UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Autonomus System

Course Project

Professor:

Giuseppe Notarstefano
Ivano Notarnicola

Students:

Andrea Alboni
Federico Calzoni
Emanuele Monsellato

Academic year 2024/2025

Abstract

A key challenge in the optimal control of robotic systems is the generation of optimal trajectories while considering the system's dynamics and constraints. This project focuses on the design and implementation of an optimal control law for a flexible robotic arm, modeled as a planar two-link robot with torque applied to the first joint.

The setup phase involves discretizing the robot's dynamics and formulating the discrete-time state-space equations.

The first task focuses on generating an optimal trajectory between two equilibrium points of the robotic arm. The equilibrium states are computed using a root-finding algorithm and a symmetric reference trajectory is defined between them.

The second task extends the first by introducing a smooth desired trajectory, for which a quasi-static trajectory is computed as an initial guess. This trajectory is refined through optimal control methods to minimize the cost function.

The third task involves trajectory tracking using Linear Quadratic Regulator (LQR) control. By linearizing the system dynamics around the optimal trajectory, a closed-loop controller is designed to track the reference trajectory while compensating for initial perturbations.

The fourth task leverages Model Predictive Control (MPC) to achieve trajectory tracking. The controller is developed to handle constraints dynamically, showcasing robustness to perturbed initial conditions.

Finally, the fifth task includes the visualization of the robotic arm's motion by animating the results of the LQR trajectory tracking task.

Contents

Chapter 1

Problem setup

1.1 Overview of the Flexible Robotic Arm Model

This work addresses the optimal control of a planar two-link robotic manipulator characterized by its underactuated nature. The system dynamics evolve in a two-dimensional plane and consist of two rigid links connected by revolute joints, with actuation provided only at the first joint through an applied torque u.

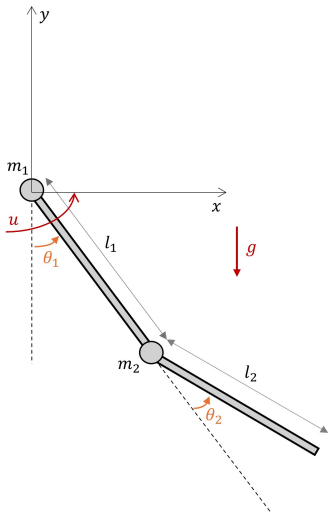


Figure 1.1: Model of the flexible arm.

The system state is defined as $\mathbf{x} = [\dot{\theta}_1, \dot{\theta}_2, \theta_1, \theta_2]^\top \in \mathbb{R}^4$, where θ_1 denotes the angle of the first link with respect to the vertical axis, θ_2 represents the relative angle between the two links, and $\dot{\theta}_1$, $\dot{\theta}_2$ are their respective angular velocities. The control input $u \in \mathbb{R}$ acts as a torque at the first joint only, making the system underactuated.

The equations of motion are described by:

$$\mathbf{M}(\theta_1, \theta_2) \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \mathbf{C}(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) + \mathbf{F} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \mathbf{G}(\theta_1, \theta_2) = \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{1.1}$$

where $\mathbf{M}(\theta_1, \theta_2) \in \mathbb{R}^{2\times 2}$ is the inertia matrix, $\mathbf{C}(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathbb{R}^2$ accounts for Coriolis and centrifugal forces, $\mathbf{G}(\theta_1, \theta_2) \in \mathbb{R}^2$ describes gravitational effects, and $\mathbf{F} \in \mathbb{R}^{2\times 2}$ represents viscous friction through a diagonal matrix. The dynamics parameters used in this analysis are presented in Table 1.1.

Table 1.1: System Dynamics Parameters

Parameter	Value
M_1	2.0 [kg]
M_2	2.0 [kg]
L_1	1.5 [m]
L_2	1.5 [m]
R_1	0.75 [m]
R_2	0.75 [m]
I_1	1.5 [kg·m^2]
I_2	1.5 [kg·m^2]
g	9.81 [m/s^2]
F_1	0.1 [N·m·s/rad]
F_2	0.1 [N·m·s/rad]

1.2 Discretization of Dynamics

1.2.1 Discretization Using Euler Method

The continuous-time dynamics described in Equation 2.1 are discretized using the Euler method for implementation in a digital control system. The time derivative is approximated as:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{\Delta t}$$

where Δt is the sampling time. Rearranging this yields the discrete-time update:

$$x_{k+1} = x_k + \Delta t \cdot dx$$

where dx is computed as:

$$dx = Ax_k + Bu_k + c$$

- A is the system matrix derived from the inertia, damping, and coupling effects, with terms dependent only on $\mathbf{x}$,
- B is the input matrix, with terms dependent only on $\mathbf{u}$,
- c encapsulates non-linear effects such as Coriolis, centrifugal, and gravitational forces.

The explicit equation of dx is the following:

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} -\mathbf{M}^{-1}\mathbf{F} & 0_{2\times 2} \\ I_{2\times 2} & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} \mathbf{M}^{-1} & 0_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\mathbf{M}^{-1}(\mathbf{C}+\mathbf{G}) \\ 0_{2\times 1} \end{bmatrix} \tag{1.2}$$

Since the system is underactuated, τ_2 is always zero.

Chapter 2

Trajectory generation (I)

2.1 Equilibrium Points

To determine the equilibrium points of the flexible robotic arm, the system is analyzed under the condition where all velocities and accelerations are zero:

$$\dot{\theta}_1 = \dot{\theta}_2 = \ddot{\theta}_1 = \ddot{\theta}_2 = 0.$$

Substituting these conditions into the system dynamics simplifies the equations to:

$$\mathbf{G}(\theta_1, \theta_2) = \begin{bmatrix} \tau \\ 0 \end{bmatrix}$$

This equation implies that the equilibrium points are the configurations (θ_1, θ_2) where the torques due to gravity are balanced by the applied torque τ.

To solve the equilibrium points numerically, Newton's method is employed to find the roots of the non-linear equations:

$$\mathbf{G}(\theta_1, \theta_2) - \begin{bmatrix} \tau \\ 0 \end{bmatrix} = 0$$

This approach iteratively refines an initial estimate of the equilibrium state to converge on a solution. The process begins with an initial guess:

$$z_0 = \begin{bmatrix} \theta_{1,0} \\ \theta_{2,0} \\ \tau_0 \end{bmatrix},$$

and updates the estimate using the formula:

$$z_{k+1} = z_k - (\nabla r(z_k))^{-1} r(z_k),$$

The algorithm should steer the decision vector towards the desired equilibrium point z_{eq}, which satisfies:

$$z_{\text{eq}} = \begin{bmatrix} \theta_{1,\text{eq}} \\ \theta_{2,\text{eq}} \\ \tau_{\text{eq}} \end{bmatrix}.$$

2.2 Reference Curve

In this task the reference curve is a step function. During the initial phase of the simulation, the system remains at the first equilibrium, after which it instantaneously transitions to the second equilibrium.

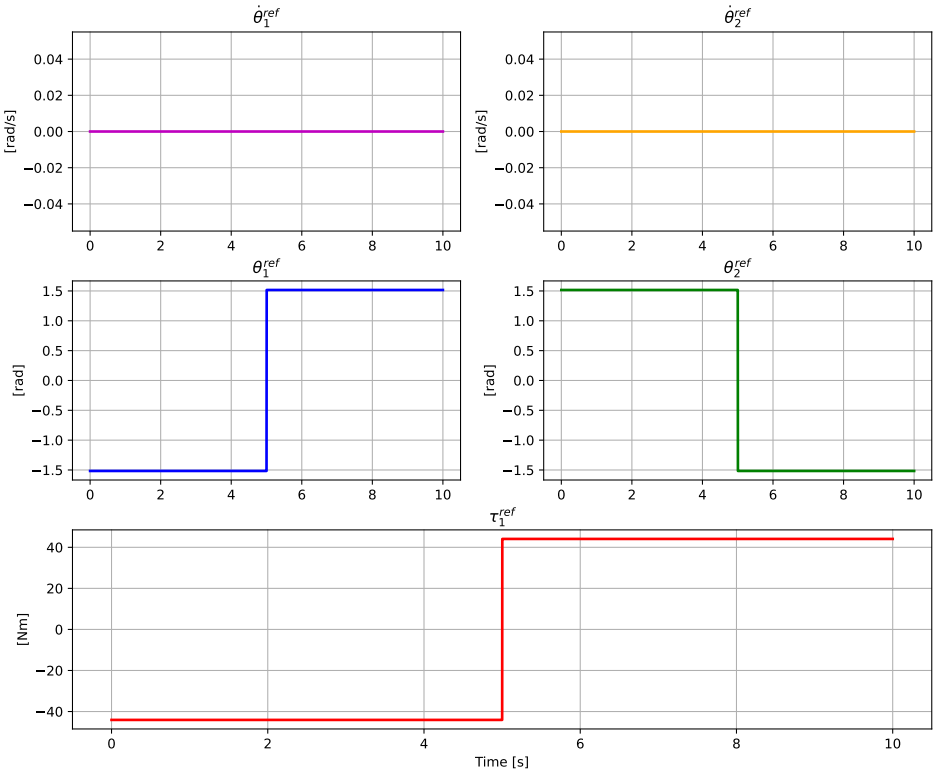


Figure 2.1: Step reference curve.

2.3 Cost Function

The cost function is quadratic and can be written as follows:

$$J(x,u) = \frac{1}{2}\sum_{t=0}^{T-1}\left[(x_t - x_{\text{ref},t})^T Q_t (x_t - x_{\text{ref},t}) + (u_t - u_{\text{ref},t})^T R_t (u_t - u_{\text{ref},t})\right] + \frac{1}{2}(x_T - x_{\text{ref},T})^T Q_T (x_T - x_{\text{ref},T}) \tag{2.1}$$

The cost matrices are time-varying to better capture the desired behavior of the system. This approach ensures precision near the equilibrium points and allows flexibility during transitions. In particular, higher weights are assigned to emphasize accuracy at equilibrium, while lower weights are used during transitions to allow smoother adjustments. This variation is implemented in a smooth manner to ensure proper and stable control. For the purposes of Task 1, the cost matrices behave as follows:

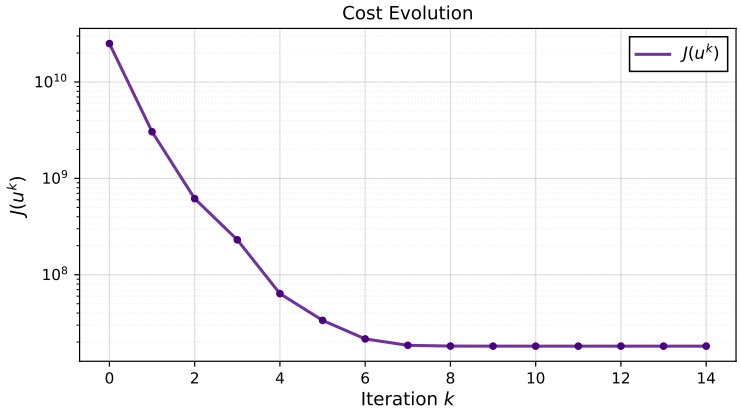


Figure 2.2: Evolution of cost matrices.

2.4 Optimal Transition: Newton's-like Algorithm

To achieve an optimal transition between equilibrium points, the Newton's-like algorithm for optimal control is applied in a closed-loop configuration. The goal is to minimize a cost function that penalizes deviations from the desired trajectory and excessive control effort. The optimization problem states as follows:

$$min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u})$$
$$s.t. \quad x_{t+1} = f(x_t, u_t), \quad t = 0, 1, ..., T-1$$

The algorithm follows a two-step iterative procedure that involves solving the co-state equations, computing the necessary feedback gains, and updating the state-input trajectory until convergence is achieved.

In the first step, the algorithm computes the descent direction, which is essential to reduce the cost function. This involves evaluating the following terms:

$$\nabla_1 f_t(x_t^t, u_t^t) \quad \nabla_2 f_t(x_t^t, u_t^t) \quad \nabla_1 \ell_t(x_t^t, u_t^t) \quad \nabla_2 \ell_t(x_t^t, u_t^t) \quad \nabla \ell_T(x_T^T)$$

Once these gradients are evaluated, the co-state equations are solved backward in time, starting from $\nabla \ell_T(x_T^T)$. The goal is to calculate the matrices Q_t^k, R_t^k, S_t^k, and Q_T^k, which are essential to obtain the feedback control law. The feedback gains K_t^k and σ_t^k are then computed for all time steps from $t = 0$ to $t = T - 1$ by solving the optimal control problem in the backward direction.

The optimal solution of the problem reads

$$\begin{aligned} \Delta u_t^* &= K_t^* \Delta x_t^* + \sigma_t^* \qquad\qquad t = 0, \dots, T-1, \\ \Delta x_{t+1}^* &= A_t \Delta x_t^* + B_t \Delta u_t^*, \end{aligned}$$

where

$$\begin{aligned} K_t^* &= -(R_t + B_t^\top P_{t+1} B_t)^{-1}(S_t + B_t^\top P_{t+1} A_t), \\ \sigma_t^* &= -(R_t + B_t^\top P_{t+1} B_t)^{-1}(r_t + B_t^\top p_{t+1} + B_t^\top P_{t+1} c_t), \\ p_t &= q_t + A_t^\top p_{t+1} + A_t^\top P_{t+1} c_t - K_t^{*\top} \left(R_t + B_t^\top P_{t+1} B_t\right) \sigma_t^*, \\ P_t &= Q_t + A_t^\top P_{t+1} A_t - K_t^{*\top} \left(R_t + B_t^\top P_{t+1} B_t\right) K_t^*, \end{aligned}$$

with $p_T = q_T$ and $P_T = Q_T$.

The second step of the algorithm involves using the computed feedback gains to update the state and control trajectories. This is done by applying the feedback control law:

$$u_t^{k+1} = u_t^k + K_t^k \left(x_t^{k+1} - x_t^k\right) + \gamma^k \sigma_t^k$$

Additionally, forward integration of the system dynamics is performed for all $t = 0, \dots, T-1$ to obtain the new trajectory. This is done by solving the system of equations:

$$x_{t+1}^{k+1} = f_t(x_t^{k+1}, u_t^{k+1})$$

To ensure convergence and prevent overly large updates that may destabilize the trajectory, a step-size selection rule is used, such as the Armijo rule. This rule adjusts the step size based on the reduction of the cost function at each iteration, ensuring that each new trajectory results in a decrease in the overall cost.

2.5 Plots of Generated Optimal Trajectory (I)

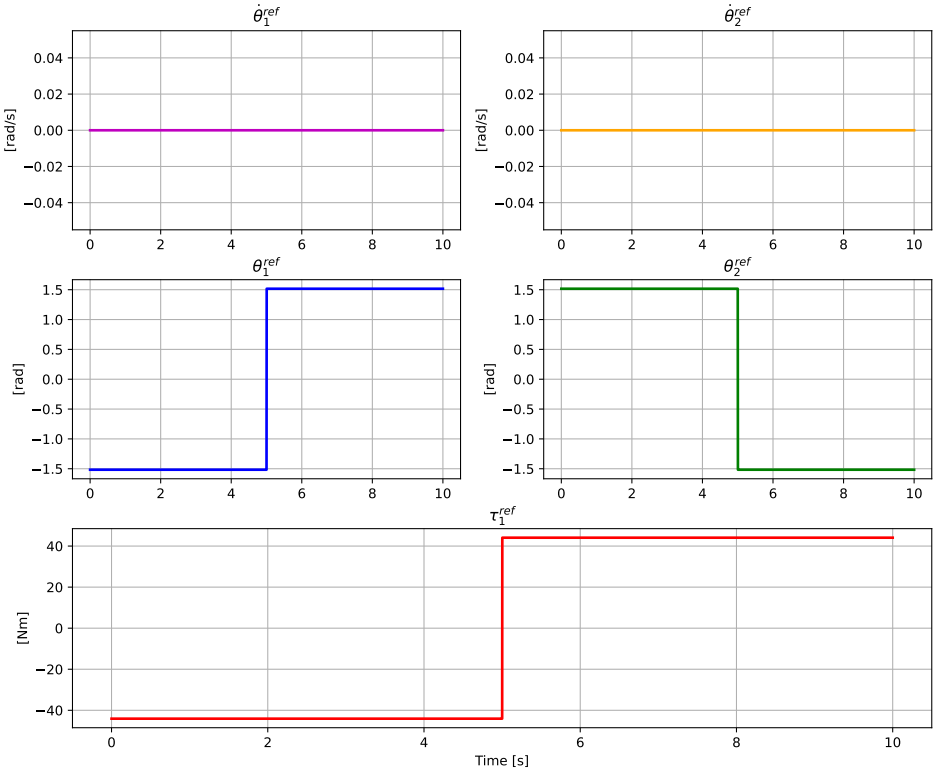


Figure 2.3: Generated optimal trajectory given a step reference.

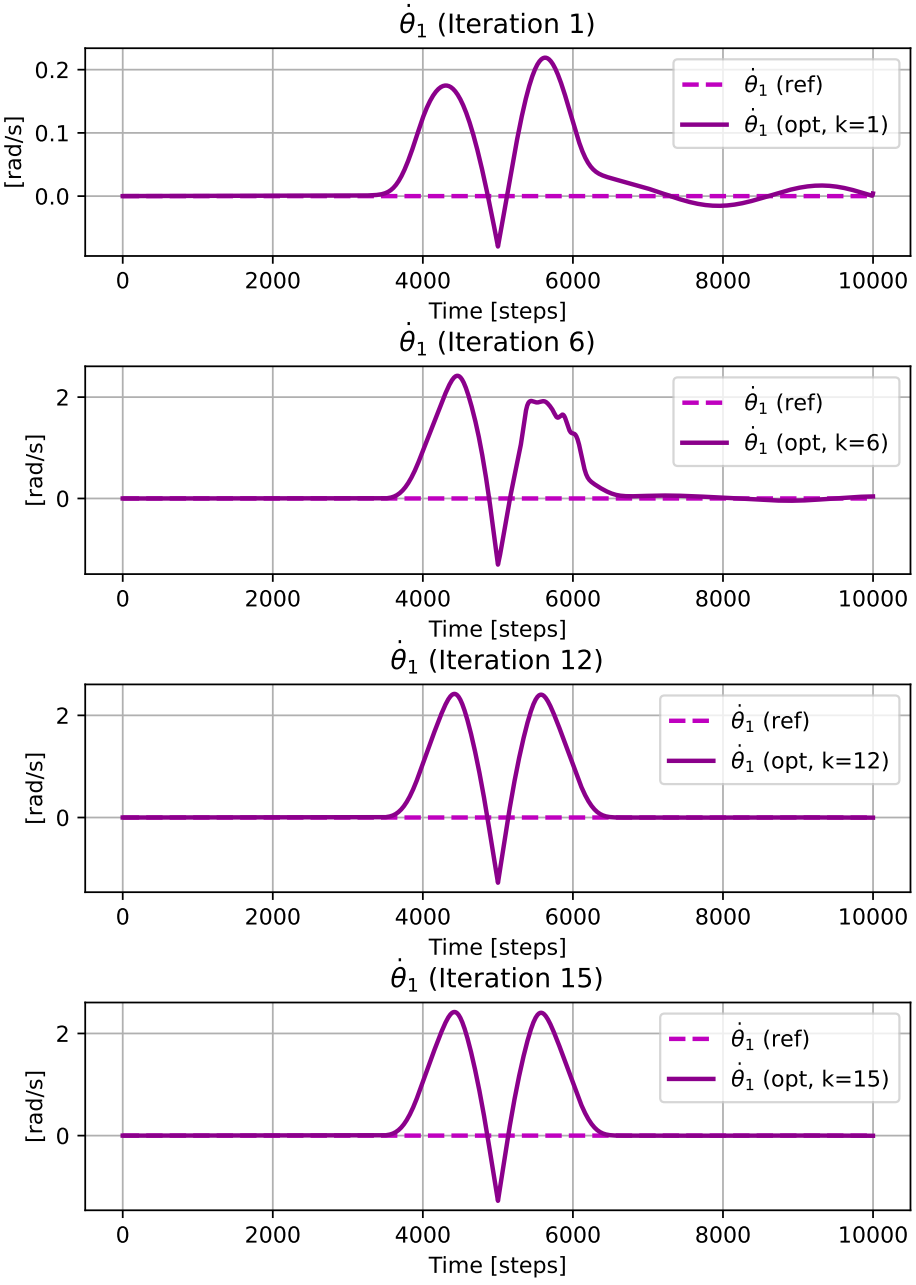


Figure 2.4: Evolution of $d\theta_1$.

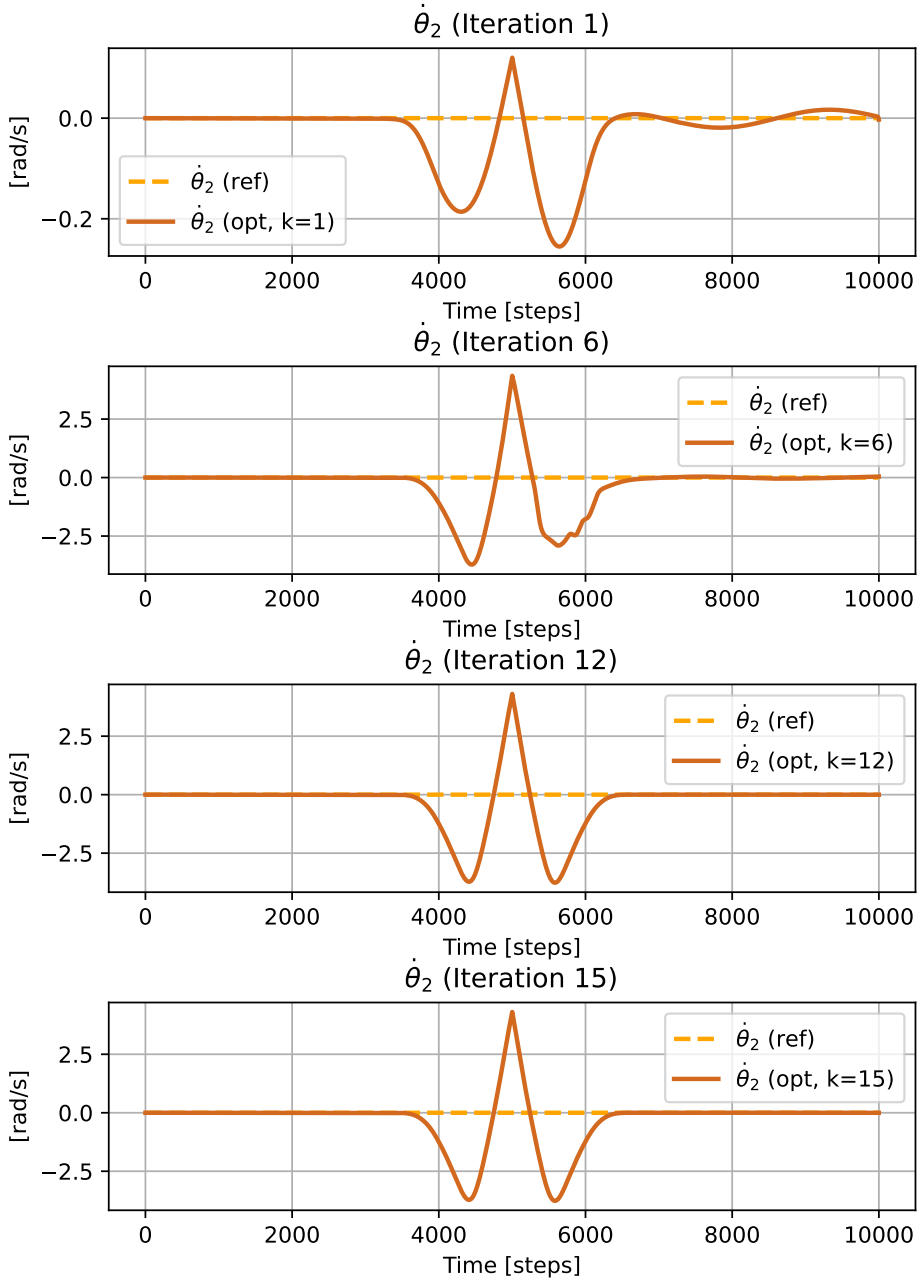


Figure 2.5: Evolution of $d\theta_2$.

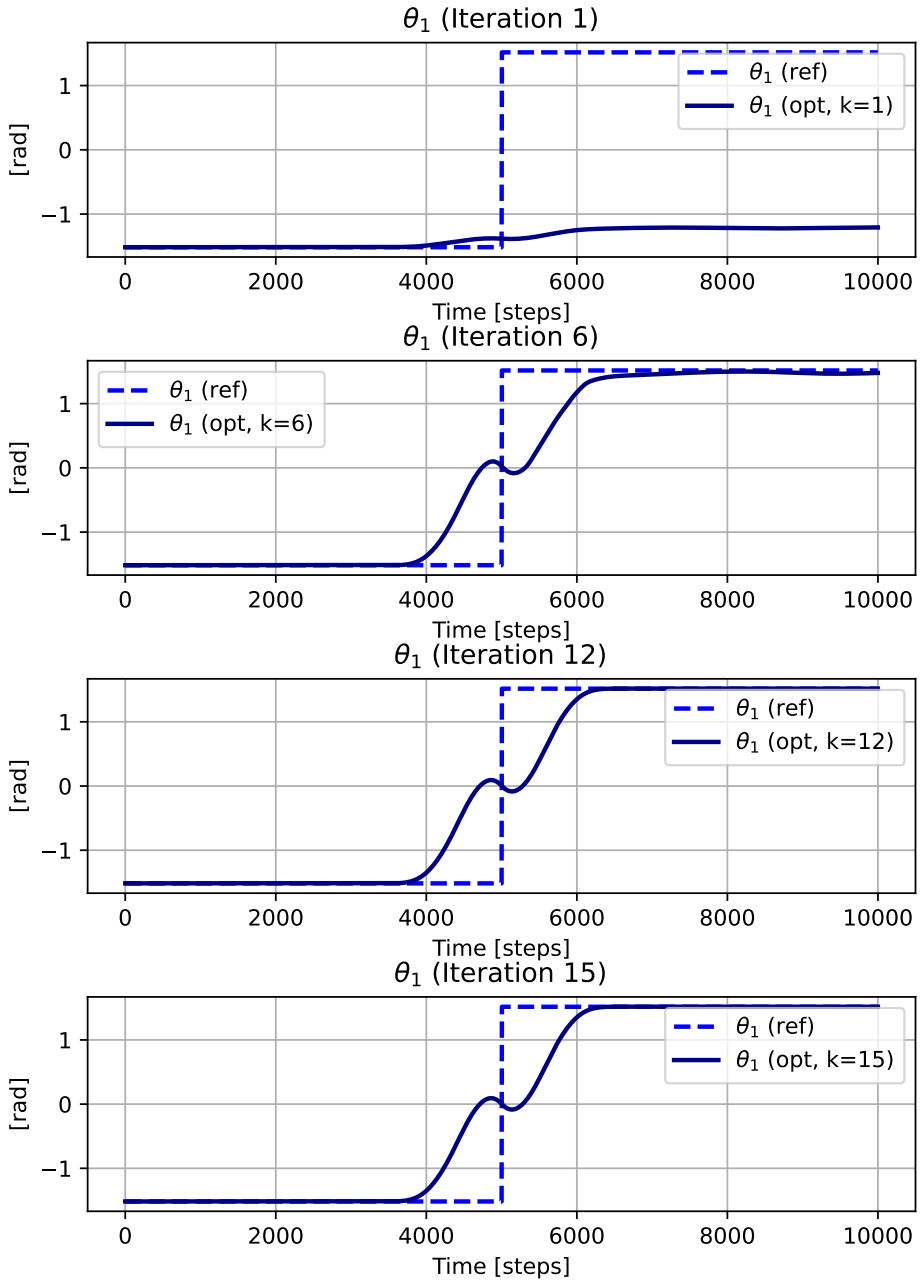


Figure 2.6: Evolution of θ_1.

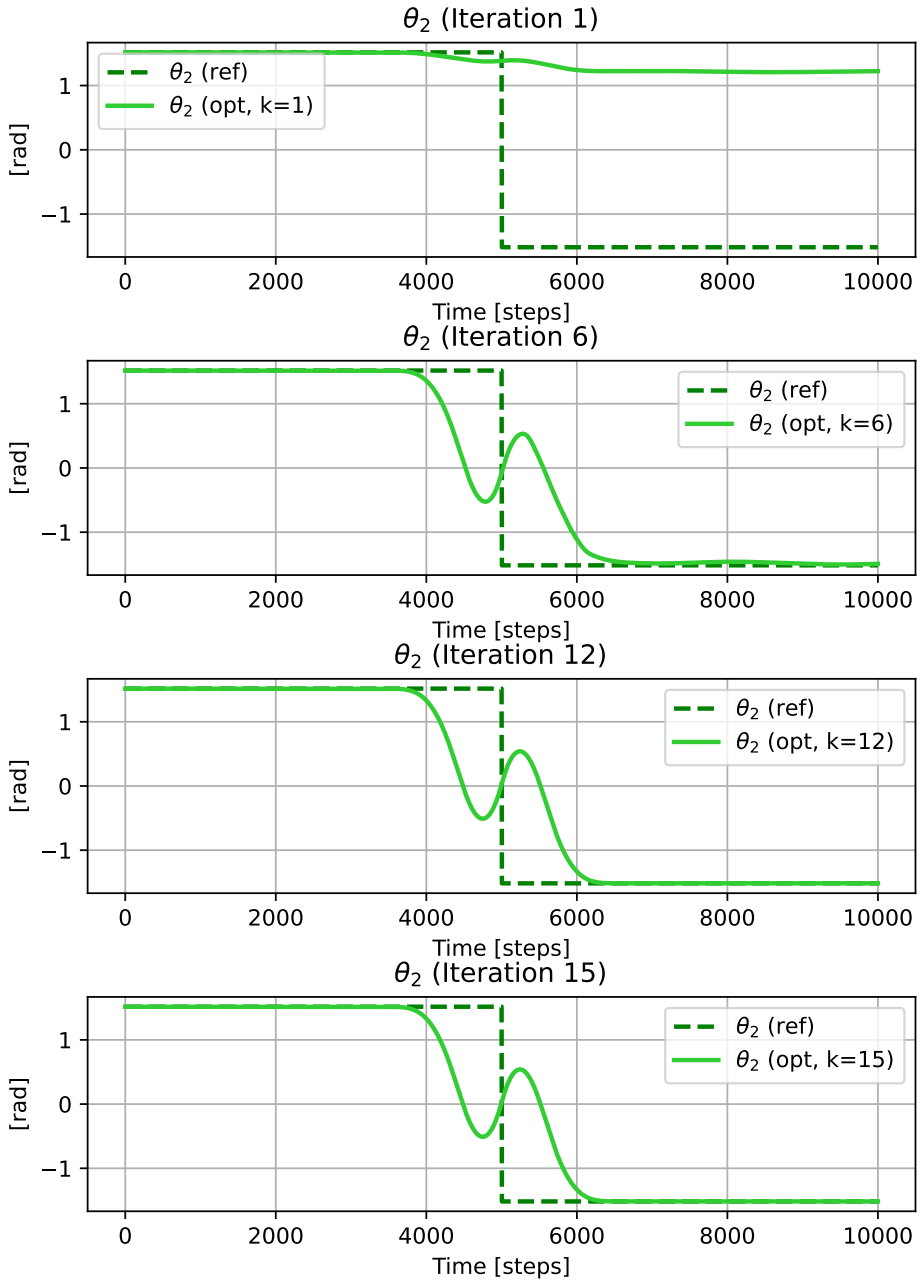


Figure 2.7: Evolution of θ_2.

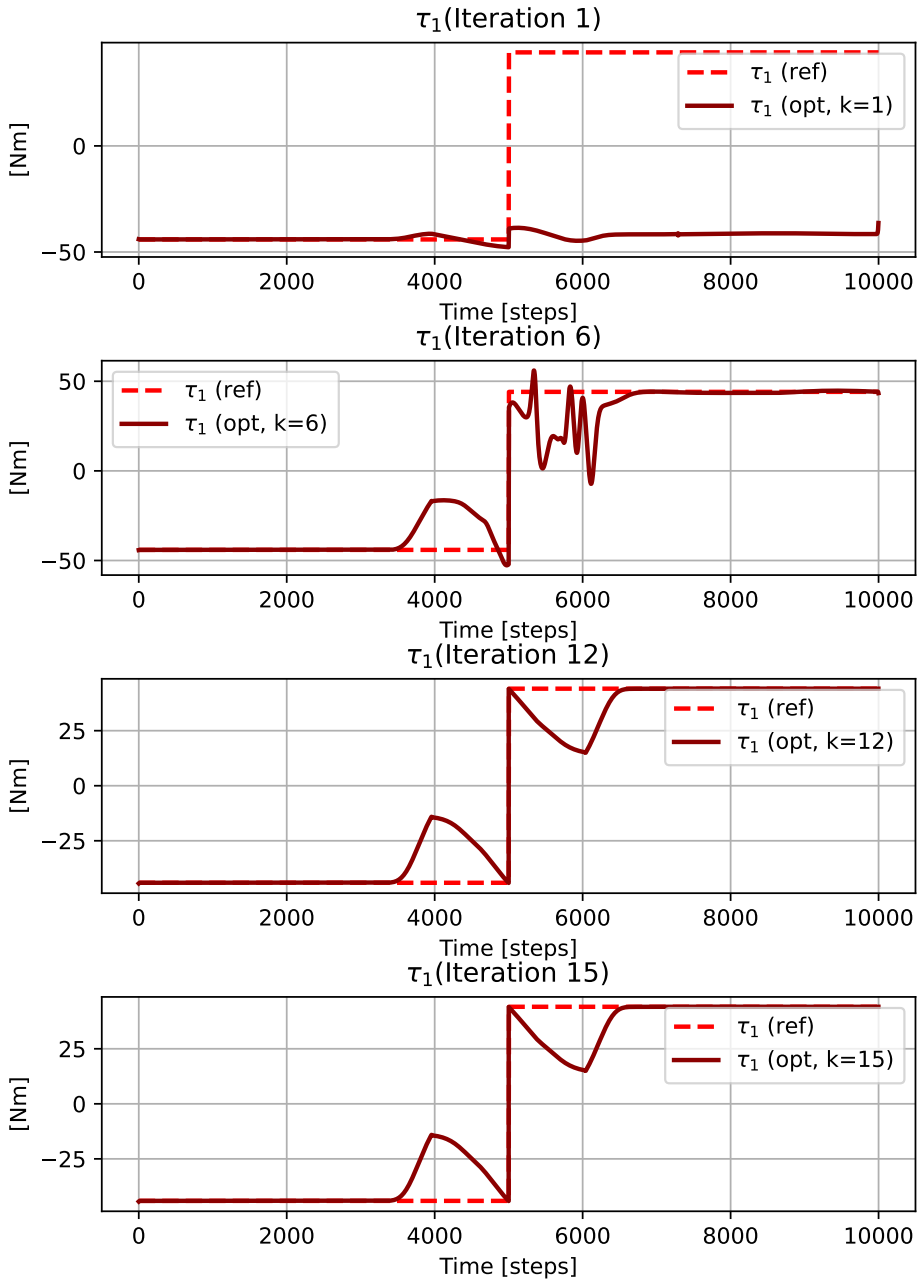


Figure 2.8: Evolution of τ_1.

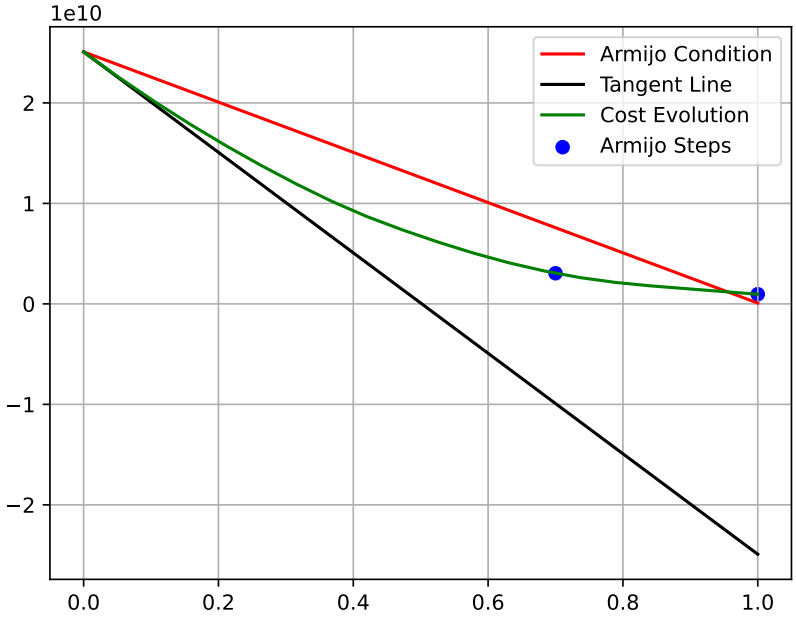


Figure 2.9: Armijo step-size selection: iteration 1.

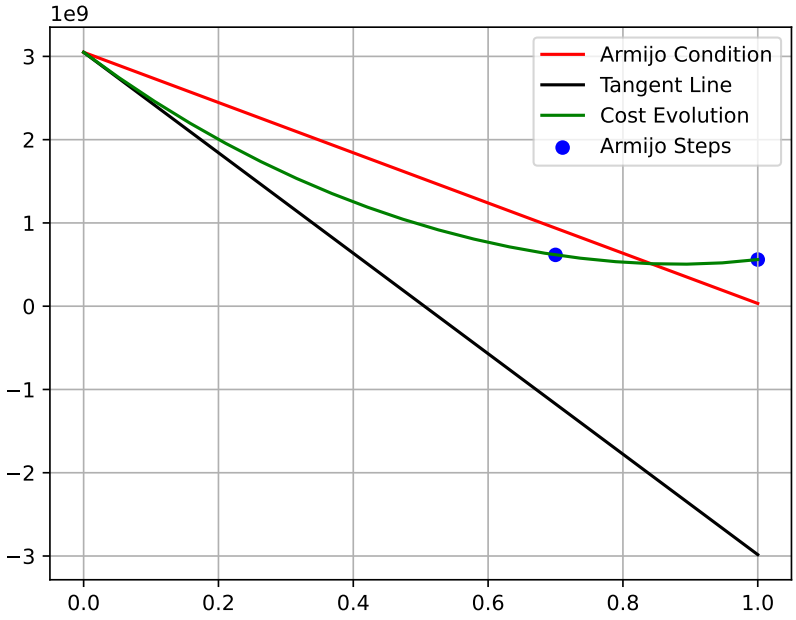


Figure 2.10: Armijo step-size selection: iteration 2.

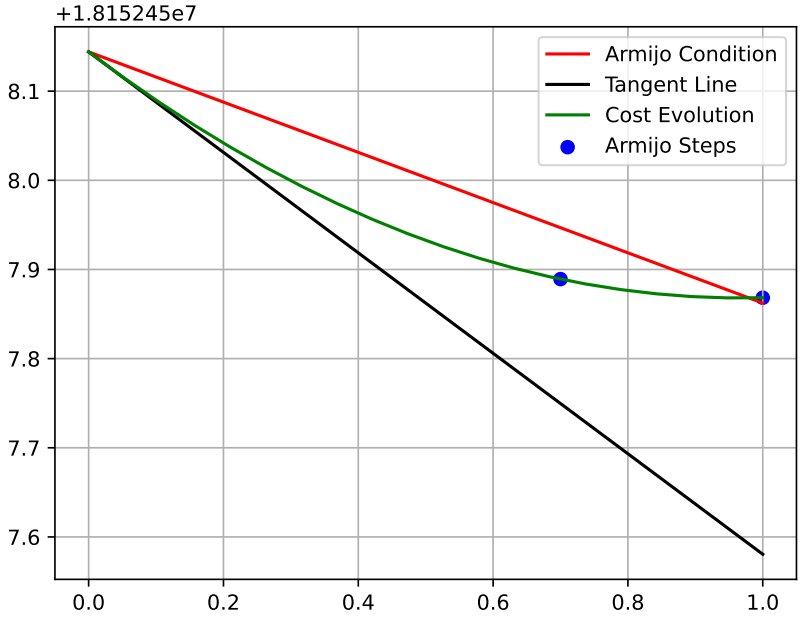


Figure 2.11: Armijo step-size selection: iteration 11.

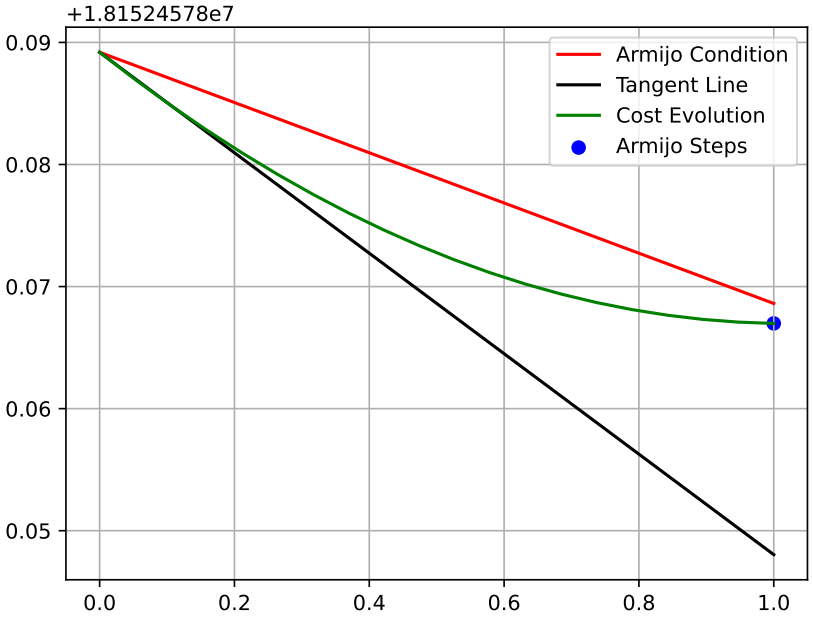


Figure 2.12: Armijo step-size selection: iteration 12.

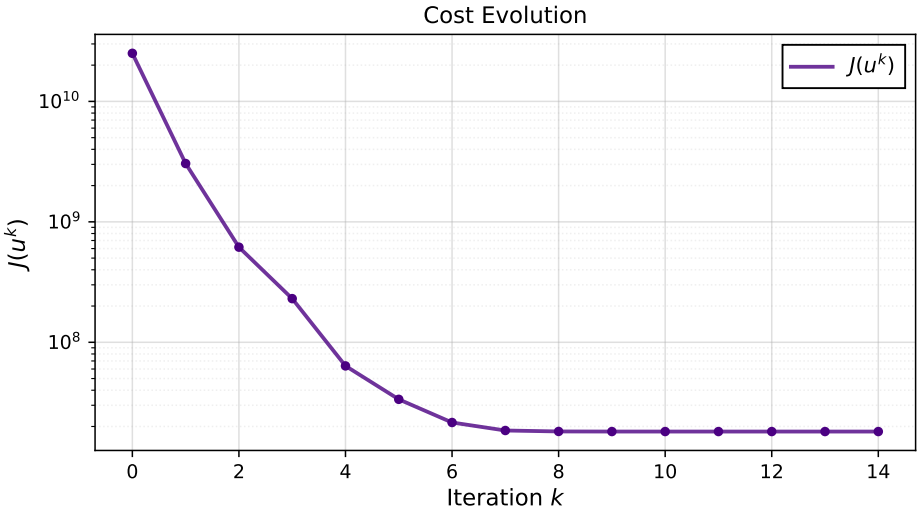


Figure 2.13: Cost evolution.

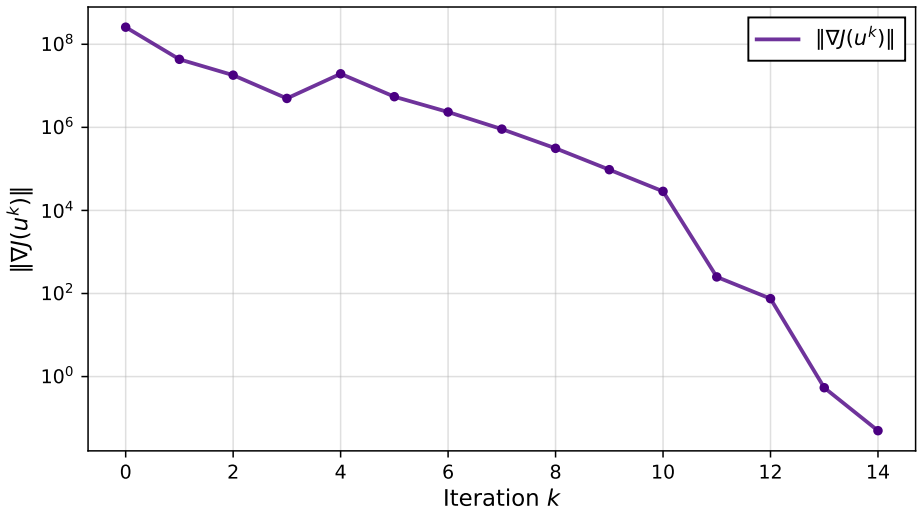


Figure 2.14: Cost gradient norm evolution.

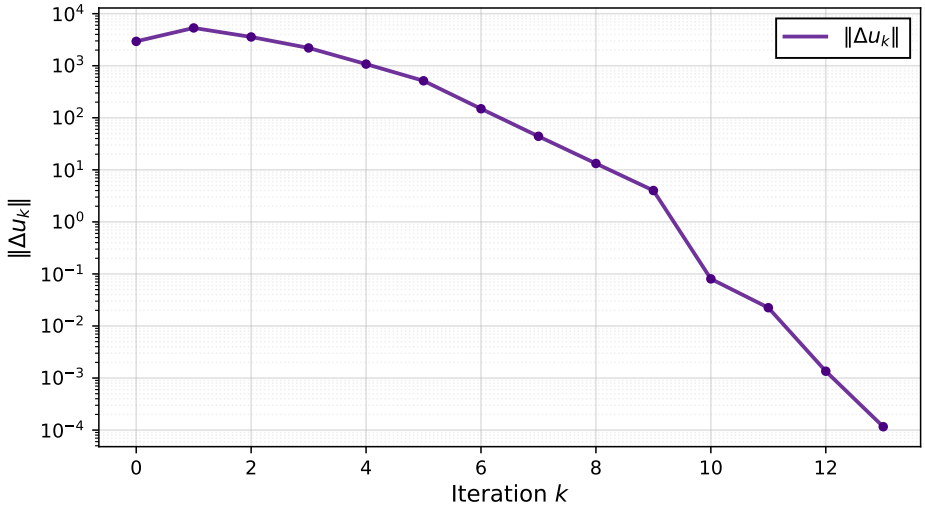


Figure 2.15: Evolution of $||\Delta u_k||$.

2.6 Constant Cost Matrices Scenario

The following cost matrices have been adopted:

$$Q_t = Q_T = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{2.2}$$

$$R_t = \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix} \tag{2.3}$$

Here, a summary of the results is presented.
The performances are obviously worse than the ones presented before.

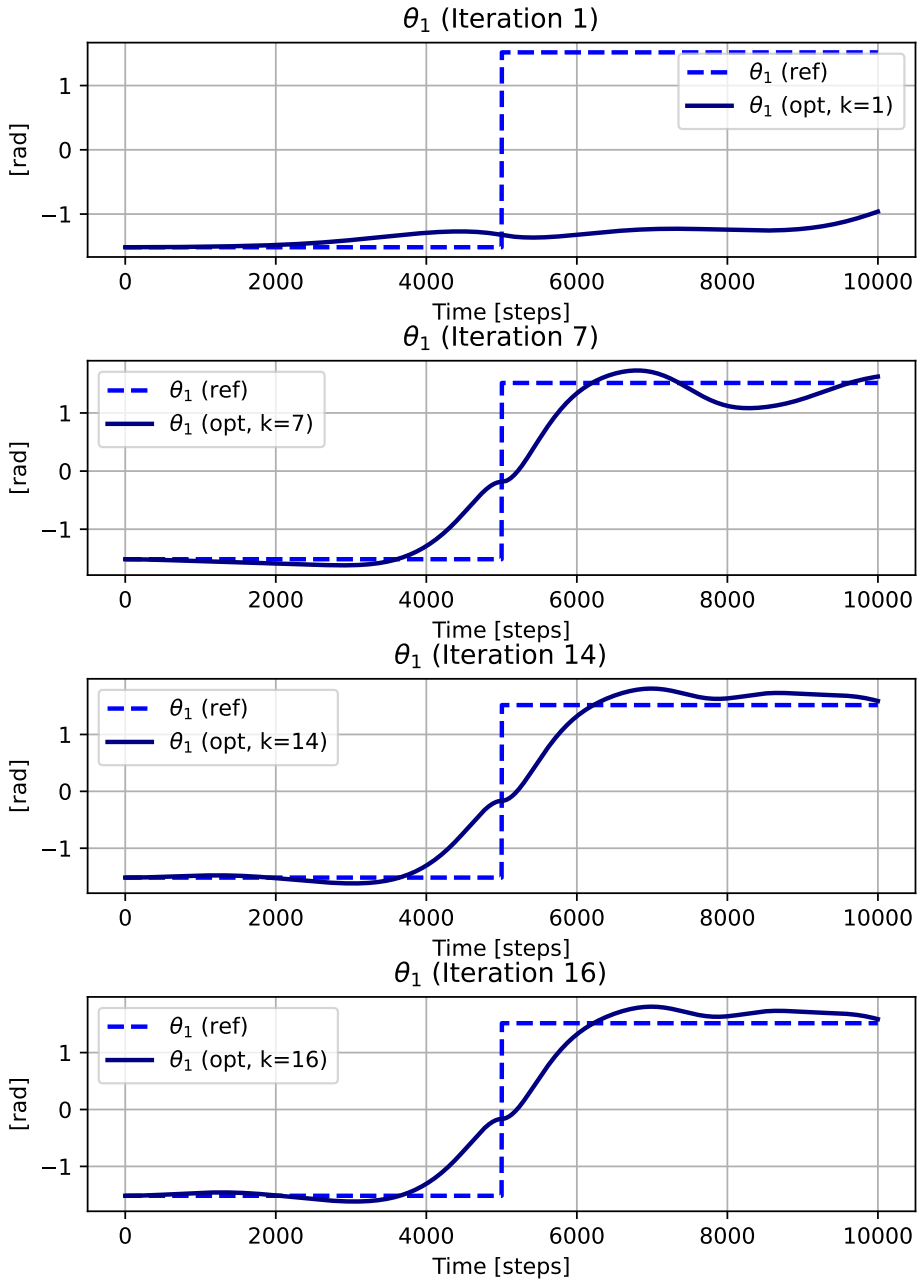


Figure 2.16: Evolution of θ_1 with constant Cost Matrices.

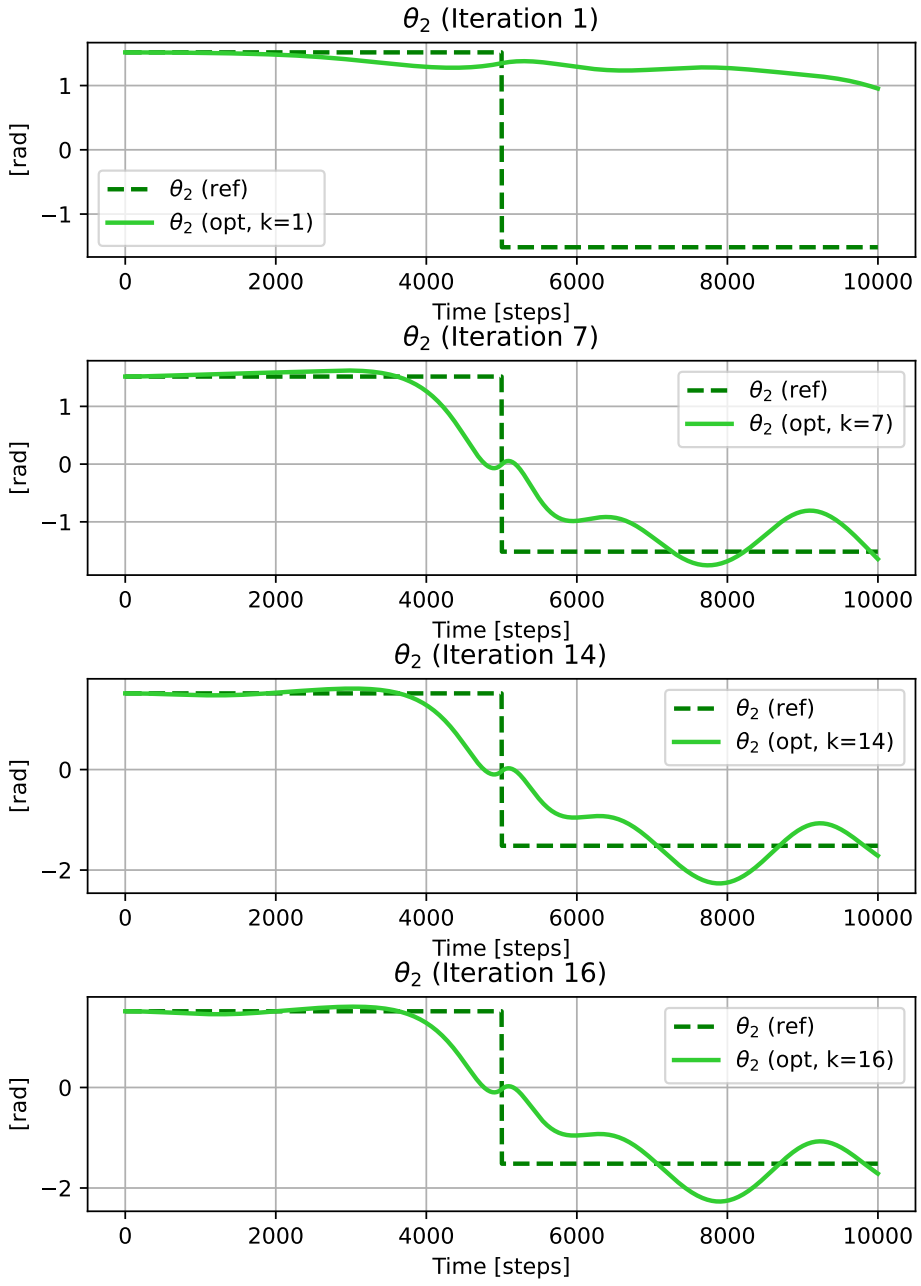


Figure 2.17: Evolution of θ_2 with constant Cost Matrices.

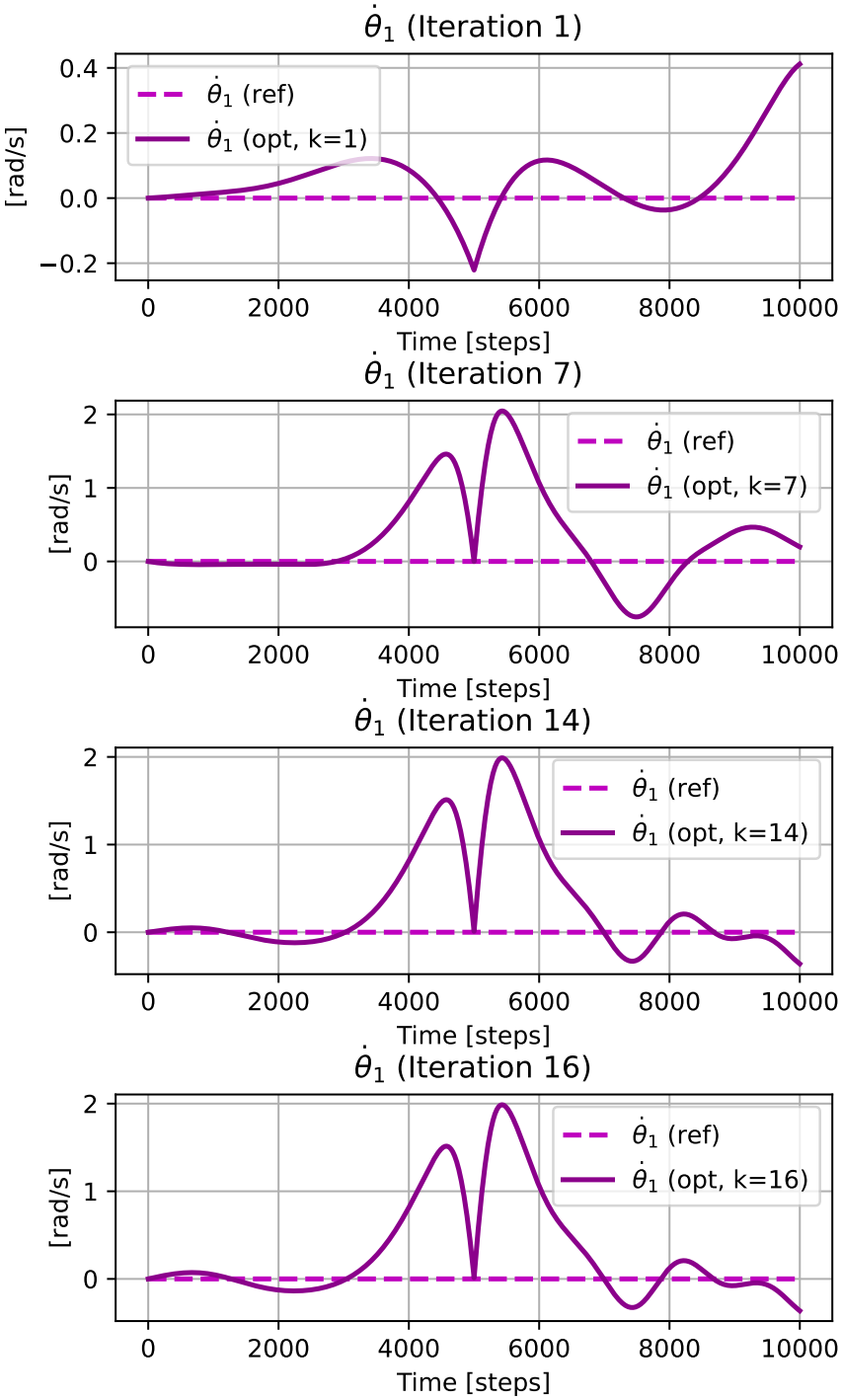


Figure 2.18: Evolution of $\dot{\theta}_1$ with constant Cost Matrices.

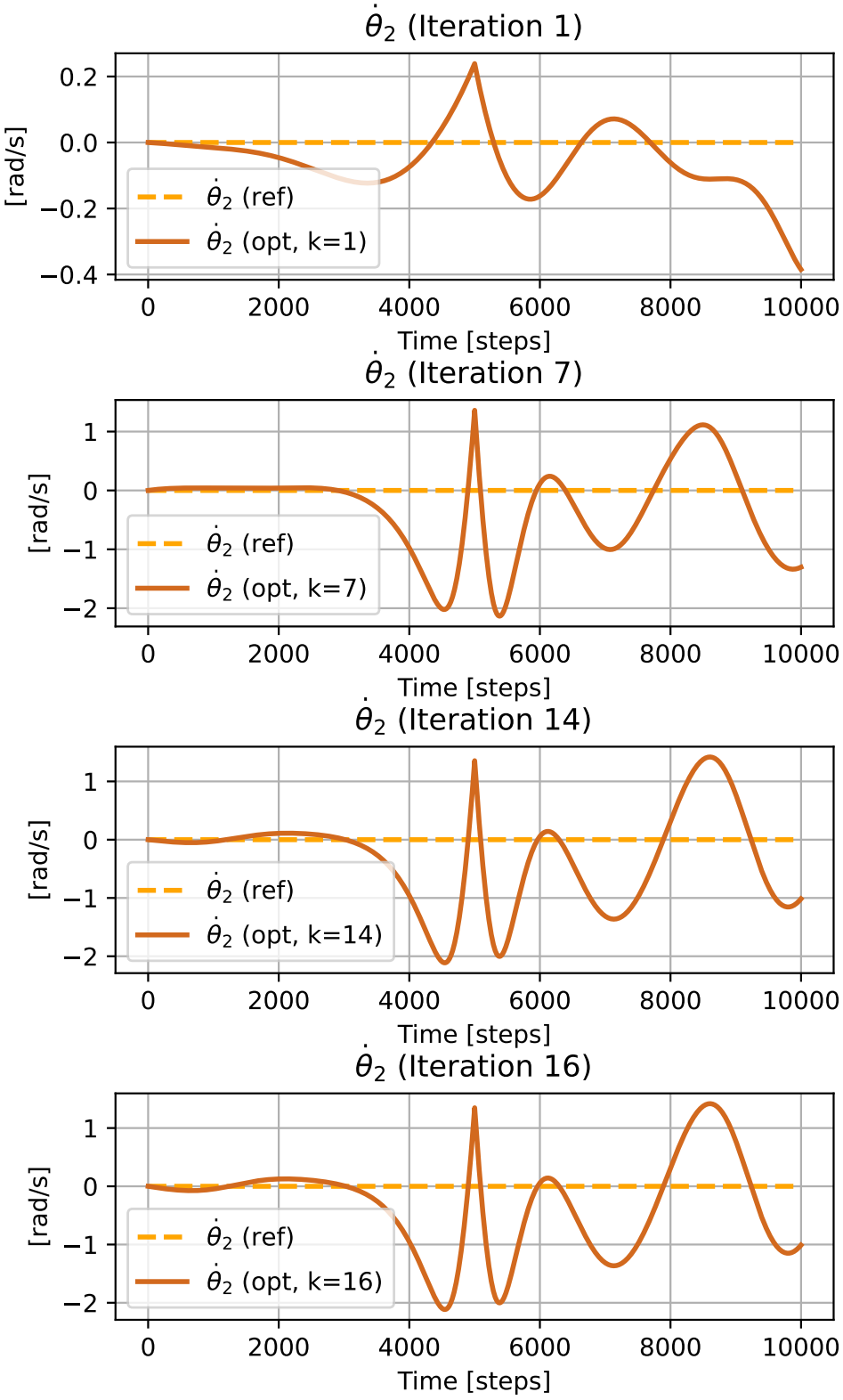


Figure 2.19: Evolution of $\dot{\theta}_2$ with constant Cost Matrices.

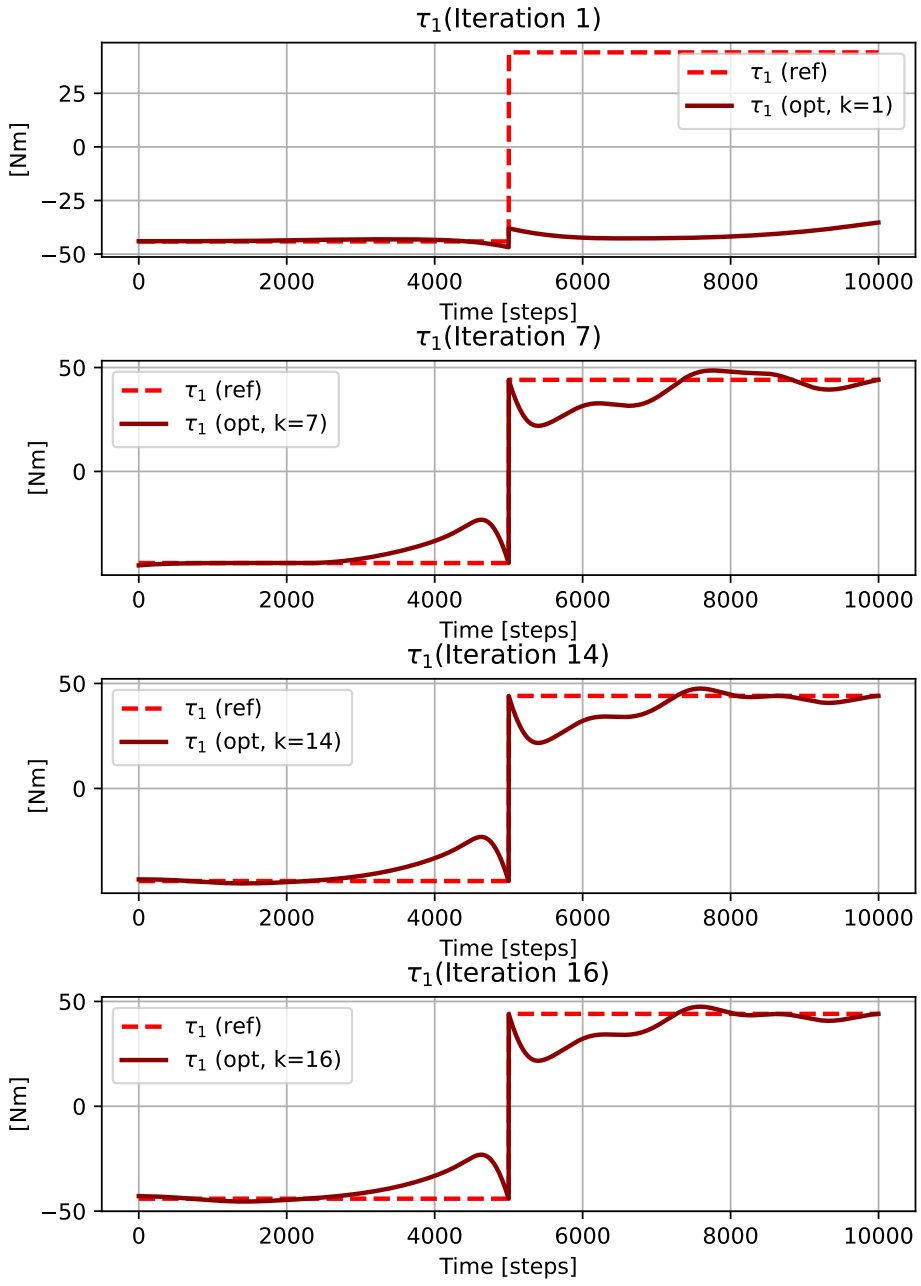


Figure 2.20: Evolution of τ with constant Cost Matrices.

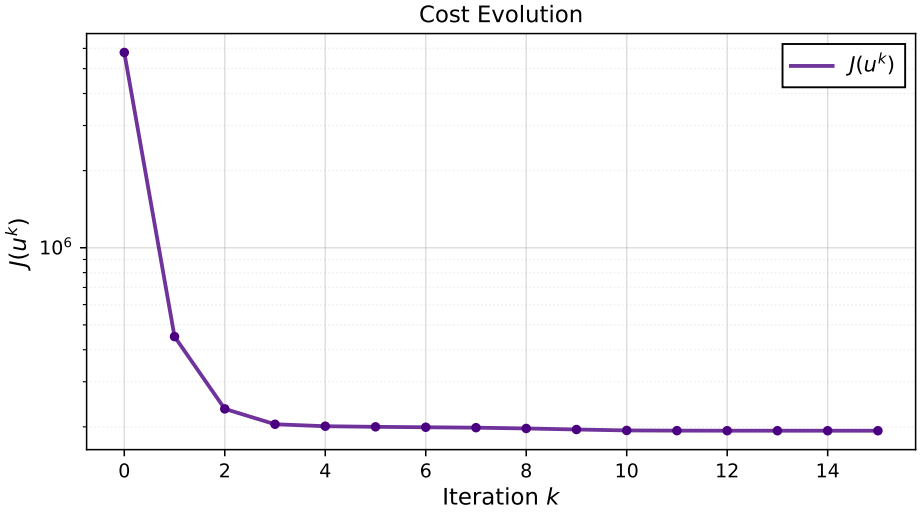


Figure 2.21: Evolution of cost function with constant Cost Matrices.

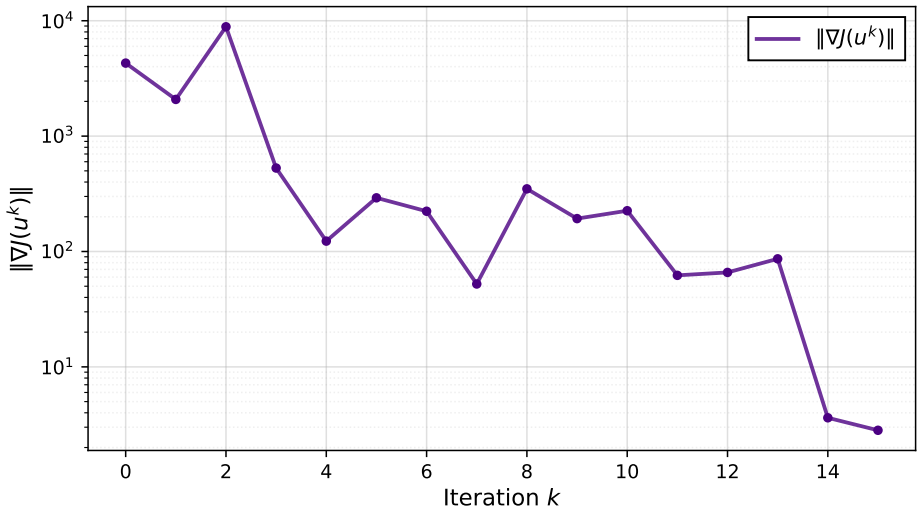


Figure 2.22: Evolution of $||\nabla J(u)||$ with constant Cost Matrices.

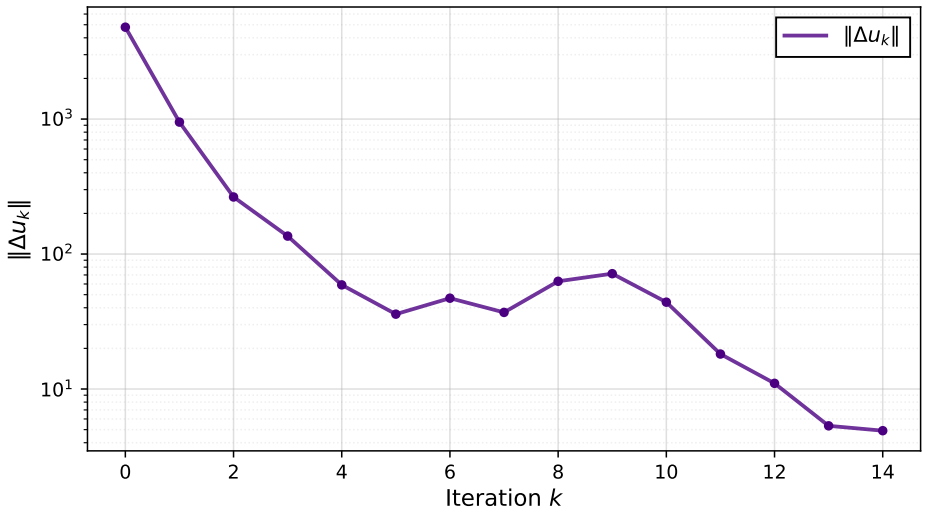


Figure 2.23: Evolution of $||\Delta u_k||$ with constant Cost Matrices.

Chapter 3

Trajectory generation (II)

3.1 Smooth Desired Trajectory

In this task, the reference curve is defined as a cubic spline that provides a smooth and continuous transition between a sequence of equilibrium points. The simulation starts with the system at the first equilibrium point. The reference trajectory guides the system through four smooth transitions between equilibrium points. At each intermediate equilibrium, the system briefly stabilizes before moving to the next. Once the system reaches the final equilibrium, it stabilizes completely and remains there for the rest of the simulation.

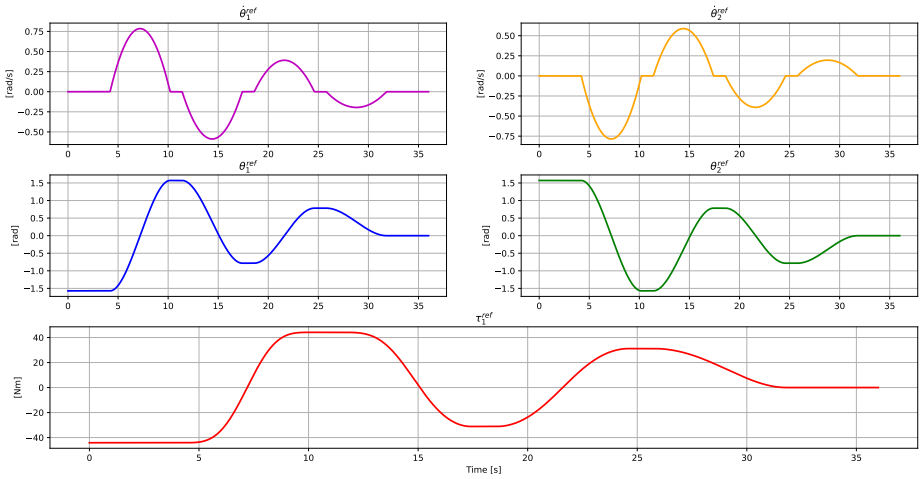


Figure 3.1: Smooth reference curve.

More precisely, this is a quasi-static trajectory; it represents a gradual and steady evolution of the system's states, during which the system remains in or near equilibrium at all times. This approach ensures that dynamic effects, such as transients or inertia, are negligible, as the system has sufficient

time to adjust to changes and maintain balance.

In the context of optimal control, quasi-static trajectories play an important role in the initialization process. They provide smooth and feasible transitions between equilibrium points, preventing abrupt changes that could lead to instability or infeasibility. By simplifying the dynamics and avoiding abrupt changes, they ensure stability, facilitate the computation of initial conditions, and improve the convergence of optimization algorithms.

3.2 Improved Optimal Transition

In Task 2, the same optimal control problem is addressed as in Task 1, with the distinction that the reference curve from which the system starts is a smooth transition between equilibrium points. The cost function is defined in Equation (2.1) and the cost matrices are time-varying, with the following behavior.

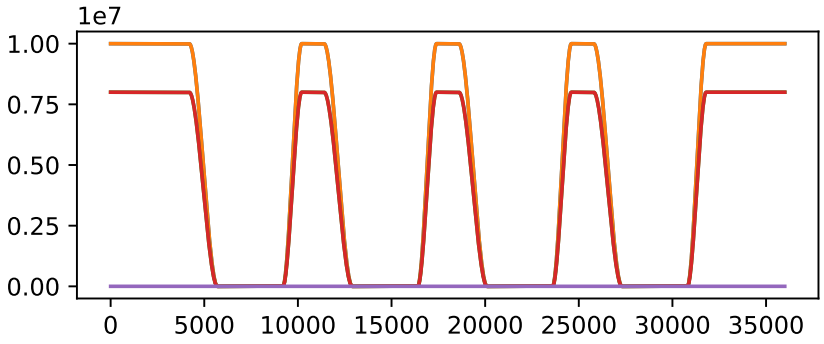


Figure 3.2: Evolution of cost matrices.

3.3 Plots for Trajectory Generation (II)

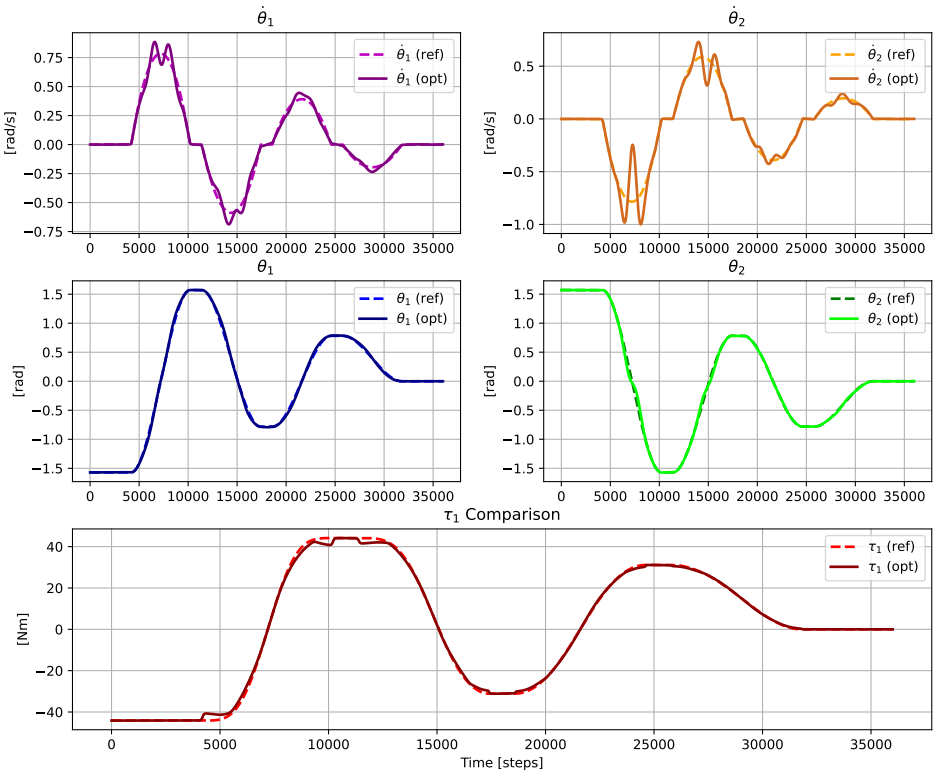


Figure 3.3: Generated optimal trajectory given a smooth reference.

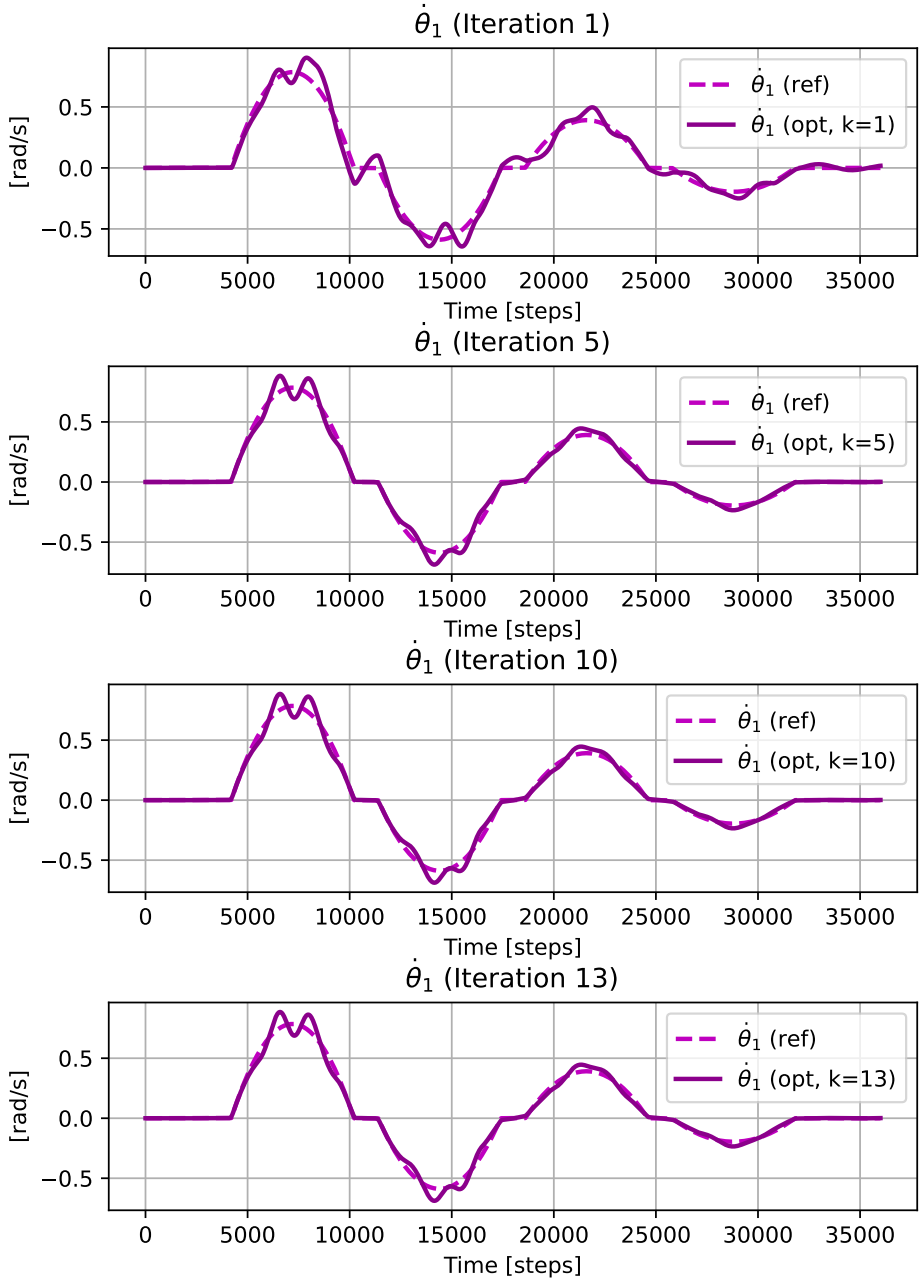


Figure 3.4: Evolution of $d\theta_1$.

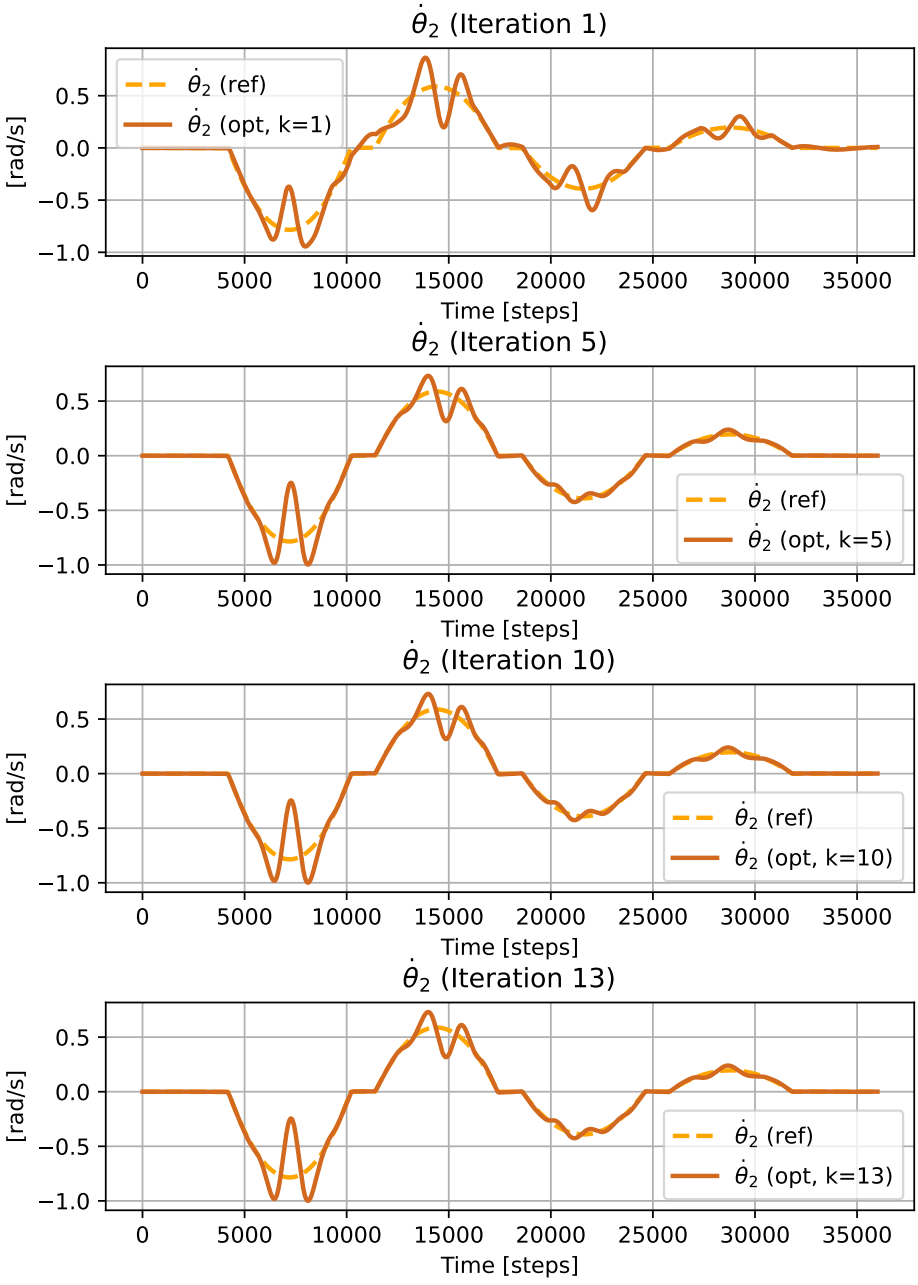


Figure 3.5: Evolution of $d\theta_2$.

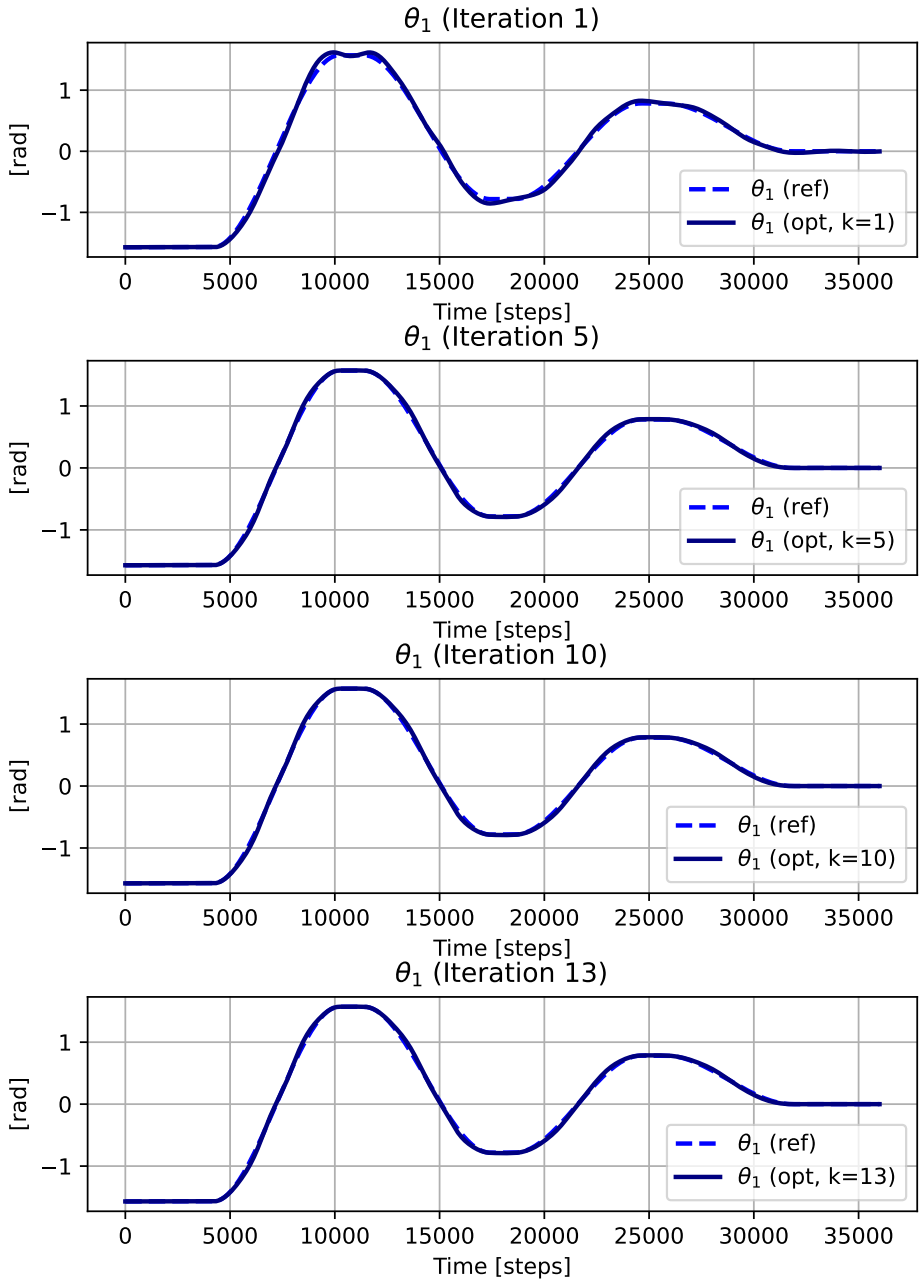


Figure 3.6: Evolution of θ_1.

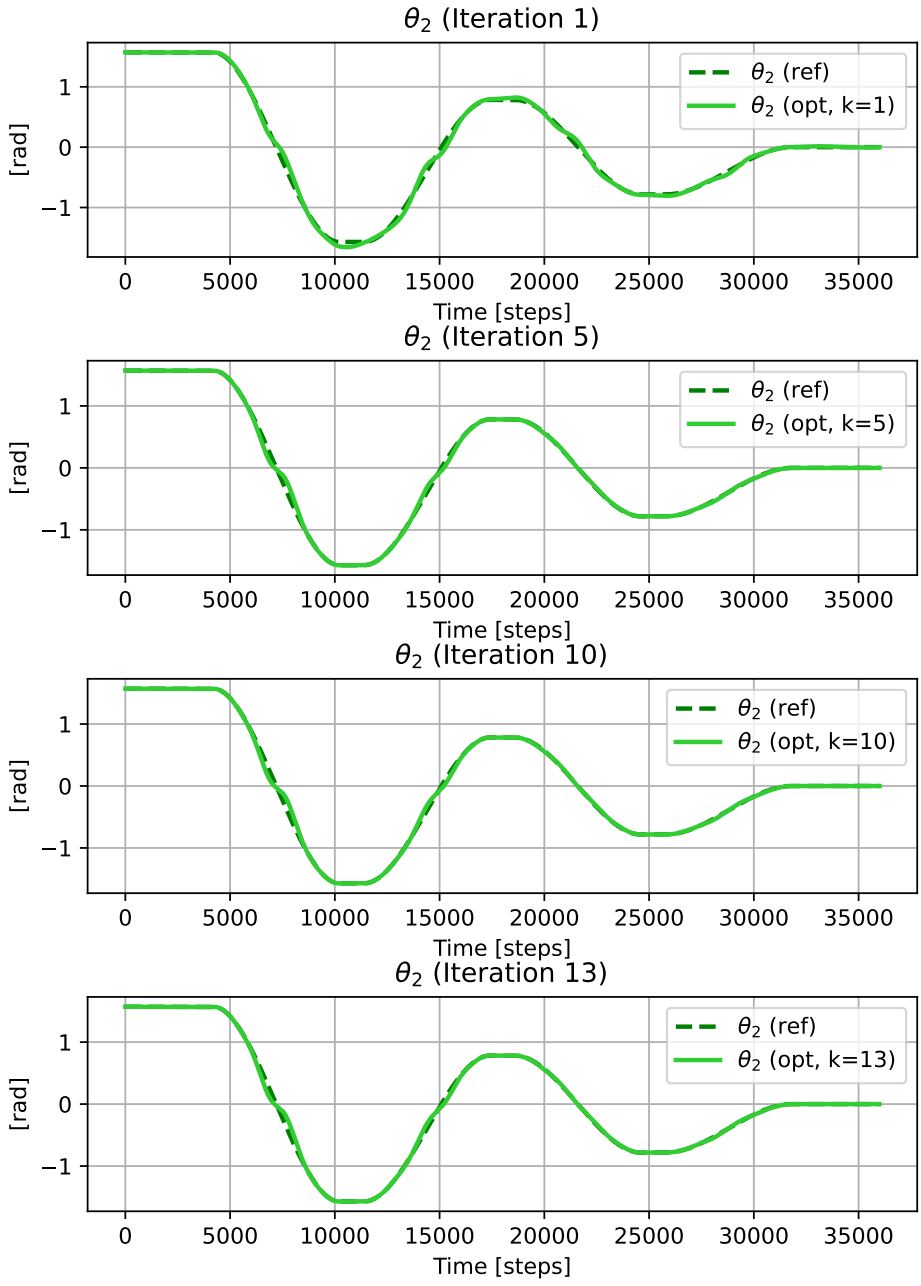


Figure 3.7: Evolution of θ_2.

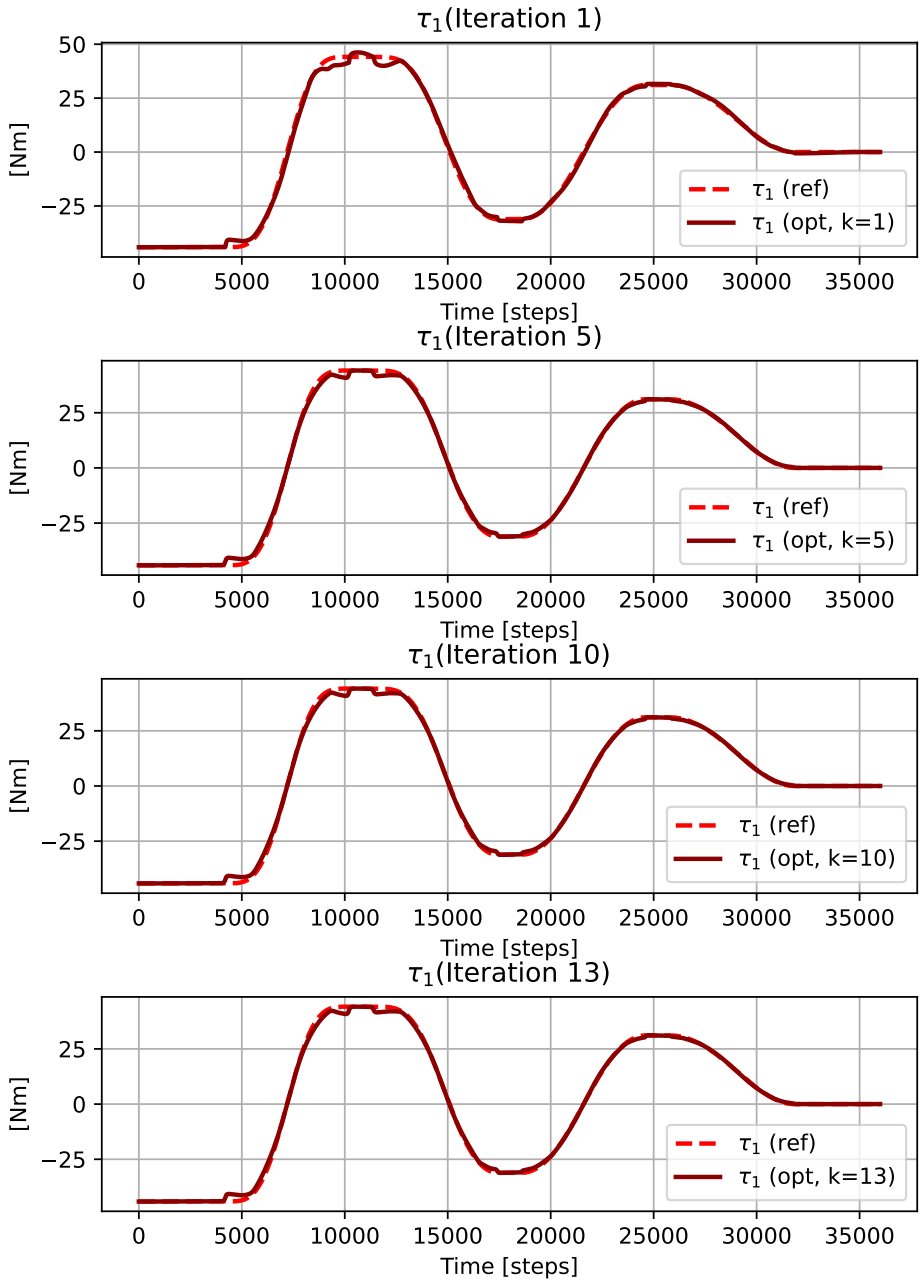


Figure 3.8: Evolution of τ_1.

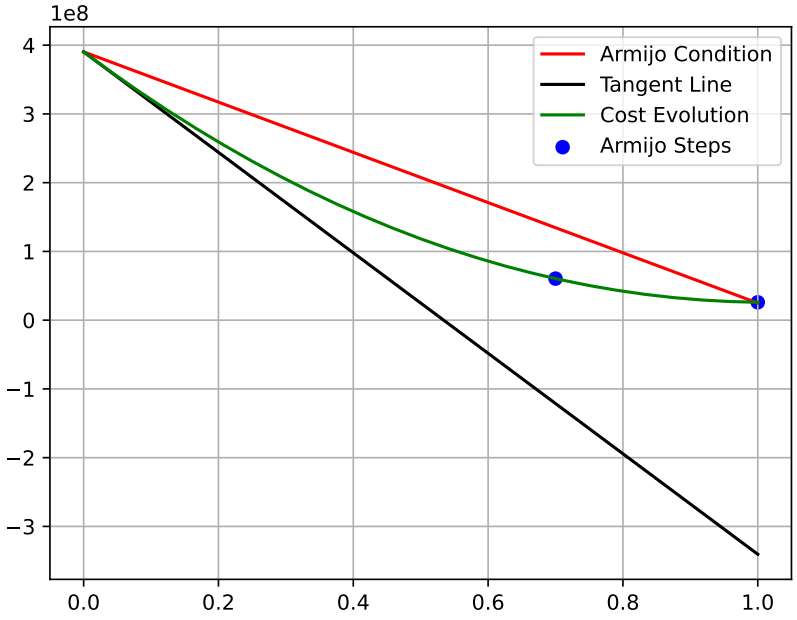


Figure 3.9: Armijo step-size selection: iteration 1

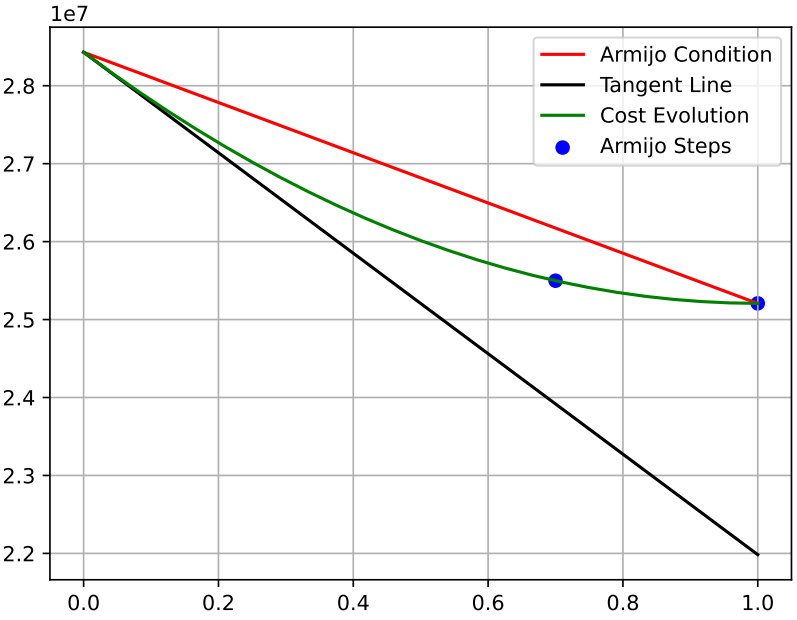


Figure 3.10: Armijo step-size selection: iteration 3

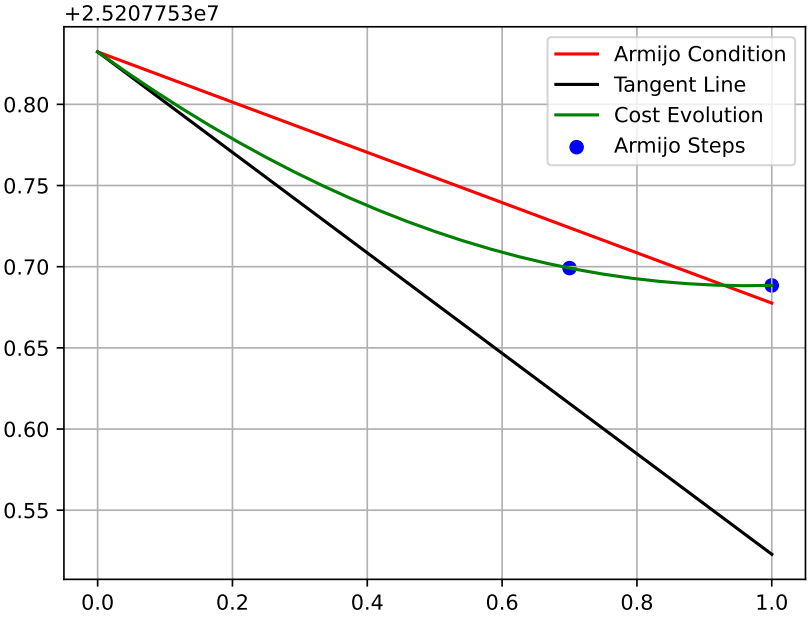


Figure 3.11: Armijo step-size selection: iteration 10

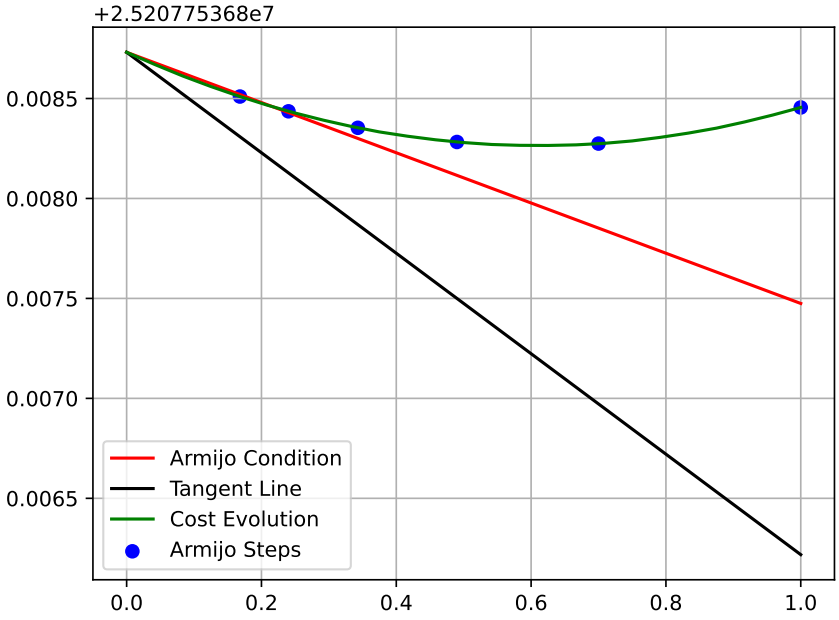


Figure 3.12: Armijo step-size selection: iteration: 12

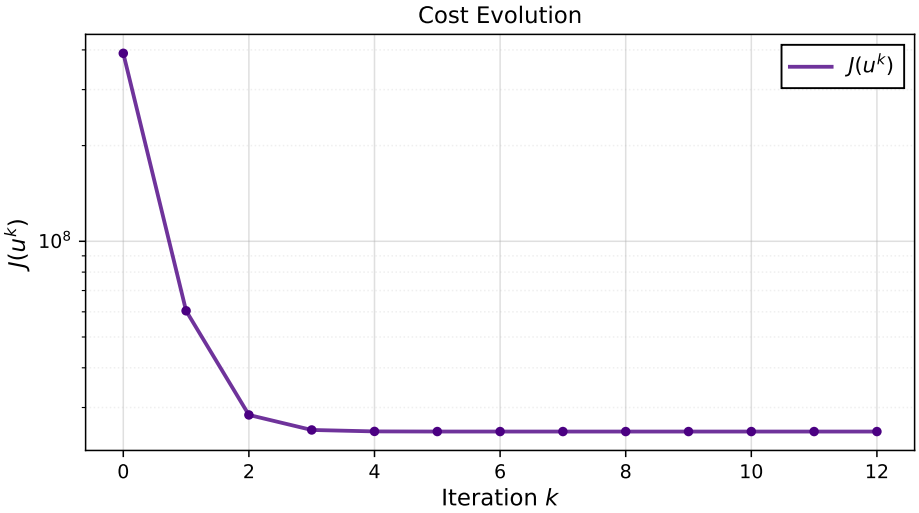


Figure 3.13: Cost evolution

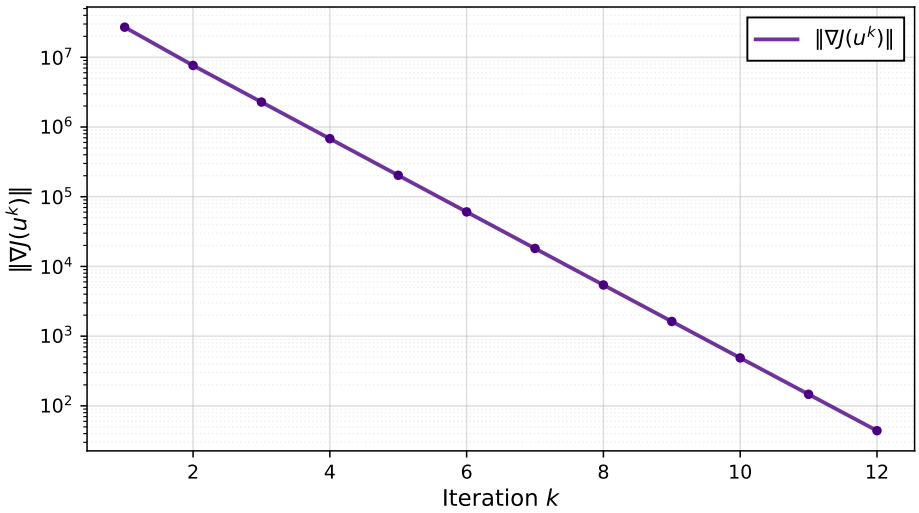


Figure 3.14: Cost gradient norm evolution

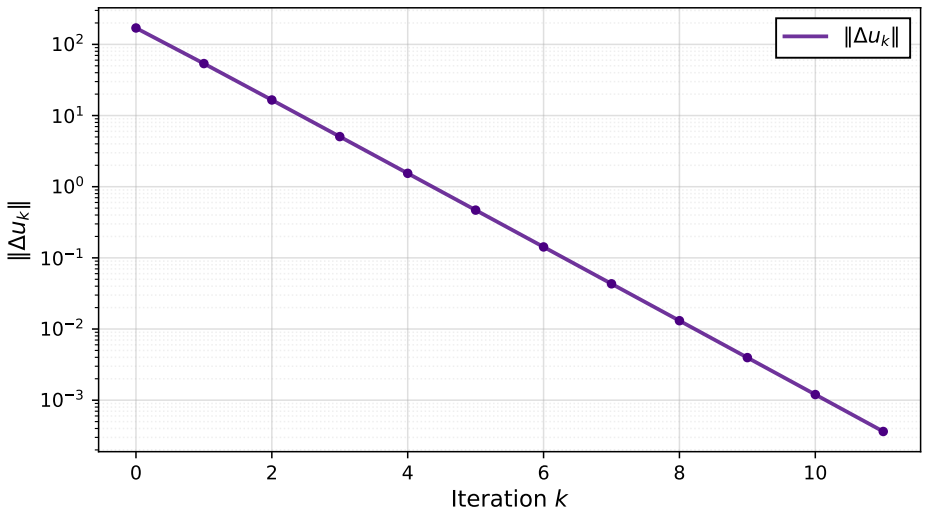


Figure 3.15: Evolution of $||\Delta u_k||$

3.4 Constant Cost Matrices Scenario

The following cost matrices have been adopted:

$$Q_t = Q_T = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.1}$$

$$R_t = \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix} \tag{3.2}$$

Here, a summary of the results is presented.
The performances are obviously worse than the ones presented before.

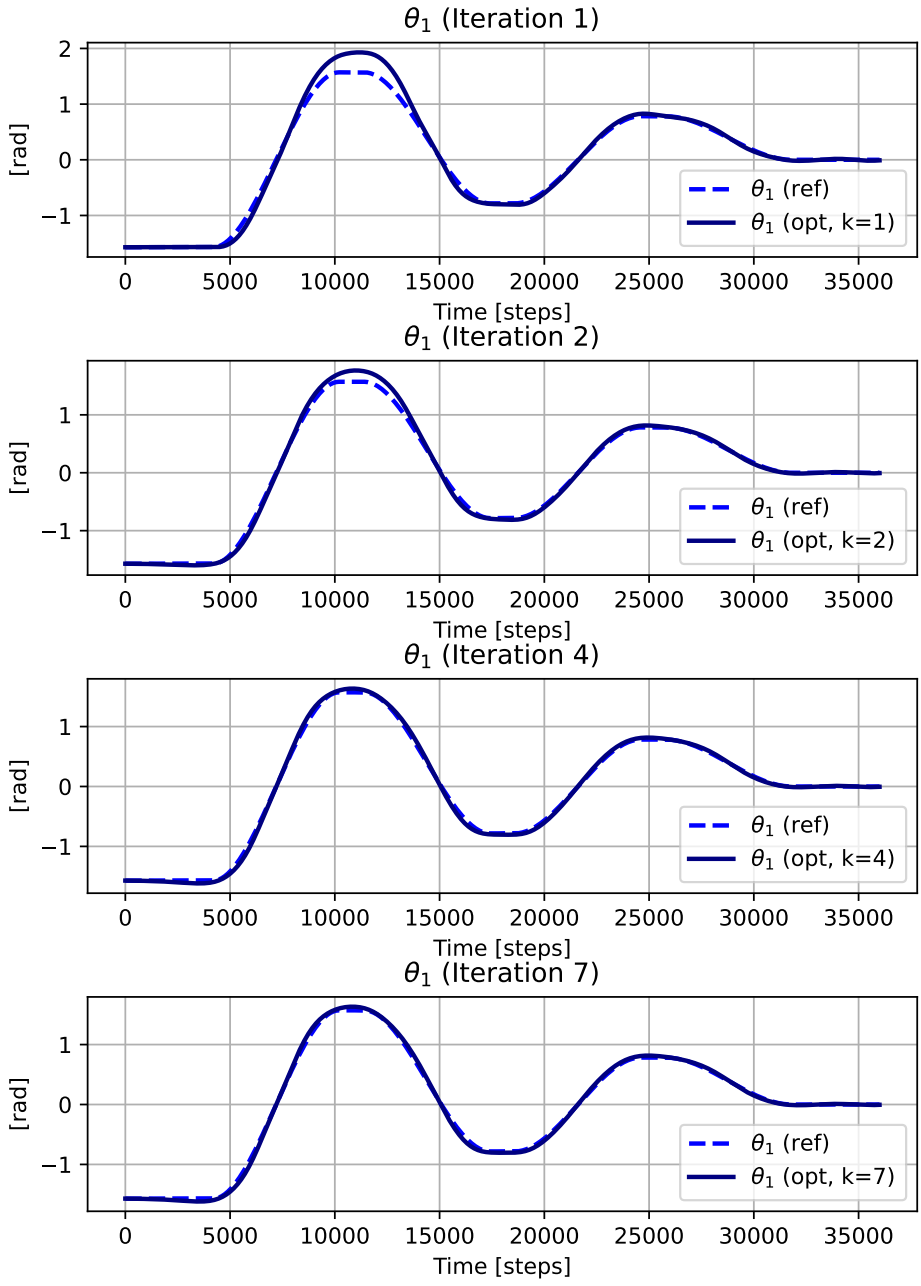


Figure 3.16: Evolution of θ_1 with constant Cost Matrices.

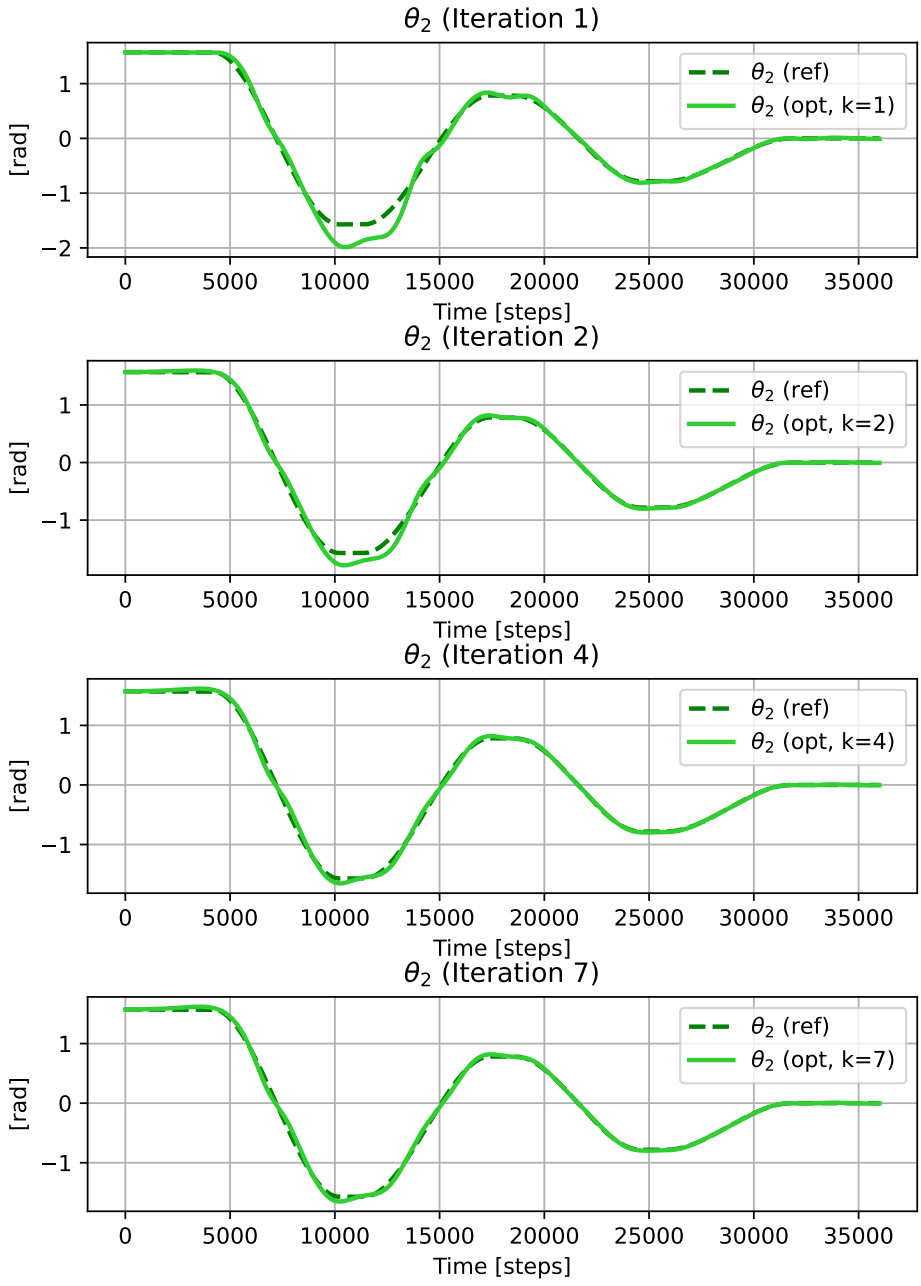


Figure 3.17: Evolution of θ_2 with constant Cost Matrices.

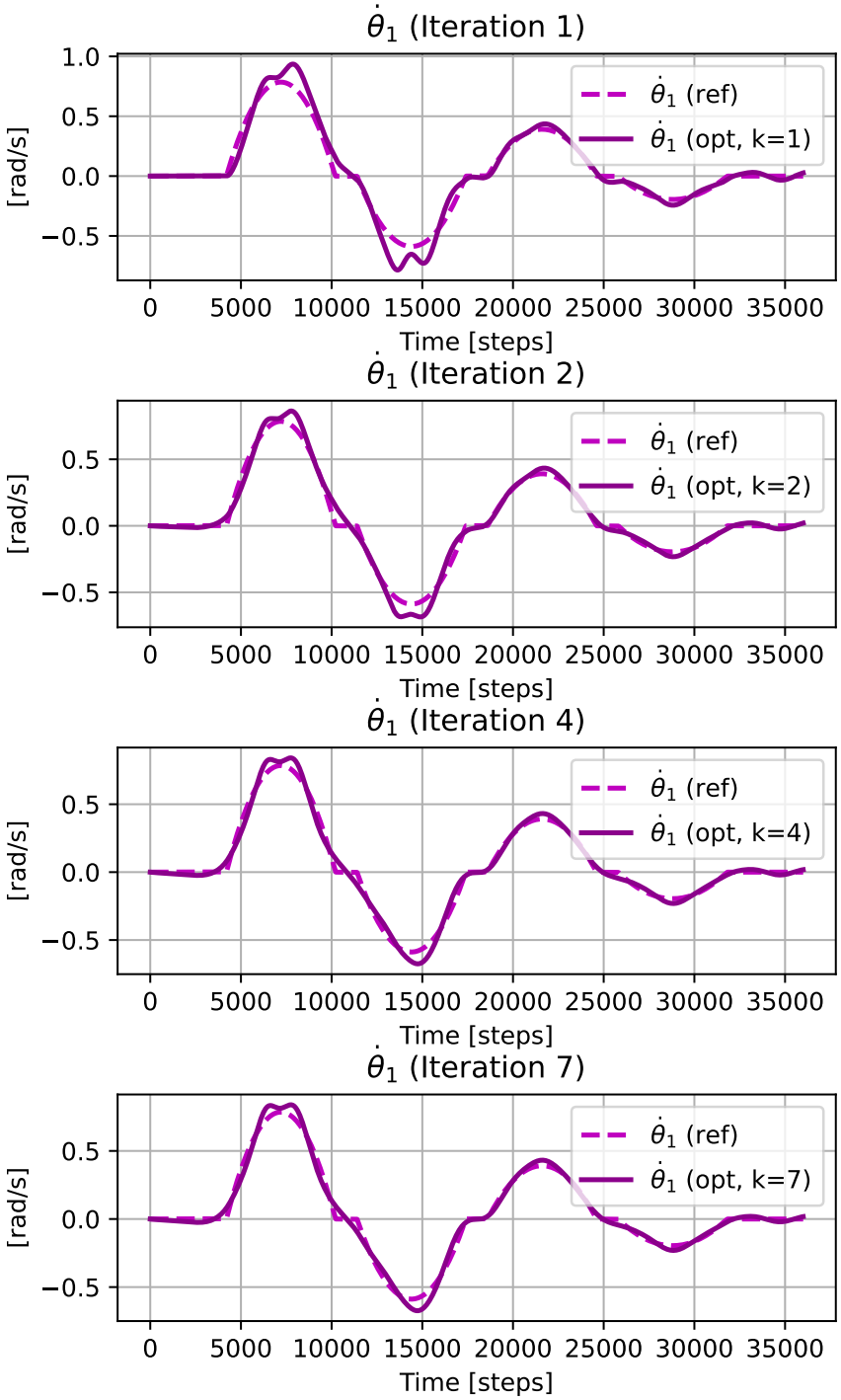


Figure 3.18: Evolution of $\dot{\theta}_1$ with constant Cost Matrices.

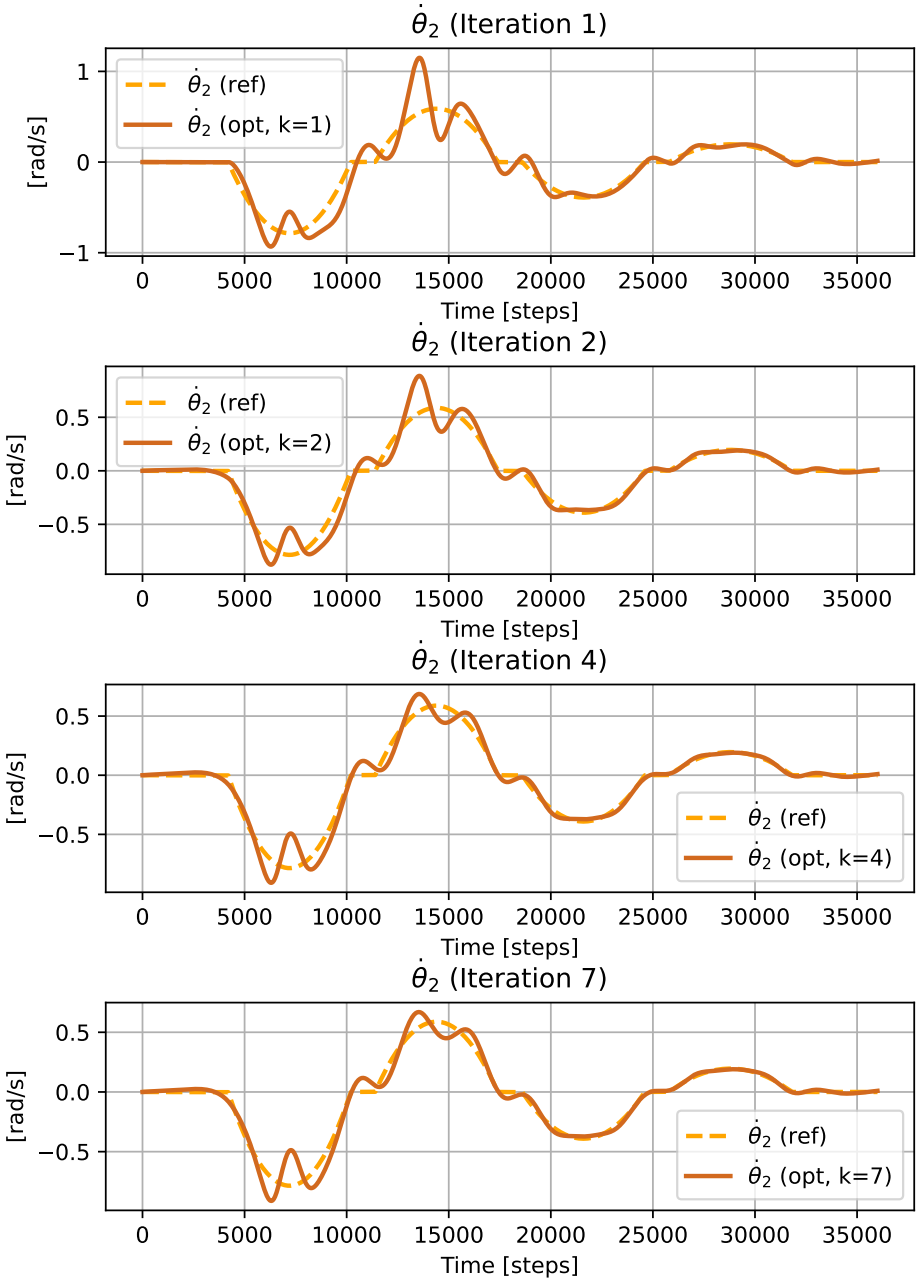


Figure 3.19: Evolution of $\dot{\theta}_2$ with constant Cost Matrices.

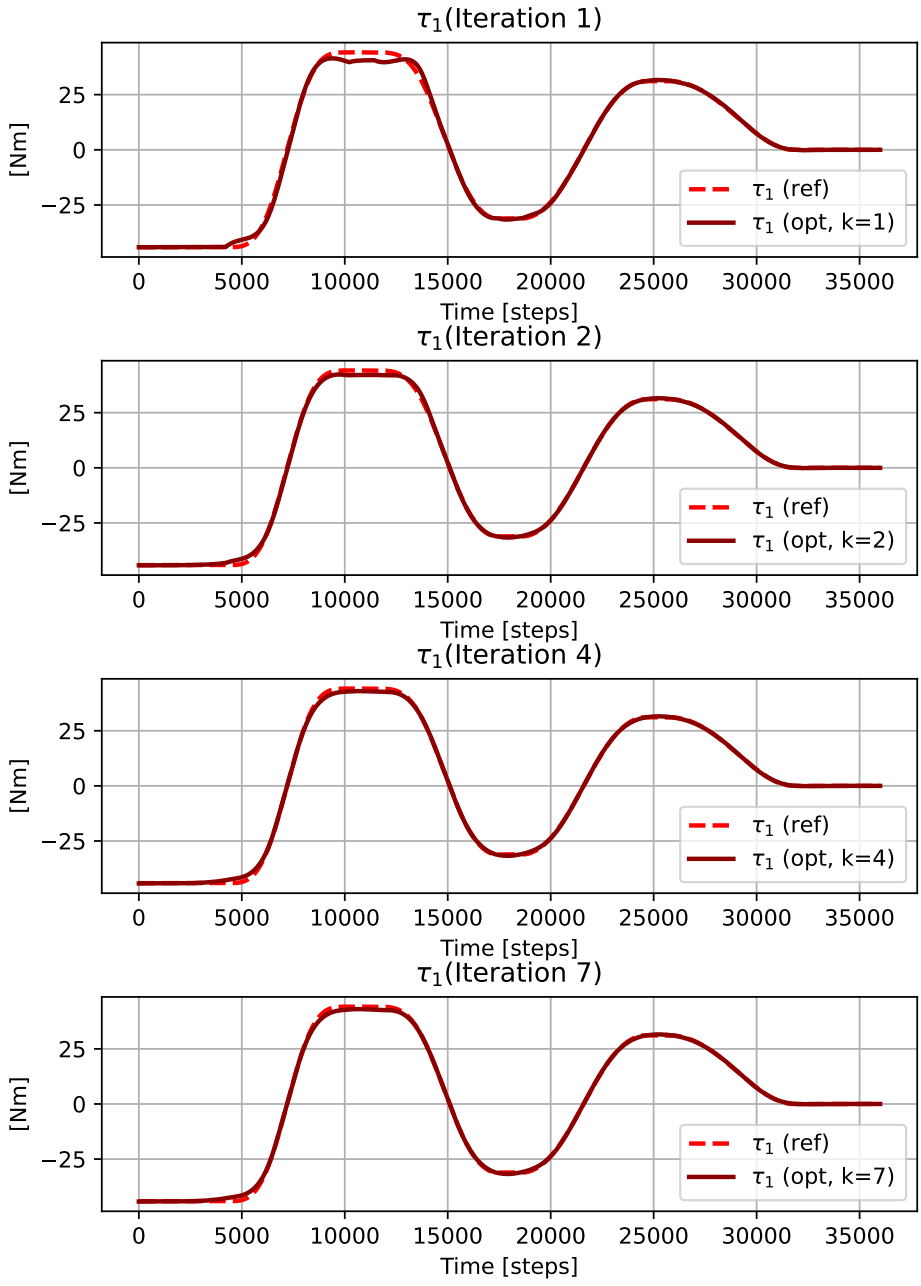


Figure 3.20: Evolution of τ with constant Cost Matrices.

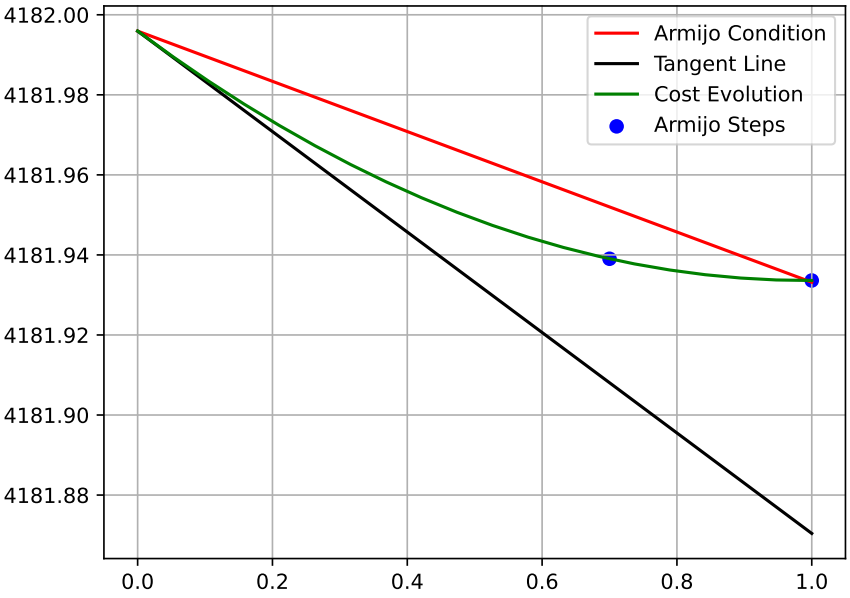


Figure 3.21: Armijo step-size selection: last iteration.

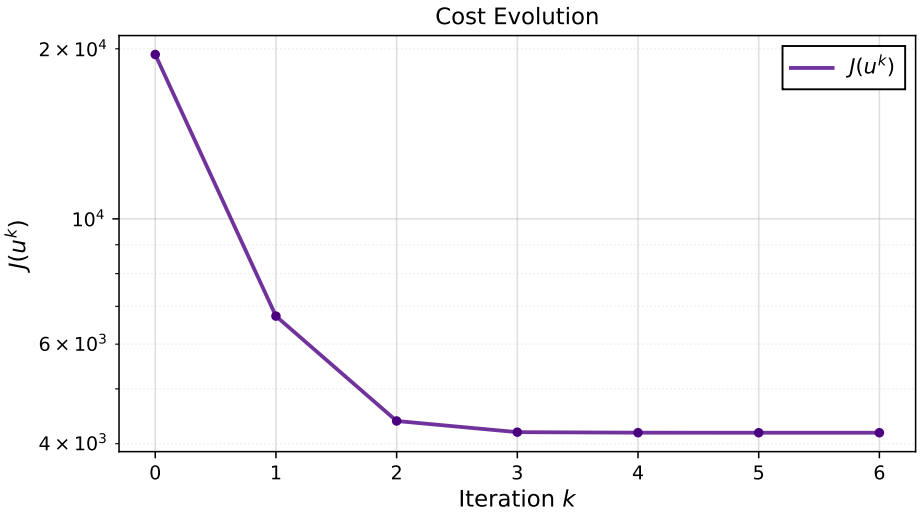


Figure 3.22: Evolution of cost function with constant Cost Matrices.

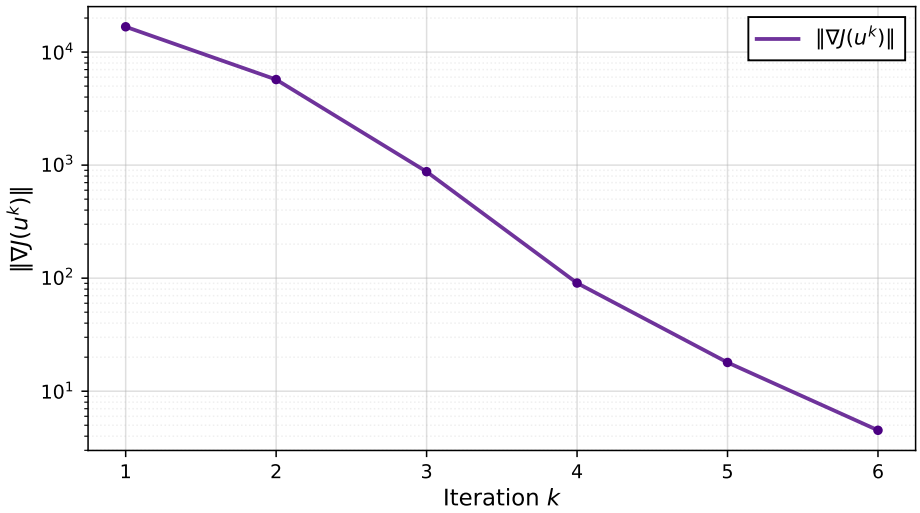


Figure 3.23: Evolution of $||\nabla J(u)||$ with constant Cost Matrices.

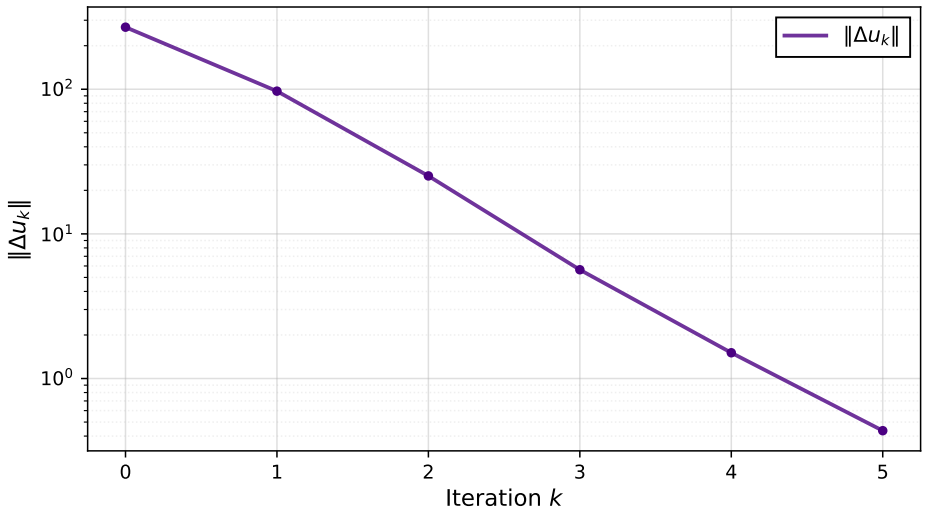


Figure 3.24: Evolution of $||\Delta u_k||$ with constant Cost Matrices.

Chapter 4

Trajectory tracking via LQR

4.1 Dynamics Linearization and LQR Design

In this task, the robot dynamics are linearized around the trajectory $(x^{\text{gen}}, u^{\text{gen}})$ computed in Task 2. Using the linearized model, the LQR algorithm is applied to design an optimal feedback controller for tracking the reference trajectory. The objective is to solve the following Linear-Quadratic (LQ) problem:

$$\min_{\substack{\Delta x_1,\ldots,\Delta x_T,\\ \Delta u_0,\ldots,\Delta u_{T-1}}} \quad \sum_{t=0}^{T-1} \Delta x_t^\top Q_t^{\text{reg}} \Delta x_t + \Delta x_t^\top R_t^{\text{reg}} \Delta u_t + \Delta x_T^\top Q_T^{\text{reg}} \Delta x_T$$

Subject to:

$$\Delta x_{t+1} = A_t^{\text{gen}} \Delta x_t + B_t^{\text{gen}} \Delta u_t, \quad t = 0, \ldots, T-1$$

$$x_0 = 0$$

Here, $\Delta x_t = x_t - x_t^{\text{gen}}$, where A_t^{gen} and B_t^{gen} are the matrices obtained by linearizing the nonlinear system dynamics around the reference trajectory $(x_{\text{gen}}, u_{\text{gen}})$. The cost matrices Q_{reg} and R_{reg} are tuning parameters that need to be defined during the regulator's setup. These matrices balance the trade-off between state deviations and control effort. In task 3 the behavior of the cost matrices is the following:

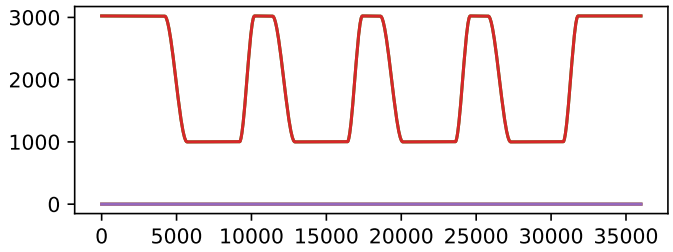


Figure 4.1: Evolution of cost matrices.

To compute the feedback gain, the discrete Riccati equation is solved backward in time. Starting with $P_T = Q_T^{\text{reg}}$, compute recursively for $t = T-1, \ldots, 0$:

$$P_t^{\text{reg}} = Q_t^{\text{reg}} + A_t^{\text{gen},\top} P_{t+1}^{\text{reg}} A_t^{\text{gen}}$$
$$- \left(A_t^{\text{gen},\top} P_{t+1}^{\text{reg}} B_t^{\text{gen}}\right) \left(R_t^{\text{reg}} + B_t^{\text{gen},\top} P_{t+1}^{\text{reg}} B_t^{\text{gen}}\right)^{-1} \left(B_t^{\text{gen},\top} P_{t+1}^{\text{reg}} A_t^{\text{gen}}\right)$$

Using the results from the Riccati equation, the feedback gain K_t^{reg} is calculated for all $t = 0, \ldots, T-1$:

$$K_t^{\text{reg}} = -(R_t^{\text{reg}} + B_t^{\text{gen},\top} P_{t+1} B_t^{\text{gen}})^{-1} (B_t^{\text{gen},\top} P_{t+1} A_t^{\text{gen}})$$

The LQR algorithm computes the control inputs in a feedback form, ensuring robustness. At each time step, the input is computed as a function of the deviation of the state from the reference trajectory:

$$u_t = u_t^{\text{gen}} + K_t^{\text{reg}} (x_t - x_t^{\text{gen}})$$

The system evolves according to:

$$x_{t+1} = f_t(x_t, u_t), \quad t = 0, 1, \ldots$$

To evaluate the tracking performance, the system should be tested with a perturbed initial condition, like an initial state that deviates from x_0^{gen}. The feedback controller should be able to compensate for this offset and guide the system back towards the reference trajectory, minimizing the tracking error over time.

4.2 Performance Analysis and Plots

In Task 3, to test the tracking performance, a perturbed initial condition is considered, with a state perturbation percentage.

Case	**State Perturbation Percentage**
Case 1	0.02
Case 2	0.05
Case 3	-0.1

Table 4.1: Perturbation values for different cases

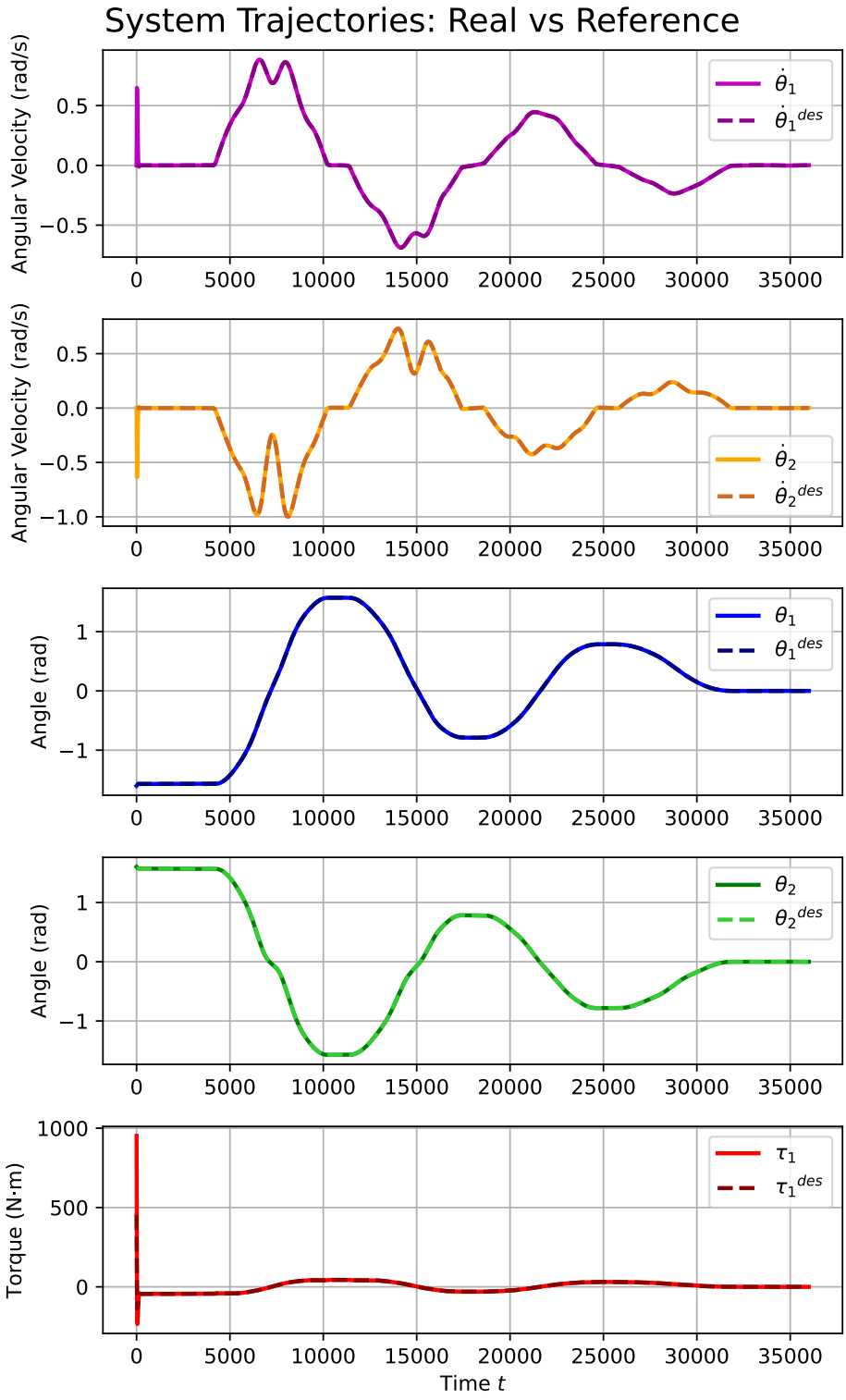


Figure 4.2: Trajectories case 1

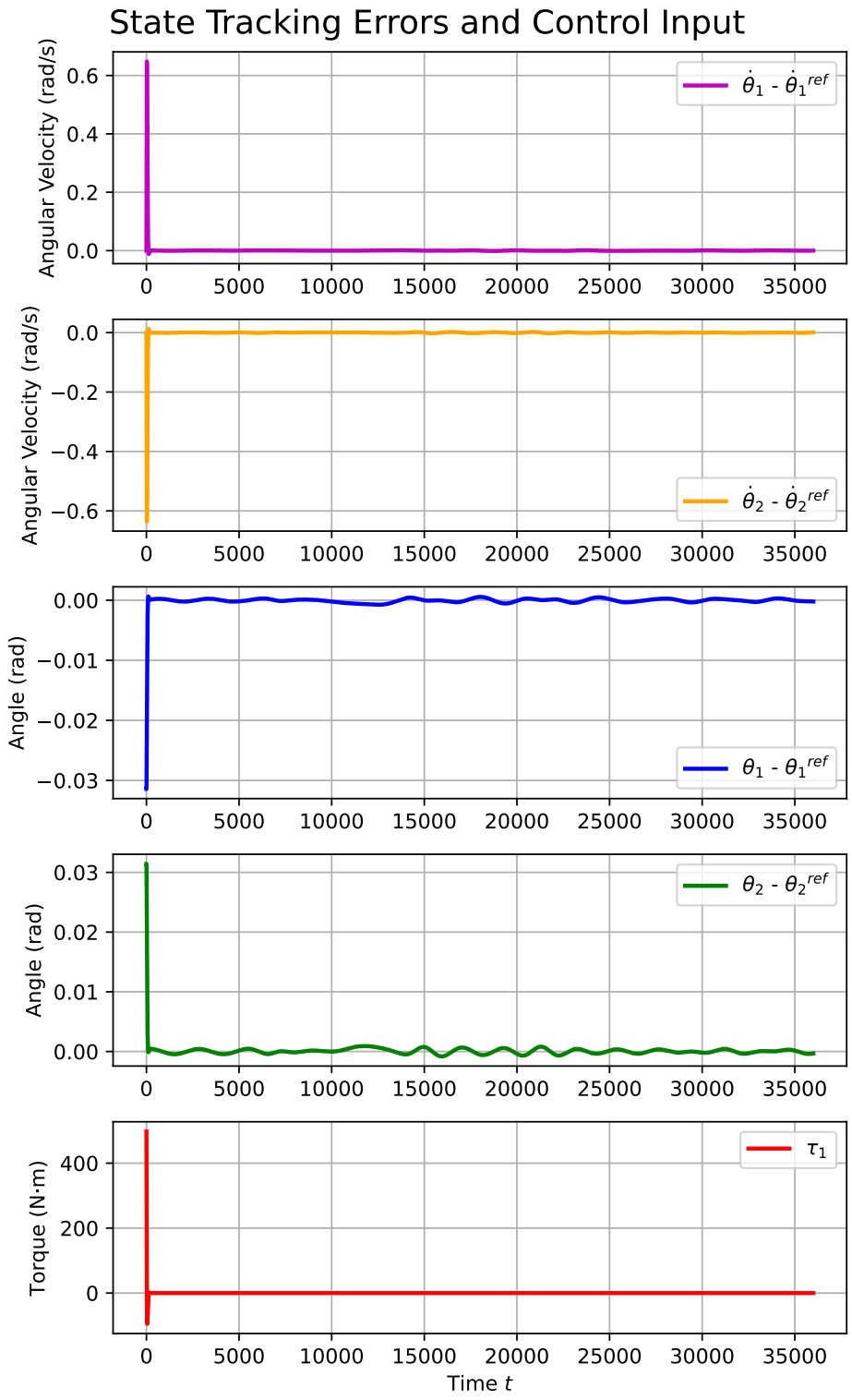


Figure 4.3: Tracking errors case 1

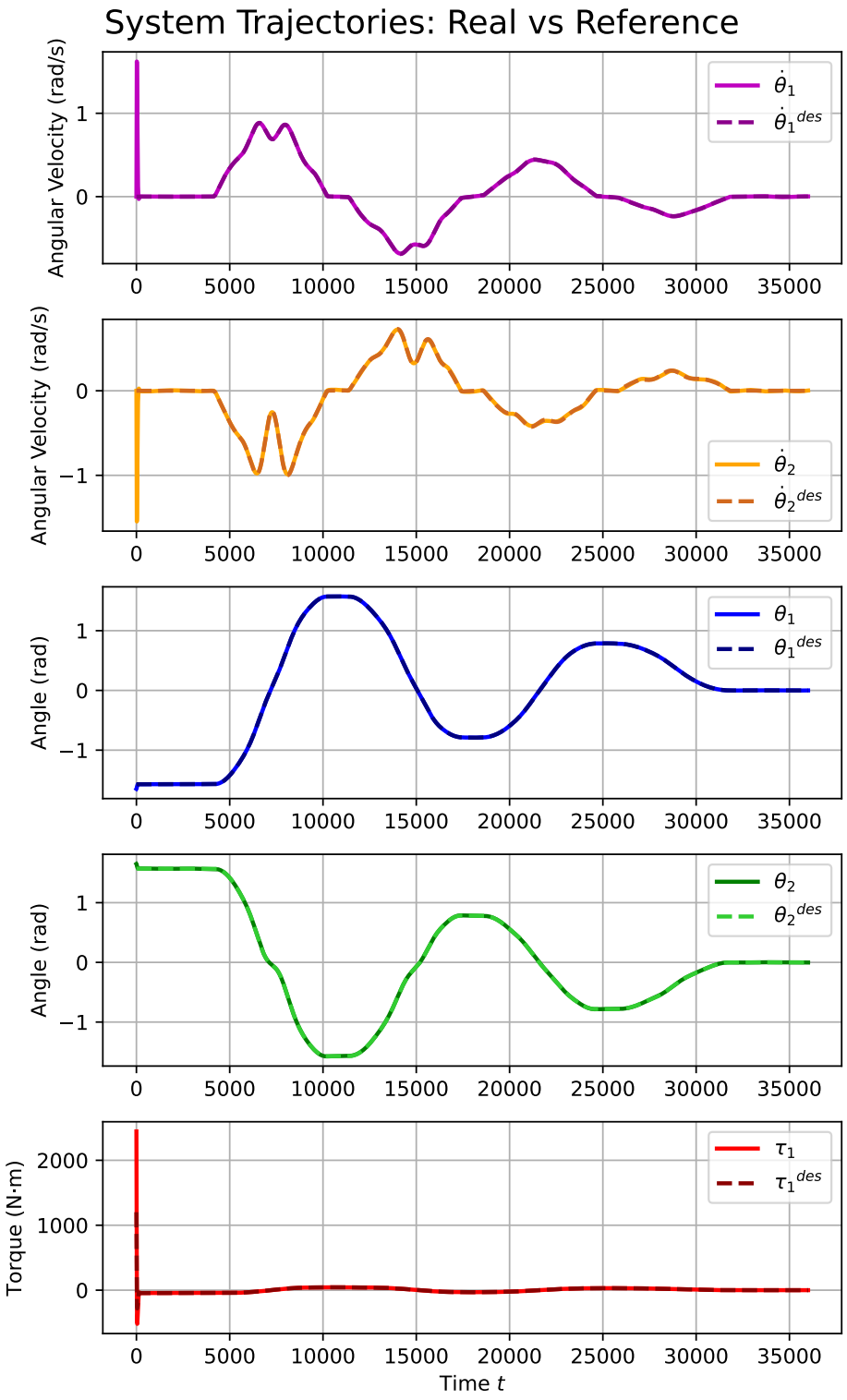


Figure 4.4: Trajectories case 2

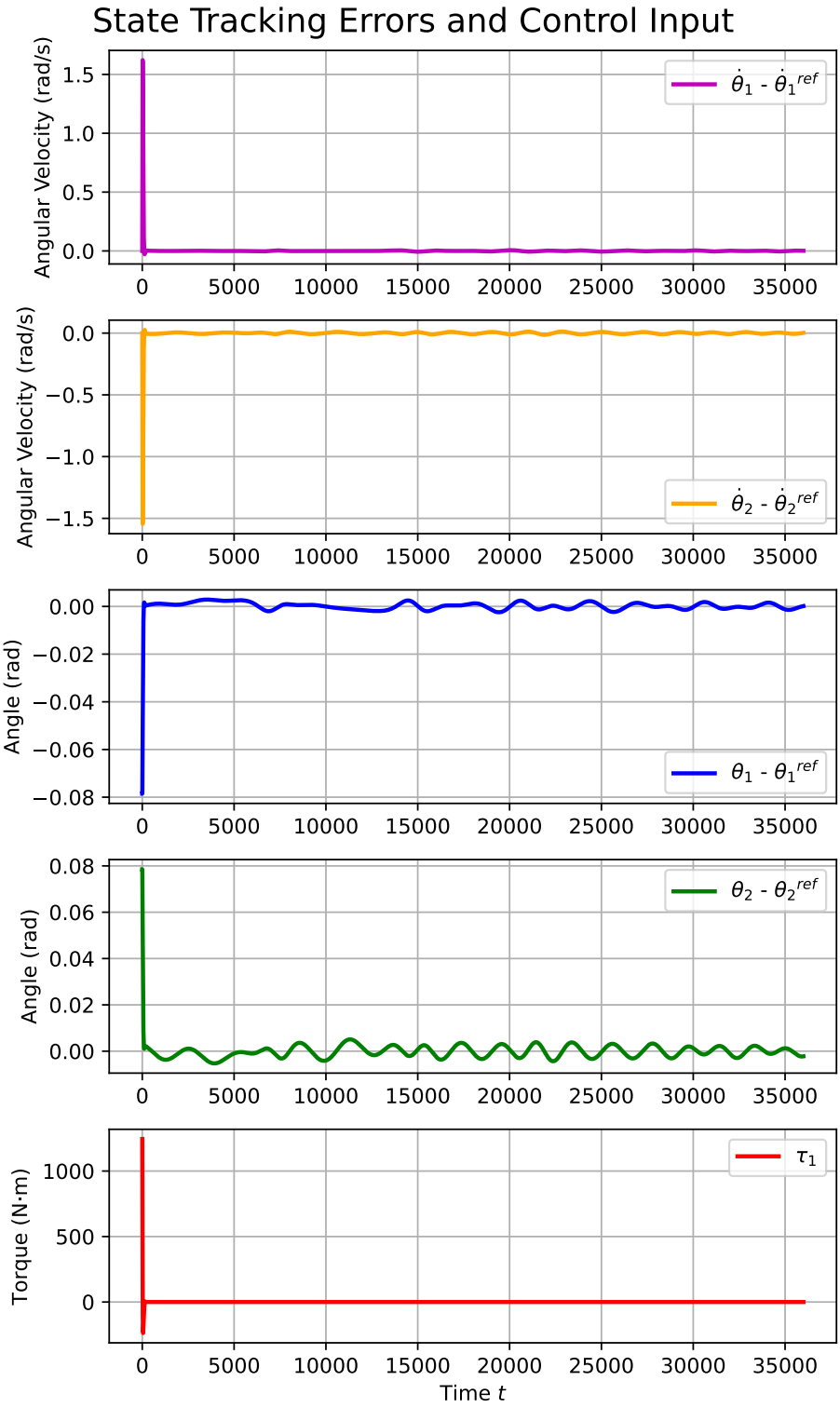


Figure 4.5: Tracking errors case 2

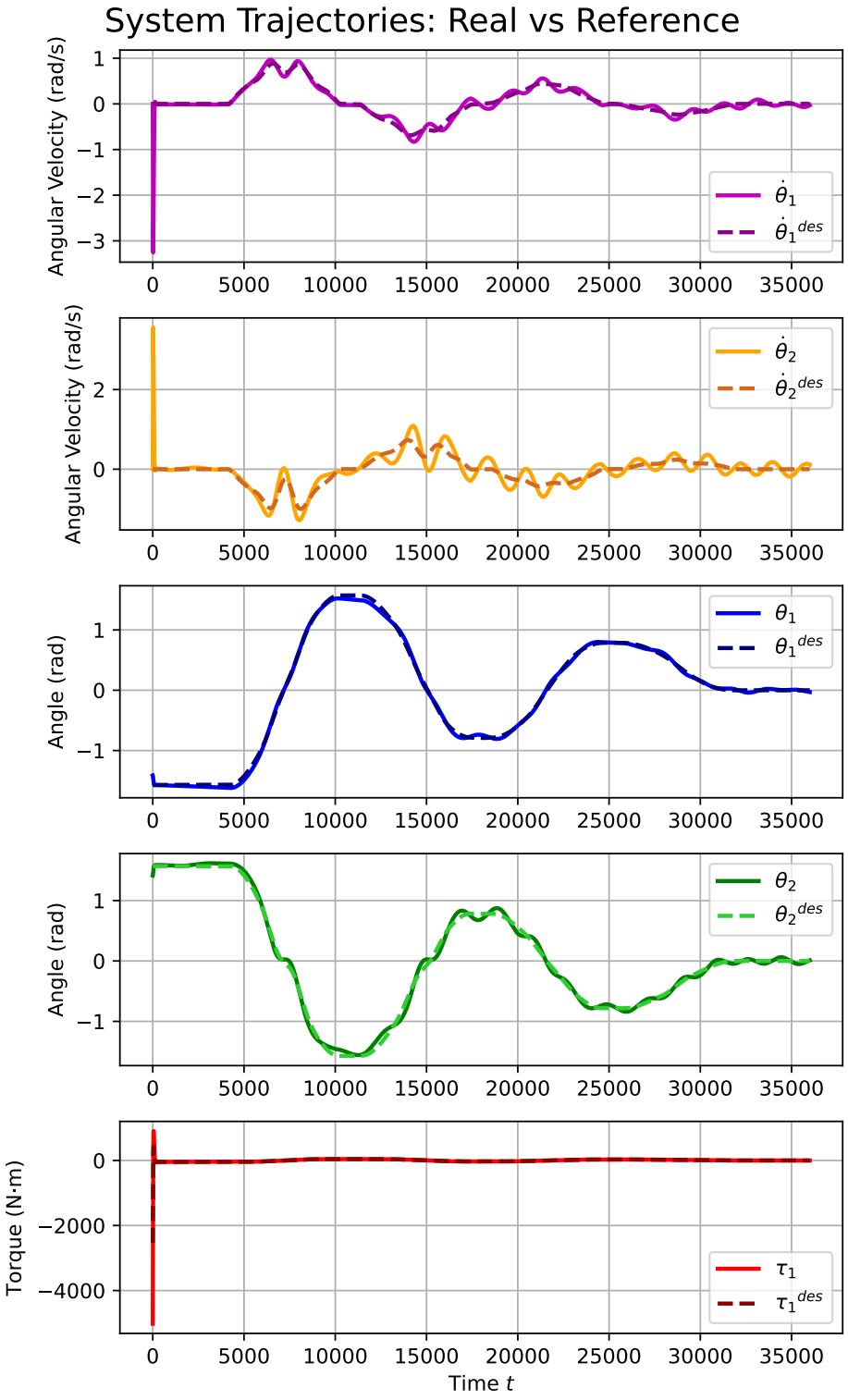


Figure 4.6: Trajectories case 3

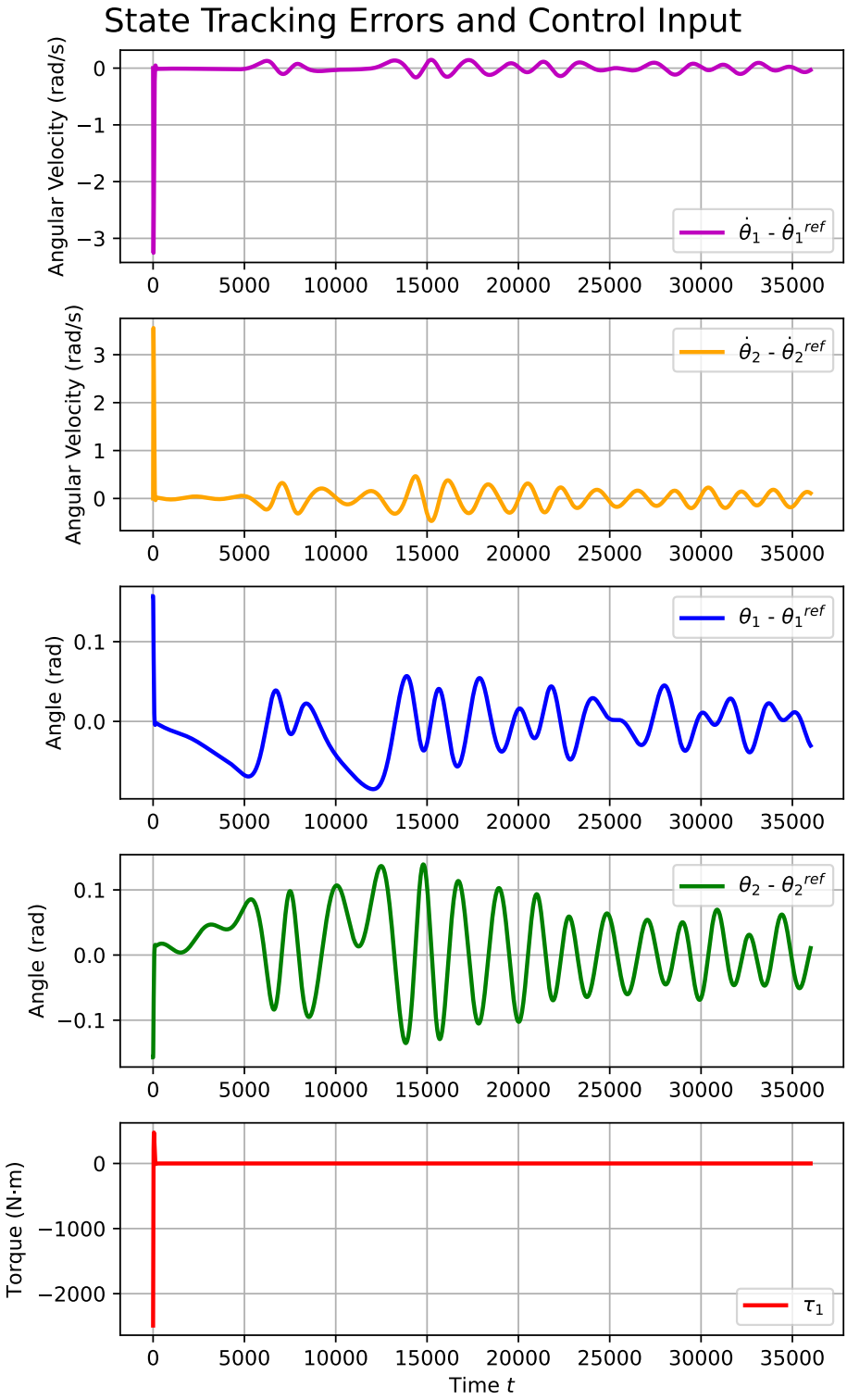


Figure 4.7: Tracking errors case 3

Chapter 5

Trajectory tracking via MPC

5.1 MPC Formulation and Implementation

The goal of this task is to apply a Model Predictive Control (MPC) framework to track the optimal trajectory $(x^{\text{opt}}, u^{\text{opt}})$, which was generated in Task 2. The MPC ensures robustness to disturbances in the perturbed initial condition x_0.

For MPC computations, the dynamics of the system is approximated around the optimal trajectory generated in the Task 2, resulting in time-dependent matrices A_t^{opt} and B_t^{opt}.

$$A_t^{\text{opt}} = \left.\frac{\partial f}{\partial x}\right|_{(x_t^{\text{opt}}, u_t^{\text{opt}})}, \quad B_t^{\text{opt}} = \left.\frac{\partial f}{\partial u}\right|_{(x_t^{\text{opt}}, u_t^{\text{opt}})} \tag{5.1}$$

The weighting matrices Q_t, R_t, and Q_T are chosen to penalize deviations in states and inputs. All these matrices are positive definite, ensuring stability and good tracking performance.

At each time step t, the initial condition for the predicted trajectory is taken as the current measured state x_t^{meas}.
The optimization problem minimizes a cost function defined over the prediction horizon T_{pred}:

$$\min_{\Delta x, \Delta u} \sum_{\tau=t}^{t+T-1} \left(\Delta x_\tau^\top Q_\tau \Delta x_\tau + \Delta u_\tau^\top R_\tau \Delta u_\tau \right) + \Delta x_{t+T}^\top Q_T \Delta x_{t+T}$$

subject to:

$$\begin{aligned} &x_{\tau+1} = A_t^{\text{opt}} x_\tau + B_t^{\text{opt}}\ u_\tau, && \tau = t, \ldots, t+T-1 \\ &x_\tau \in \mathcal{X}, \quad u_\tau \in \mathcal{U}, && \tau = t, \ldots, t+T \\ &x_t^{\text{mpc}} = x_t^{\text{meas}} \end{aligned}$$

Here, $\Delta x = x_t^{\text{mpc}} - x_t^{\text{opt}}$ and $\Delta u = u_t^{\text{mpc}} - u_t^{\text{opt}}$ represent deviations from the optimal reference trajectory.

The algorithm enforces specific constraints on the states and inputs to ensure safe and feasible operation:

$$u_{\min} \leq u_\tau \leq u_{\max}, \quad x_{\min} \leq x_\tau \leq x_{\max}, \quad \forall \tau \in [t, t+T] \tag{5.2}$$

These constraints include limits on angular velocities, angles, and control inputs.

At each time step t, the algorithm solves the optimization problem to compute the predicted control inputs and state trajectories. However, only the first control input u_t^{mpc} is applied to the system. At the next time step $t+1$, the state x_{t+1}^{meas} is re-measured, and the optimization problem is solved again. This process allows the MPC to dynamically adapt the control actions, ensuring robustness to disturbances and modeling inaccuracies.

5.2 Performance Analysis and Plots

In Task 4, to test the tracking performance, a perturbed initial condition is considered, with a state perturbation percentage. In addition, the model accounts for both measurement and actuation noise to provide a more realistic representation of real-time implementation challenges.

Case	**State Perturbation**	**Measurement Noise**	**Actuation Noise**
1	0.05	0	0
2	-0.1	0.005	0.005
3	-0.2	0.05	0.05

Table 5.1: State perturbation values, measurement noise, and attenuation noise for different cases

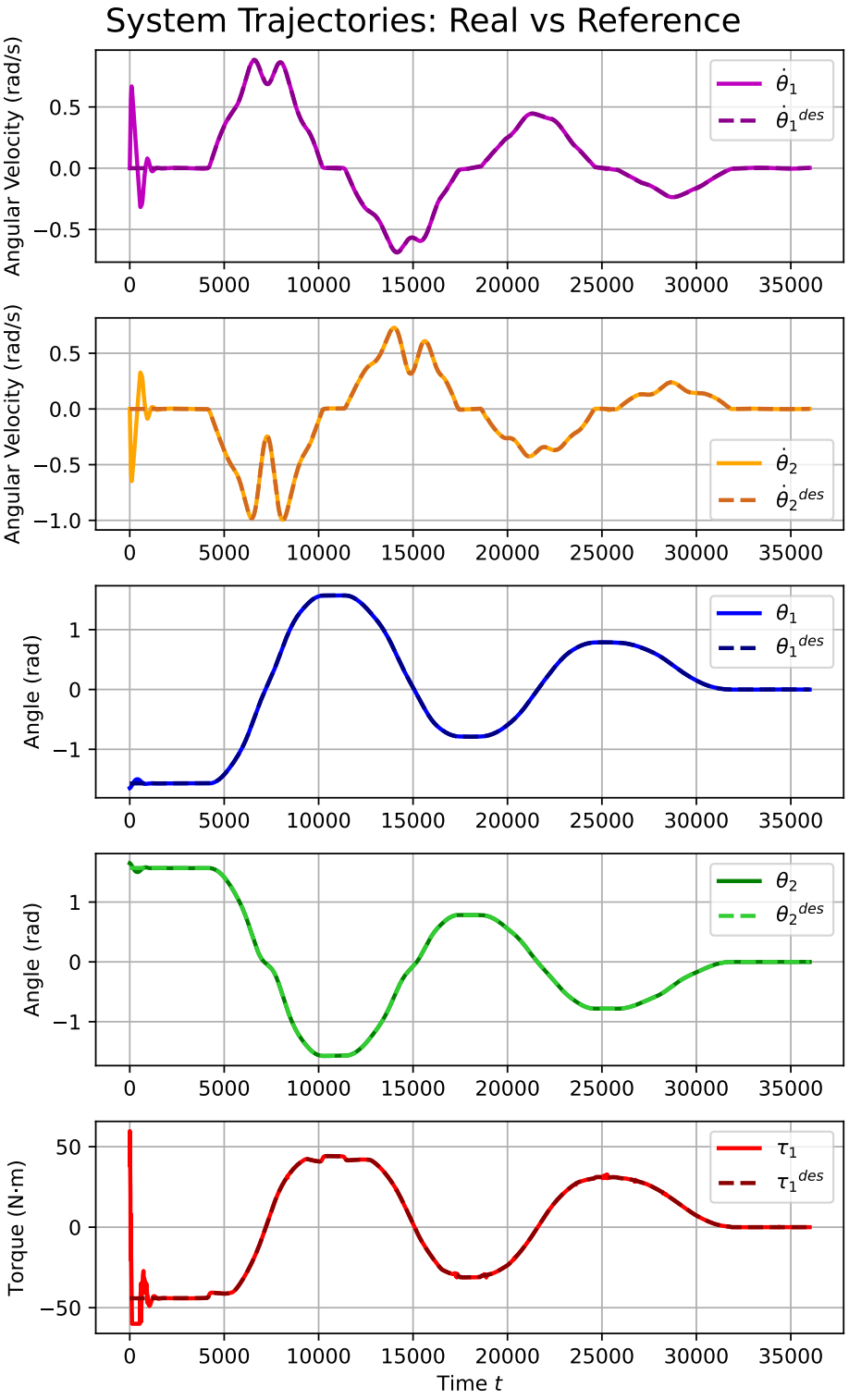


Figure 5.1: Trajectories case 1

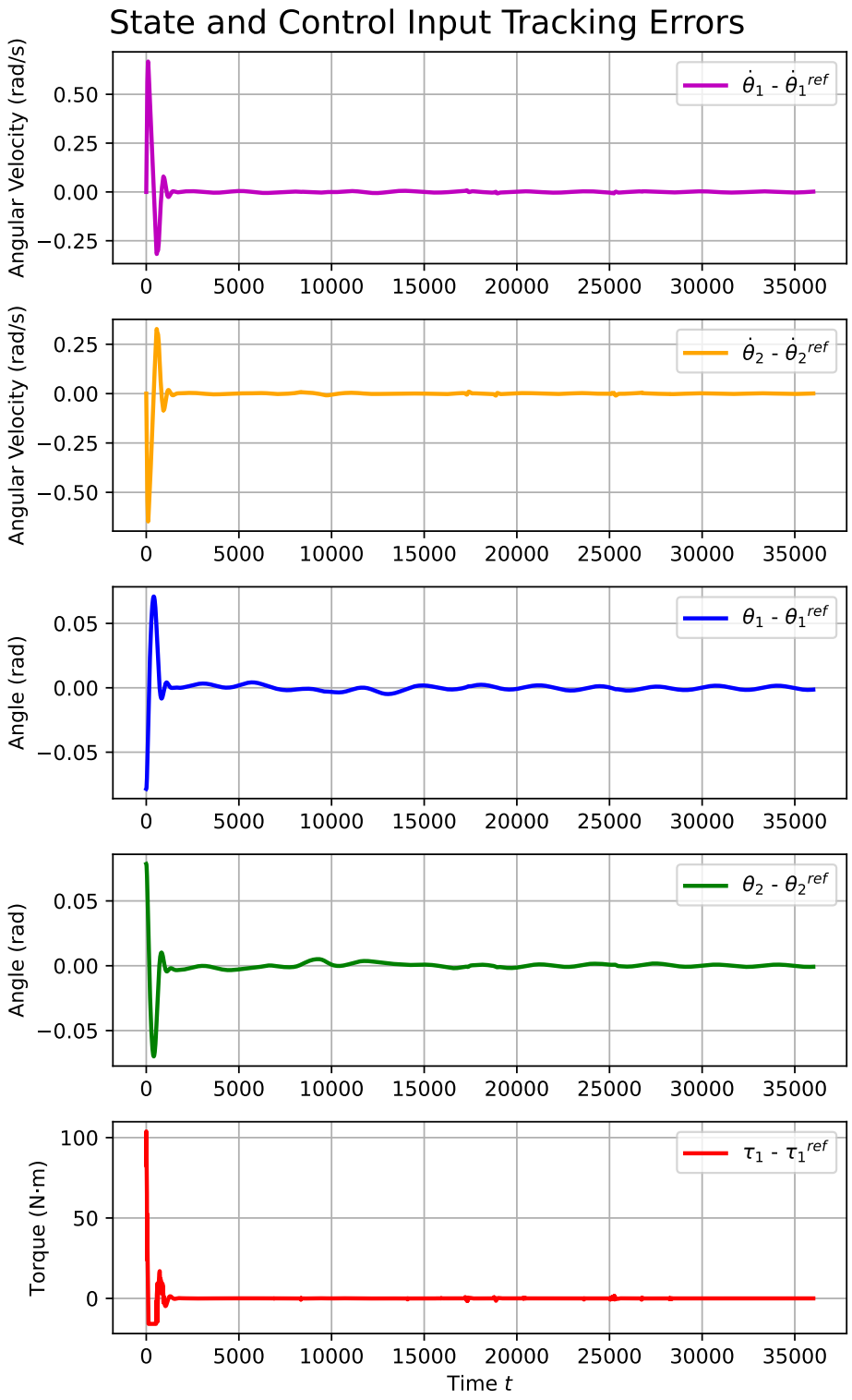


Figure 5.2: Tracking errors case 1

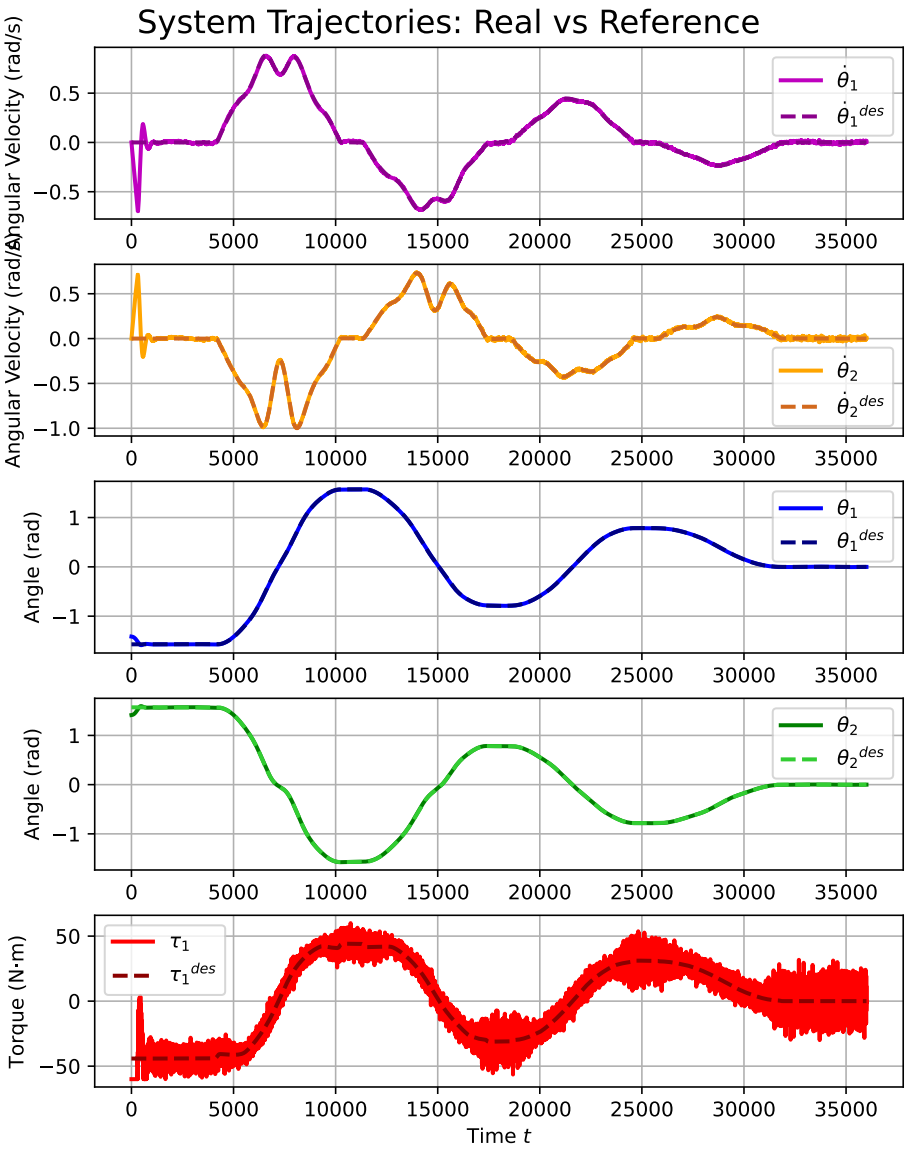


Figure 5.3: Trajectories case 2

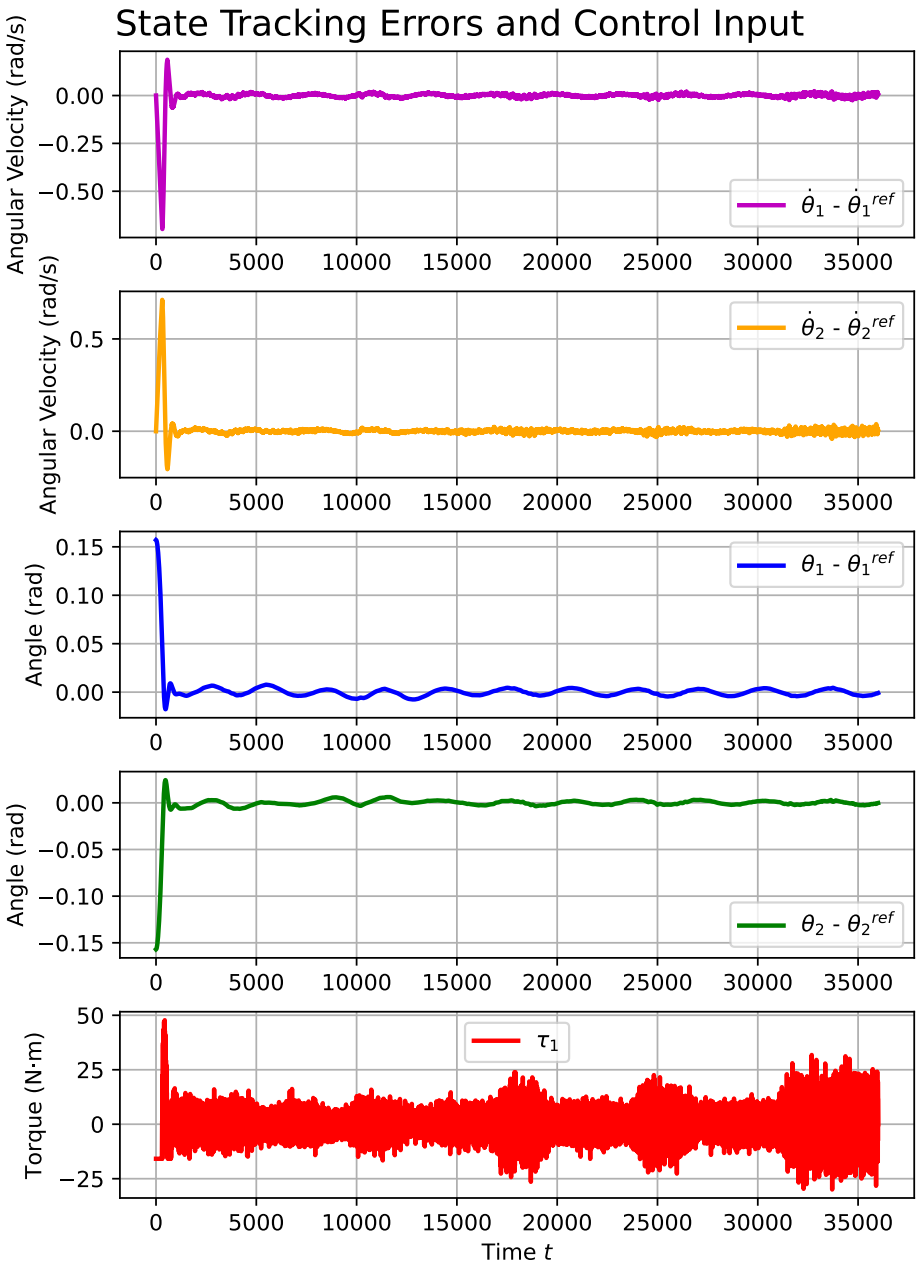


Figure 5.4: Tracking errors case 2

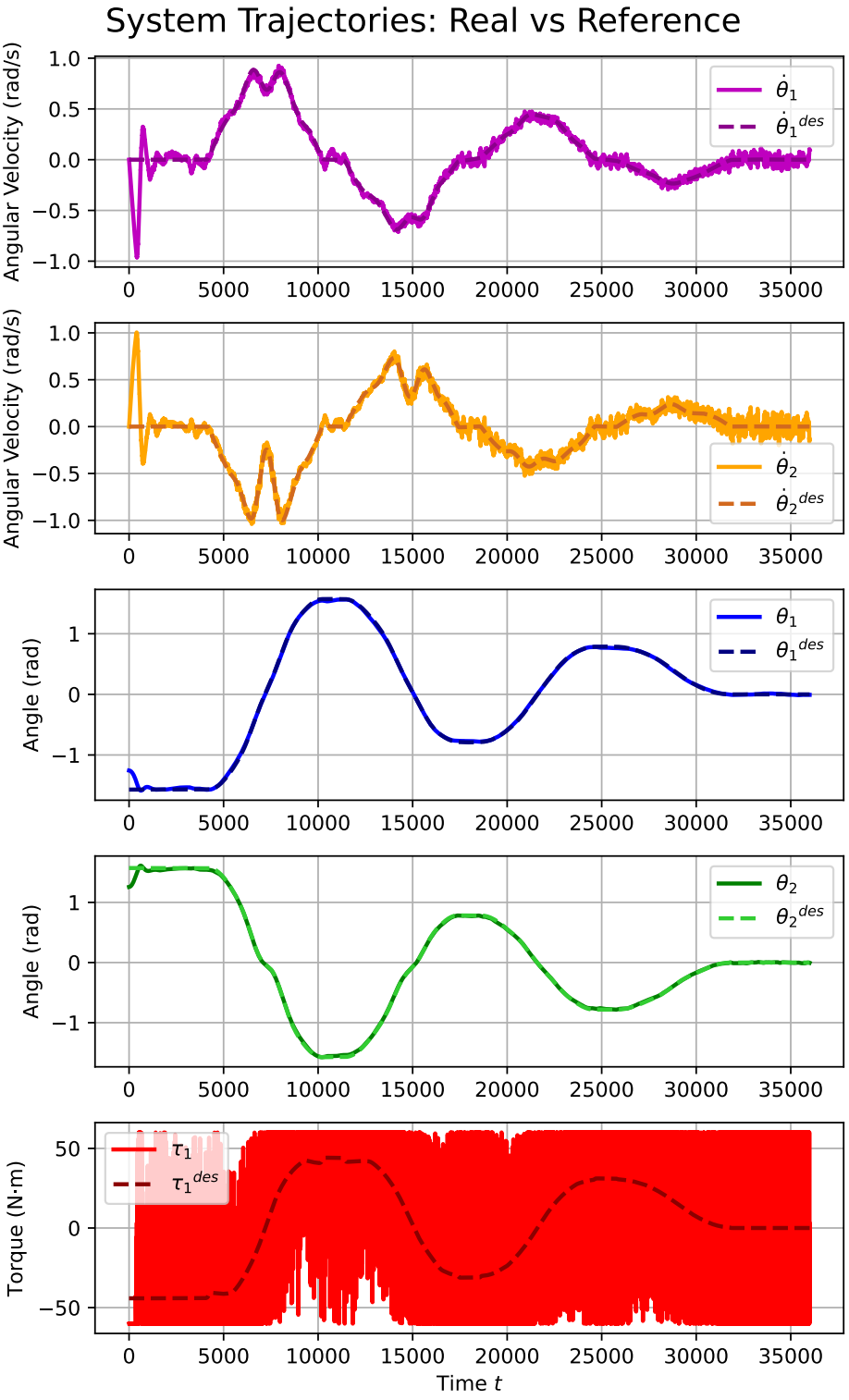


Figure 5.5: Trajectories case 3

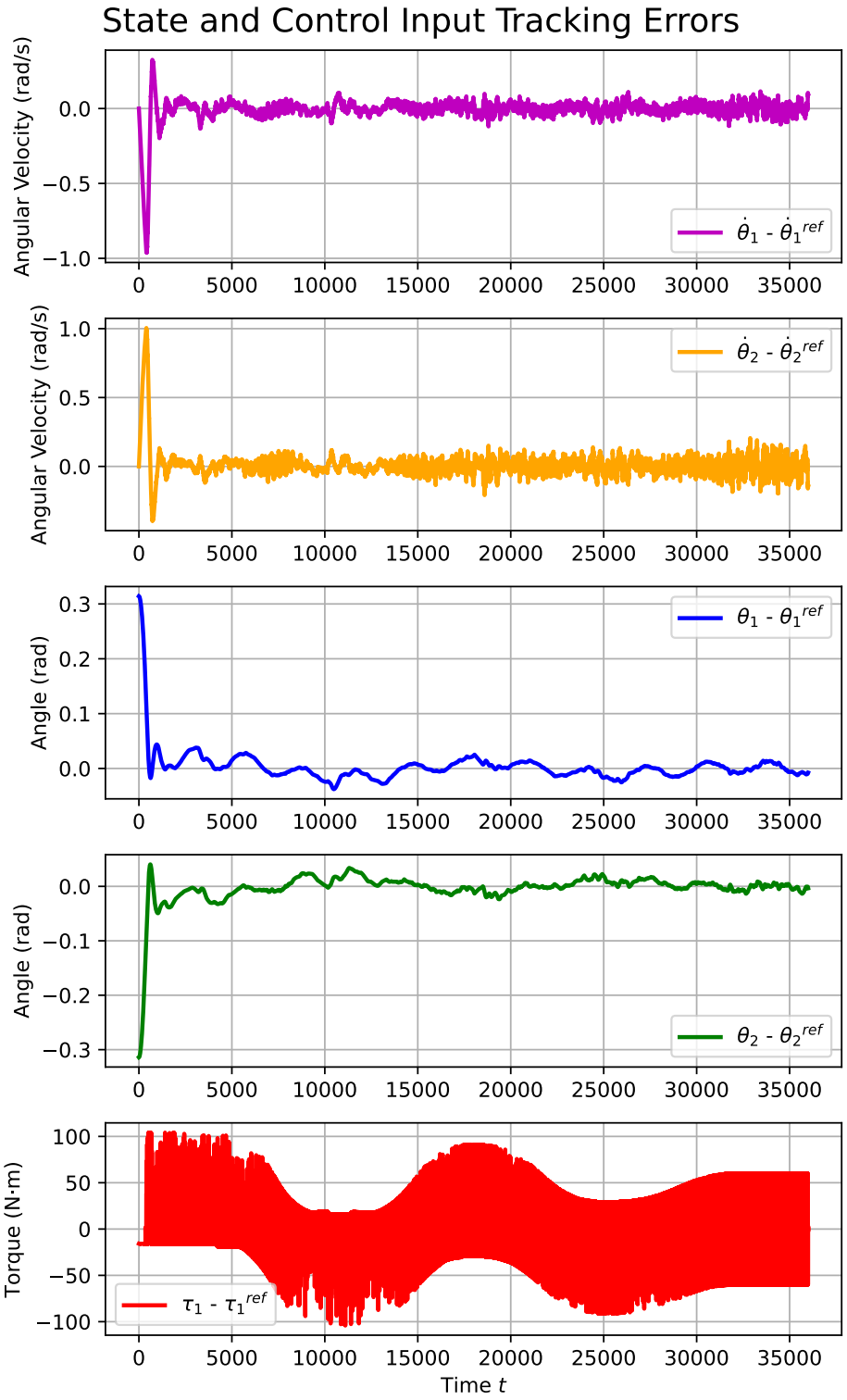
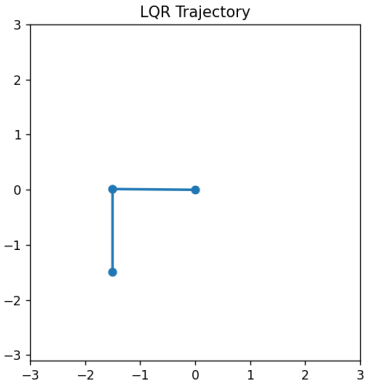


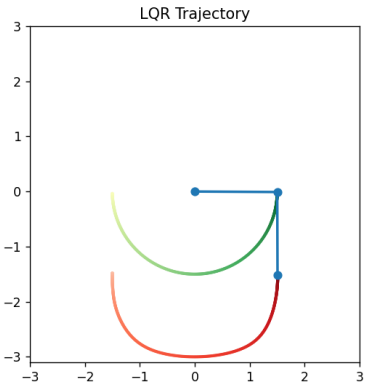
Figure 5.6: Tracking errors case 3

Chapter 6

Animation

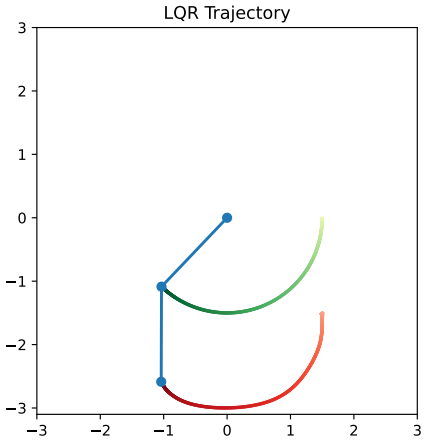


(a) Equilibrium point 1

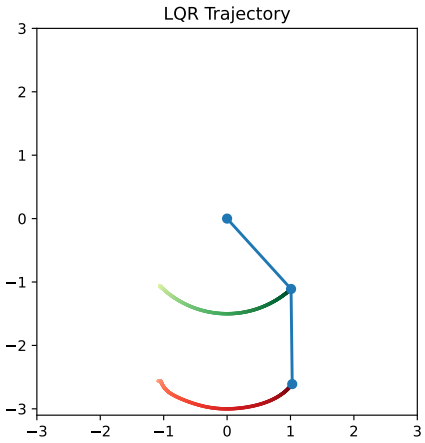


(b) Equilibrium point 2

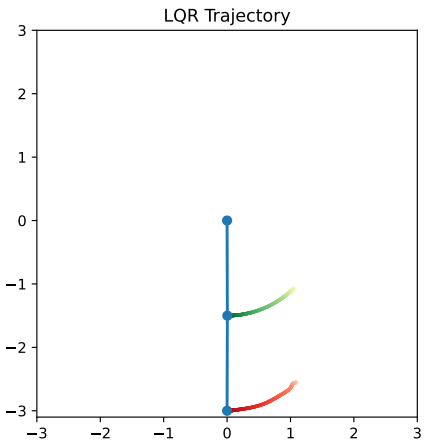
Figure 6.1



(a) Equilibrium point 3



(b) Equilibrium point 4



(c) Equilibrium point 5

Figure 6.2

Conclusions

This work addresses the optimal control of a planar underactuated two-link robotic manipulator. The system dynamics were discretized, and the system's evolution was observed to be consistent and promising. Newton's method for root finding, used to compute the equilibria, proved to be highly effective. Also the reference generation for a quasi-static trajectory also demonstrated strong performance. Additionally, the formulation of a quadratic cost function effectively penalized errors in both states and inputs.

In Task 1, although significant results were achieved, the Armijo plots were sometimes not representative in the last iterations of the algorithm. Several potential causes could explain this issue. One possibility is numerical instability, which can occur when handling extremely large or small values and the precision is crucial. The use of a step reference may exacerbate this problem due to its lack of continuity.

In Task 2 a smooth reference curve was employed. In this case, the Armijo plots appeared accurate, with the descent direction consistently tangent to the cost function. Furthermore, the plot's scale indicated that the cost function's slope decreased over time, resulting in the Armijo method taking more steps as expected when nearing convergence.

The trajectory tracking task using Linear Quadratic Regulator (LQR) control was successfully implemented to design an optimal feedback control. The results demonstrated that the controller could rapidly eliminate disturbances applied at the initial condition. Since the inputs were not constrained, the control was highly aggressive initially, quickly removing oscillations. However, when disturbances increased, some oscillations were not entirely eliminated.

Trajectory tracking using Model Predictive Control (MPC) demonstrated robustness to both state and input perturbations. In cases 1 and 2, the same disturbance values were applied at the initial condition as in cases 2 and 3 of the LQR. A comparison revealed that LQR quickly mitigated perturbations but struggled with larger disturbances. In contrast, MPC effectively compensated for even larger disturbances, as observed in case 3. Additionally, MPC managed Gaussian noise in both sensor readings and actuation effectively while adhering to constraints on input and

state variations. However, as noise levels increased, noticeable oscillations in actuation arose, a result of the controller's aggressive behavior. These oscillations could potentially be mitigated by imposing constraints on input variation over time. Another drawback of the MPC was the limited prediction window. While this constraint enables real-time applications, it reduces the controller's overall effectiveness.

The animations provided demonstrated satisfactory results across all tasks, as the references were closely followed and oscillations were eliminated through actuation.