Rheinische Friedrich-Wilhelms-Universität Bonn
Prof. Dr. Maren Bennewitz

Institut für Informatik
Abteilung VI
Humanoid Robots Lab
Adresse:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

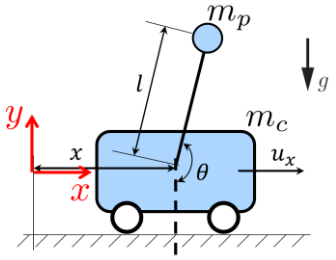
Humanoid Robotics

Assignment 10

Due Thursday, July 10th, before class.

Legged Robots Locomotion

1. LQR Control: Cart-Pole Robot **(Total: 6 points)**
In the lecture, we discussed linearization techniques for nonlinear dynamical systems. Our main example was the **cart-pole** model — a classic benchmark for underactuated control.



a. What are the key limitations of applying linear control methods to nonlinear systems like the cart-pole? How do LQR limitations affect the performance of the LQR controller in the cart-pole example? (1 point)

b. Derive and implement **nonlinear dynamics** of the cart-pole (2 points)
In the lecture, we derived the nonlinear equations of motion for the cart-pole system using the Lagrangian formulation. The system is defined by the configuration vector: $q = \begin{bmatrix} x \\ \theta \end{bmatrix}$, where *"x"* is the horizontal position of the cart, and "θ" is the angle of the pendulum. The control input is the force "u_x" applied to the cart.
In the provided Python code, complete the missing functions to numerically compute the dynamics of the system. These dynamics will be used to simulate the robot's behavior under feedback control.

c. Linearize the system dynamics (1 points)
Complete the function to linearize the nonlinear dynamics of the cart-pole system around the **upright** equilibrium point $(x = 0,\ \theta = \pi)$. This linearized model will be used to design an LQR controller to stabilize the pendulum in the upright position.

d. Visualize the results (2 points)
Once your simulation and controller are working:
- Plot the **control input** over time.
- Plot the **pendulum angle** over time.

Use separate subplots or figures for clarity.

Briefly discuss how the choice of cost matrices "*Q*" and "*R*" in the LQR design affects the performance. Specifically, how different weights influence control effort, convergence speed, and stability?

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institut für Informatik
Abteilung VI
Humanoid Robots Lab

2. Legged Robots Modeling **(Total: 3 points)**

In the class, we discussed the dynamics of legged robots. By making a set of simplifying assumptions, we were able to derive a simplified model for humanoid walking, known as the Linear Inverted Pendulum Model (LIPM).

a. Why is it acceptable in the LIPM with dynamic walking to let the projection of the COM move outside the support polygon, while requiring the ZMP to stay inside? (1 point)

b. Why is the assumption of constant angular momentum ($\dot{L} = 0$) important in deriving the ZMP equation, and what kind of robot motion would violate this assumption? (1 point)

c. What is the physical interpretation of the equation $c_x - \frac{c_z}{g}\ddot{c}_x = z_x$? How does this tell us about the direction of COM acceleration in relation to ZMP? (1 point)

3. Motion Planning **(Total: 6 points)**

In the lecture, we discussed **ZMP-based motion planning and control**, which is a fundamental technique for generating dynamically feasible walking motions for humanoid robots. This exercise focuses on implementing a **ZMP-based Model Predictive Controller** (**MPC**) based on **Section (II)** of the following paper:

Pierre-Brice Wieber, "**Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations**," 2007.

In this method, the Linear Inverted Pendulum Model (LIPM) is used to simplify the dynamics of the robot's Center of Mass (CoM), and the Zero Moment Point (ZMP) serves as a stability criterion. The main idea is to generate a sequence of jerk inputs that steer the CoM while keeping the ZMP within a predefined support region.

Task 3.a) Why do we use Model Predictive Control (MPC) to track a desired ZMP trajectory instead of a simpler control method (e.g., PD control)? What advantage does MPC provide in the context of dynamic walking stability?" (1 point)

ZMP-based MPC

In the following tasks, we will implement an MPC controller in Python. Using a set of predefined footstep locations, we will apply the controller to generate a feasible CoM motion for humanoid walking. The MPC controller is formulated as an optimization problem that minimizes a cost function subject to system dynamics and (optional) ZMP constraints.

Task 3.b) Complete the \`build_dynamics()\` function to construct the discrete-time state-space model. (2 point)

Task 3.c) Implement the \`run_mpc()\` function to formulate and solve the MPC problem with a cost function and constraints. You may use cvxpy, or any QP solver you're comfortable with. (2 point)

Task 3.d) Plot the evolution of calculated ZMP and CoM along with the footstep locations over time. Based on the plot briefly discuss how their evolution relates to the convex hull of the supporting feet. Is it acceptable for the CoM to leave the convex hull? Why? (1 point)