Legged Robots Practicals – Week 2

Contents

1 Introduction

In the first weeks we will explore the two-link pendulum to control a quadruped leg. Please refer to the slides for additional details. In the first week we derived the kinematics and dynamics of the double pendulum. This week we will use the Jacobian and inverse kinematics to control a quadruped leg in simulation.

2 Code Installation

The code is hosted at `https://gitlab.epfl.ch/lgevers/lr-practicals`. The `README.md` gives details on installation and code structure. The repository contains the code for all the practicals of week 1-3. In this session you will only work on `practical2_jacobian.py` and `practical3_ik.py`. This week we will start using the `pybullet` physics engine. It's a good idea to have it installed and running this session as we will be using `pybullet` throughout the rest of the practicals.

3 Theory Review

3.1 Modeling

The quadruped leg we consider in `pybullet` has 2 joints (thigh, calf). Refer to the slides for a visualization and the coordinate systems. In many cases, we will be interested in the location of the foot in the leg frame. As an example, let's consider the position of the foot at a "neutral" position (i.e., the foot is directly under the hip) when standing at 0.3 m. The foot position in the leg frame is then $(x, z) = (0, -0.3)$ m.

Forward kinematics: recall from the first practical, the relationship between the joint angles $\boldsymbol{q}$ and foot position $\boldsymbol{p}$:

$$\boldsymbol{p} = f(\boldsymbol{q}) \tag{1}$$

where the function $f(\cdot)$ denotes the forward kinematics of a single leg with respect to the leg frame.

Inverse kinematics: Determining this inverse relationship maps a desired foot position back to desired joint angles:

$$\boldsymbol{q} = f^{-1}(\boldsymbol{p}) \tag{2}$$

For general robotic systems this relationship will not be unique (i.e., there are multiple sets of joint configurations $\boldsymbol{q}$ that satisfy $\boldsymbol{p} = f(\boldsymbol{q})$). For the quadruped leg, we will use two different methods to solve for each solution.

Foot linear velocity: Differentiating the forward kinematics relationship gives the foot velocity $\boldsymbol{v}$:

$$\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3}$$

where $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{2\times 2}$ is the single-leg Jacobian of the foot position w.r.t. the leg frame.

Force relationship: The Jacobian can also be used to map a desired force $\boldsymbol{F}$ at the end effector (foot) to joint torques:

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{F} \tag{4}$$

These relationships will be used repeatedly throughout this course.

3.2 Control

Last week we merely simulated a double pendulum. This week and throughout the course, you will implement and use controllers in both joint and Cartesian space. We first start with describing a typical controller for position control in joint space:

$$\boldsymbol{\tau}_{\text{joint}} = \boldsymbol{K}_{p,joint}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_{d,joint}(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) \tag{5}$$

where $\boldsymbol{q}_d$ are the desired joint angles, $\dot{\boldsymbol{q}}_d$ are the desired joint velocities, and $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the current joint angles and joint velocities, respectively. $\boldsymbol{K}_{p,joint}$ and $\boldsymbol{K}_{d,joint}$ are vectors of proportional and derivative gains. Please note the dimensions of each of these vectors.

In order to control our system in the world, we cannot directly plan in joint space! (Why?) Cartesian PD controllers can be used to directly control our system in the world frame, for example usually used in model-predictive control approaches for legged systems. Consider planning a foot trajectory ($\boldsymbol{p}_d$) (and foot velocity ($\boldsymbol{v}_d$)) over some time-based array of indices $K \in \{0...N\}$. How can we track such a trajectory? Given the current foot position ($\boldsymbol{p}$) and foot velocity ($\boldsymbol{v}$), the corresponding motor torques to track the desired quantities can be computed with:

$$\boldsymbol{\tau}_{\text{Cartesian}} = \boldsymbol{J}^T(\boldsymbol{q})\left[\boldsymbol{K}_{p,Cartesian}(\boldsymbol{p}_d - \boldsymbol{p}) + \boldsymbol{K}_{d,Cartesian}(\boldsymbol{v}_d - \boldsymbol{v})\right] \tag{6}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the foot Jacobian at joint configuration $\boldsymbol{q}$, and $\boldsymbol{K}_{p,Cartesian}$ and $\boldsymbol{K}_{d,Cartesian}$ are diagonal matrices of proportional and derivative gains. Recall also that the Jacobian can be used to map a desired force $\boldsymbol{F}$ at the end effector (foot) to joint torques:

$$\boldsymbol{\tau}_{\text{FF}} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{F} \tag{7}$$

4 Assignment: Jacobian

Fill in `practical2_jacobian.py` to control the foot position of a quadruped leg using the Jacobian. Explore the following questions:

1. Begin with the `on_rack` variable equal to true, to suspend the leg in the air. Fill in the code blocks to compute the absolute (double pendulum) and relative (pybullet's coordinate system) Jacobians, and use a Cartesian PD controller to keep the foot at $(x, z) = (0, -0.3)$ m.

2. Try changing the gain matrices $\boldsymbol{K}_{p,Cartesian}$ and $\boldsymbol{K}_{d,Cartesian}$. How important are these? What happens if you set `on_rack` to false - can you use the same gains? How does this affect the foot position? How about when dragging the leg up and letting it fall? It will be helpful to plot the foot position/velocity, joint positions/velocities, and motor torques.

3. With `on_rack` set to False, now try to apply force control by compensating for gravity and the mass of the system (use `env.robot.total_mass`, and be careful with the sign). What are the minimum gains you can use?

4. Plan a trajectory in task space and track it with the Cartesian PD controller. For example, decrease the hip height from 0.3 m to 0.2 m and vice versa over different time windows. Would you be able to plan such a trajectory in joint space?

5. What benefits do you see from using impedance control (i.e. Cartesian PD + force control)? What could be improved? (i.e. what if a particular joint configuration is required, for example if the leg should look like < or >. Try starting with different initial angle configurations.)

5 Assignment: Inverse Kinematics

Fill in both analytical and iterative inverse kinematics methods for a two-link quadruped leg in `practical3_ik.py`. Subsequently use the computed inverse kinematics to control the leg in joint space. Note that this relies on the correct implementation of `jacobian_rel` from the previous assignment. Explore the following questions:

1. **Analytical IK**: For the same desired end effector position, can you select two different knee angles?

2. **Iterative IK**: How can we find both sets of solutions with the algorithm (i.e. leg looking like < or >)?

3. What benefits do you see from tracking task space trajectories with inverse kinematics (and joint PD control) vs. impedance control (with Cartesian PD + force control)? When might you use one over the other? What if a particular joint configuration is required, for example if the leg should look like < or >? What about motor-related considerations (torque sensing, bandwidth)?