Robot Learning

Winter Semester 2020/2021, Homework 1

Prof. Dr. J. Peters, J. Watson, J. Carvalho, J. Urain and T. Dam

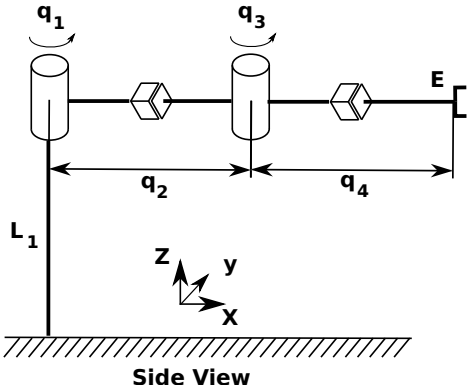
TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 43
Due date: Midnight, Tuesday, 01 December 2020

Name, Surname, ID Number

Problem 1.1 Robotics in a Nutshell [12 Points]

You are considering to buy a new multi-purpose robot platform. Its kinematic chain has two rotational $q_{\{1,3\}}$ and two linear $q_{\{2,4\}}$ degrees of freedom (DoFs), as shown in the figure below. These four joints are actuated with forces and torques of u_i, $i \in \{1,2,3,4\}$. A gripper is mounted on the end of the robot, indicated by the letter E. The robot's base is mounted on a table. We assume that the base Cartesian coordinates at the mount are $x_{\text{base}} = [0,0,0]$.



a) Forward Kinematics [2 Points]

Compute the kinematic transformation in the global coordinate system from the base $\boldsymbol{x}_{\text{base}}$ to the end-effector E. Write the solution for the $\boldsymbol{x}_{\text{end-eff}} = [x,y,z]^T$ according to the joint values q_i, where $i \in \{1,2,3,4\}$.

b) Inverse Kinematics [2 Points]

Define briefly in your own words the inverse kinematics problem in robotics. Can we always accurately model the inverse kinematics of a robot with a function?

c) Differential Kinematics [4 Points]

Compute the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ of the robot such that $\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $\dot{\boldsymbol{q}}$ is the first time derivatives of the state vector $\boldsymbol{q}$ of the robot. Explain in a sentence the physical meaning of the Jacobian.

d) Singularities [3 Points]

What is the kinematic singularity in robotics? How can you detect it? When does our robotic arm, which was defined above, enter a kinematic singularity?

e) Workspace [1 Points]

If your task is to sort items placed on a table, would you buy this robot? Briefly justify your answer.

Name, Surname, ID Number

Problem 1.2 Control [31 Points]

In robotic locomotion it is common to abstract from the robot by using inverted pendulum models. In this exercise we will use a planar double inverted pendulum to test different control strategies. Our robot can be controlled by specifying the torque $\mathbf{u} = [u_1, u_2]$ of its motors. Consider that in mechanical systems the torque $\boldsymbol{u}$ is a function of the joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$, as given by

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}),$$

where $\boldsymbol{M}$ denotes the inertial matrix, $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis and centripetal forces, and $\boldsymbol{g}$ the gravity terms. In the following exercises assume that these terms are given.

For the programming exercises you will use the attached code. We provide skeletons for controlling the system either in joint space (`my_ctl.py`) or in task space (`my_taskSpace_ctl.py`) and a basic functionality for plotting. You can invoke either mode by running `jointCtlComp.py` or `taskCtlComp.py` respectively. Attach a printout with plots and a snippet of your source code for each programming exercise.

a) PID Controller [2 Points]

What is the form of a proportional-integral-derivative (PID) controller and how could you use it to control a robot, i.e. what physical quantities could you control? Name one positive and one negative aspect of PID controllers.

b) Gravity Compensation and Inverse Dynamics Control [4 Points]

Suppose that you would like to create a control law to set the joint angles on the double inverted pendulum model by controlling the torque of the motors. Write a feedback control law which additionally gravity compensates and then extend it to full inverse dynamics control.

c) Comparison of Different Control Strategies [12 Points]

In the following exercise you will investigate the differences of the following control algorithms, P, PID, PD with gravity compensation, and full inverse dynamics. The double pendulum is initiated hanging down, with state $\boldsymbol{q}_{\text{start}} = [-\pi, 0]$. We simulate the system with a time-step $dt = 0.002$ seconds using symplectic Euler integration and run the simulation for $t_{\text{end}} = 3s$.

Implement the control laws by filling the skeleton file `my_ctl.py`. Use the following feedback gains $K_P = 60, K_D = 10, K_I = 0.1$ for the first joint and $K_P = 30, K_D = 6, K_I = 0.1$ for the second one. The target state of the double pendulum is set to $\boldsymbol{q}_{\text{des}} = [-\pi/2, 0]$.

Create (max. 4) plots that compare the different control strategies and analyze the results. It is your choice how to illustrate your results. In your analysis you should include a discussion on the overall performance of each controller. Which controllers manage to go to the desired point, and how does the choice of a controller affects the behavior of the second joint of the pendulum? Additionally discuss which controller you would choose and why. The provided code is able to generate plot but feel free to modify it if you like. Points will be deducted for confusing plots. Do not forget to include your source code in your solutions.

d) Tracking Trajectories [4 Points]

Repeat the same experiment but this time use the provided time-varying target trajectory. Create (max 4) plots that compare the different control strategies and analyze the results. In your analysis discuss the overall performance and which controllers track the desired trajectory nicely. Additionally discuss which controller you would choose and why.

e) Tracking Trajectories — High Gains [4 Points]

Repeat the same experiment (using the provided trajectory) but this time multiply the gains by ten. Create plots that compare the different control strategies and analyze the results. In your analysis discuss the overall performance and compare it to the previous case. Are there any drawbacks of using high gains?

f) Task Space Control [5 Points]

The robot must now reach a desired position in task space $\boldsymbol{x}_{\text{end}} = [-0.35, 1.5]$. In class we derived the Jacobian transpose, Jacobian pseudo-inverse, and Jacobian pseudo-inverse with damping methods. All of them are implemented in `my_taskSpace_ctl.py`. You are asked to implement also the null-space task prioritization method with a null-space resting posture $\boldsymbol{q} = [0, \pi]$. Run the simulation and plot the initial and final configuration of the robot. Then, change the resting posture to $\boldsymbol{q} = [0, -\pi]$ and redo the plots. Analyze in a couple of sentences your observation. Use the same damping coefficient 10^{-6} and include a code snippet to your solutions.