Robotic Systems II: Homework II Report

Student: Vangelis Tsiatsianas

December 6, 2024

Contents

Introduction

This report focuses on modeling and control of a double pendulum system. The tasks include deriving the system dynamics, discretizing them using Runge-Kutta integration, and implementing control strategies such as Linear Quadratic Regulator (LQR) and Convex Model Predictive Control (MPC). The goal is to evaluate the performance of these controllers in stabilizing the system, including under disturbances and noise, and to explore their ability to perform complex motions like swing-up.

1 Modeling

The double pendulum system is a non-linear system consisting of two connected links, each with a mass and length, and torques applied at the joints. The system is modeled in the $\hat{x}$-$\hat{y}$ plane, under the influence of gravity g. The generalized coordinates q_1 and q_2 represent the angles of the two links, and the generalized velocities $\dot{q}_1$ and $\dot{q}_2$ represent their angular velocities. The parameters used are $g = 9.81\,\mathrm{m/s}^2$, $m_1 = m_2 = 1\,\mathrm{kg}$, and $l_1 = l_2 = 0.5\,\mathrm{m}$.

1.1 Dynamics Function

The dynamics of the double pendulum system are described by the continuous equations of motion:

$$\dot{x} = f(x, u) = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix},$$

where the generalized coordinates and velocities are:

$$x = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \in \mathbb{R}^4, \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in \mathbb{R}^2.$$

The system is implemented in the `dynamics(x, u)` function, which computes the time derivative of the state vector $\dot{x}$ based on the given control inputs u.

1.2 Dynamics Derivation

To derive the equations of motion for the system, we begin with the kinetic and potential energy components.

The kinetic energy consists of two components:

$$K = \frac{1}{2} m_1 \left(\dot{x}_1^2 + \dot{y}_1^2 \right) + \frac{1}{2} m_2 \left(\dot{x}_2^2 + \dot{y}_2^2 \right),$$

where:

$$\dot{x}_1 = -l_1 \sin(q_1)\dot{q}_1, \tag{1.1}$$
$$\dot{y}_1 = l_1 \cos(q_1)\dot{q}_1, \tag{1.2}$$
$$\dot{x}_2 = -l_1 \sin(q_1)\dot{q}_1 - l_2 \sin(q_1 + q_2)\dot{q}_2, \tag{1.3}$$
$$\dot{y}_2 = l_1 \cos(q_1)\dot{q}_1 + l_2 \cos(q_1 + q_2)\dot{q}_2. \tag{1.4}$$

The potential energy due to gravity for the two links is:

$$P = m_1 g l_1 \sin(q_1) + m_2 g \left(l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \right).$$

The Lagrangian $L(q, \dot{q})$ is defined as the difference between the kinetic and potential energy of the system:

$$L = K - P.$$

Using the Euler-Lagrange equation for each coordinate q_1 and q_2, the equations of motion can be written as:

$$u_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \quad i = 1, 2,$$

where u_1 and u_2 are the generalized torques applied at each joint.

After calculating the derivatives and simplifying the terms, the dynamic equations for the double pendulum in matrix form are:

$$u = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q),$$

where:

- $M(q)$ is the mass (inertia) matrix,

$$M = \begin{bmatrix} l_1^2 m_1 + l_2^2 m_2 + l_1^2 m_2 + 2 l_1 l_2 m_2 \cos q_2 & l_2^2 m_2 + l_1 l_2 m_2 \cos q_2 \\ l_2^2 m_2 + l_1 l_2 m_2 \cos q_2 & l_2^2 m_2 \end{bmatrix},$$

- $C(q, \dot{q})$ is the Coriolis and centrifugal forces matrix,

$$C(q, \dot{q}) = \begin{bmatrix} -m_2 L_1 L_2 \sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2) \\ m_2 L_1 L_2 \sin(q_2)\dot{q}_1^2 \end{bmatrix}.$$

- $g(q)$ is the gravitational force vector.

$$g(q) = \begin{bmatrix} (m_1 + m_2) g L_1 \cos(q_1) + m_2 g L_2 \cos(q_1 + q_2) \\ m_2 g L_2 \cos(q_1 + q_2) \end{bmatrix}.$$

Thus, the accelerations $\ddot{q}$ are given by:

$$\ddot{q} = M^{-1}\left(u - C(q, \dot{q})\dot{q} + G(q)\right),$$

These equations describe the dynamics of the double pendulum system under the influence of torques applied at the joints, and gravitational forces.

1.3 Discretization

To discretize the continuous system dynamics, Runge-Kutta 4th Order (RK4) integration was used, allowing precise integration over a fixed time step $\Delta T = 0.05$. This method iteratively calculates the state at each time step on the basis of intermediate evaluations of the continuous dynamics.

1.4 Dynamics Validation

To confirm the correct implementation of the dynamics, different test cases were created. Two of them are shown bellow.

1.4.1 Test Case 1: Stable position at $q = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

As seen in Figure 1, the system remains in equilibrium in this configuration, confirming that the dynamics model behaves as expected.

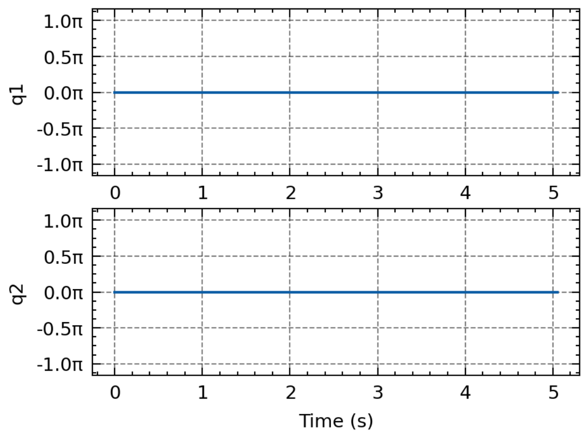


Figure 1: Test Case 1 position at $q_1 = 0, q_2 = 0$

1.4.2 Test Case 2: Stable position at $q = \begin{bmatrix} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{bmatrix}$

In this test case, Newton's law was used to compute the torques required for specific configurations so that the system stays in place.

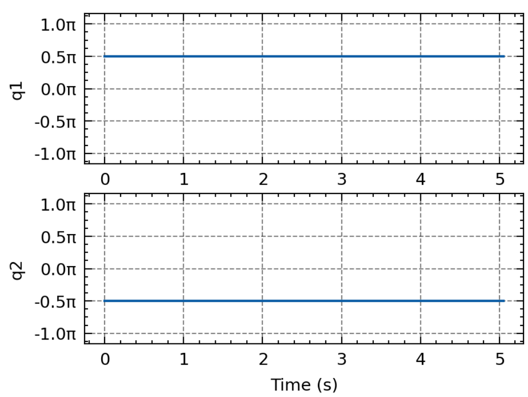


Figure 2: Test Case 2 at $q_1 = \frac{\pi}{2}, q_2 = -\frac{\pi}{2}$

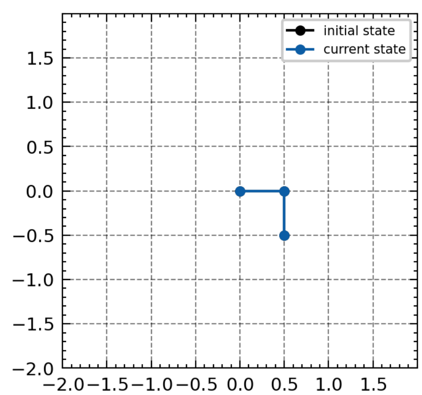


Figure 3: System visualization

The results indicate that the system stabilizes at the given position with the correct torque values, demonstrating the accuracy of the implemented dynamics in handling non-zero angle configurations.

2 Linearization and LQR

2.1 Linearization Around a Fixed Point

To apply LQR, firtsly the nonlinear dynamics were linearized around the fixed point:

$$\bar{x} = \begin{bmatrix} \pi & 0 & 0 & 0 \end{bmatrix}^T, \quad \bar{u} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T.$$

The discrete dynamics are given by:

$$x_{k+1} = f_{\text{discrete}}(x_k, u_k),$$

where x_k and u_k are the state and control inputs at time step k. Linearization involves expanding the Taylor series of $f_{\text{discrete}}(x, u)$, around the fixed point:

$$x_{k+1} = f_{\text{discrete}}(\bar{x}, \bar{u}) + \left.\frac{\partial f_{\text{discrete}}}{\partial x}\right|_{\bar{x},\bar{u}} (x_k - \bar{x}) + \left.\frac{\partial f_{\text{discrete}}}{\partial u}\right|_{\bar{x},\bar{u}} (u_k - \bar{u}) \tag{2.1}$$

where:

$$A = \left.\frac{\partial f_{\text{discrete}}}{\partial x}\right|_{\bar{x},\bar{u}} \quad B = \left.\frac{\partial f_{\text{discrete}}}{\partial u}\right|_{\bar{x},\bar{u}}$$

By defining:

$$\Delta x_k = x_k - \bar{x}, \quad \Delta u_k = u_k - \bar{u}$$

the linearized system dynamics become:

$$\Delta x_{k+1} = A \Delta x_k + B \Delta u_k$$

2.2 LQR Controller

The Linear Quadratic Regulator (LQR) optimally stabilizes the linearized system by minimizing the following cost function:

$$J = \sum_{k=1}^{K-1} \left(\Delta x_k^T Q_k \Delta x_k + \Delta u_k^T R_k \Delta u_k \right) + \Delta x_K^T Q_K \Delta x_K, \tag{2.2}$$

$$\text{s.t.} \quad \Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k.$$

where for the time-invariant case:

$$A = A_k, \quad B = B_k, \quad Q = Q_k, \quad R = R_k, \quad \forall k$$

- Q penalizes state deviations,
- R penalizes control effort.

The problem stated in equation 2.2, can be formulated as a Quadratic Programming Problem as shown:

$$\arg\min_z J(z) = z^T H z \tag{2.3}$$

$$\text{s.t.} \quad Gz = d$$

Assuming that Δx_1 (initial conditions) is given, and define:

$$z = \begin{bmatrix} \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \vdots \\ \Delta x_K \end{bmatrix}, \quad H = \begin{bmatrix} R_1 & & & & \\ & Q_2 & & & \\ & & R_2 & & \\ & & & \ddots & \\ & & & & Q_K \end{bmatrix}$$

$$G = \begin{bmatrix} B_1 & -I & 0 & 0 & & & \\ 0 & A_2 & B_2 & -I & \cdots & & \\ & & & \vdots & \ddots & & \\ & & & & A_{K-1} & B_{K-1} & -I \end{bmatrix}, \quad d = \begin{bmatrix} -A_1 \Delta x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The problem can be solved using the Lagrangian and the optimality conditions from which the KKT system can be defined:

$$\begin{bmatrix} H & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}$$

Using the Riccati recursion, the solution is given by:

$$\Delta u_k = -K_k \Delta x_k \tag{2.4}$$

$$\lambda_k = -P_k \Delta x_k \tag{2.5}$$

where:

$$P_K = Q_K \tag{2.6}$$

$$K_k = \left(R_k + B_k^T P_{k+1} B_k\right)^{-1} B_k^T P_{k+1} A_k, \tag{2.7}$$

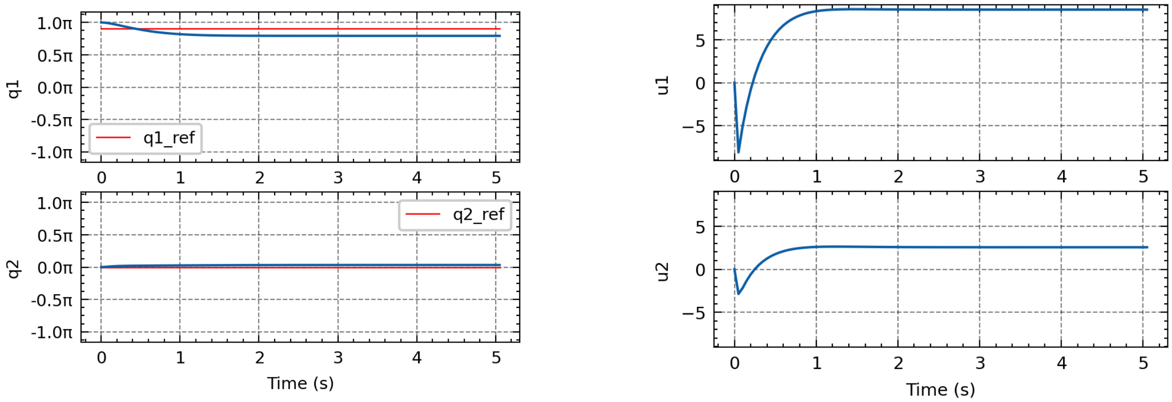
$$P_k = Q_k + A_k^T P_{k+1} \left(A_k - B_k K_k\right). \tag{2.8}$$

By solving the infinite horizor LQR for the time invariant system an optimal control law occurs, as:

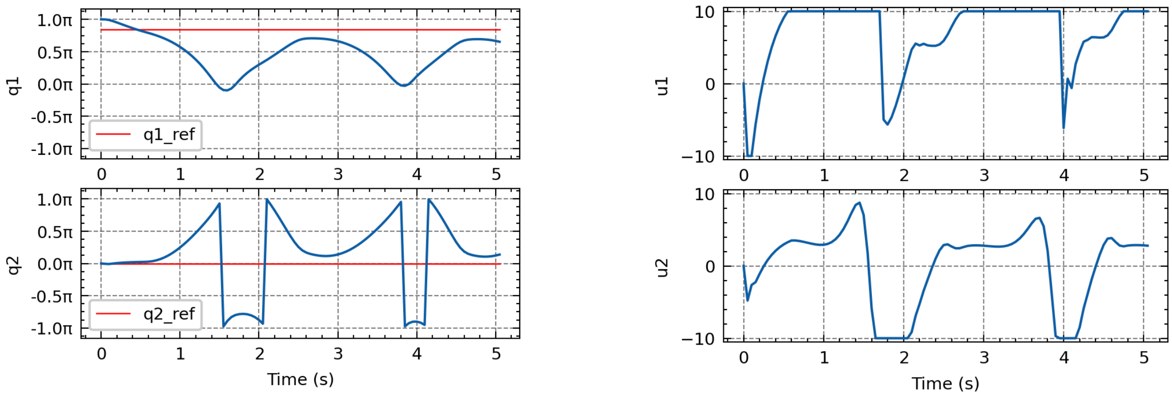
$$u_k = \bar{u} - K \Delta x_k \tag{2.9}$$

2.3 Simulation

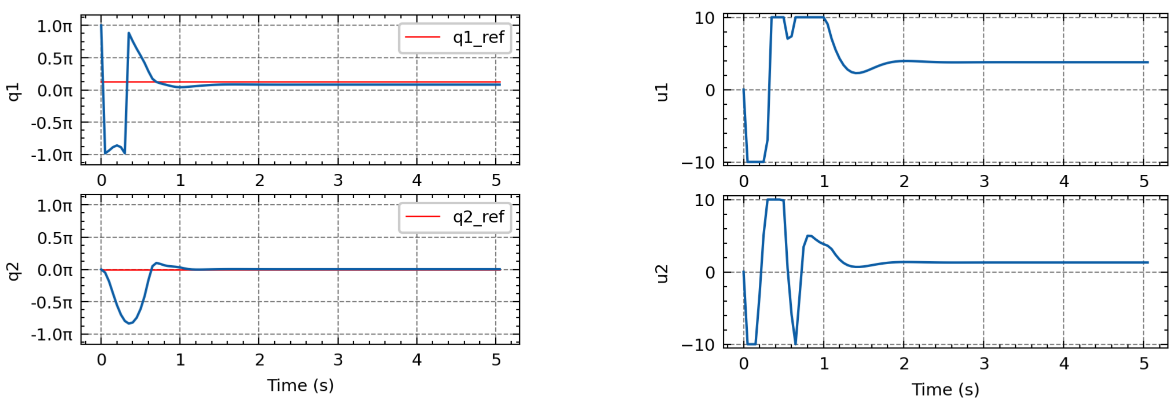
The controller derived in equation 2.9, was used for simulating the system with a time step of $\Delta t =$ 0.05sec and the system's non-linear dynamics. In addition, controls were limited in $[-10, 10]$ range. After performing the evaluation, it was observed that the controller is able to effectively control the system when operating close to the linearization point. However, as the system deviates further from the linearization point, both the accuracy and controllability of the system decrease, as demonstrated in the figures below.



(a) $q_{ref} = \begin{bmatrix} \pi - 0.3, & 0 \end{bmatrix}^T$



(b) $q_{ref} = \begin{bmatrix} \pi - 0.5, & 0 \end{bmatrix}^T$



(c) $q_{ref} = \begin{bmatrix} 0.2, & 0 \end{bmatrix}^T$

Simulation cases, in its case the configuration $q = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^T$ and the applied torques $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$ are plotted over time

3 Controlling via Convex MPC

3.1 Introduction

Model Predictive Control (MPC) is a control strategy that optimizes control inputs over a finite horizon while considering system dynamics, constraints, and a cost function. In this section, convex MPC is applied to the double pendulum system.

3.2 Problem Formulation

The problem is formulated by solving the LQR problem stated in 2.3 for a smaller horizon H. The optimality of the solution is ensured by using a Value Function deriving from the infinite horizon LQR:

$$V_k(x) = \frac{1}{2} x_k^T P_{\text{inf}} x_k \tag{3.1}$$

So the convex MPC problem is defined as:

$$\min_{x_{1:H}, \Delta u_{1:H}} \sum_{k=1}^{H-1} \left(\frac{1}{2}(x_k - x_{ref})^T Q (x_k - x_{ref}) + \frac{1}{2} \Delta u_k^T R \Delta u_k \right) + \frac{1}{2}(x_H - x_{ref})^T P_{inf} (x_H - x_{ref}) \tag{3.2}$$

$$\begin{aligned} \text{s.t} \quad \Delta x_{k+1} &= A_k \Delta x_k + B_k \Delta u_k \\ u_k &\in \mathcal{U} \end{aligned}$$

with:

- $x_k = \bar{x} + \Delta x_k$
- $u_k = \bar{u} + \Delta u_k$
- x_1 the current state
- x_{ref} the reference state

The control problem can be formulated as a single quadratic programming (QP) problem as:

$$\arg\min_z J(z) = z^T H z - hz \tag{3.3}$$

$$\begin{aligned} \text{s.t.} \qquad Gz &= d \\ lb \le & z \le ub \end{aligned}$$

where:

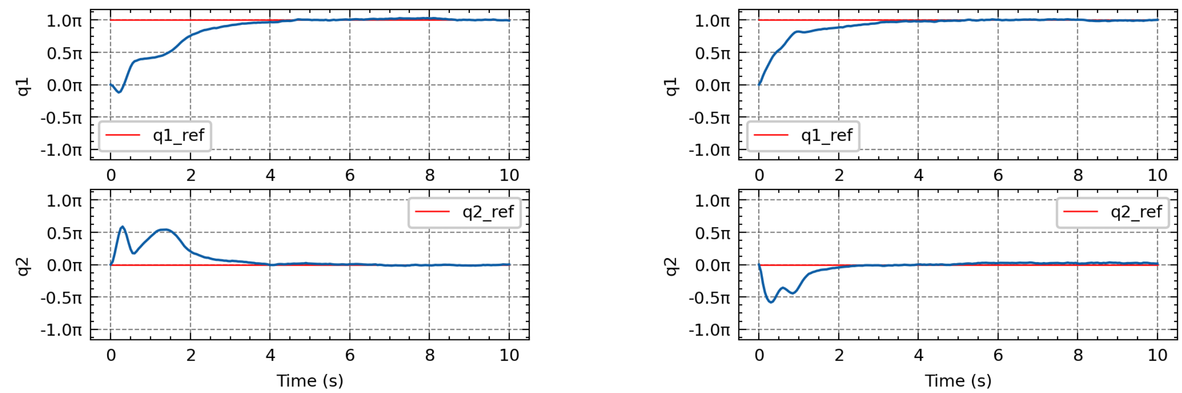
$$z = \begin{bmatrix} \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \vdots \\ \Delta x_H \end{bmatrix}, \quad H = \begin{bmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & P_{inf} \end{bmatrix}, \quad h = \begin{bmatrix} 0 \\ (\bar{x} - x_{ref})^T Q \\ 0 \\ \vdots \\ (\bar{x} - x_{ref})^T P_{inf} \end{bmatrix}$$

$$G = \begin{bmatrix} B & -I & 0 & 0 & & & \\ 0 & A & B & -I & \cdots & & \\ & & & \vdots & \ddots & & \\ & & & & A & B & -I \end{bmatrix}, \quad d = \begin{bmatrix} -A(x_1 - \bar{x}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$lb = \begin{bmatrix} u_{min} - \bar{u} \\ -\text{inf} \\ u_{min} - \bar{u} \\ \vdots \\ -\text{inf} \end{bmatrix}, \quad ub = \begin{bmatrix} u_{max} - \bar{u} \\ \text{inf} \\ u_{max} - \bar{u} \\ \vdots \\ \text{inf} \end{bmatrix}$$

3.3 Results

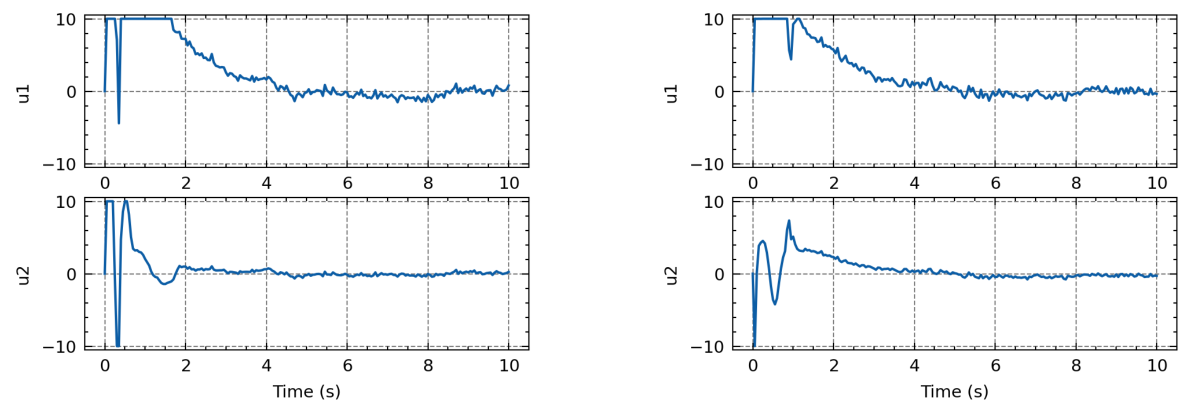
The performance of the convex MPC was evaluated by applying it to the nonlinear system and simulating the system in the same way as in section 2.3. In addition noise was added to the observations (Gaussian with zero mean and diagonal covariance $\mathcal{N}(0, 10^{-3})$), to confirm the robustness of the controllers. Many different cases were simulated, all with the same linearization, around $x_{bar} = \left[\pi, 0, 0, 0\right]^T$ and $u_{bar} = \left[0, 0\right]^T$. The results revealed that both the LQR and the MPC Controller could perform the "swing-up" motion (Figures 5, 6). Convex MPC controller showcased better reference tracking accuracy further away from the lineriazation point in its close area as seen in Figure 7, while trying to minimize the applied torques. The LQR derived controller demonstrated enhanced capability of controlling the system close to equilibriums far away from the linearization point, in contrast to the MPC as showed in Figure 9 where $q_{ref} = \left[\pi - 0.8, \quad 0\right]^T$.



(a) LQR

(b) MPC

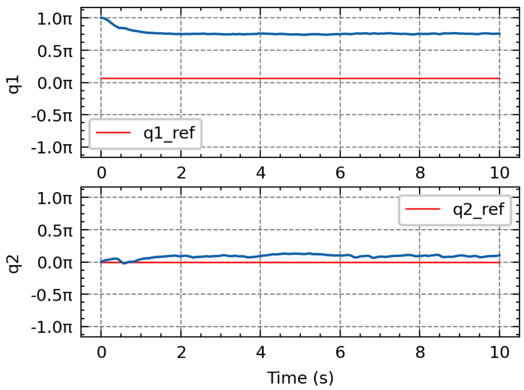
Figure 5: Swing up trajectories



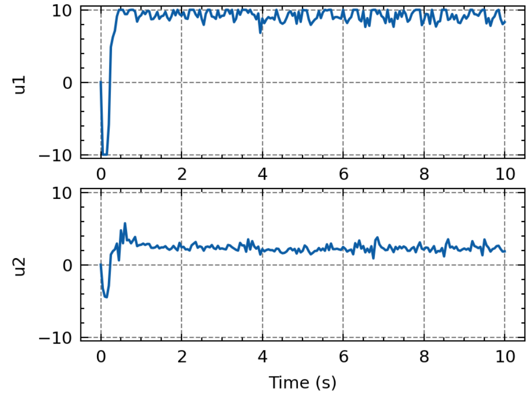
(a) LQR

(b) MPC

Figure 6: Swing up torques

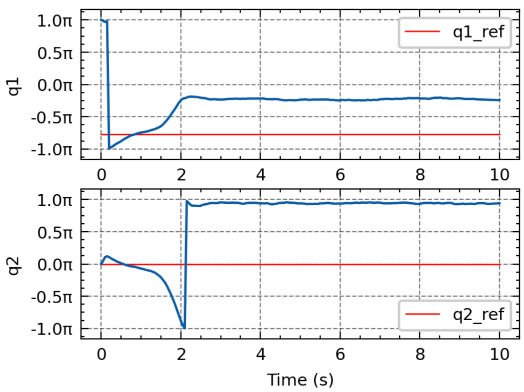


(a) trajectory

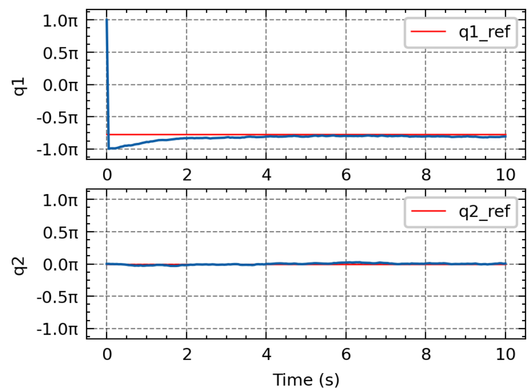


(b) controls

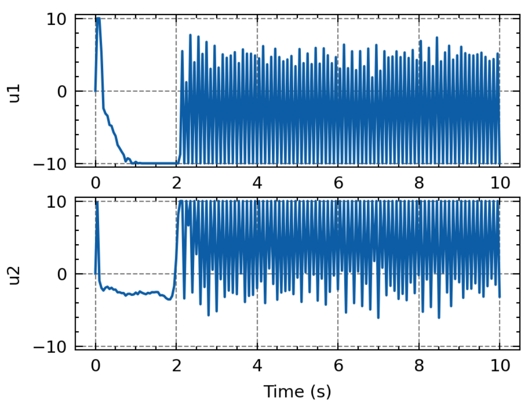
Figure 7: MPC far from linearization, x_{ref} near stable equilibrium point



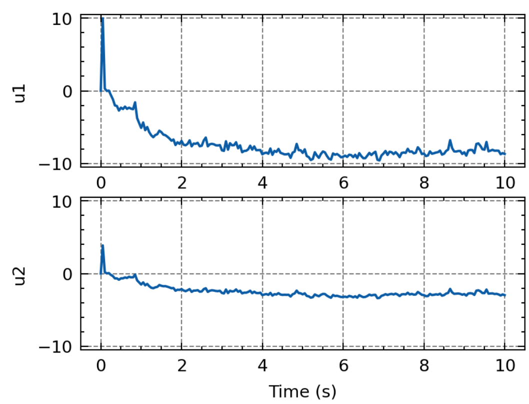
(a) LQR trajectory



(b) MPC trajectory



(a) LQR controls



(b) MPC controls

Figure 9: Tracking accuracy far inside the linearization area