EE160 Project (Fall 2024)

November 13, 2024

1 Introduction

The EE160 Project is about the modeling, analysis, design and simulation of a control system of your choice (including but not limited to mechanical, circuit, thermodynamic system). Here, you have two options:

1. Pick at least one of the application examples (with given physical models) from the appendix of this project sheet;
2. Propose a control problem of your own choice. If you want to go for the second option, please contact Prof. Chen. We will accept this as a project topic as long as the project is related to the methods that are discussed in the lecture.

You should form your team of no more than 3 members and submit your initial report before Dec. 22th, independent of whether you go for one of the default project or come up with your own project, you will need to complete a final report ($\geq$ 4 pages) and a presentation. And your presentation should consist of several slides as well as on-site simulation display about 10 minutes per group. The goal of this project is to practice how to use control method such as root locus method, bode analysis to analyze your system and design a feedback controller for a practical application. Moreover, you will learn how to analyze the performance of the the closed-loop system, then show the effectiveness of your design.

2 Main requirements on the initial report

The main tasks of the initial report are as follows:

1. **You should clarify your team member and the control problem you choose.**
2. Modeling the system in the form of

$$\dot{x}(t) = f(t, x(t), u(t)),$$

Explain what x, u, and f are in your application and define your preliminary control goal. Calculate the transfer function and find the time domain solution of the system

3 Main requirements on the final report

1. Model the system in both Time and Frequency domain.

2. Analyze the performance of the open-loop system.
3. Design a feedback controller or use the controller given in each appendix and analyze the performance of the closed-loop system.
4. Analyze the stability of the system and show the performance with simulation. Verify the robustness, sensitivity, relative stability and other performance of the system.

4 Some specifications for the report

Write a short report (preferably in Latex) containing the following sections:

1. Title and Authors (find a good title + name of the author)
2. Introduction (describe the problem that you want to solve and cite relevant literature)
3. Problem Formulation (introduce a suitable mathematical notation to define the problem that you are trying to solve)
4. Open-loop performance analysis (use the method in class to describe the performance)
5. Controller Design and closed-loop performance analysis (explain how you design your controller and why the controller is able to achieve your control objective.)
6. Numerical Results (plot/visualize and explain your numerical results)
7. Conclusion (analyze and summarize the highlights of your results)

Other requirements:

1. The length should be at least 4 single-column pages with 10pt font. Be brief.
2. The contribution/work of both students involved should be clearly stated
3. **No plagiarism or self-plagiarism. The student is never allowed to reuse his/her own published papers as the final project.**
4. **The citations should indicate references.** (IEEE format is recommended)

5 Presentation

Every group will get a $15 \sim 20$ minute time slot to present your project. The presentation should consist of at least 5 slides, preferably in pdf format. In order to "pass", you should:

1. prepare clean and efficient slides in English, contains
   - the project title and the name of all authors
   - introducing/explaining the problem that you are solving
   - introducing your design logic for the controller
   - the numerical results showing the effectiveness of your control system
   - one conclusion slide summarizing and assessing your results

   Do not use small fonts!
2. present your work freely in English or Chinese by using your own words.

3. answer questions by the Lecturer, TAs, or other people from the audience.

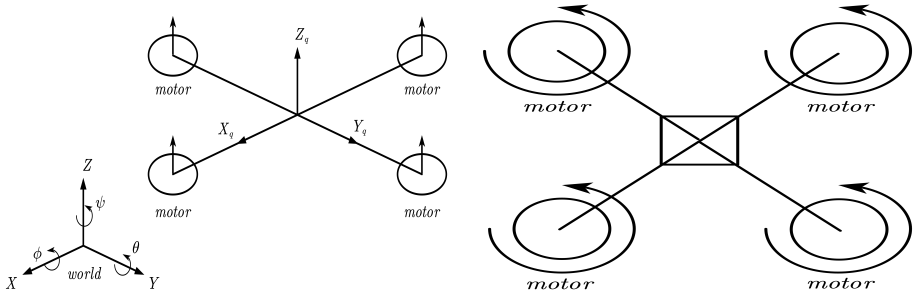
Appendix

This appendix collects a few suggestions for possible project topics and some hints on how to derive a differential equation model. For each proposal, we have listed a few basic control objectives and related simulation requirements. If you propose your own project, please contact the lecturer to discuss about task details.

A Quadrotor Control (Supervisor: Wu, Bo)

A.1 Introduction

A quadrotor is a mainly flying robot that composed of motor, electric control, battery, propeller, frame, remote control (manual control) and flight control. The dynamics of quadrotor mainly includes position dynamics and attitude dynamics. For position dynamics, we have three states p_x, p_y, p_z representing the three-dimensional coordinates of the quadrotor in the world coordinate system, and three other states v_x, v_y, v_z representing the velocities in each direction. And for attitude dynamics, we have 3 more states for angular velocity of the quadrotor which are $\omega_x, \omega_y, \omega_z$ and ϕ, θ, ψ are the additional 3 states for the Euler angle.



(a) Coordinate system (b) Quadrotor model

Figure 1: Quadrotor

A.2 Simplified Model

To simplify the model, this topic only discusses the tilt control of the quadrotor along one of the supporting axes,and does not consider the altitude, rotation, and tilt of the quadrotor along the other axis, nor the disturbance caused by the rotor on the other axis. The control objective is regulate the quadrotor bank angle to zero degrees (steady, wings level) and maintain the wings-level orientation in the presence of unpredictable external disturbances.

For our purposes, a simplified dynamic model is required for the quadrotor design process. A simplified model might consist of a transfer function describing the input-output relationship between the Motor voltage and the quadrotor bank angle. By making the simplifying assumptions and linearizing about the steady, wings-level flight condition, we can obtain a transfer function model describing the bank angle output, $C(s)$, to the motor voltage input, $U(s)$,

$$G(s) = \frac{C(s)}{U(s)} = \frac{2s + 0.1}{s(s^2 + 0.1s + 4)}, \tag{1}$$

where $U(s)$ is the output of the select key controller/compensator $G_c(s)$ to be designed. The system configuration is shown in figure:

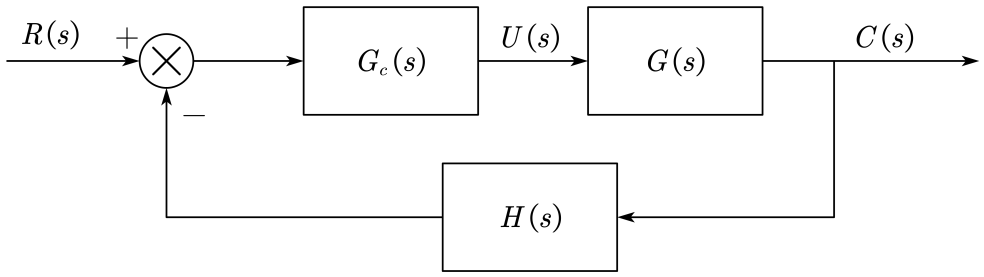


Figure 2: Quadrotor attitude control system

Requirement:

1. The controller we select for this question is a proportional controller $G_c(s) = K$. Sketch and analyze the root locus for open-loop system. Choose an appropriate K, sketch and analyze the unit step response, Bode plot and Nyquist plot.(Assumed that $H(s) = 1$)

2. We want to make the overshoot and the peak time are as small as possible and want to eliminate the oscillation mode. Try to design an appropriate compensator $G_c(s)$ such that the closed-loop system has a pair of dominant poles whose location can be chosen by yourself properly. Then we consider there exists a disturbance $d(t) = t$ added between $G_c(s)$ and $G(s)$, verify whether the controller can eliminate the influence of the disturbance n, if not, redesign your controller to satisfy the above requirements at the same time. Write the form of $G_c(s)$ and give the closed-loop transfer function. Describe the root locus of the compensated system, and use Matlab to test the transient response characteristics of the system.

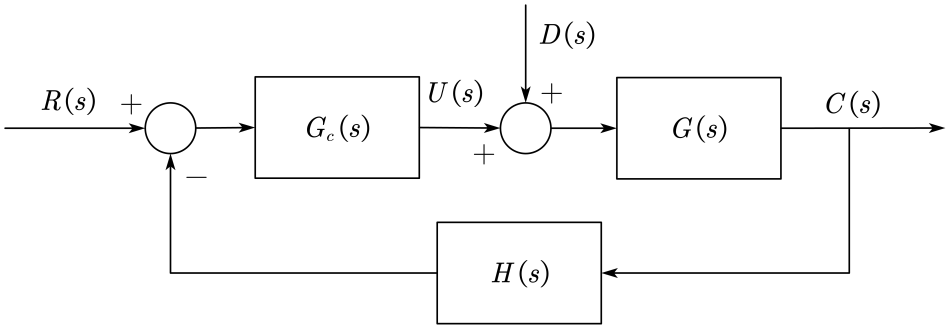


Figure 3: Quadrotor attitude control system with disturbance

A.3 Compelte Model

Now, you have controlled one of the tilt angles of the quodrator, now we can give a complete model. The position dynamics model of quadrotor is given as follows:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ -mg\end{bmatrix}+RT_B=\begin{bmatrix}0\\ 0\\ -mg\end{bmatrix}+R\begin{bmatrix}0\\ 0\\ f\end{bmatrix} \tag{2}$$

R and T_B are as follows:

$$R=\begin{bmatrix}\cos(\theta)\cos(\psi) & \cos(\psi)\sin(\theta)\sin(\phi)-\sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi)+\sin(\psi)\sin(\phi)\\ \cos(\theta)\cos(\psi) & \sin(\psi)\sin(\theta)\sin(\phi)+\cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi)-\cos(\psi)\sin(\phi)\\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta)\end{bmatrix}. \tag{3}$$

where $p=[x,y,z]^T\in\mathbb{R}^3$ is the position of the quadrotor, R is the rotation matrix from the body frame to the inertial frame, ϕ,θ,ψ represent the attitude angles roll, pitch, and yaw. f is the total thrust on the quadrotor. m is the mass of the quadrotor and g is acceleration of gravity.

The attitude dynamics model of quadrotor is given as follows:

$$I\dot{\omega}+\omega\times(I\omega)=\tau\Rightarrow\begin{bmatrix}\dot{\omega}_x\\ \dot{\omega}_y\\ \dot{\omega}_z\end{bmatrix}=\begin{bmatrix}\tau_x{I_{xx}}^{-1}\\ \tau_y{I_{yy}}^{-1}\\ \tau_z{I_{zz}}^{-1}\end{bmatrix}+\begin{bmatrix}\frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z\\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z\\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y\end{bmatrix}, \tag{4}$$

where $\omega=[\omega_x,\omega_y,\omega_z]^T$ is the angular velocity vector of the quadrotor. I_{xx},I_{yy},I_{zz} represent the moment of inertia of the three axes. τ_x,τ_y,τ_z represent the torques generated on three axes.

In order to convert angular velocities $\begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ into the angular velocity vector $\omega=[\omega_x,\omega_y,\omega_z]^T$, we can use the following relation:

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}=\begin{bmatrix}1 & 0 & -\sin(\theta)\\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi)\\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi)\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=R_\omega\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}, \tag{5}$$

and we can also convert angular velocity vector into angular velocities by calculating the R_ω^{-1}. The angular velocity of the each propeller are $\omega_1,\omega_2,\omega_3,\omega_4$, which are also the input of the system. The torque τ_x,τ_y,τ_z and force f are defined as follows:

$$\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\\ f\end{bmatrix}=\begin{bmatrix}\frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T\\ \frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T & -\frac{\sqrt{2}}{2}dc_T\\ c_M & -c_M & c_M & -c_M\\ \frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T & \frac{\sqrt{2}}{2}dc_T\end{bmatrix}\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix}, \tag{6}$$

where d is shaft length of the quadrotor, c_T,c_M are tension coefficient and torque coefficient. The values of system parameters are summarized in Table 2.

Symbol	Value
c_T	1.11×10^{-5}
c_M	1.56×10^{-7}
d	0.225
m	1.4
g	9.8
I_{xx}	0.021
I_{yy}	0.022
I_{zz}	0.37

Table 1: Parameters of complete quadrotor

Requirement: try to find a proper control law to drive quadrotor is to hover at a fixed position under all sorts of initial conditions.

B Active Suspension Control (Supervisor: Wang, Yifan ; Wu,Bo)

For road vehicle users, comfort is an important issue. Automotive suspension systems are intended to absorb and decrease the shocks and vibrations transferred from the ground to the passengers as well as the vehicle body. The suspension system shown below represents the vehicle system at each wheel. It consists of a spring k_s, a damper b_s and a hydraulic actuator F_a. The tire stiffness and damping properties are also shown by k_t and b_t, respectively. The effective vehicle body mass is shown by M_s (spring mass), and M_u (unsprung mass) represents the effective mass for the wheel and axle. The vertical displacements from the static equilibrium for Mu and Ms are shown by x_u and x_s, respectively. The road profile is represented by x_r. The suspension travel x_s - $x_u$is measured and compared to the set point ($r = 0$). The required actuator force is determined by the controller to eliminate the error, and thus, to reduce the vehicle oscillations. The actuator can provide a maximum force of 3000 N.

Q1.Establish a state-space model for the system with the x_r and F_a as the system input, and x_s and x_u as the system outputs.

Q2.Simulate system on various road profiles, plot x_r, x_s, x_u and $x_s - x_u$ over time. When PID controller is not activated, a constant $F_a = 1500$ N is applied)

PS: Using sinusoidal bumpy with amplitude is 5 cm (max), frequency is 1 rad/s to simulate uneven road surfaces.

Using sinusoidal bumpy with amplitude is 5 cm (max), frequency is 1 rad/s with a switch to simulate pot-hole road profile.

Q3. Design a PID Controller for F_a: Tuning PID parameters to achieve desired characteristics:

(1) Response time: 2.00 s $\sim$ 2.03 s

(2) Robustness: 0.6

Then plot x_r, x_s, x_u and $x_s - x_u$ over time. Compare with the performance of the system without PID control.

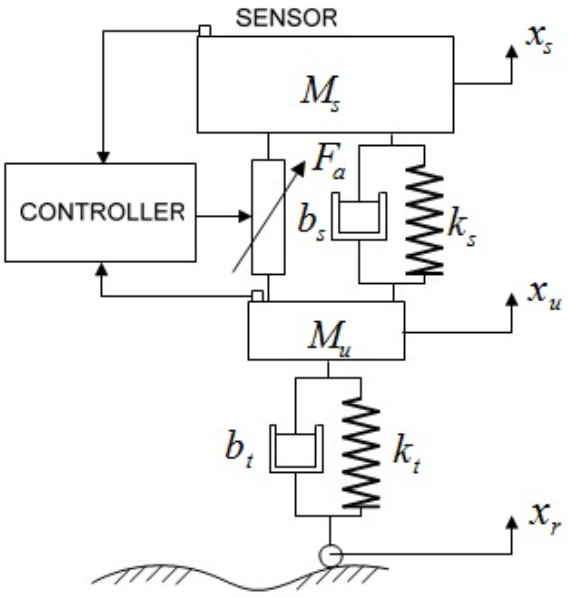


Figure 4: Active suspension model

Symbol	Value
M_s	300
M_u	50
k_s	18000
k_t	180000
b_s	1200
b_t	0

Table 2: Parameters of active suspension

C DC-DC Converter Control (Supervisor: Xie, Yuhao)

C.1 Introduction

DC-DC converter is an electronic circuit that converts a direct current (DC) source from one voltage level to another with very little energy loss. In this section, we will discuss the closed-loop control design of buck converter. It is a simple circuit shown in the figure.

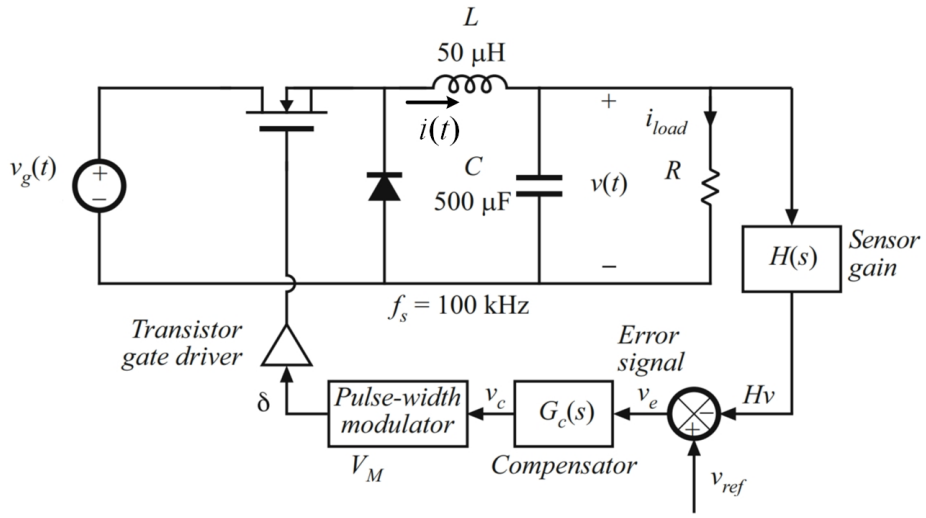


Figure 5: buck converter and control loop

This project is recommended for students interested in circuits or power electronics. If you are interested in DC-DC converter or have questions, you can read chapters 2,7,8,9 of the reference book (Genuine download: *https://link.springer.com/book/10.1007/978-3-030-43881-4*)

C.2 Model and Parameters

The converter implements the conversion by switching on and off at high frequencies, d is the duty cycle that commanded by controller that $0 \leq d \leq 1$. vg is the source voltage and v is the output voltage, and i **is the current of inductance**. The average model of the circuit is provided:

$$L(\frac{di(t)}{dt}) = d(t) * vg(t) - v(t) \tag{7}$$

$$C(\frac{dv(t)}{dt}) = i(t) - \frac{v(t)}{R} \tag{8}$$

Tips: 1. It is recommended to solve the DC component first 2. The second-order terms are nonlinear and negligible.
The model of the control loop is :

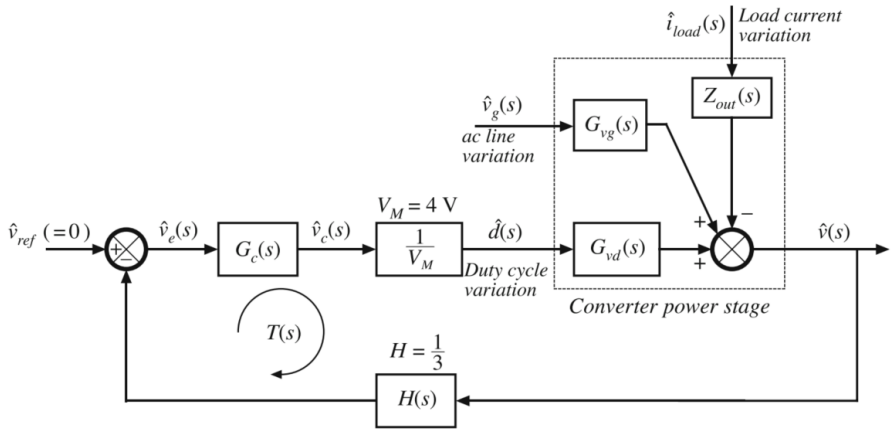


Figure 6: small signal block diagram

Parameters: The output voltage is expected to be 15 V and the load is set to 3 Ω.

Symbol	Value
V_g	27 V
V_{ref}	5 V
V_M	4 V
L	50 μH
C	500 μF

C.3 Requirement

1. Derive the linearized state space model of buck circuit. Realize time-domain simulation of open-loop system by codes or circuit simulation software.
2. Calculate the open-loop response $G_{vd}(s)$ and draw the bode diagram.
3. Calculate the loop gain $T(s)$ without compensator, and draw the bode diagram of the system. Analyze the performance in frequency domain, such as stability, DC gain and phase margin.
4. Design a compensator to improve the performance of the closed-loop system. Analyze the performance of in frequency domain, and compare it with the previous. Finally, try to demonstrate the step response of your system (step signal of v_{ref}).
5. (optional) Test the closed-loop system's response to the fluctuations of voltage source v_g. Analyze its gain $G_{vg}(s)$

D *2D humanoid robot walking (Supervisor: Zhu, Junlei)

In this part, we will implement a program to make a 2D humanoid robot walk. The human shape is simplified, divided into the upper body, left and right thighs, left and right calves, a total of five heel links. The simplified model is shown in Fig 7.

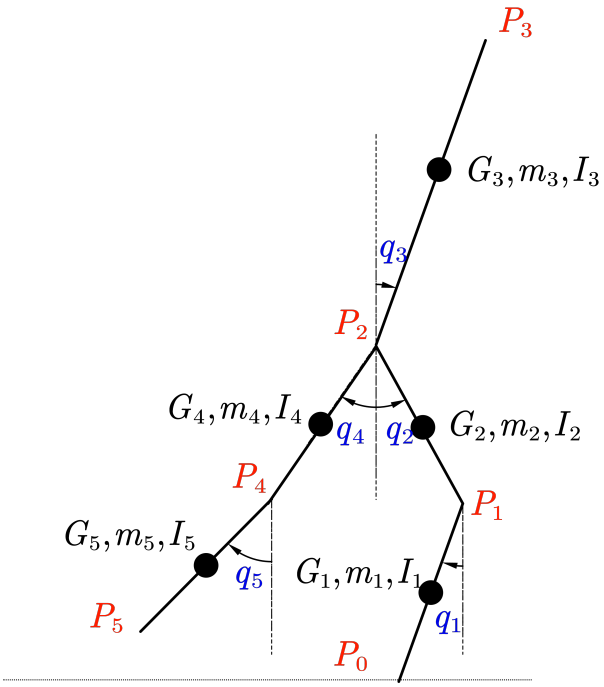


Figure 7: Robot Dynamics Model

D.1 Introduction

We focus on exploring the walking strategy of a biped robot in a two-dimensional world (the xz plane), ignoring the dynamic influences in the lateral (y-axis) direction. Therefore, we simplify the robot model to a two-dimensional representation for ease of analysis and computation.

We divide the walking control of the two-dimensional robot into two parts. The first part is the generation of the optimal trajectory [1], where the trajectory includes the positions and orientations that the robot needs to achieve at each joint during each time interval. The second part is trajectory tracking using the simulation software Drake [2]. Fortunately, we do not need to concern ourselves with generating the trajectories; all trajectories have been preprocessed and are provided in a Python-readable file.

This project requires participants to have a certain level of programming and analytical ability. The experiment needs to be conducted under a Linux system (or WSL, the Windows Subsystem for Linux), so participants are also expected to have a basic understanding of Linux.

D.2 Derivation of dynamic model

Based on Kelly's tutorial on dynamic modeling [3], we will derive the dynamics of a robot model consisting of five links. As shown in Figure 7, we define the joint positions P_i to identify the coordinates of each joint node, and use G_i to represent the positions of the centers of mass of each link. The angle vector $q = q_1\ q_2\ q_3\ q_4\ q_5]^T$ is defined to represent the angles between the links and the vertical direction, with the positive direction as shown in the figure.

In the analysis of the robot's walking motion, we distinguish between the supporting leg and the swinging leg, where the toe position of the supporting leg is denoted as P_0. During walking, the robot's joint positions P and centers of mass G can be expressed in terms of the joint angles q: $P_i = P_i(q)$, $G_i = G_i(q)$. To accurately

describe the dynamic behavior of the robot, the joint angular velocities are defined as $\dot{q} = [\dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ \dot{q}_5]^T$. Based on these angular velocities, the velocities and accelerations of the joints and centers of mass can be calculated using the following mathematical formulas:

$$\begin{aligned} \dot{P}_i &= \left(\frac{\partial P_i}{\partial q}\right)\dot{q}, \qquad \dot{G}_i = \left(\frac{\partial G_i}{\partial q}\right)\dot{q}, \\ \ddot{P}_i &= \left(\frac{\partial \dot{P}_i}{\partial z}\right)\dot{z}, \qquad \ddot{G}_i = \left(\frac{\partial \dot{G}_i}{\partial z}\right)\dot{z}. \end{aligned} \tag{9}$$

We represent the robot's dynamic model as a series of dynamic equations based on D'Alembert's principle. We first define the state vector $\dot{z} = [q\ \dot{q}]^T$, and then express the behavior of the system through the following dynamic equation:

$$M(q)\ddot{q} = F(q, \dot{q}, u), \tag{10}$$

where $M(q)$ is the mass matrix of the robot joints, and $F(q, \dot{q}, u)$ is a function that includes external forces, control inputs, and other dynamic effects.

For the five-link robot system, we establish the following five dynamic equations separately:

$$\begin{aligned} u_1 + \hat{k}\cdot\sum_{i=1}^{5}\left((G_i - P_0)\times(-m_i g\hat{j})\right) &= \hat{k}\cdot\sum_{i=1}^{5}\left((G_i - P_0)\times(m_i\ddot{G}_i) + \ddot{q}_i I_i\hat{k}\right), \\ u_2 + \hat{k}\cdot\sum_{i=2}^{5}\left((G_i - P_1)\times(-m_i\ddot{G}_i)\right) &= \hat{k}\cdot\sum_{i=2}^{5}\left((G_i - P_1)\times(m_i\ddot{G}_i) + \ddot{q}_i I_i\hat{k}\right), \\ u_3 + \hat{k}\cdot\sum_{i=3}^{5}\left((G_i - P_2)\times(-m_i\ddot{G}_i)\right) &= \hat{k}\cdot\sum_{i=3}^{5}\left((G_i - P_2)\times(m_i\ddot{G}_i) + \ddot{q}_i I_i\hat{k}\right), \\ u_4 + \hat{k}\cdot\sum_{i=4}^{5}\left((G_i - P_3)\times(-m_i\ddot{G}_i)\right) &= \hat{k}\cdot\sum_{i=4}^{5}\left((G_i - P_3)\times(m_i\ddot{G}_i) + \ddot{q}_i I_i\hat{k}\right), \\ u_5 + \hat{k}\cdot\left((G_5 - P_4)\times(-m_5\ddot{G}_5)\right) &= \hat{k}\cdot\left((G_5 - P_4)\times(m_5\ddot{G}_5) + \ddot{q}_5 I_5\hat{k}\right). \end{aligned} \tag{11}$$

In these equations, u_i represents the control torque at the i-th joint, serving as the system input and optimization objective; m_i is the mass of link i, and I_i is its moment of inertia.

Additionally, to simulate the transition between the supporting leg and the swinging leg, we introduce a transformation matrix T_{change}, which defines the mapping from the joint angles before the transition q^- to the joint angles after the transition q^+:

$$q^+ = T_{\text{change}} q^- \tag{12}$$

$$T_{\text{change}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

Optimization Objective

To achieve efficient and smooth walking of the robot, this study adopts a trajectory optimization method. The optimization goal is to minimize the integral of the square of the torque u, thereby producing smooth and efficient motion solutions. The objective function is defined as follows:

$$J = \int_0^T \left(\sum_{i=1}^{5} u_i^2(t)\right) dt \tag{14}$$

where T represents a complete walking cycle, and $u_i(t)$ is the torque at the i-th joint at time t.

Constraints

The constraints set during the trajectory optimization process mainly include:

- System State Continuity Constraint: It requires that after completing one walking cycle, the system's state should remain consistent with the initial state. We define the system state as $x = [q\ \dot{q}]^T$, where the initial state x_0 and the final state x_f are related through a transition function T_b, which considers the exchange between the supporting leg and the swinging leg and the change in velocity when the swinging leg contacts the ground.
- Ground Contact Constraint: The supporting leg must remain in contact with the ground throughout the entire walking cycle, i.e., at any time $P_0(t) = [0\ 0]^T$.
- Target Position Constraint: At the end of the cycle, the swinging leg should reach the predetermined walking distance D, and its toe needs to contact the ground, i.e., $P_5(T) = [D\ 0]^T$.

These constraints are processed using the OptimTraj software to obtain the optimal trajectory. In this study, we set the walking distance per step to 0.5 meters and the walking cycle to 0.7 seconds. Through this optimization process, we obtained the trajectories of the robot's joint angles and angular velocities.

The modeling data is as follows, in python you can directly import the urdf file to load the model.

Table 3: Robot Hardware Parameters

Link Name	Torso	Left Thigh	Left Shank	Right Thigh	Right Shank
Link Length $[m]$	0.625	0.4	0.4	0.4	0.4
Distance(Mass, Axis of Rotation) $[m]$	0.200	0.163	0.128	0.163	0.128
Moment of Inertia $[\mathrm{kg} \cdot \mathrm{m}^2]$	2.22	1.08	0.93	1.08	0.93
Link Mass [kg]	20.0	6.8	3.2	6.8	3.2

We will provide the URDF file; however, you are welcome to create it yourself if you prefer.

D.3 Requirement

D.3.1 Designing a PD Controller for Robot Walking and Conducting In-Depth Analysis of Controller Parameters

To enable the robot to perform stable and accurate walking motions, it is imperative to design a Proportional-Derivative (PD) controller for each joint. Initially, we need to conduct a detailed analysis and tuning of the controller's proportional (P) and derivative (D) parameters to optimize the system's response speed and stability.

D.3.2 PID vs. PD

Analyze the impact of incorporating integral control into a Proportional-Derivative (PD) controller to form a Proportional-Integral-Derivative (PID) controller for robotic joint control. Compare the performance of PID control with PD control.

D.3.3 Employing Phase Plane Analysis and Introducing Poincaré Maps

To gain a deeper understanding of the robot's dynamic behavior during walking, we will utilize phase plane analysis and introduce the concept of Poincaré maps. This method aids in studying the system's

periodicity and stability, allowing us to determine the evolution of the system's state over each walking cycle. By referencing relevant coursework from the University of Arizona, we can effectively apply these theoretical tools to analyze the robot's walking dynamics.

E Integral Pre-Filter (Supervisor: Wu, Bo)

In practical serve(motor) system control, the encoder is the most used sensor to have the accurate position of the rotor. If we want to implement a speed control of the motor, ACTUALLY, a differential operation is adopted to rebuild the motor speed signal based on the position signal by encoder. The classical dual speed loop control structure of motor is shown in Figure 8[4].

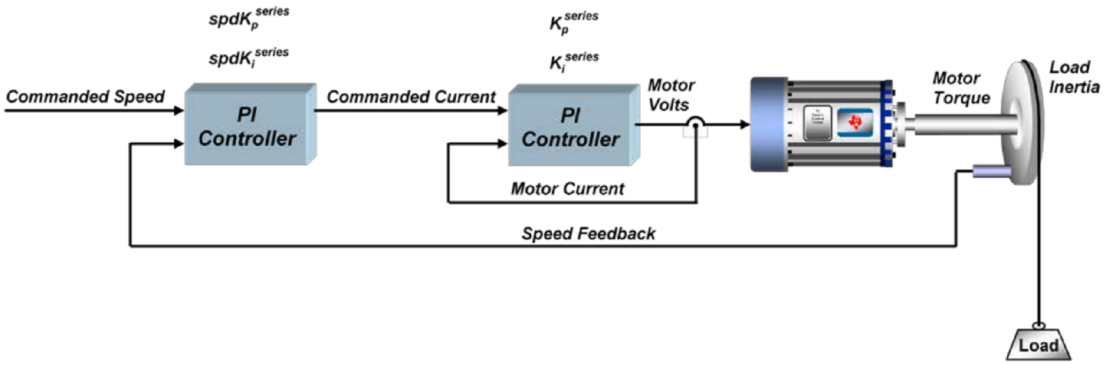


Figure 8: Classical Dual Speed Loop Control Structure

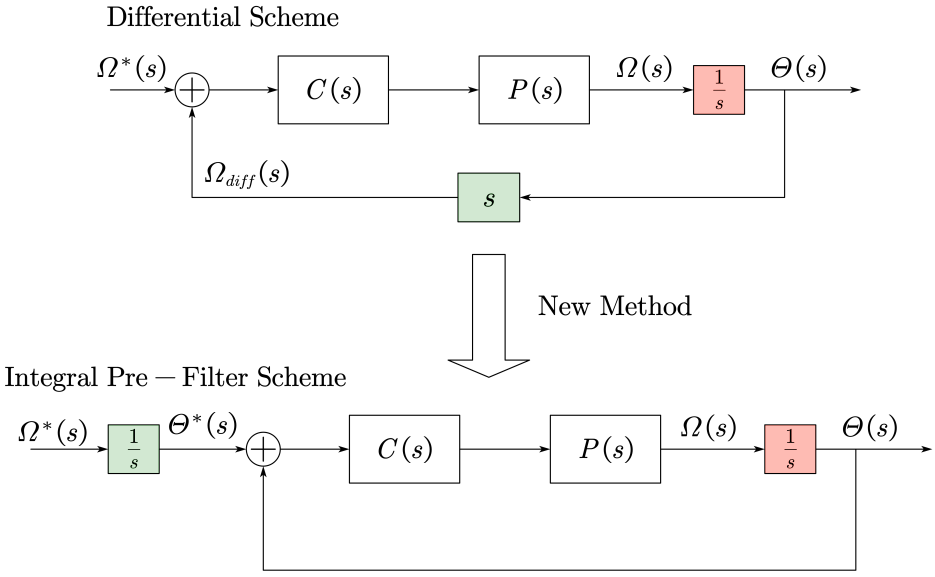


Figure 9: The Two Control Structures

But the differential operation is quite sensitivity to lots of things, for example, the noise of the original signal and quantization error[5]. While a integral pre-filter is able to avoid the differential operation and is

still able for the speed regulation of motors via encoder, the comparison between the integral pre-filter control scheme and classical differential operation scheme is shown in Figure 9.

Noticing that the green part is the main different thing of these two control schemes. $C(s)$ is the controller and $P(s)$ is the control plant, Ω_* is the motor speed reference, Ω is the motor speed and Θ is the motor position. The encoder here is considered as 1 and we do not show it as a block in the Figure 9.

E.1 Question

You can start with the simple first order plant $\frac{1}{Ts+1}$ and proportional controller K_P, discuss the advantage and disadvantage of the Integral Pre-Filter scheme compared to the classical one.

Then what about a PI controller or PID controller and what about a second order plant? Even you can try to add a nonlinearity to the controller like controller output saturation.

Hint: You are required to adopt discrete differential method like Euler Method[**?**], and add some noise and quantization error to the feedback loop. And integral pre-filter is also desired to be done in discretization.

F Adaptive Proportional Control (Supervisor: Wu, Bo)

F.1 Introduction

Adaptive control is a advanced control method to deal with the parameter uncertainty of the control plant. Basically, classical adaptive control evolves the idea of Model Identification, while the Gradient-Descent is the most common way to do model identification of the control plant. More detail you can check the Chapter 3 and 5 in [6].

A classical adaptive proportional control method is implemented[7] for a first-order system $\frac{1}{s+1}$with an unmodeled dynamic showed as $\frac{229}{s^2+30s+229}$ to indicate that the first order nonlinear simple proportional controller is enough for such system unless the bound of proportional K is carefully chosen.

The idea behind this adaptive control law is simple. First, notice that y_p is actually the error of the your system, do not mix up with the system real output which is usually denoted as C or Y in EE160. It is intuitive to tune the K based on the y_p, large error means we need to make K grows fast enough to control the output catch the reference as quick as possible while the error becomes smaller, K needs also to change mildly. And when K hits a maximum value, the controller is desired to stop working. The above can be concluded as (15), same as (5), (6) in [7].

$$\begin{aligned} u_p(t) &= -K(t)y_p(t) \\ \dot{K}(t) &= \begin{cases} y_p^2(t)\gamma & K < K_{\max} \\ 0 & K \geq K_{\max} \end{cases} \end{aligned} \tag{15}$$

F.2 Question

1. You are required to replicate the adaptive control method mentioned in [7] for a first-order system with an unmodeled dynamic. including the root locus analysis and time-domain response.

2. Feel free to share any thoughts when you finish the Q1.

3. Discuss the results in the paper and show advantages and disadvantages of the proposed control law

4. This control law is a classical one and mentioned as a beginner example in many modern adaptive control textbooks. For more detail information you can check Chapter 5.2 in [6].

G Koopman Lifting for Passive Walker

G.1 Introduction

How do you control bipedal robots to walk? This is a relatively hard question because controllers need to handle the nonlinear dynamics of robots while dealing with discontinuous from breaking and making contact, which require modern techniques such as optimal control and reinforcement learning. However, these methods are a little bit away from what we have learned in EE160, which mainly focus on linear systems and controllers. Thanks to Koopman lifting, we can somehow treat a walker as a linear system with higher dimensions than original system. In this project, you will analyze the stability of a passive walker by rewriting it in a linear system and then using what you have learned (e.g. poles, zeros, root locus, etc.) to get the conclusion. Similar methods can be applied not only to robot systems but also to human walking or other nonlinear behaviors of nature. Notice that koopman lifting is a popular research topic in control and robotics communities. Therefore, it is hard and new (at least at MIT, this is an extended part of a graduated-level course), and you may not get enough assistance from either the TA or lecturer. On the other hand, if you happen to find something interesting, that may be publishable. Last word, *We choose to go to the moon not because they are easy, but because they are hard; because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept.*

G.2 Potential Problems

There is no requirement you need to actually reach to get pass (because of the difficulty), but I will provide some problems that I think are interesting. **Notice if you don't like walking, find other systems you like (e.g., quadrotor, yo-yo). The key of this project is koopman lifting.**

1. How does a passive walker work? (I will mention passive walker in the material list.) Can you present a manipulation equation for this?
2. Find a local linear model. What can and cannot you get from it?
3. What is koopman lifting? Derive the equation yourselves; you need to make your hand dirty!
4. Why can we find a global linearization of a nonlinear system? What's the consequence of doing that? Intuitive understanding is enough.
5. Can you control it? Why you can or cannot?
6. Is it possible to get gait characteristics through Kooman lifting? How?

G.3 Materials

You may need to read the following list first to have a big picture.

1. You may need to watch Harry Asada's talk: Click here.
2. You may want to know something about passive walker: Click here.
3. You can find more information about walking: Click here.
4. You may need to read a few papers: Click here.

You can ask Prof. Chen or me for more information. One small suggestion: you should search more information online or find examples on GitHub. Others's codes are fine, but your thoughts should be your own.

References

[1] M. P. Kelly, "OptimTraj: Trajectory Optimization for Matlab," 12 2022. [Online]. Available: https://github.com/MatthewPeterKelly/OptimTraj

[2] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[3] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[4] T. INSTRUMENTS, "Instaspin-foc and instaspin-motion user's guide," TEXAS INSTRUMENTS, Tech. Rep.

[5] T. Wang, Z. Zhu, N. M. A. Freire, Z. Wu, M. Foster, and D. A. Stone, "Study on noise and disturbance issues of generalized predictive speed control for permanent magnet synchronous machines," vol. 15, no. 1, pp. 63–78.

[6] P. Ioannou and B. Fidan, *Adaptive control tutorial*. SIAM.

[7] I. Bar-Kana, "Rohrs examples and robustness of simple adaptive control," in *IEEE International Conference on Systems Engineering*. IEEE.