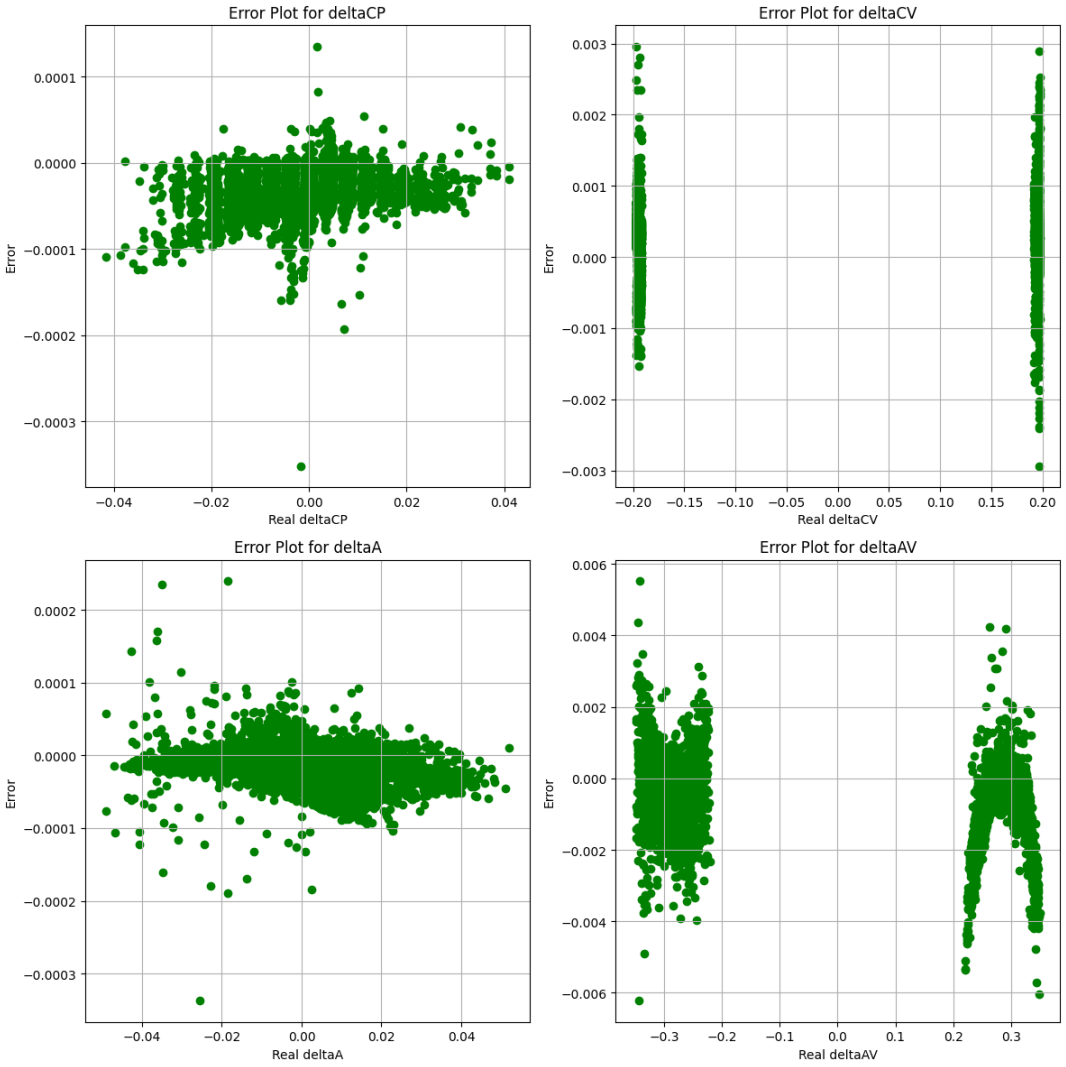

Figure 4.13: Scatter plot with the data on the x-axis and the error in MLP's prediction on the y-axis.

We use the rollout data mentioned in Chapter 3 as a reference to evaluate the model rollouts. We initiate the process by feeding a scaled initial state action pair from the rollout data to the model. The model then predicts delta state, which is the difference between the next state and the initial state. We unscale the delta state predicted by the model and add the initial state to obtain the next state. The following action from the rollout data is appended to this next state and is fed into the model again to predict the new state. This process is repeated for 50 steps to observe the deviations of the models on all four state parameters.

The rollouts of the models are depicted in Figure 4.14, along with the corresponding absolute error values. We observe that the classical LSTM deviates less than the quantum LSTM, especially in position and angle plots. MLP deviates the most out of the three models. However, it is the author's viewpoint that the error in predictions is very minimal for all three models, meaning the models are within the acceptable range of errors to be used for MPC.

4.5 Prediction of Partially Observable Cart-Pole

Similar to Section 4.4, we now evaluate the performance of our quantum LSTM, classical LSTM and MLP models trained on the POCP dataset as described in Chapter 3 in terms of their ability to predict future states of the system. Because we have now omitted the cart-pole system's velocity and angular velocity information during the training, it is interesting to observe if our models, especially quantum LSTM, can learn the underlying dynamics of the system by considering the data from the previous time steps.

Unlike the FOCP case, where a single model is trained to predict the entire delta state given the complete state-action pair, the POCP problem is approached using an ensemble modeling technique. Specifically, separate models are trained to individually predict the cart's position and pole's angle.

For the *Angle model*, the input vector is represented as $\mathbf{u} = \begin{bmatrix}\text{Position} & \text{Angle} & \text{Action}\end{bmatrix}$ and the output is given by $\mathbf{y} = \begin{bmatrix}\text{Angle}\end{bmatrix}$. Similarly, for the *Position model*, the input vector is defined as $\mathbf{u} = \begin{bmatrix}\text{Position} & \text{Angle} & \text{Action}\end{bmatrix}$ with the output being $\mathbf{y} = \begin{bmatrix}\text{Position}\end{bmatrix}$.

This ensemble-based approach was adopted due to the improved loss values and more accurate state rollouts observed for each model, compared to a single model jointly predicting both the position and angle.

The configuration of the models is detailed in Table 4.5.

Table 4.5: Angle and position model configuration details

Model	Qubits	Hidden Units	Batch Size	Learning Rate
Quantum LSTM	4	16	64	0.05
Classical LSTM	-	5	64	0.0002
MLP	-	10	64	0.001

Figure 4.15 shows the test loss values obtained for both angle and position models for quantum LSTM, classical LSTM, and MLP for three repetitive trainings. The first observation

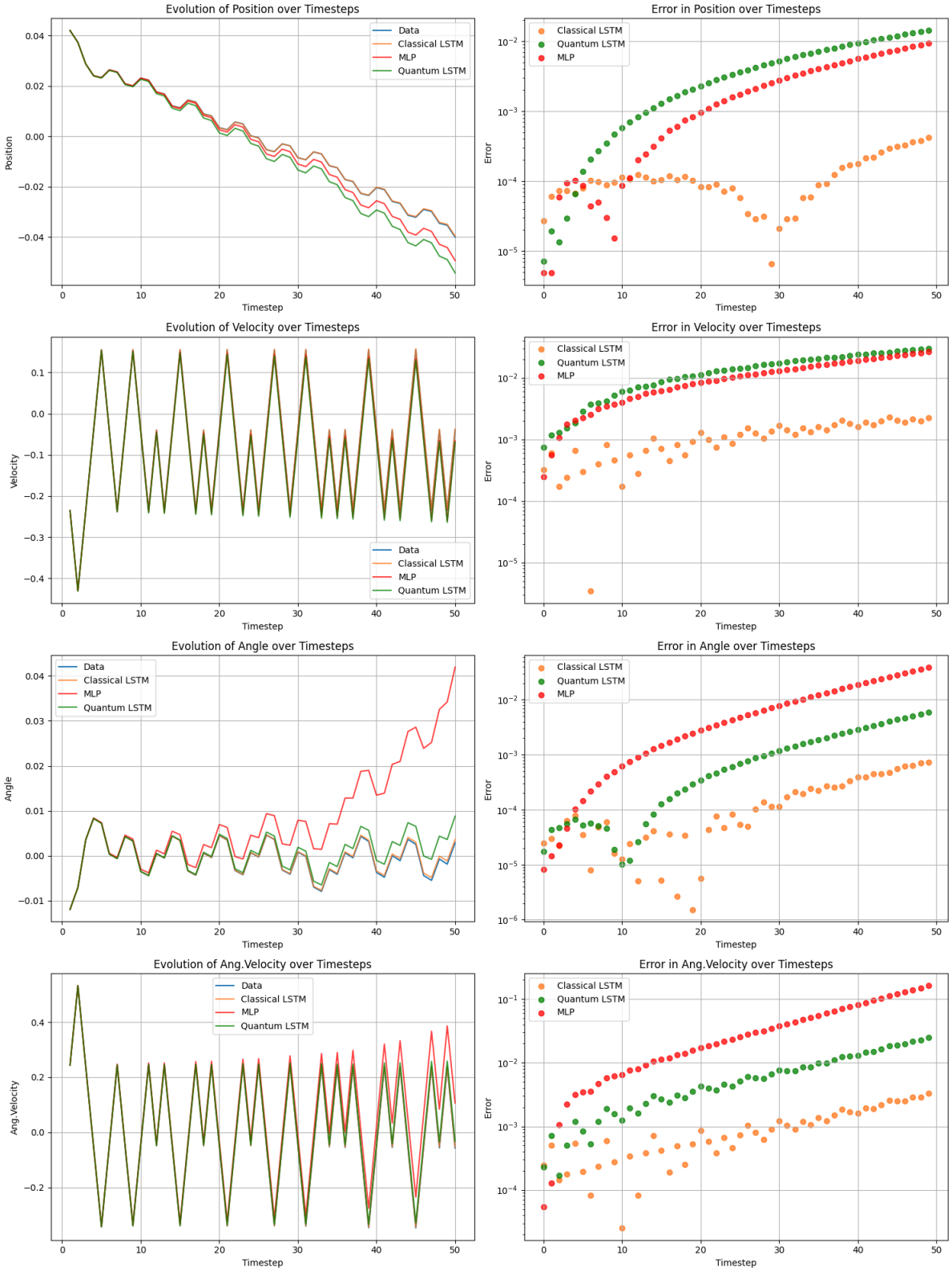

Figure 4.14: Cart-pole rollouts and their absolute error comparison between classical LSTM, quantum LSTM, and MLP

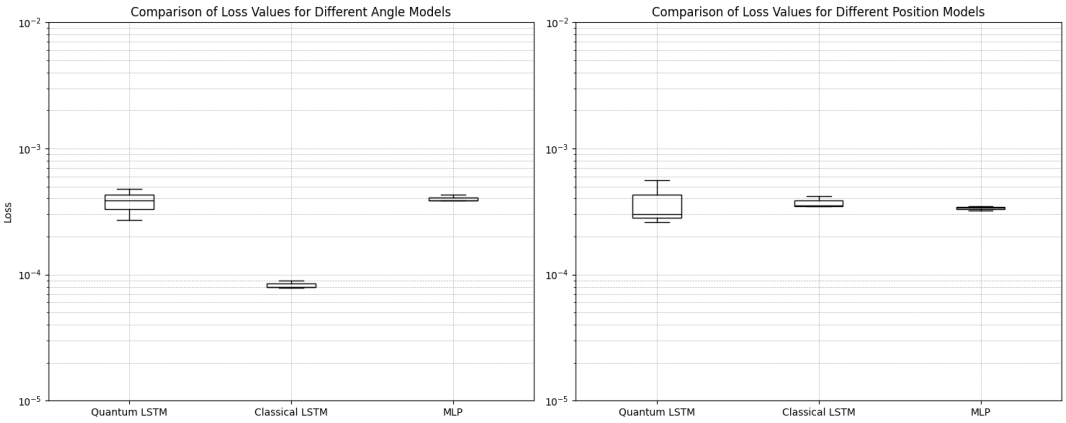

Figure 4.15: Loss comparison for quantum LSTM, classical LSTM, and MLP for POCP benchmark

is that the loss values are much higher when compared to the FOCP case due to the missing velocity information from the state tuple. We observe that, for the position model, the loss values obtained from quantum LSTM, classical LSTM, and MLP are in the same range. classical LSTM shows a better loss in the angle model, followed by quantum LSTM and MLP.

The rollouts of the models are depicted in Figure 4.6, along with the corresponding absolute error values. Unlike the FOCP case, we now see significant deviations in position and angle for classical LSTM and quantum LSTM as we move forward. Up to the first 20 timesteps, the rollouts of the models stay close to the original data. As prediction errors start accumulating, we observe the deviation in the state.

We observe that the classical LSTM does not deviate much from the original position data, unlike the quantum LSTM. However, both the models deviate heavily from the original angle data, with classical LSTM deviating less than quantum LSTM. MLP deviates the most in both position and angle.

4.6 Model Predictive Control for Fully Observable Cart-Pole

After establishing the capability of all three models to predict future states through rollouts, we look at the performance of the MPC strategy incorporating these models in balancing the FOCP environment in this section. The objective of the MPC strategy is to balance the cart-pole by applying forces and keeping it within its bounds. We evaluate the effectiveness of the MPC strategy based on its ability to maintain system stability over time.

Ten experiments per model have been conducted, where each experiment begins with a randomly drawn starting state obtained from the OpenAI Gym cart-pole environment. The results presented in this section depict the statistics of the MPC strategy across all ten experiments. 100 particles have been used and 5 PSO iterations have been conducted. Same number of fitness function evaluations were applied for each model.

It is important to note that, just like the training of a quantum LSTM model, using the

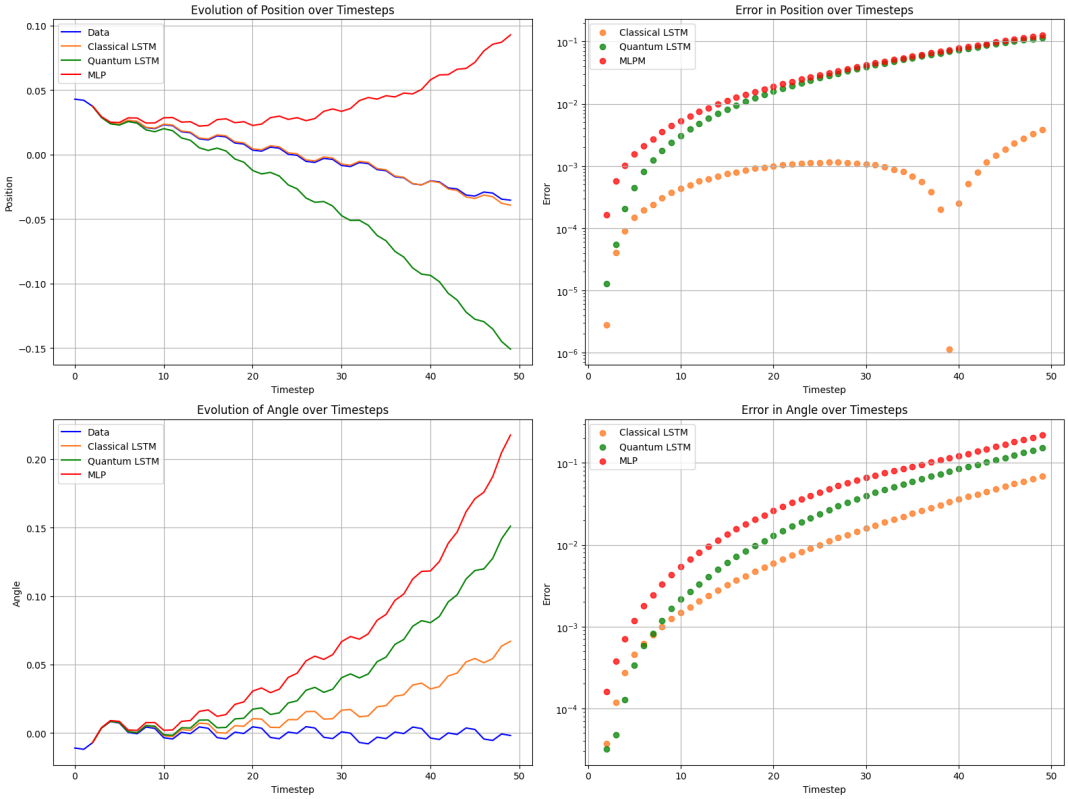

Figure 4.16: POCP rollouts and their absolute error comparison between classical LSTM, quantum LSTM, and MLP

model in the PSO to predict a batch of states corresponding to each particle is equally time-consuming. The average time taken to optimize one action for each model across all ten experiments is tabulated in Table 4.6.

Table 4.6: Time taken to optimize single action

	Time taken (in seconds)
Quantum LSTM	240
Classical LSTM	0.5
MLP	0.5

Figure 4.17 presents the evolution of the cart-pole state variables along with the rewards obtained in one of the ten experiments conducted for the quantum LSTM model over 5000 timesteps. We can infer from the results that the MPC strategy successfully maintained the pole angle within a narrow range of around zero degrees, preventing it from falling. Similarly, the cart position remained within the defined bounds, ensuring the system remained stable throughout the trial. Figure 4.18 and Figure 4.19 show the cart-pole balanced using the classical LSTM and MLP models respectively.

The reward trajectories across all 5000 steps for each model is depicted in Figure 4.20 for all the 10 experiments conducted with the mean value shown by the thicker line and one standard deviation above and below the mean line shown by the shaded region. Because the reward function aims to keep the cart and the pole close to zero and penalizes any deviations from the zero position or zero angle, we notice in the plots that the reward (or, in our case, penalty) stays close to zero. Though we see regular drifts of the reward away from zero, meaning the position or the angle is drifting away from their desired targets, the MPC strategy tries to rebalance them by penalizing their actions. This behaviour can also be observed in the state variables in Figure 4.17, Figure 4.18, and Figure 4.19, where we notice the state drifting away from the zero position but returning as time progresses.

A box plot of the average reward obtained from balancing the cart-pole environment for 5000 steps is depicted in Figure 4.21. The orange line in the box plot in Figure 4.21 shows the median, the box edges are the interquartile range (IQR) which is the 25th and 75th percentiles, and the whiskers extend to to the most extreme data points not considered outliers. It represents overall spread of the returns across all the ten experiments conducted for each model.

We can infer from the box plots that the classical LSTM model delivers the highest average reward, indicating that it consistently performs better in balancing the cart-pole. The performance of classical LSTM is more stable across all experiments depicted by the narrow IQR compared to quantum LSTM and MLP. Quantum LSTM falls second, followed by MLP. There is a higher performance variability observed for the quantum LSTM that is indicated by the more extensive spread in the box plot. This variability might suggest that the MPC strategy implemented using the model is more sensitive to initial conditions or parameters. The MLP model indicates slightly lower average reward compared to the quantum LSTM, indicating a

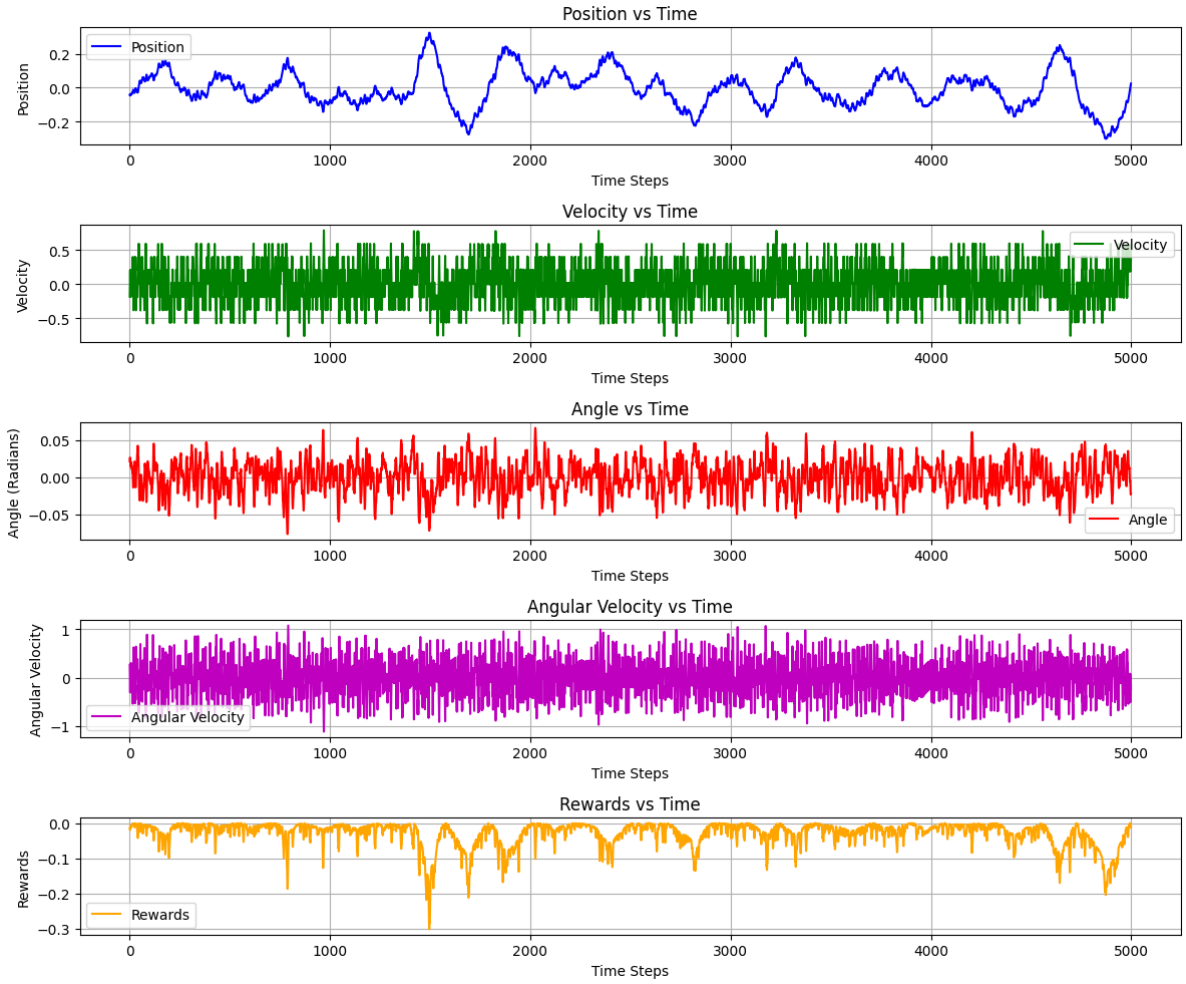

Figure 4.17: Balanced cart-pole using a trained quantum LSTM model over 5000 steps

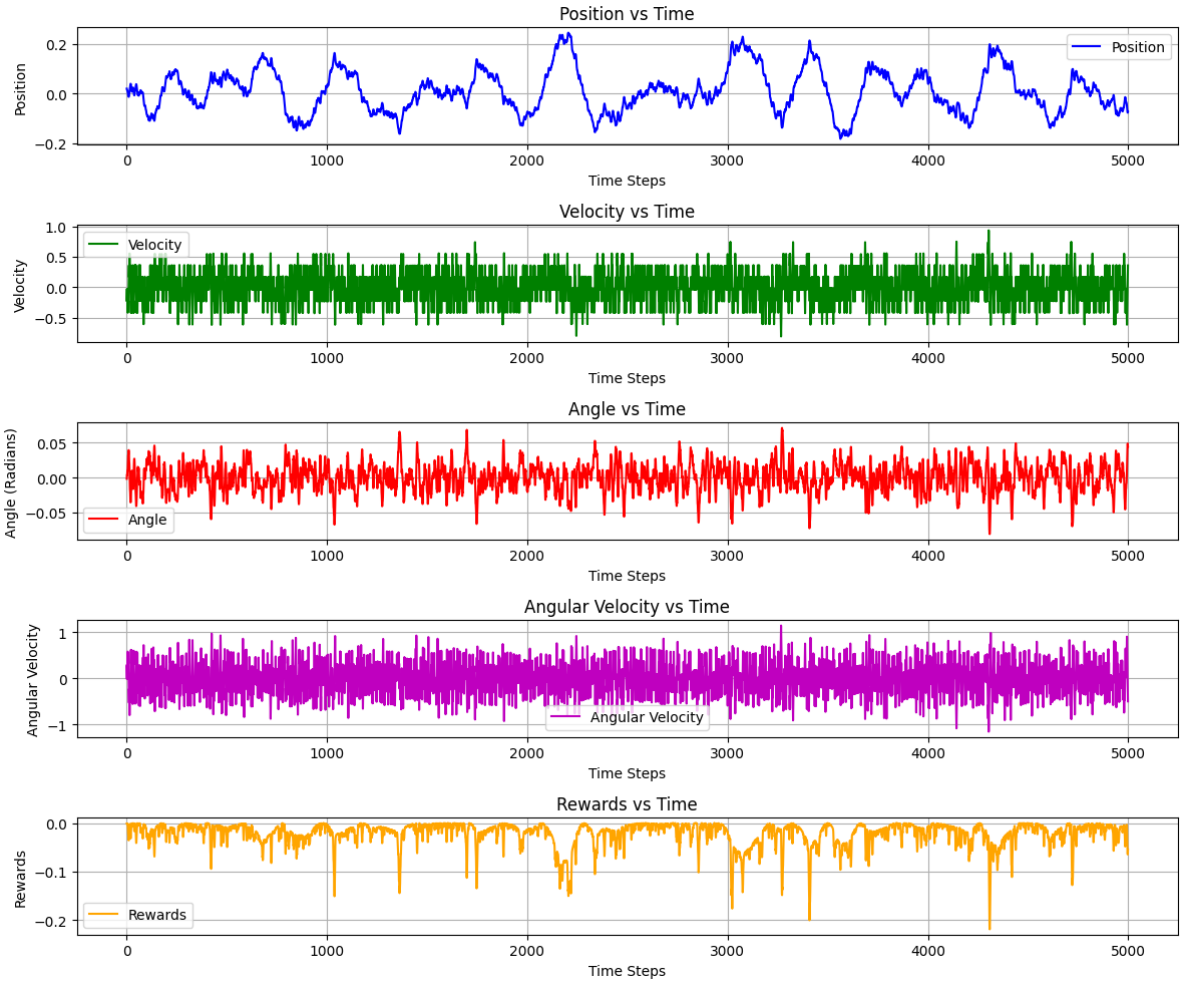

Figure 4.18: Balanced cart-pole using a trained classical LSTM model over 5000 steps

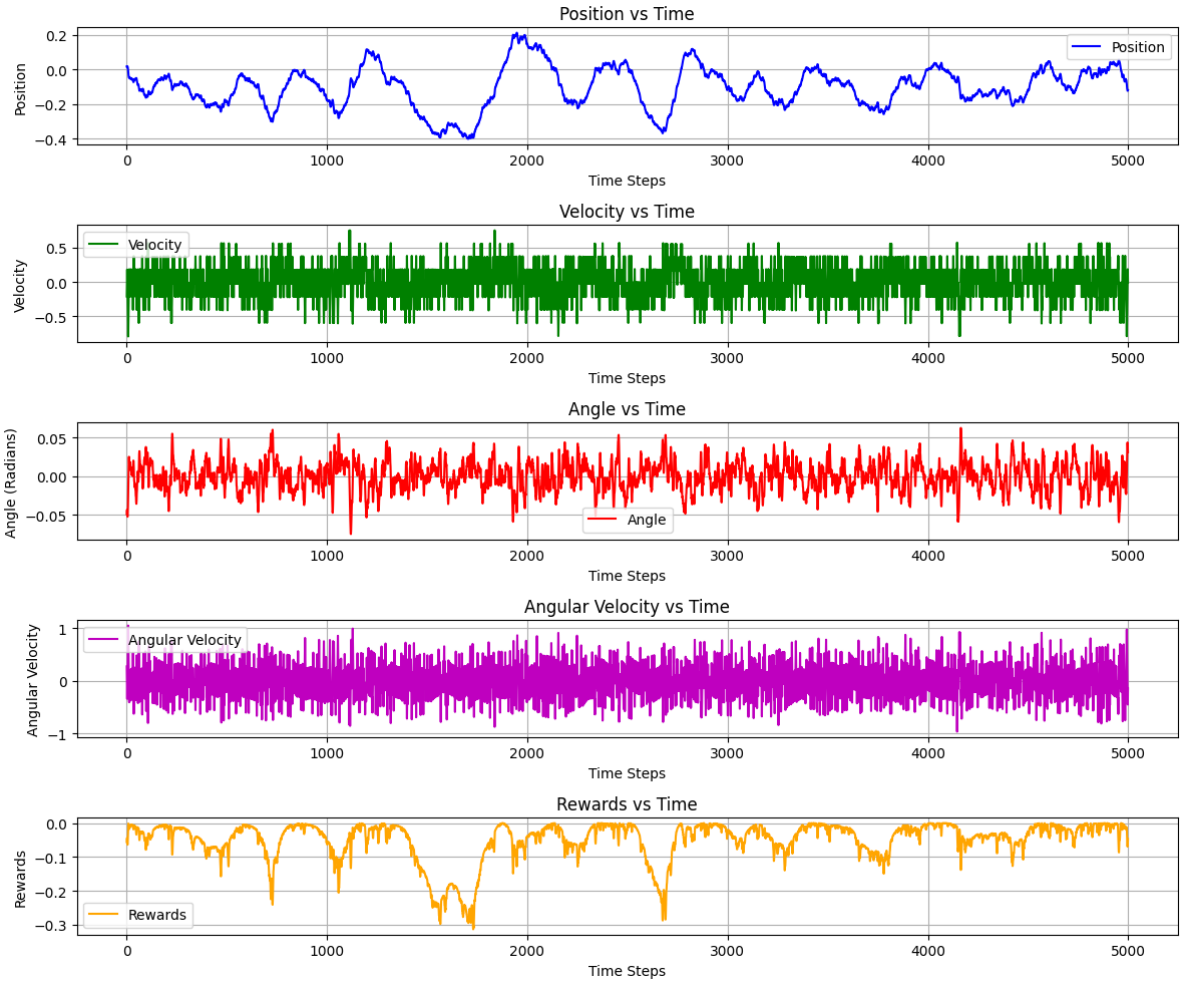

Figure 4.19: Balanced cart-pole using a trained MLP model over 5000 steps

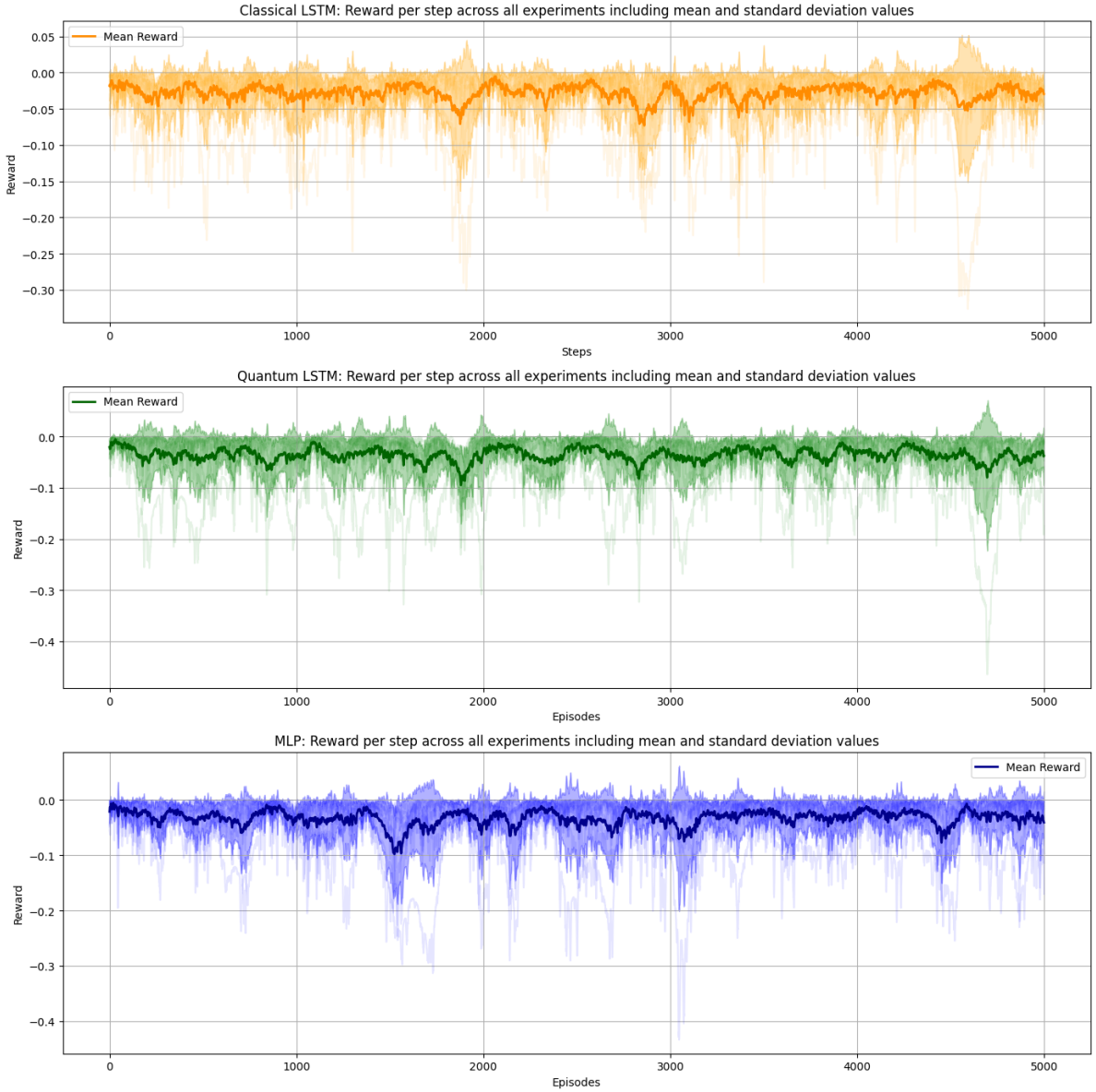

Figure 4.20: Visualization of reward per step across all 10 experiments with mean and standard deviation for classical LSTM, quantum LSTM and MLP

suboptimal control strategy.

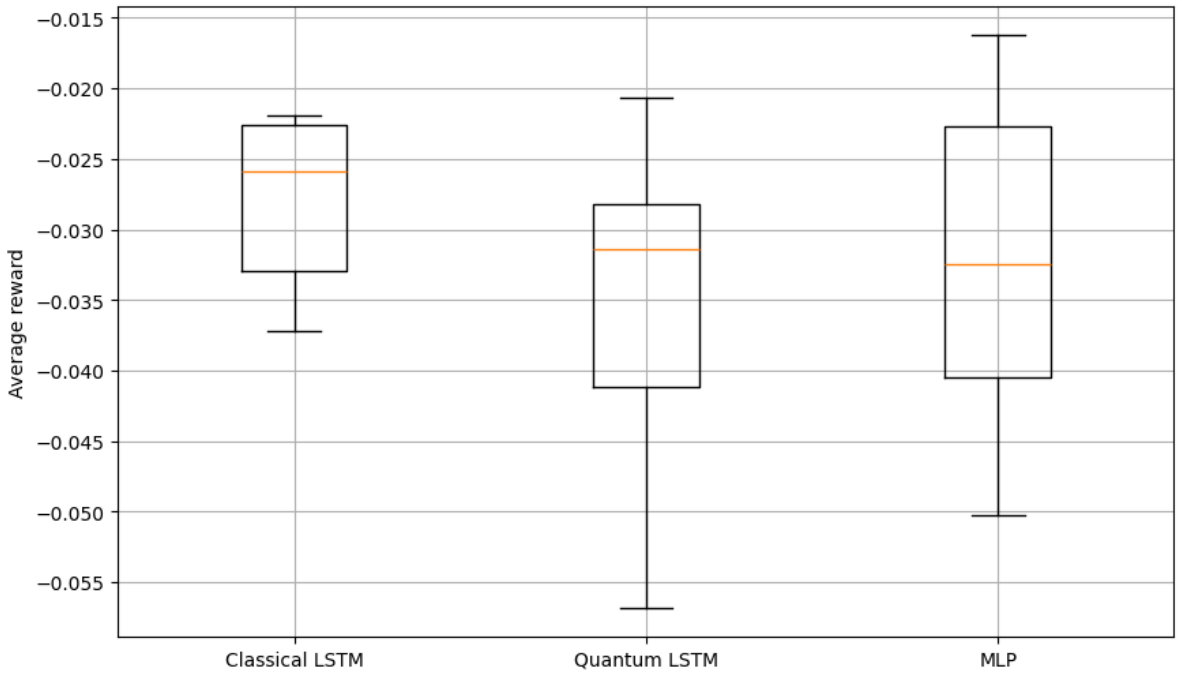

Figure 4.21: Boxplots representing the average reward over 5000 timesteps collected from 10 experiments each for classical LSTM, quantum LSTM and MLP

To conclude, the classical LSTM model consistently delivered better rewards when compared to the quantum LSTM and MLP models. Although the quantum LSTM shows promise, its greater variability indicates room for improvement, particularly in refining its training and stabilizing its control strategy. MLP model proved to be stable but had a slightly lower average reward than the quantum LSTM indicating a suboptimal control strategy.

4.7 Model Predictive Control for Partially Observable Cart-Pole

We now look at the performance of the MPC strategy incorporating the models trained on the POCP environment in balancing the cart-pole in this section. Again, the objective of the MPC strategy here is to balance the cart-pole by applying forces and keeping it within its bounds. We evaluate the effectiveness of the MPC strategy based on its ability to maintain system stability over 1000 steps.

Ten experiments per model have been conducted again with same number of fitness function evaluations for every model, where each experiment begins with two consecutive randomly drawn starting states obtained from the OpenAI Gym cart-pole environment. This is because our quantum and classical LSTM models and the MLP model have been trained with an input sequence length of two, i.e., one past time step and one current time step of the cart-pole, to learn the temporal dynamics of the environment and account for the missing

velocity information. Also, because our models are only trained on the position and angle data, we extract only that information from the starting state. We then duplicate it across all particles for optimization.

Hence, the input to our model would be a batch of state-action pairs again, but only with the position and angle information of the cart-pole. The results presented in this section depict the statistics of the MPC strategy across all ten experiments. One of the main reasons for running the experiment for only 1000 steps here, unlike the 5000 steps in the FOCP case, is the increase in optimization time for the POCP case. We have used 100 particles and 10 PSO iterations for the POCP case.

One of the main reasons for the increase in the optimization time for the POCP case could be due to the ensemble modeling approach, where we now have two quantum LSTM models to pass our input state-action vector to get our position and angle output. Another reason is due to the increase in fitness function evaluations to balance the cart-pole in comparison with the FOCP scenario.

The average time taken to optimize one action for each model across all ten experiments is tabulated in Table 4.7.

Table 4.7: Time taken to optimize single action

	Time taken (in seconds)
Quantum LSTM	720
Classical LSTM	1.5
MLP	1.25

Figure 4.22 presents the evolution of the cart-pole state variables (position and angle) along with the rewards obtained in one of the ten experiments conducted for the quantum LSTM model over 1000 timesteps. We can infer from the results that the MPC strategy successfully maintained the pole angle within a narrow range of around zero degrees, preventing it from falling. Similarly, the cart position remained within the defined bounds, ensuring the system remained stable throughout the trial. Figure 4.23 and Figure 4.24 show the cart-pole balanced using the classical LSTM and MLP models respectively.

The reward trajectories across all 1000 steps for each model is depicted in Figure 4.25 for all the 10 experiments conducted with the mean value shown by the thicker line and one standard deviation above and below the mean line shown by the shaded region.

A box plot of the average reward obtained from balancing the cart-pole environment for 1000 steps is depicted in Figure 4.26. We can infer from the box plot that the quantum LSTM has a slightly higher average reward compared to the classical LSTM and MLP models. The classical LSTM and MLP show wider interquartile ranges (IQR), suggesting higher variability in their average rewards across the 10 experiments. A more compact IQR of the quantum LSTM suggests that it consistently achieved more stable results in this task. The quantum LSTM has a single outlier below its lower whisker, indicating that in one of the experiments, it produced an unusually low average reward compared to the other runs. Overall, the quantum

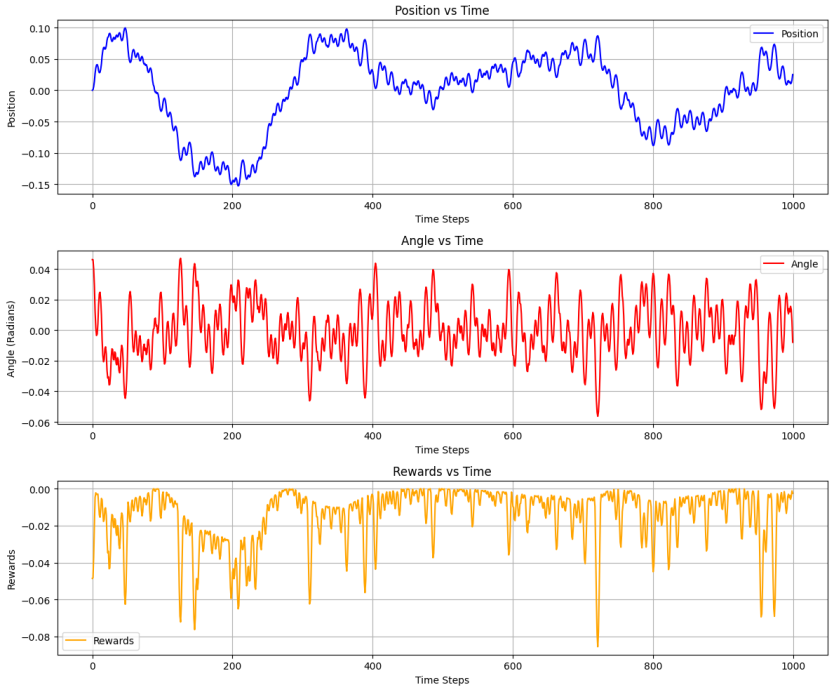

Figure 4.22: Balanced cart-pole using a trained quantum LSTM model over 1000 steps

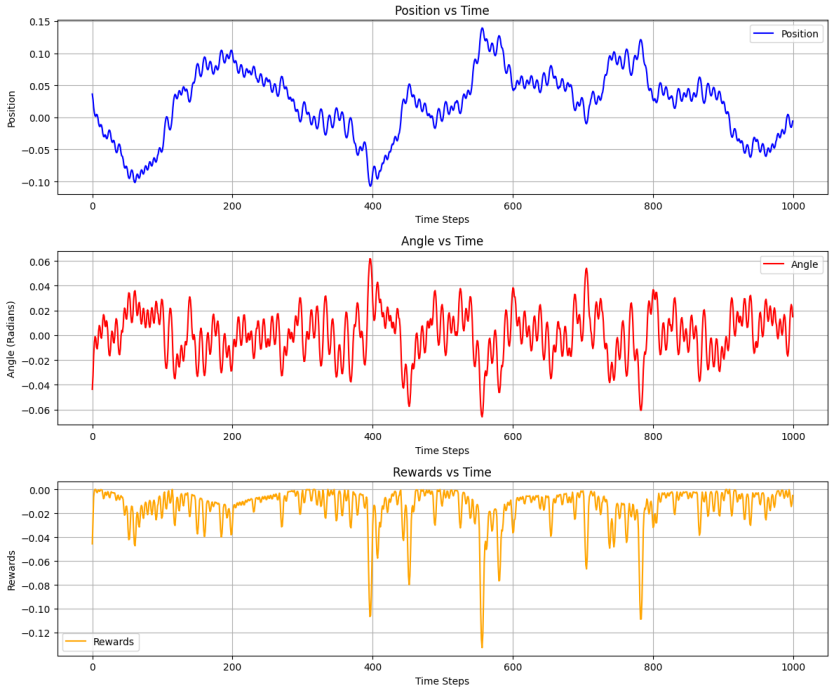

Figure 4.23: Balanced cart-pole using a trained classical LSTM model over 1000 steps

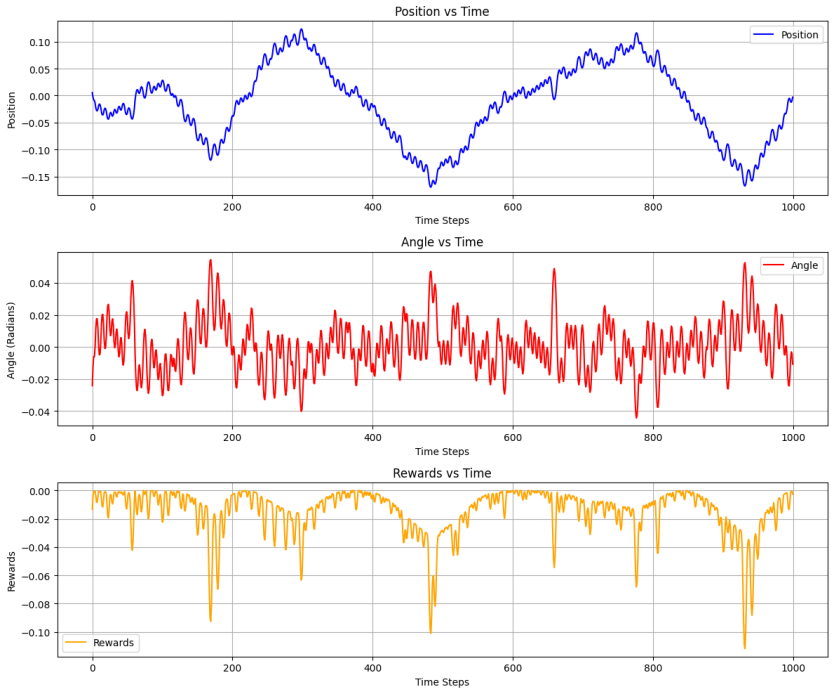

Figure 4.24: Balanced cart-pole using a trained MLP model over 1000 steps

LSTM appears to be a slightly more reliable model in comparison to the classical LSTM and MLP in the POCP task in terms of consistency and average reward, while the other two models show more variability in their results.

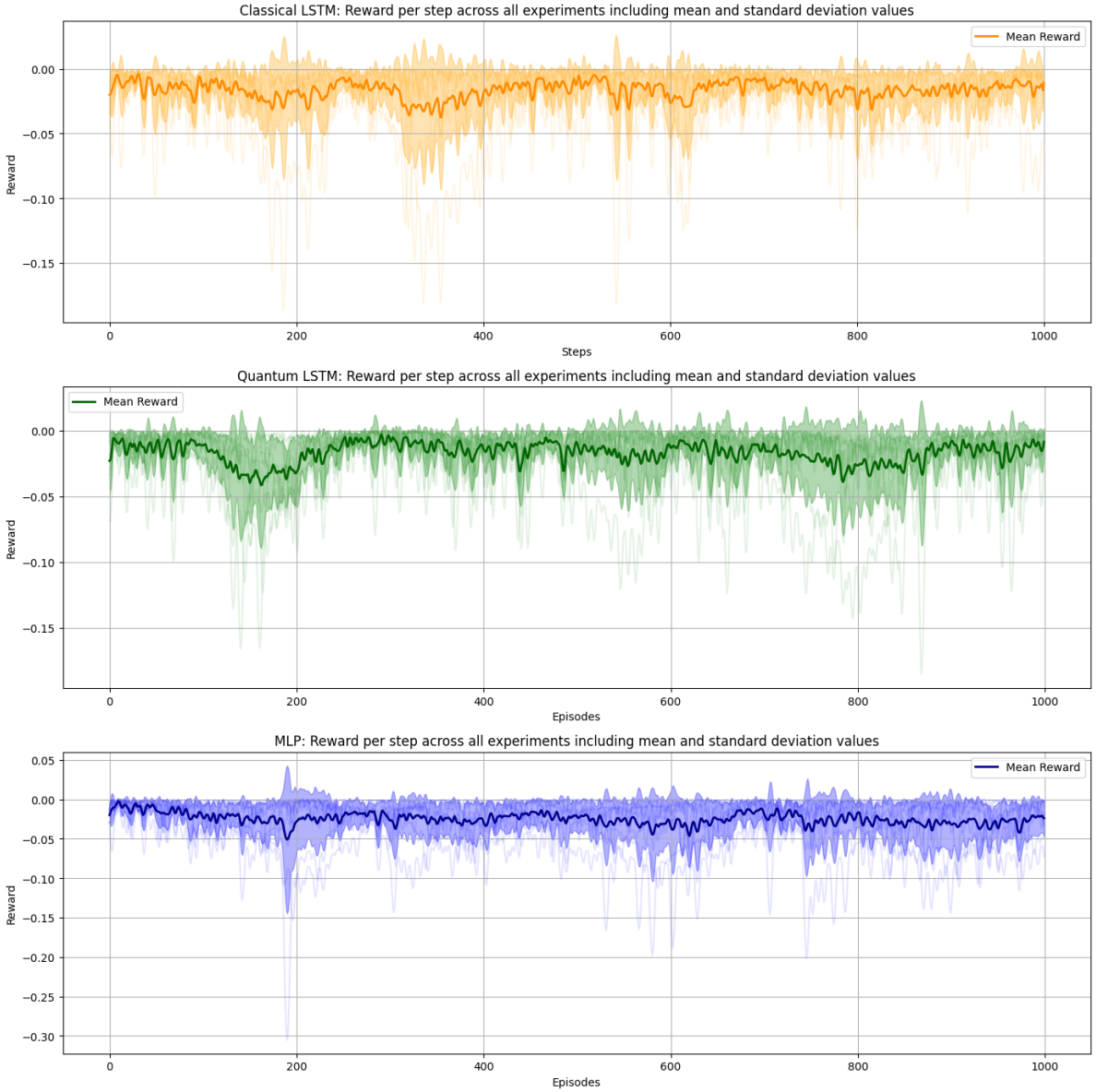

Figure 4.25: Visualization of reward per step across all 10 experiments with mean and standard deviation for classical LSTM, quantum LSTM and MLP

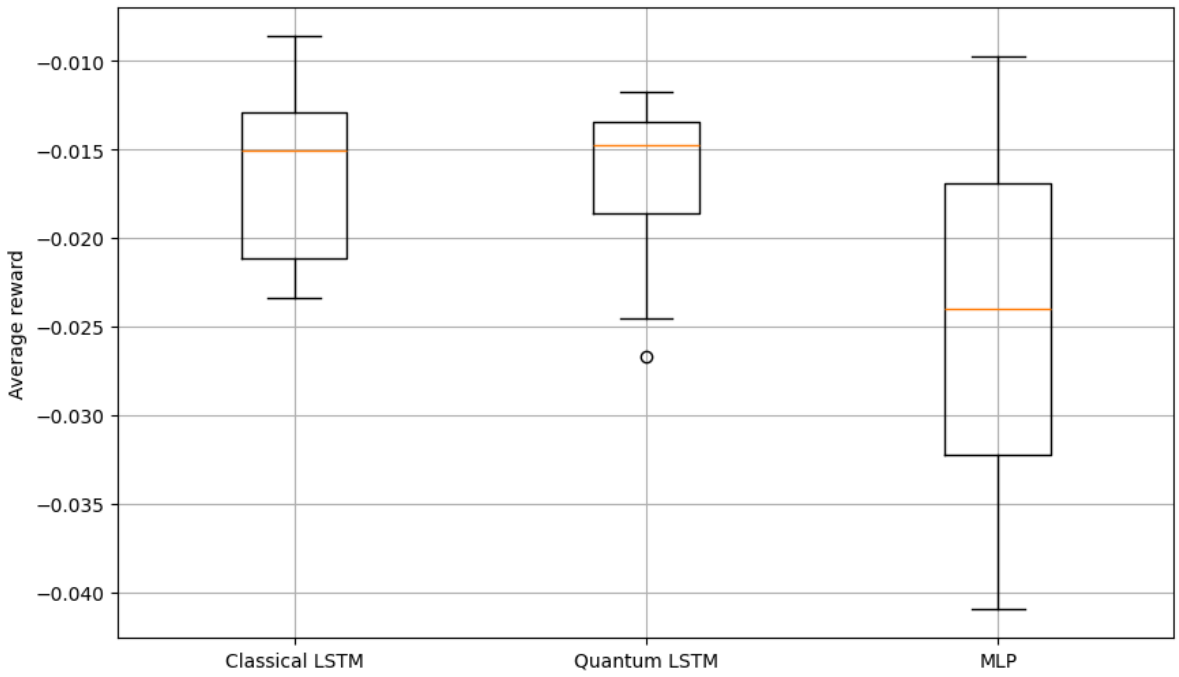

Figure 4.26: Boxplots representing the average reward over 1000 timesteps collected from 10 experiments each for classical LSTM, quantum LSTM and MLP

4.8 Discussion

In this thesis, we extensively experimented with quantum LSTMs. We compared their performance with classical models, with the primary motivation to investigate their feasibility as a precise model in the MPC framework. We began the experimentation by testing the model on the standard function approximation tasks like the sine function, DHO, and the second-order Bessel functions and compared its performance with its classical counterpart. We have obtained reasonably low loss values and good function approximations that validated the model's performance on simple function approximation tasks.

We then trained the quantum LSTM on the data generated from the OpenAI Gym's cart-pole environment, a classic RL benchmark. The test loss values obtained upon three training repetitions of the quantum LSTM show that its loss is approximately an order of magnitude higher than the classical LSTM and MLP trained on the same dataset. However, the loss values are minimal, centered around 10^{-5} for the classical models and 10^{-4} for the quantum LSTM. To test the model's ability to predict future states, we performed rollout experiments, predicting 50 future states using the models one step at a time. The rollout results showed that the classical LSTM deviated the least, followed by quantum LSTM, especially in position and angle plots. The author believes the prediction error is minimal enough for the quantum LSTM to be used as a model in MPC experiments.

A similar study of the model was performed on the POCP benchmark. In this case, the

information from the previous time step of the cart-pole is used along with the current time step in state prediction to account for the missing velocity information in the benchmark. The ensemble modeling approach is used in the POCP benchmark as it produced better loss values and rollouts than a single model. We observe that, for the position model, the loss values obtained from quantum LSTM, classical LSTM, and MLP are in the same range centered around 10^{-3}. Classical LSTM shows a better loss in the angle model, followed by quantum LSTM and MLP. The rollout resutls showed a significant deviation in position and angle for classical LSTM and quantum LSTM from the 20th time step as time progresses due to accumulated errors. We observed that the classical LSTM does not deviate much from the original position data, unlike the quantum LSTM. However, both the models deviate heavily from the original angle data, with classical LSTM deviating less than quantum LSTM. MLP has shown the maximum deviation in both position and angle when compared to the classical and quantum LSTM.

Results from the MPC experiments for the FOCP benchmark indicated that the quantum LSTM successfully balanced the cart-pole in all ten experiments with different starting states. Box plot results of the average reward of different models show that classical LSTM delivered the highest average reward, followed by quantum LSTM and then MLP. A higher performance variability is shown by the quantum LSTM, which might indicate that the model is more sensitive to initial conditions and parameters. The MLP model indicates slightly lower average reward than the quantum LSTM, indicating a suboptimal control strategy. The time taken to optimize a single action is also significantly higher when using the quantum LSTM, which might improve in the future upon introducing parallelization of batch predictions in the optimization step.

Results of the MPC for the POCP environment showed that quantum LSTM has a slightly higher average reward than the classical LSTM and MLP models for 1000 time steps. Also, the interquartile range of the quantum LSTM is compact compared to classical LSTM and MLP, suggesting that the model consistently achieved more stable results in this task. The quantum LSTM also has a single outlier, which shows that a low average reward was obtained in one of the experiments compared to other runs.

5 Conclusion and Outlook

In this thesis, we presented the potential of quantum LSTM networks compared to their classical counterparts for system identification in MPC for the FOCP and POCP benchmarks. We have leveraged PSO for action sequence optimization in our MPC strategy. PSO performs an online optimization of an action sequence and outputs an action for a given system state. Our trained models are used to predict future states during this optimization process, given a batch of state-action pairs as inputs.

We first presented the theoretical background on MPC, RL in control systems, and PSO, followed by introducing fundamentals of quantum computing theory and the usage of VQCs in LSTMs to construct the quantum LSTM network in Chapter 2. We then discussed the methodology and implementation details in Chapter 3, understanding the cart-pole benchmark, system identification process with our quantum and classical LSTM networks, and integrating our trained quantum and classical LSTM models in the MPC strategy to balance the FOCP and POCP environment. Chapter 4 discusses the results of various experiments conducted using our quantum and classical LSTM networks, and MLP. The presented results show that quantum LSTMs can successfully balance FOCP and POCP benchmarks. The findings indicate that while the classical LSTM achieved higher average rewards with greater consistency, the quantum LSTM demonstrated potential but suffered from increased variability in the FOCP benchmark. In the POCP problem, quantum LSTM achieved a slightly higher average reward than the classical LSTM and MLP models for 1000 time steps. The results also suggested that the model consistently achieved more stable results in this task.

This thesis makes the following key contributions with regards to quantum LSTM networks:

- Developed a quantum LSTM-based system identification approach and integrated it into a MPC framework for the FOCP and POCP benchmarks.
- Conducted a comparative analysis of classical LSTM, quantum LSTM, and MLP models in terms of prediction accuracy and control performance.
- Demonstrated the effectiveness of using PSO for optimizing control actions within both classical and quantum MPC frameworks.

Many avenues for future work arise from this thesis. Firstly, optimizing the training and inference process of quantum LSTM networks could significantly help develop better networks for control tasks and other applications. One of the major obstacles to working with quantum models on quantum simulators is the time-consuming training and inference processes. As the complexity of the network increases, the waiting time increases substantially.

Optimizing the network to parallelize the process can be highly beneficial in testing various network parameters and optimization techniques for longer steps to better understand the models' performance. Utilizing Pennylane's lightning backend has shown a 2x speedup in the computations in this research compared to the default qubit backend. With the rising interest in QML, several other quantum simulator backends and efficient gradient computation schemes for quantum circuits are being deployed regularly. Detailed research can be conducted to select the optimal simulator backends and differentiation methods based on the problem size. This can help in prototyping useful QML methods that can potentially be implemented and tested on real quantum hardware. Another research direction is to consider more complicated environments that resemble more realistic use cases, like the Industrial benchmark [32], and test the quantum LSTMs on such sophisticated environments, which could demonstrate the true potential of quantum models in real-world control problems. Finally, studying the effect of preprocessing on the encoding strategies and the resulting expressivity of the quantum LSTM is another exciting research direction to consider.

The findings of this thesis can have several implications for future research in model-based control using quantum-enhanced system models. The demonstrated potential of quantum LSTM suggests that further work on optimizing and stabilizing quantum circuit-based models could yield promising results. The quantum LSTM can be applied to more complex RL environments to test its potential. Additionally, integrating quantum-enhanced models in real-world control applications could become feasible as quantum hardware advances. These insights provide a basis for further exploration into the development of hybrid classical-quantum algorithms for dynamic system identification and control.

List of Figures

List of Tables

Bibliography

[1] J. B. Rawlings. "Tutorial Overview of Model Predictive Control". In: *IEEE Control Systems Magazine* 20.3 (2000), pp. 38–52. DOI: 10.1109/37.845037.

[2] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[3] I. Hammoud, S. Hentzelt, T. Oehlschlaegel, and R. Kennel. "Learning-based model predictive current control for synchronous machines: An LSTM approach". In: *European Journal of Control* 68 (2022), p. 100663.

[4] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: http://dx.doi.org/10.1038/s42254-021-00348-9.

[5] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang. "Quantum Long Short-Term Memory". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), pp. 8622–8626. URL: https://api.semanticscholar.org/CorpusID:221470351.

[6] S. Y.-C. Chen. *Quantum deep recurrent reinforcement learning*. 2022. arXiv: 2210.14876 [quant-ph]. URL: https://arxiv.org/abs/2210.14876.

[7] Farama Foundation. *Cart Pole - Gymnasium Documentation*. Accessed: 2023-10-15. 2023. URL: https://gymnasium.farama.org/environments/classic_control/cart_pole/.

[8] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. "Learning finite-state controllers for partially observable environments". In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. UAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., 1999, pp. 427–436. ISBN: 1558606149.

[9] D. Hein, A. Hentschel, T. Runkler, and S. Udluft. "Particle Swarm Optimization for Model Predictive Control in Reinforcement Learning Environments". In: Feb. 2018, pp. 401–427. ISBN: 9781522551348. DOI: 10.4018/978-1-5225-5134-8.ch016.

[10] M. Jung, P. R. da Costa Mendes, M. Önnheim, and E. Gustavsson. "Model Predictive Control when utilizing LSTM as dynamic models". In: *Engineering Applications of Artificial Intelligence* 123 (2023), p. 106226. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106226. URL: https://www.sciencedirect.com/science/article/pii/S0952197623004104.

[11] S. Eisenmann, D. Hein, S. Udluft, and T. A. Runkler. *Model-based Offline Quantum Reinforcement Learning*. 2024. arXiv: 2404.10017 [quant-ph]. URL: https://arxiv.org/abs/2404.10017.

[12] M. Periyasamy, M. Hölle, M. Wiedmann, D. D. Scherer, A. Plinge, and C. Mutschler. *BCQQ: Batch-Constraint Quantum Q-Learning with Cyclic Data Re-uploading*. 2024. arXiv: 2305.00905 [quant-ph]. URL: https://arxiv.org/abs/2305.00905.

[13] A. Asadi, A. Dusko, C.-Y. Park, V. Michaud-Rioux, I. Schoch, S. Shu, T. Vincent, and L. J. O'Riordan. *Hybrid quantum programming with PennyLane Lightning on HPC platforms*. 2024. arXiv: 2403.02512 [quant-ph]. URL: https://arxiv.org/abs/2403.02512.

[14] E. S. Meadows and J. B. Rawlings. "Model Predictive Control". In: *Nonlinear Process Control*. Ed. by M. A. Henson and D. E. Seborg. Prentice Hall, 1997, pp. 233–310.

[15] S. Li, Y. Liu, and X. B. Qu. "Model Controlled Prediction: A Reciprocal Alternative of Model Predictive Control". In: *IEEE/CAA Journal of Automatica Sinica* 9.6 (June 2022), pp. 1107–1110. DOI: 10.1109/JAS.2022.105611.

[16] W.-H. Kwon, A. Bruckstein, and T. Kailath. "Stabilizing state-feedback design via the moving horizon method". In: vol. 37. Mar. 1983, pp. 234–239. DOI: 10.1109/CDC.1982.268433.

[17] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer London, 2011. ISBN: 9780857295019. URL: https://books.google.de/books?id=kBnz6OFxO8wC.

[18] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[19] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[20] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, and et al. "Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis". In: *Machine Learning* 110.9 (2021), pp. 2419–2468. DOI: 10.1007/s10994-021-05961-4.

[21] B. Singh, R. Kumar, and V. P. Singh. "Reinforcement Learning in Robotic Applications: A Comprehensive Survey". In: *Artificial Intelligence Review* 55 (2022), pp. 945–990. DOI: 10.1007/s10462-021-09997-9.

[22] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN: 9781139495486.

[23] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, 2021. ISBN: 9783030830984. URL: https://books.google.de/books?id=-N5IEAAAQBAJ.

[24] J. Cunningham and J. Zhuang. *Investigating and Mitigating Barren Plateaus in Variational Quantum Circuits: A Survey*. 2024. arXiv: 2407.17706 [quant-ph]. URL: https://arxiv.org/abs/2407.17706.

[25] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[26] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang. *Quantum Long Short-Term Memory*. 2020. arXiv: 2009.01783 [quant-ph]. URL: https://arxiv.org/abs/2009.01783.

[27] J. Lindsay and R. Zand. *A Novel Stochastic LSTM Model Inspired by Quantum Machine Learning*. 2023. arXiv: 2305.10212 [cs.LG]. URL: https://arxiv.org/abs/2305.10212.

[28] M. Schuld, R. Sweke, and J. J. Meyer. "Effect of data encoding on the expressive power of variational quantum-machine-learning models". In: *Physical Review A* 103.3 (Mar. 2021). ISSN: 2469-9934. DOI: 10.1103/physreva.103.032430. URL: http://dx.doi.org/10.1103/PhysRevA.103.032430.

[29] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. ISSN: 1476-4687. DOI: 10.1038/s41586-019-0980-2. URL: http://dx.doi.org/10.1038/s41586-019-0980-2.

[30] T. Jones and J. Gacon. *Efficient calculation of gradients in classical simulations of variational quantum algorithms*. 2020. arXiv: 2009.02823 [quant-ph]. URL: https://arxiv.org/abs/2009.02823.

[31] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[32] D. Hein, S. Depeweg, M. Tokic, S. Udluft, A. Hentschel, T. A. Runkler, and V. Sterzing. "A benchmark environment motivated by industrial control problems". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2017, pp. 1–8. DOI: 10.1109/ssci.2017.8280935. URL: http://dx.doi.org/10.1109/SSCI.2017.8280935.