

# A Quadratic Control Framework for Dynamic Systems

Igor Ladnik\*

## Abstract

This article presents a unified approach to quadratic optimal control for both linear and nonlinear discrete-time systems, with a focus on trajectory tracking. The control strategy is based on minimizing a quadratic cost function that penalizes deviations of system states and control inputs from their desired trajectories.

For linear systems, the classical Linear Quadratic Regulator (LQR) solution is derived using dynamic programming, resulting in recursive equations for feedback and feedforward terms. For nonlinear dynamics, the Iterative Linear Quadratic Regulator (iLQR) method is employed, which iteratively linearizes the system and solves a sequence of LQR problems to converge to an optimal policy.

To implement this approach, a software service was developed and tested on several canonical models, including: Rayleigh oscillator, inverted pendulum on a moving cart, two-link manipulator, and quadcopter. The results confirm that iLQR enables efficient and accurate trajectory tracking in the presence of nonlinearities.

To further enhance performance, it can be seamlessly integrated with Model Predictive Control (MPC), enabling online adaptation and improved robustness to constraints and system uncertainties.

arXiv:2504.15396v2 [eess.SY] 24 Apr 2025

---

\*LinkedIn, email

# 1 Introduction

Quadratic optimal control remains one of the most fundamental and practically significant problems in control theory, with wide-ranging applications from robotics and aerospace systems to economics and biological motion modeling. Classical Linear Quadratic Regulator (LQR) theory provides an elegant closed-form solution for optimal control in linear systems with quadratic cost, offering both efficiency and robustness.

However, many real-world systems exhibit nonlinear dynamics, rendering direct application of LQR ineffective. To address this, the iterative Linear Quadratic Regulator (iLQR) [1] method extends LQR to nonlinear domains through a sequence of linearizations and optimizations. This iterative approach preserves the computational efficiency of LQR while offering improved performance in nonlinear settings.

This article presents a unified treatment of both LQR and iLQR within a discrete-time trajectory tracking framework. The cost function is expressed relative to desired trajectories for both states and controls. The resulting method is applicable to a wide range of practical systems and is demonstrated through detailed simulations of Rayleigh oscillator, an inverted pendulum on a moving cart, a two-link robotic manipulator and a quadcopter stabilization problem.

In addition, the iLQR method can be efficiently combined with *Model Predictive Control* (MPC). MPC is an advanced control technique that solves a finite-horizon optimization problem at each time step using the current system state as the initial condition. By integrating iLQR into the MPC framework, one can rapidly compute optimal control sequences for nonlinear systems while continuously updating the control policy based on real-time feedback. This hybrid iLQR–MPC scheme is particularly advantageous for systems with constraints, disturbances, or time-varying dynamics, where both predictive adaptation and computational tractability are critical.

## 2 Problem Statement

We begin by formulating the optimal control problem for a nonlinear dynamical system governed by the differential equation:

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [0, T], \quad (2.1)$$

where  $x(t) \in \mathbb{R}^n$  is the system state, and  $u(t) \in \mathbb{R}^m$  is the control input (Figure 1).

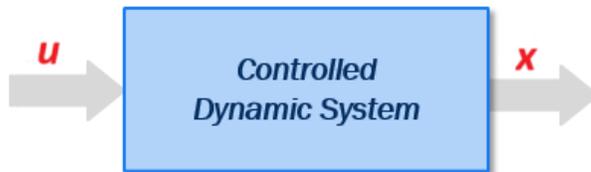


Figure 1: Open-loop control system.

Our objective is to minimize the following quadratic cost function:

$$J(u) = \int_0^T \left[ (\hat{x}(t) - x(t))^\top Q(t) (\hat{x}(t) - x(t)) + (\hat{u}(t) - u(t))^\top R(t) (\hat{u}(t) - u(t)) \right] dt, \quad (2.2)$$

where  $\hat{x}(t)$  and  $\hat{u}(t)$  denote the desired state and control trajectories. The matrices  $Q(t)$  and  $R(t)$  are symmetric and time-dependent, with  $Q(t)$  positive semi-definite and  $R(t)$  positive definite.

This formulation corresponds to a standard *trajectory tracking problem*, commonly encountered in control theory. The goal is to determine an optimal control input  $u(t)$  such that the system follows the desired trajectories as closely as possible while minimizing the cost.

In the following sections, we first examine the linear case, deriving the optimal solution using dynamic programming in discrete time. We then extend this approach to nonlinear systems using an iterative linearization technique, corresponding to the iLQR method [1].

### 3 Linear Case

First, we consider a *linear dynamic system* governed by the well-known differential equation:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)u(t)$$

where  $x(t)$  is the state vector,  $u(t)$  is the control input, and  $\bar{A}(t)$  and  $\bar{B}(t)$  are the system and input matrices, respectively.

In discrete form, this equation can be approximated as:

$$x_{k+1} = A_k x_k + B_k u_k \quad (3.1)$$

with the discrete-time matrices  $A_k$  and  $B_k$  defined by:

$$\begin{aligned} A_k &\approx I + \bar{A}(k\Delta t) \Delta t \\ B_k &\approx \bar{B}(k\Delta t) \Delta t \end{aligned}$$

where  $I$  is the identity matrix and  $\Delta t$  is the discretization time step.

Suppose that  $\hat{x}_0 = x_0$ . Then the discretization of the cost function (2.2) can be presented with the following expression

$$J_N(u) = \sum_{k=0}^{N-1} [(\hat{x}_{k+1} - x_{k+1})^\top Q_k (\hat{x}_{k+1} - x_{k+1}) + (\hat{u}_k - u_k)^\top R_k (\hat{u}_k - u_k)] \quad (3.2)$$

Let's denote minimum of  $J_N(u)$  as

$$m_N(x_0) = \min_{(u_0, u_1, \dots, u_{N-1})} J_N(u)$$

and the minimum of the cost function on the last  $N - (k + 1)$  steps of the process is  $m_{N-(k+1)}(x_{k+1})$ .

Consider the  $k$ -th step of the process. Assuming that the control over the last  $N - (k + 1)$  steps is optimal, and applying Bellman's principle of optimality, the following relation is obtained:

$$m_{N-k}(x_k) = \min_{u_k} [(\hat{x}_{k+1} - x_{k+1})^\top Q_k (\hat{x}_{k+1} - x_{k+1}) + (\hat{u}_k - u_k)^\top R_k (\hat{u}_k - u_k) + m_{N-(k+1)}(x_{k+1})] \quad (3.3)$$

where  $k = 0, 1, 2, \dots, N - 1$ .

At  $k = N$ , we have  $m_0(x_N) = 0$ .

Let's assume that  $m_{N-k}(x_k)$  can be presented as

$$m_{N-k}(x_k) = x_k^\top P_{N-k} x_k - 2v_{N-k}^\top x_k + a_{N-k} \quad (3.4)$$

where  $P$  is symmetric positively semi-defined matrix,  $v$  is vector and  $a$  is a value independent on  $x$ .

Replacing in (3.3)  $m_{N-(k+1)}(x_{k+1})$  with expression (3.4) for  $(k + 1)$ -th step, getting

$$\begin{aligned} m_{N-k}(x_k) = \min_{u_k} [ &(\hat{x}_{k+1} - x_{k+1})^\top Q_k (\hat{x}_{k+1} - x_{k+1}) + (\hat{u}_k - u_k)^\top R_k (\hat{u}_k - u_k) + \\ &x_{k+1}^\top P_{N-(k+1)} x_{k+1} - 2v_{N-(k+1)}^\top x_{k+1} + a_{N-(k+1)}] \end{aligned} \quad (3.5)$$

The control sequence  $u_k$  is determined by substituting the expression for  $x_{k+1}$  from equation (3.1) into equation (3.5), and setting to zero the partial derivative with respect to  $u_k$  of the expression in square brackets in (3.5).

$$u_k = c_{N-k} - F_{N-k} x_k \quad (3.6)$$

where

$$\begin{aligned} W_{N-k} &= B_k^\top (Q_k + P_{N-(k+1)}) B_k + R_k \\ c_{N-k} &= W_{N-k}^{-1} [B_k^\top (Q_k \hat{x}_{k+1} + v_{N-(k+1)}) + R_k \hat{u}_k] \\ F_{N-k} &= W_{N-k}^{-1} [B_k^\top (Q_k + P_{N-(k+1)}) A_k] \end{aligned} \quad (3.7)$$

If matrix  $W$  is near-singular, then a product of the regularization parameter (as in Levenberg–Marquardt regularization) and the identity matrix may be added to it.

Substituting (3.6), (3.7) and (3.1) into (3.5), an expression for  $m_{N-k}(x_k)$  is derived in the form corresponding to (3.4), where

$$\begin{aligned} Z_k &= A_k - B_k F_{N-k} \\ P_{N-k} &= Z_k^\top (Q_k + P_{N-(k+1)}) Z_k + F_{N-k}^\top R_k F_{N-k} \\ v_{N-k} &= Z_k^\top (Q_k \hat{x}_{k+1} + v_{N-(k+1)}) - (Q_k + P_{N-(k+1)}) B_k c_{N-k} + F_{N-k}^\top R_k (c_{N-k} - \hat{u}_k) \end{aligned} \quad (3.8)$$

Expressions (3.6)–(3.8) define the optimal control of the open-loop system  $u_k$ , as illustrated in Figure 2, where  $c_{N-k}$  represents the control vector of the closed-loop system, and  $F_{N-k}$  is the feedback gain matrix applied to all coordinates of the state vector. Both of these parameters are generally time-dependent.

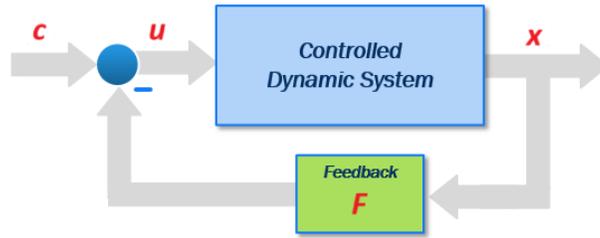


Figure 2: Closed-loop control system.

Theoretically, different combinations of  $c_{N-k}$  and  $F_{N-k}$  can yield the same values of  $u_k$  in equation (3.6). However, such variations can significantly affect the system’s response to unmodeled disturbances.

Clearly, not all coordinates of the state vector are directly measurable. The unmeasured components are estimated using various observers, such as the Kalman filter.

The computation process proceeds as follows: First, during the backward pass, the parameters  $c_{N-k}$  and  $F_{N-k}$  are computed using equations (3.7) and (3.8), starting from the final time step. Next, in the forward pass, the control vector  $u_k$  is computed using equation (3.6), and the corresponding state vector is then updated via the system dynamics described in equation (2.1). The overall procedure is illustrated in Figure 3.

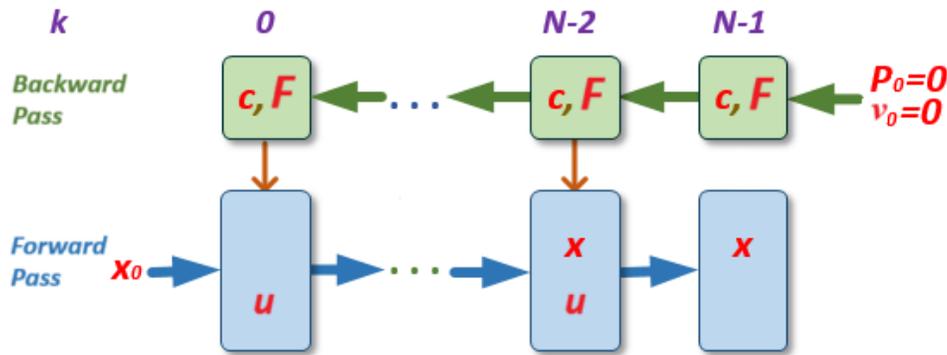


Figure 3: Computation schema.

## 4 Nonlinear Case

Now we extend the above result to the nonlinear systems. The main idea is using the above recurrent mechanism, to obtain control policy with iterations for linearized control object. For that matter we linearize discrete representation of (2.1)

$$x_{k+1} = f(k, x_k, u_k) \quad (4.1)$$

For  $i$ -th iteration (trajectory) the controlled object is presented in the following form:

$$x_{k+1}^{i+1} = A_k^i x_k^{i+1} + B_k^i u_k^{i+1} \quad (4.2)$$

where transfer matrices at the  $k$ -th point of  $i$ -th trajectory are given with the following Jacobians

$$\begin{aligned} A_k^i &= \nabla_x f(k, x_k^i, u_k^i) = \left. \frac{\partial f}{\partial x} \right|_{x_k^i, u_k^i}, \\ B_k^i &= \nabla_u f(k, x_k^i, u_k^i) = \left. \frac{\partial f}{\partial u} \right|_{x_k^i, u_k^i}. \end{aligned} \quad (4.3)$$

State and control vectors on  $(i + 1)$ -th trajectory may be presented as:

$$\begin{aligned} x_k^{i+1} &= x_k^i + \delta x_k^i \\ u_k^{i+1} &= u_k^i + \delta u_k^i \end{aligned} \quad (4.4)$$

Cost function (3.2) may be formulated for increments  $\delta x$  and  $\delta u$ . In this case, in the cost function (3.2), vectors of desirable values  $\hat{x}$  and  $\hat{u}$  should be replaced with new vectors  $\delta \hat{x}$  and  $\delta \hat{u}$  respectively:

$$\begin{aligned} \delta \hat{x}_{k+1}^{i+1} &= \hat{x}_{k+1} - x_{k+1}^i \\ \delta \hat{u}_k^{i+1} &= \hat{u}_k - u_k^i \end{aligned} \quad (4.5)$$

This allows us to reduce the nonlinear problem to a sequence of linear iterations.

## 5 Computational Algorithm for Nonlinear Case

The algorithm consists of the following steps:

1. Assume initial (for iteration 0) vectors  $x_k^0$  and  $u_k^0$  (normally all zero values except for initial conditions  $x_{k=0}^0$ , but the choice is up to a user).
2. Calculate linearized  $A_k^i$  and  $B_k^i$  for all  $k$  at the  $i$ -th trajectory according to (4.3). This can be performed either analytically, by deriving Jacobians, or numerically using finite differences or automatic differentiation.
3. Calculate new vectors  $\delta \hat{x}_{k+1}^i$  and  $\delta \hat{u}_k^i$  of desirable values for the  $i$ -th iteration according to (4.5).
4. Calculate  $\delta x_k^i$  and  $\delta u_k^i$  with inverse and direct runs as for the linear case, replacing in (3.6)–(3.8), and (3.1) actual and desirable state and control vectors with their respective increments:

$$x_k \Rightarrow \delta x_k^i, \quad u_k \Rightarrow \delta u_k^i, \quad \hat{x}_k \Rightarrow \delta \hat{x}_k^i, \quad \hat{u}_k \Rightarrow \delta \hat{u}_k^i.$$

Regularization may be applied in this step to ensure numerical stability.

5. Calculate  $x_k^{i+1}$  and  $u_k^{i+1}$  for  $(i + 1)$ -th iteration with (4.4).
6. Check the condition to stop iterations. This condition may be based, e.g., on the difference between values of the cost function compared to the previous iteration, or simply contain a fixed number of iterations. If the condition is not satisfied, then steps 2 - 6 should be repeated until the stop condition will be satisfied. After the condition was satisfied, policy  $u_k$  and states  $x_k$  is considered as the final solution.

It is important to note that even in the nonlinear case, when using a single iLQR iteration with constant desired state  $\hat{x}$ , control vector  $\hat{u}$ , and weight matrices  $Q$  and  $R$ , both the input vector  $c$  and the feedback matrix  $F$  in equation (3.6) remain constant. This observation enables the system designer to simplify, in some cases, the closed-loop control of nonlinear systems by reducing it to time-invariant input and feedback terms.

Several numerical examples illustrating different scenarios are presented below.

## 6 Numeric Examples

The quadratic optimization algorithm was implemented as a software service and tested on several canonical models of dynamic systems.

### 6.1 Rayleigh Oscillator

This dynamic system is described as following [2, 3].

Table 1: State and Control Variables

	#	Variable	Description
<b>State Vector</b>	1	$x_0$	Position
	2	$x_1$	Velocity
<b>Control Input</b>	1	$u$	Control input

Table 2: System Parameters and Equations

Symbol	Description
$a$	System nonlinearity parameter ( $a = 1.4$ )
$b$	Control gain ( $b = 4$ )
$\dot{x}_0$	$x_1$
$\dot{x}_1$	$-x_0 + a(1 - \frac{x_1^2}{10})x_1 + bu$

Initial state of the system is presented in the Table 3.

Table 3: Initial State Vector

Parameter	$x$ (m)	$v_x$ (m/s)
<b>Value</b>	-5	-5

The objective of the control in this case is to drive the system to the zero state with minimal control effort. Constant desired values of state and control and their weight factors are presented in the Table 4.

Table 4: Desired Values and Weights

Variable	$x_0$	$x_1$	$u$
<b>Desired Value</b>	0	0	0
<b>Weight</b>	1	0	1

Two cases of control design are shown below. In the left-hand side of Figure 4 a single iteration variant of control is presented. In this case the control may be implemented with constant negative feedbacks (Table 5) without input of close-loop system (since the steady state is zero). The three iterations case is presented in the right-hand side of Figure 4. Although the dynamics of the system state coordinates are similar in both cases, the initial control signal in the open-loop system with constant feedback gains is significantly higher—approximately 9 vs. 6 m/s<sup>2</sup>.

Table 5: The Single Iteration Case

Symbol	Value
$c$ (closed-loop input)	0
$F$ (feedback matrix)	$[0.7591 \quad 1.064]$

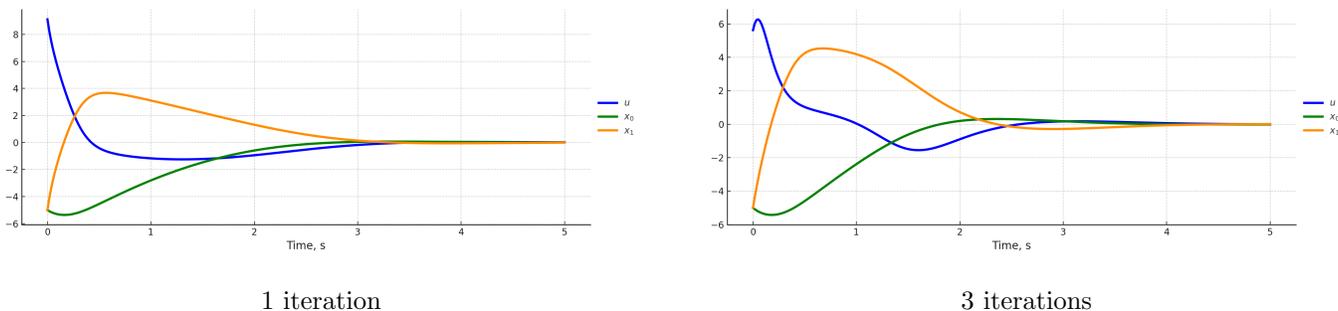


Figure 4: Comparison of Rayleigh system dynamics under different scenarios.

## 6.2 Inverted Pendulum on a Moving Cart

The inverted pendulum on a moving cart (Figure 5) is a classic nonlinear control system consisting of a pendulum hinged on a horizontally moving cart [4, 5]. The goal is to apply a horizontal force to the cart in order to stabilize the pendulum in its upright (inverted) position, despite its inherent instability. This system often serves as a benchmark problem for validating control algorithms in robotics, automation, and feedback system design.

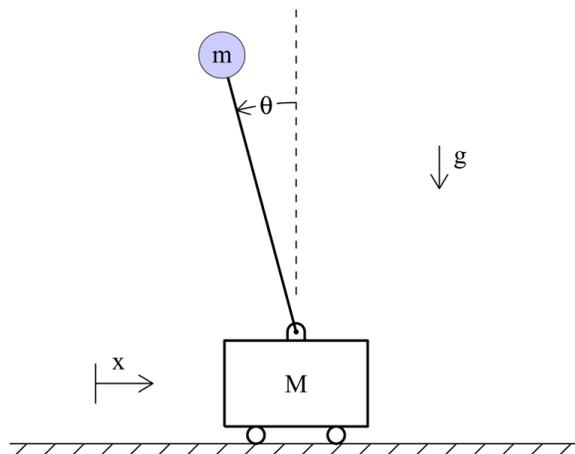


Figure 5: Schema of Inverted Pendulum on a Moving Cart [4].

Table 6: State and Control Variables

	#	Variable	Units	Description
<b>State Vector</b>	1	$x$	m	Cart position
	2	$\dot{x}$	m/s	Cart velocity
	3	$\theta$	$^\circ$	Pendulum angle from vertical
	4	$\dot{\theta}$	$^\circ/s$	Angular velocity of pendulum
<b>Control Input</b>	1	$u$	N	Horizontal force on the cart

Table 7: System Dynamics and Parameters

Symbol	Description
$g$	Acceleration due to gravity (9.81 m/s <sup>2</sup> )
$m$	Mass of the pendulum (1 kg)
$M$	Mass of the cart (1 kg)
$l$	Length to pendulum center of mass (1 m)
$\dot{x}$	Cart velocity
$\dot{\theta}$	Angular velocity of pendulum
$\ddot{x}$	$\frac{-ml\dot{\theta}^2 \sin \theta + mg \sin \theta \cos \theta + u}{M+m(1-\cos^2 \theta)}$
$\ddot{\theta}$	$\frac{-ml\dot{\theta}^2 \sin \theta \cos \theta + (M+m)g \sin \theta + u \cos \theta}{l(M+m(1-\cos^2 \theta))}$

The control objective is to shift the cart by 10 meters while stabilizing the pendulum in the upright position. The corresponding constant desired values for the state and control vectors, along with their weights, are provided in Table 8.

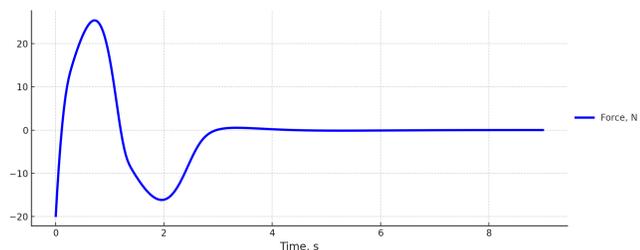
Table 8: Desired Values and Weights for Inverted Pendulum

Variable	$x$	$\dot{x}$	$\theta$	$\dot{\theta}$	$u$
<b>Desired Value</b>	10	0	0	0	0
<b>Weight</b>	100	1	1	1	10

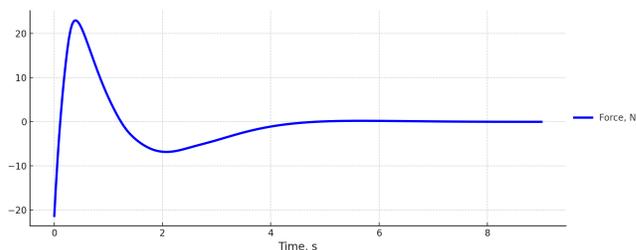
Figure 6 presents the control results using constant desired state, control inputs, and weight values for a single iteration (left column) and for three iterations (right column).

Table 9: The Single Iteration Case

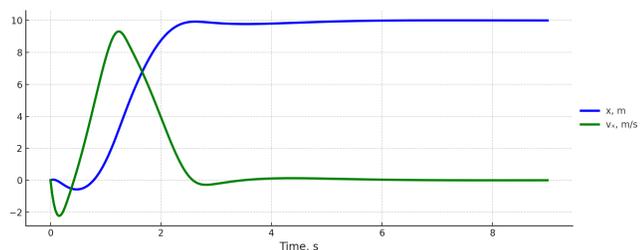
Symbol	Value
$c$ (closed-loop input)	-19.84
$F$ (feedback matrix)	$[-1.984 \quad -3.595 \quad 49.87 \quad 12.83]$



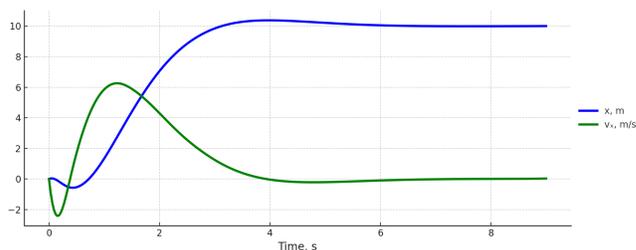
Force, 1 iteration



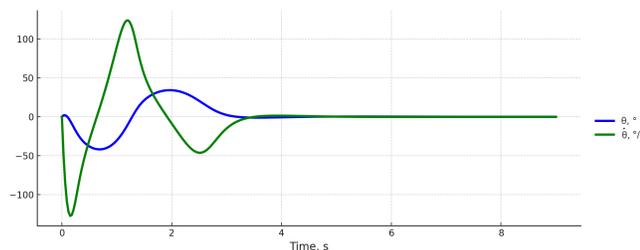
Force, 3 iterations



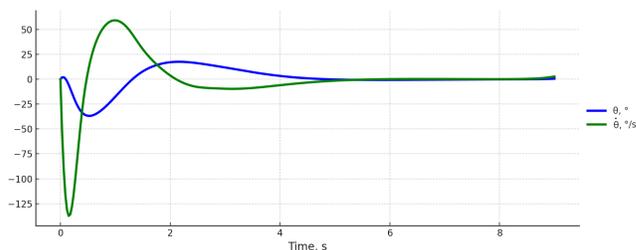
Position and velocity, 1 iteration



Position and velocity, 3 iterations



Angle and angular velocity, 1 iteration



Angle and angular velocity, 3 iterations

Figure 6: Comparison of pendulum system behavior under different scenarios.

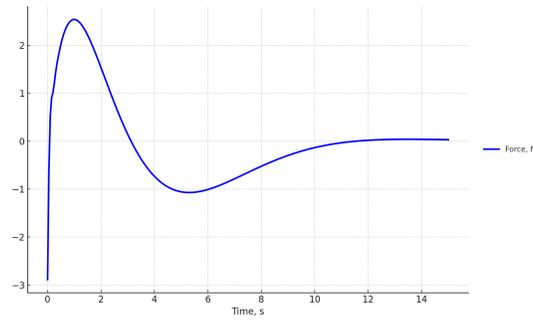
Another selection of weight gains, presented in Table 10, results in a softened control input (force). In this case, the transients (Figure 7) are significantly attenuated and require substantially more time to reach steady state. A single iteration is sufficient here, as the variations in  $\theta$  are relatively small and the system behaves almost linearly.

Table 10: Desired Values and Weights for Softened Control Input in the Inverted Pendulum System

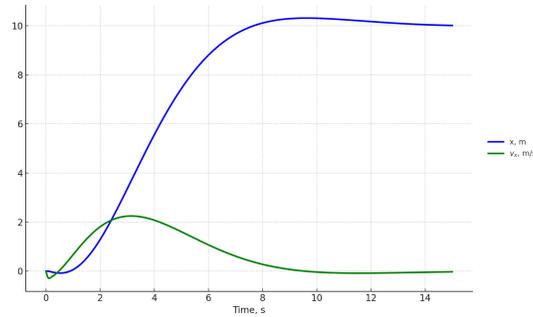
Variable	$x$	$\dot{x}$	$\theta$	$\dot{\theta}$	$u$
Desired Value	10	0	0	0	0
Weight	1	1	1000	1000	0

Table 11: The Single Iteration Case for Softened Control Input

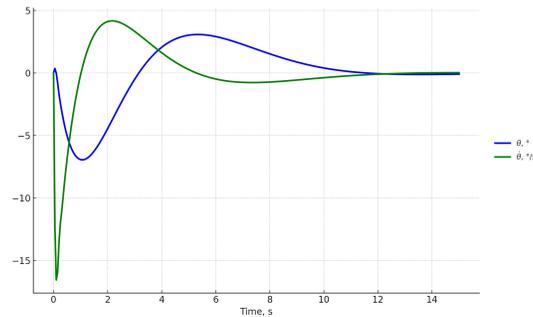
Symbol	Value
$c$ (closed-loop input)	-2.889
$F$ (feedback matrix)	$[-0.2889 \quad -1.103 \quad 38.36 \quad 12.86]$



Force, 1 iteration



Position and velocity, 1 iteration



Angle and angular velocity, 1 iteration

Figure 7: Dynamic response of the pendulum system under softened control input.

### 6.3 Two-Link Manipulator

We consider a simplified dynamic model of a two-link manipulator depicted in Figure 8 [6]. The state and control variables are summarized in Table 12, while the corresponding mathematical model is presented in Table 13.

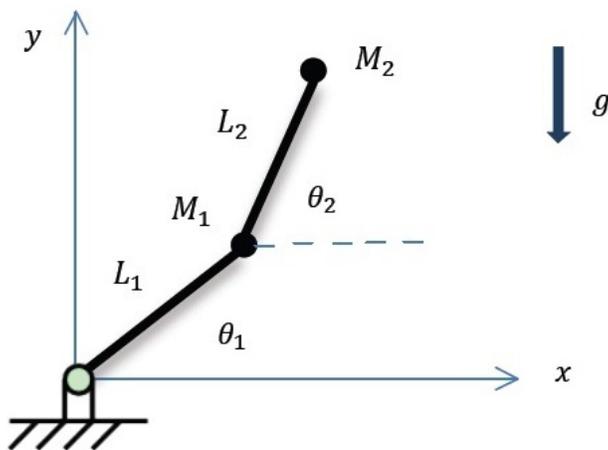


Figure 8: Schema of Two-Link Manipulator.

Table 12: State and Control Vector for Two-Link Manipulator

	#	Variable	Units	Description
<b>State Vector</b>	1	$\theta_1$	rad	Angle of link 1
	2	$\dot{\theta}_1$	rad/s	Angular velocity of link 1
	3	$\theta_2$	rad	Angle of link 2
	4	$\dot{\theta}_2$	rad/s	Angular velocity of link 2
<b>Control Inputs</b>	1	$T_1$	Nm	Torque at joint 1
	2	$T_2$	Nm	Torque at joint 2

Table 13: Two-Link Manipulator Dynamic Parameters and Equations

Symbol	Description
$g$	Gravity constant $9.81m/s^2$
$M_1, M_2$	Masses of links (each 1 kg)
$L_1, L_2$	Lengths of links (each 1 m)
$m$	$M_2/(M_1 + M_2)$
$\dot{\theta}_1$	Angular velocity of link 1
$\dot{\omega}_1$	$\frac{mT_1/(M_2L_1) - mL_2\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g \cos(\theta_1) - m \cos(\theta_1 - \theta_2)(T_2/(M_2L_2) + L_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g \cos(\theta_2))}{L_1(1 - m \cos^2(\theta_1 - \theta_2))}$
$\dot{\theta}_2$	Angular velocity of link 2
$\dot{\omega}_2$	$\frac{T_2/(M_2L_2) + L_1\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - g \cos(\theta_2) - \cos(\theta_1 - \theta_2)(mT_1/(M_2L_1) - mL_2\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - g \cos(\theta_1))}{L_2(1 - m \cos^2(\theta_1 - \theta_2))}$

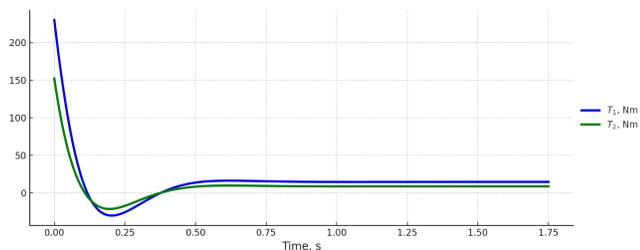
The objective is to compute an optimal control policy that drives the manipulator from zero initial conditions to the final configuration with  $\theta_1 = \pi/4$  and  $\theta_2 = \pi/6$  radians. The corresponding desired state and control values, along with their weights, are provided in Table 14.

Table 14: Desired Values and Weights (States vs Inputs)

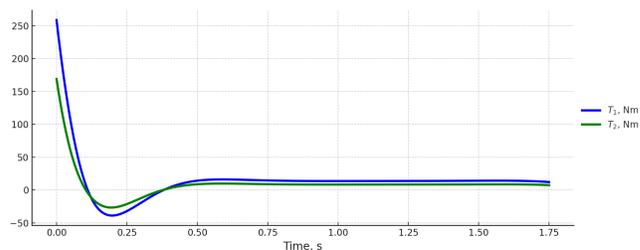
Variable	$\theta_1$	$\dot{\theta}_1$	$\theta_2$	$\dot{\theta}_2$	$T_1$	$T_2$
<b>Desired Value</b>	$\pi/4$	0	$\pi/6$	0	0	0
<b>Weight</b>	100000	0	100000	0	1	1

Table 15: The Single Iteration Case

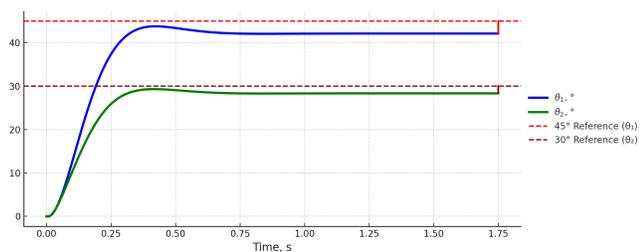
Symbol	Value
$c$ (closed-loop input)	$\begin{bmatrix} 230.3 \\ 152.3 \end{bmatrix}$
$F$ (feedback matrix)	$\begin{bmatrix} 283.0 & 33.64 & 15.54 & 11.29 \\ 15.53 & 11.29 & 267.4 & 22.35 \end{bmatrix}$



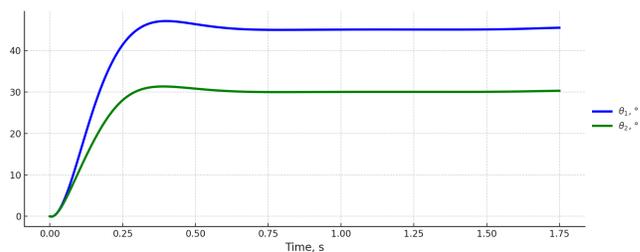
Joint torques  $T_1, T_2$ , 1 iteration



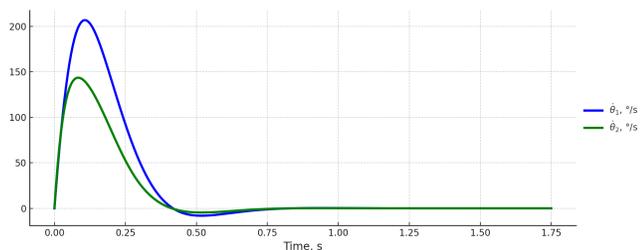
Joint torques  $T_1, T_2$ , 3 iterations



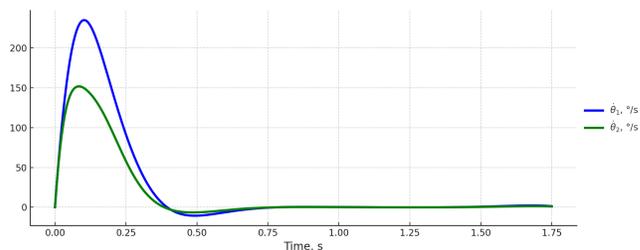
Joint angles  $\theta_1, \theta_2$ , 1 iteration. Reference deviations.



Joint angles  $\theta_1, \theta_2$ , 3 iterations.



Angular velocities  $\dot{\theta}_1, \dot{\theta}_2$ , 1 iteration.



Angular velocities  $\dot{\theta}_1, \dot{\theta}_2$ , 3 iterations.

Figure 9: Two-Link Manipulator control response comparisons.

It can be observed that under similar settings, the single-iteration case results in a steady-state deviation of the joint angles from their reference values. Although this deviation can be corrected through parameter tuning, doing so complicates the controller design process. In contrast, when three iterations are used, the system achieves zero steady-state error.

## 6.4 Quadcopter

We consider a simplified dynamic model of a quadcopter, illustrated in Figure 10 [7, 8], which includes integrators in the control input channels to enhance control smoothness. The associated state and control variables are listed in Table 16, and the corresponding dynamic equations are detailed in Table 17. The numerical values of the model parameters used in the simulations are provided in Table 18.

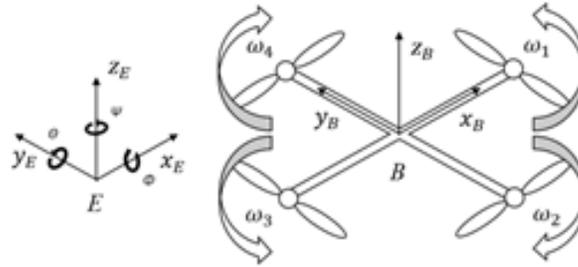


Figure 10: Schema of Quadcopter.

Table 16: State and Control Vectors for the Quadcopter

	#	Variable	Units	Description
<b>State Vector</b>	1	$\phi$	rad	Roll angle
	2	$\dot{\phi}$	rad/s	Roll rate
	3	$\theta$	rad	Pitch angle
	4	$\dot{\theta}$	rad/s	Pitch rate
	5	$\psi$	rad	Yaw angle
	6	$\dot{\psi}$	rad/s	Yaw rate
	7	$x$	m	Position along $x$ -axis
	8	$v_x$	m/s	Velocity along $x$ -axis
	9	$y$	m	Position along $y$ -axis
	10	$v_y$	m/s	Velocity along $y$ -axis
	11	$z$	m	Position along $z$ -axis
	12	$v_z$	m/s	Velocity along $z$ -axis
	13	$T$	N	Total thrust
	14	$\tau_\phi$	Nm	Torque around $x$ -axis (roll)
	15	$\tau_\theta$	Nm	Torque around $y$ -axis (pitch)
	16	$\tau_\psi$	Nm	Torque around $z$ -axis (yaw)
<b>Control Inputs</b>	1	$u_T$	N/s	Input to thrust integrator
	2	$u_\phi$	Nm/s	Input to $\tau_\phi$ integrator
	3	$u_\theta$	Nm/s	Input to $\tau_\theta$ integrator
	4	$u_\psi$	Nm/s	Input to $\tau_\psi$ integrator

Table 17: Quadcopter Dynamic Parameters and Equations

Category	Symbol	Units	Description
Auxiliary Coefficients	$a_1 = \frac{J_y - J_z}{J_x}$	-	Gyroscopic coefficient (roll)
	$a_2 = \frac{J_z - J_x}{J_y}$	-	Gyroscopic coefficient (pitch)
	$a_3 = \frac{J_x - J_y}{J_z}$	-	Gyroscopic coefficient (yaw)
	$b_1 = \frac{l}{J_x}$	m/kg·m <sup>2</sup>	Input gain (roll)
	$b_2 = \frac{l}{J_y}$	m/kg·m <sup>2</sup>	Input gain (pitch)
	$b_3 = \frac{l}{J_z}$	m/kg·m <sup>2</sup>	Input gain (yaw)
Rotational Dynamics	$\dot{\phi} = \omega_\phi$	rad/s	Angular velocity around $x$ -axis
	$\dot{\omega}_\phi = a_1 \omega_\theta \omega_\psi + b_1 \tau_\phi$	rad/s <sup>2</sup>	Angular accel. (roll)
	$\dot{\theta} = \omega_\theta$	rad/s	Angular velocity around $y$ -axis
	$\dot{\omega}_\theta = a_2 \omega_\phi \omega_\psi + b_2 \tau_\theta$	rad/s <sup>2</sup>	Angular accel. (pitch)
	$\dot{\psi} = \omega_\psi$	rad/s	Angular velocity around $z$ -axis
	$\dot{\omega}_\psi = a_3 \omega_\phi \omega_\theta + b_3 \tau_\psi$	rad/s <sup>2</sup>	Angular accel. (yaw)
Translational Dynamics	$\dot{x} = v_x$	m/s	Velocity along $x$ -axis
	$\dot{v}_x = \frac{T}{m} (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi)$	m/s <sup>2</sup>	Acceleration along $x$
	$\dot{y} = v_y$	m/s	Velocity along $y$ -axis
	$\dot{v}_y = \frac{T}{m} (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi)$	m/s <sup>2</sup>	Acceleration along $y$
	$\dot{z} = v_z$	m/s	Velocity along $z$ -axis
	$\dot{v}_z = \frac{T}{m} \cos \phi \cos \theta - g$	m/s <sup>2</sup>	Acceleration along $z$
Control Dynamics	$\dot{T} = u_T$	N/s	Thrust control rate
	$\dot{\tau}_\phi = u_\phi$	Nm/s	Torque input (roll)
	$\dot{\tau}_\theta = u_\theta$	Nm/s	Torque input (pitch)
	$\dot{\tau}_\psi = u_\psi$	Nm/s	Torque input (yaw)

Table 18: Model Parameters

Parameter	Value	Units	Description
$g$	9.81	m/s <sup>2</sup>	Gravitational acceleration
$l$	0.23	m	Arm length
$m$	0.65	kg	Mass
$J_x$	$7.5 \cdot 10^{-3}$	kg · m <sup>2</sup>	Moment of inertia about $x$ -axis
$J_y$	$7.5 \cdot 10^{-3}$	kg · m <sup>2</sup>	Moment of inertia about $y$ -axis
$J_z$	$1.3 \cdot 10^{-2}$	kg · m <sup>2</sup>	Moment of inertia about $z$ -axis

The objective is to compute an optimal control policy that stabilizes the quadcopter (i.e., ensures hovering), starting from the initial conditions specified in Table 19.

Table 19: Initial State Vector  $x_0$ 

Parameter	$\phi$ (rad)	$\dot{\phi}$ (rad/s)	$\theta$ (rad)	$\dot{\theta}$ (rad/s)	$\psi$ (rad)	$\dot{\psi}$ (rad/s)	$x$ (m)	$v_x$ (m/s)
<b>Value</b>	0.3	1	-0.4	1	0.2	1	0	1
Parameter	$y$ (m)	$v_y$ (m/s)	$z$ (m)	$v_z$ (m/s)	$T$ (N)	$\tau_\phi$ (Nm)	$\tau_\theta$ (Nm)	$\tau_\psi$ (Nm)
<b>Value</b>	0	1	0	-1	0	0	0	0

The computational results after five iterations, using the desired state and control values along with their corresponding weights from Table 20, are presented in Figure 11.

Table 20: Desired Values and Weights

Variable	$\phi$	$\dot{\phi}$	$\theta$	$\dot{\theta}$	$\psi$	$\dot{\psi}$	$x$	$v_x$	$y$	$v_y$	$z$	$v_z$	$T$	$\tau_\phi$	$\tau_\theta$	$\tau_\psi$	$u_T$	$u_\phi$	$u_\theta$	$u_\psi$	
Desired Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Weight	10	1	10	1	10	1	10	1	10	1	50	5	0	0	0	0	0.5	0.5	0.5	0.5	

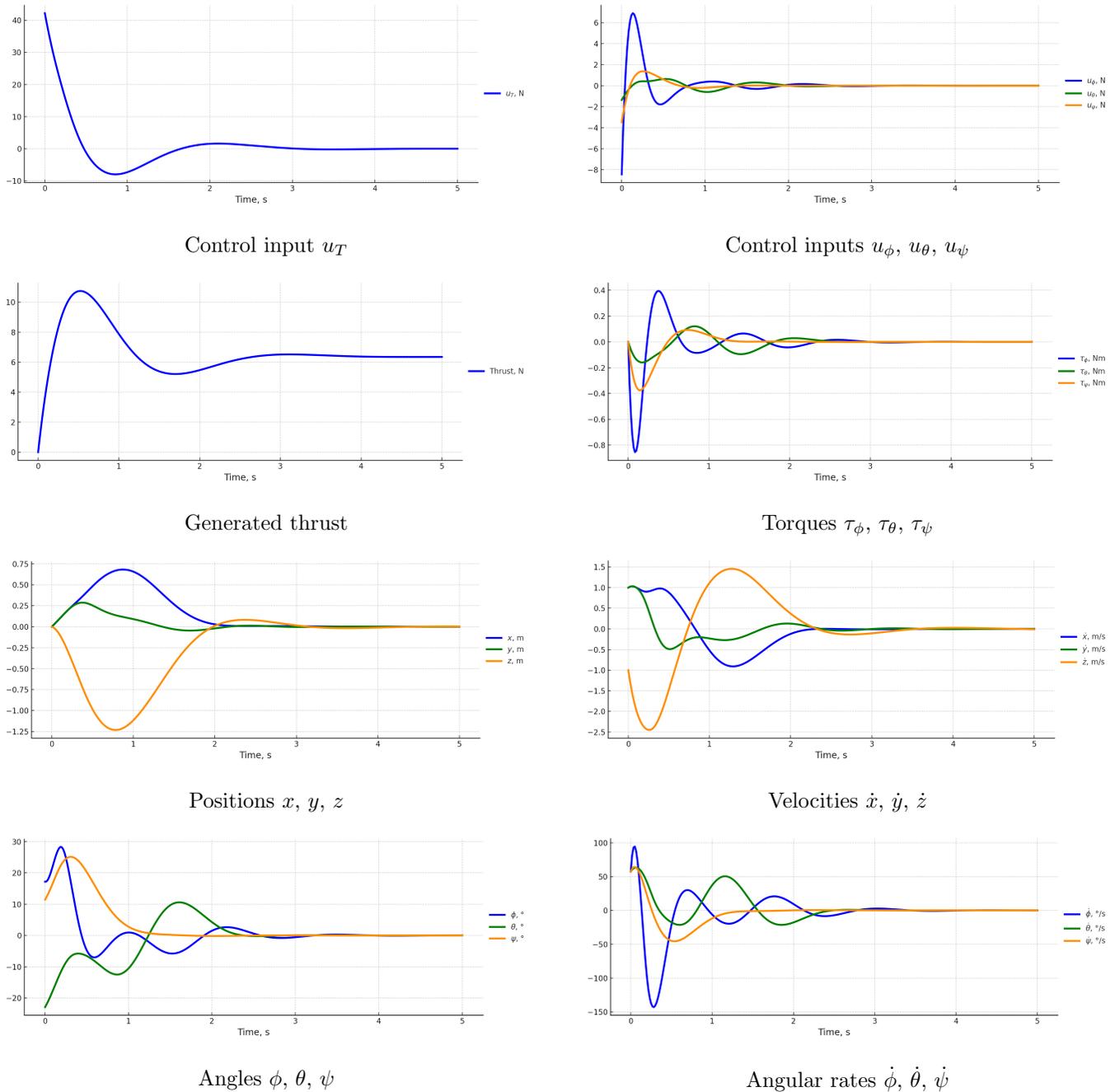


Figure 11: Quadcopter control and state evolution: inputs, forces, torques, position, velocity, and orientation.

A single iteration, which yields constant input and feedback, is insufficient to transfer the quadcopter to the hovering position. With two iterations, the system reaches a hovering state, though with some deviation from the initial location. Increasing the number of iterations results in a solution that ensures final hovering precisely at the initial point.

## Conclusions

This paper presented a unified framework for solving quadratic optimal control problems using both classical LQR and the iterative LQR (iLQR) algorithm. The approach is applicable to linear and nonlinear systems alike, leveraging discrete-time dynamic programming and system linearization.

We demonstrated the effectiveness of the iLQR approach through detailed simulations involving Rayleigh oscillator, an inverted pendulum, a two-link manipulator, and a quadcopter model. The results confirm the method's tracking accuracy under nonlinear conditions.

The modular structure of iLQR makes it well-suited for integration with Model Predictive Control (MPC) architectures, further enhancing its utility in real-time, constrained, or adaptive control environments. Future work may explore real-world implementation scenarios, especially for robotics and aerial vehicle control.

## References

- [1] Li, W. & Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pp. 222–229.
- [2] J. Liang. (2003). Lecture notes for ece/mae 7360, robust and optimal control (part 2). Technical report, Utah State University at Logan.
- [3] Rayleigh Unconstrained
- [4] E. Burdet. (2010). Inverted Pendulum on a Cart: Equations of Motion.
- [5] B. L. Widjiantoro & M. K. Wafi. (2023). Discrete-time state-feedback controller with canonical form on inverted pendulum (on a cart).
- [6] Mahboub Baccouch. (2012). A two-link manipulator simulation and control design.
- [7] Bouabdallah, S., Murrieri, P., & Siegwart, R. (2004). Design and control of an indoor micro quadrotor. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*.
- [8] Shi Lu. (2018). Modeling, Control and Design of a Quadrotor Platform for Indoor Environments.