Robot Learning

Winter Semester 2020/2021, Homework 1

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

Total points: 43
Due date: Midnight, Tuesday, 01 December 2020
Übungsblatt 1

Aufgabe 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 $x_{\text{base}} = [0, 0, 0]$.

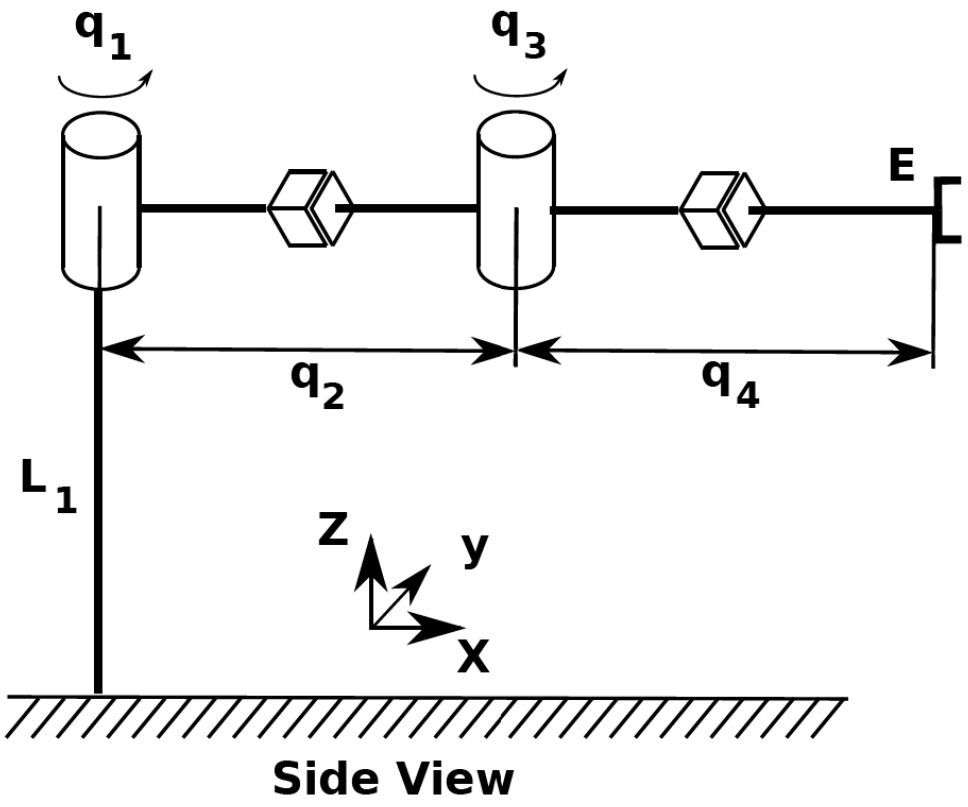

Abbildung 1: Side View

Robot Learning Homework 1

Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

1.1a) Forward Kinematics (2 Punkte)

Lösungsvorschlag:

In DH (Denavit - Hartenberg) Description:

Link	a_i	α_i	d_i	θ_i
1	0	0	L_1	q_1
2	q_2	0	0	0
3	0	0	0	q_3
4	q_4	0	0	0

$$\underline{\underline{A}}_1 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \underline{\underline{A}}_2 = \begin{bmatrix} 1 & 0 & 0 & q_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\underline{A}}_3 = \begin{bmatrix} \cos q_3 & -\sin q_3 & 0 & 0 \\ \sin q_3 & \cos q_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \underline{\underline{A}}_4 = \begin{bmatrix} 1 & 0 & 0 & q_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\underline{H}}_4^0 = \underline{\underline{A}}_1 \cdot \underline{\underline{A}}_2 \cdot \underline{\underline{A}}_3 \cdot \underline{\underline{A}}_4 = \begin{bmatrix} \cos(q_1+q_3) & -\sin(q_1+q_3) & 0 & q_2 \cdot \cos q_1 + q_4 \cdot \cos(q_1+q_3) \\ \sin(q_1+q_3) & \cos(q_1+q_3) & 0 & q_2 \cdot \sin q_1 + q_4 \cdot \sin(q_1+q_3) \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\hat{x}}_{end} = \underline{\underline{H}}_4^0 \cdot \underline{\hat{x}}_E = \begin{bmatrix} \cos(q_1+q_3) & -\sin(q_1+q_3) & 0 & q_2 \cdot \cos q_1 + q_4 \cdot \cos(q_1+q_3) \\ \sin(q_1+q_3) & \cos(q_1+q_3) & 0 & q_2 \cdot \sin q_1 + q_4 \cdot \sin(q_1+q_3) \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{pmatrix} q_2 \cdot \cos q_1 + q_4 \cdot \cos(q_1+q_3) \\ q_2 \cdot \sin q_1 + q_4 \cdot \sin(q_1+q_3) \\ L_1 \\ 1 \end{pmatrix} \Rightarrow \underline{x}_{end} = \begin{pmatrix} q_2 \cdot \cos q_1 + q_4 \cdot \cos(q_1+q_3) \\ q_2 \cdot \sin q_1 + q_4 \cdot \sin(q_1+q_3) \\ L_1 \end{pmatrix}$$

Abbildung 2: Task 1a

Robot Learning Homework 1 Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

1.1b) Inverse Kinematics (2 Punkte)

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?

Lösungsvorschlag:

The inverse kinematic problem is a mapping from joint task space to joint space.
In other words, we have known the position of Endeffektor in global coordinate (x), and the mission is to calculate the translation or rotation in each joints (q).
In the calculation process, the foreseeable challenges will be the singularity problem. It could lead to infinite result of inverse kinematic function.So a function of inverse kinematic can not guarantee the accuracy model.

1.1c) Differential Kinematics (4 Punkte)

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

Lösungsvorschlag:

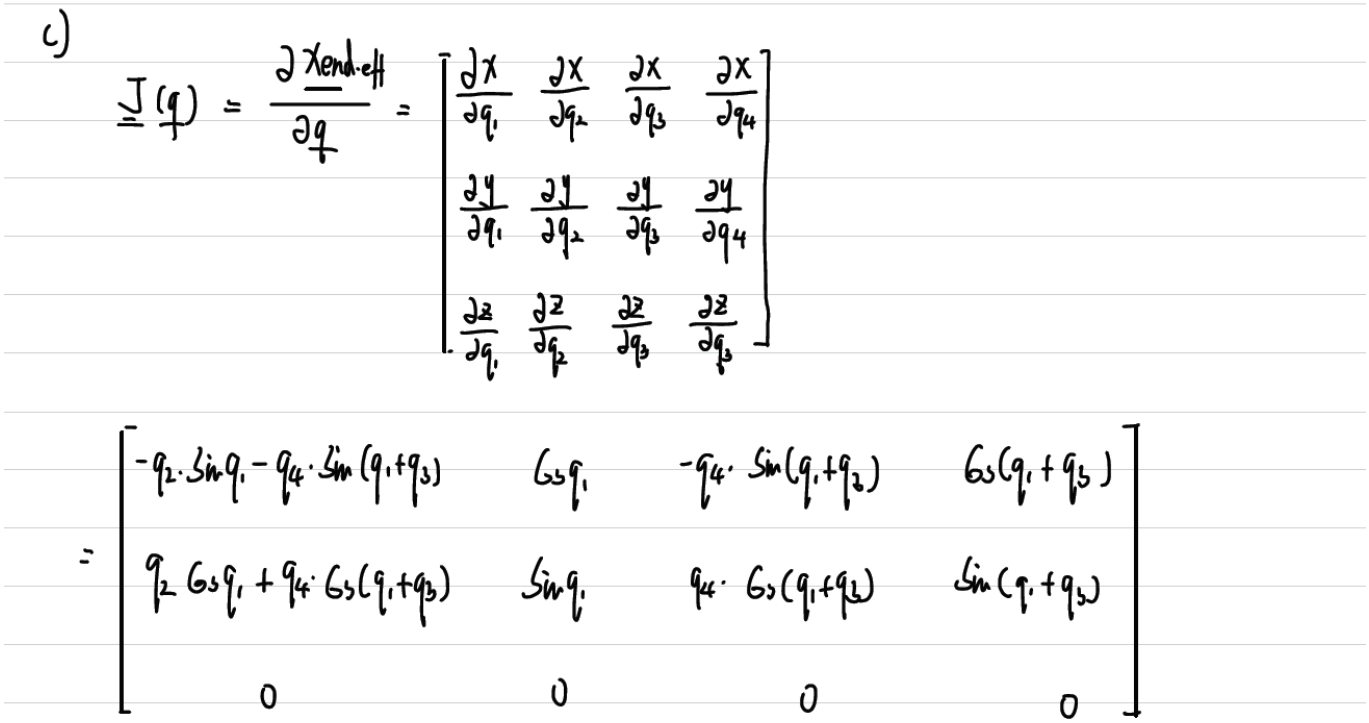

Abbildung 3: Task 1c

Jacobian is the mapping from joints velocity to end-effector velocity. It could be seen as a coordinate system distortion (from joint coordinate to global coordinate).

Robot Learning Homework 1

Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

1.1d) Singularities (3 Punkte)

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?

Lösungsvorschlag:

The kinematic singularity could be explained as the linear dependence in columns of the Jacobian.
It could be detected, when the determination of Jacobian is zeros, which means the rank reduction of matrix.

In our Jacobian, the third raw is zeros, it means a kinematic singularity.
In addition, if $q_3 = 0$ or $q_3 = \pi$, the second column in Jacobian matrix has a linear correlation with forth column.

1.1e) Workspace (1 Punkt)

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

Lösungsvorschlag:

No, i won't buy this robot.
The spatial motion of this robot end-effector is just the x-y plane. If the Items on table have different heights, this robot could not grab them in a fixed horizontal plane.

Robot Learning Homework 1 Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

Aufgabe 1.2: Differential Kinematics (31 Punkte)

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 u is a function of the joint positions q, velocities $\dot{q}$ and accelerations $\ddot{q}$, as given by

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

where M denotes the inertial matrix, $c(q, \dot{q})$ the Coriolis and centripetal forces, and 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.

1.2a) PID Controller (2 Punkte)

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.

Lösungsvorschlag:

The overall control function $u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e\,(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$
where $K_\mathrm{p}, K_\mathrm{i}$, and K_d, all non-negative, denote the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted P_1, l, and D).[**ref1**]

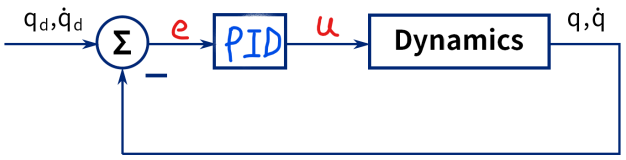

Abbildung 4: PID Controller in robot

As shown above, a PID controller continuously calculates an error value $e(t)$ as the difference between a desired trajectory point (q_d,$\dot{q_d}$) and a measured process variable (q,$\dot{q}$) and applies a correction based on proportional, integral, and derivative terms.

Physical quanties:
The output physical quantity $u(t)$ of PID Controller is torque, which means the acceleration $\ddot{q}$ of robot can be directly controlled, i.e. velocity $\dot{q}$ and spatial translation q will be integrally computed.

Positive Aspect:
Useful if not good model is known!
Negative Aspect:
For tracking control, it may create havoc and disaster!, which means the Real-time response could be unstable. Moreover, it's hart to identify Control Parameters in practice.

Robot Learning Homework 1 Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

1.2b) Gravity Compensation and Inverse Dynamics Control (4 Punkte)

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.

Lösungsvorschlag:

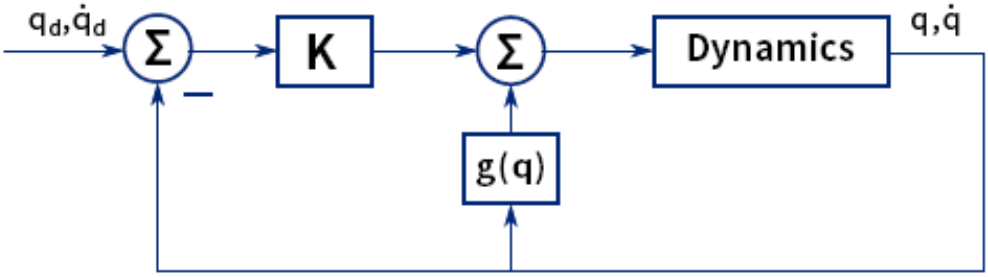

Abbildung 5: Linear PD Control with Gravity Compensation

PD-Controller with gravity compensation:

$$u_t = K_P\left(q_d - q_t\right) + K_D\left(\dot{q}_d - \dot{q}_t\right) + g(q)$$

where $g(q) = \begin{bmatrix} g{*}m_2 * (\frac{1_2 * \cos(q_1+q_2)}{2} + 1_1 * \cos(q_1)) + \frac{g * 1_1 * m_1 * \cos(q_1)}{2} \\ g{*}1_2 * m_2 * \cos(q_1+q_2)_{\frac{}{2}} \end{bmatrix}$

extend it to full inverse dynamics control:

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

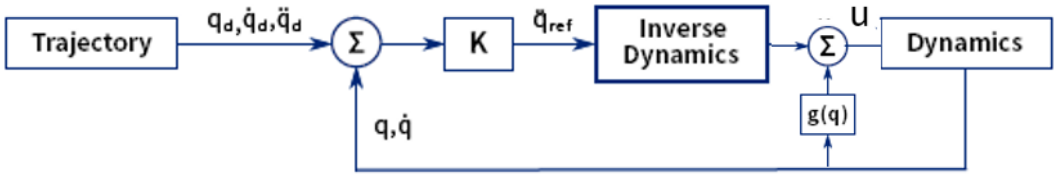

Abbildung 6: Full inverse dynamics control with Gravity Compensation

where $M(q) = \begin{bmatrix} \theta_1 + \theta_2 + m_2 * ({1_1}^2 + 1_1 * 1_2 * \cos(q_2) + 1_2) + \frac{{1_1}^2 * m_1}{4} & \theta_2 + m_2 * (\frac{l_2^2}{4} + \frac{1_1 * 1_2 * \cos(q_2)}{2}) \\ \theta_2 + m_2 * (\frac{l_2^2}{4} + \frac{1_1 * 1_2 * \cos(q_2)}{2}) & \theta_2 + \frac{m_2 * {l_2}^2}{4} \end{bmatrix}$

where $c(\dot{q}, q) = \begin{bmatrix} f_1 * \dot{q_1} + 1_1 * 1_2 * m_2 * (\sin(q_2) * {\dot{q_1}}^2 + \dot{q_(}1) * \dot{q_(}2) * \sin(q_2)) \\ 1_1 * 1_2 * m_2 * \sin(q_2) * {\dot{q_1}}^2 + f_2 * \dot{q_2} \end{bmatrix}$

1.2c) Comparison of Different Control Strategies (12 Punkte)

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 $q_{\text{start}} = [-\pi, 0]$. 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 $my\_ct.py$. 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 $q_{\text{des}} = [-\pi/2, 0]$ 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.

Lösungsvorschlag:

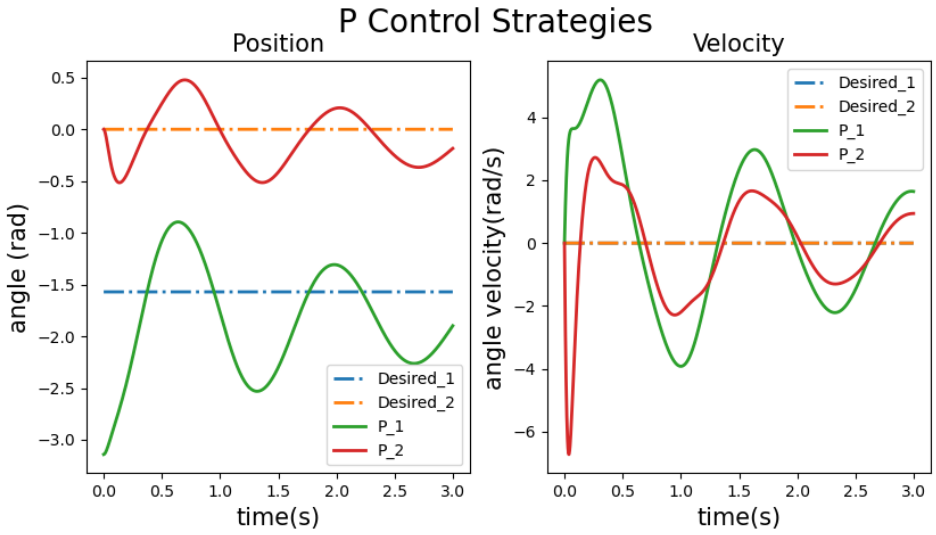

Abbildung 7: P Control Strategy

P Control Strategy will lead to a Oscillation at the second joint, which has a long convergence period and a significant steady state error. It has a poor stability and a bad dynamic character.

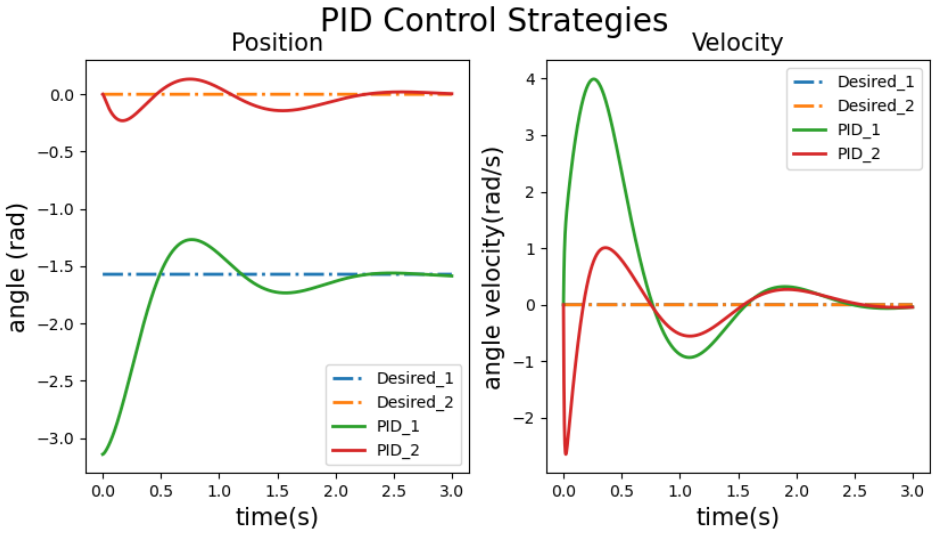

Abbildung 8: PID Control Strategy

PID Control Strategy perform better than P Control.It improves the stability and dynamic character of robot system. The convergence peroid is nearly 2s, simultaneously the overshoot reduced in half.

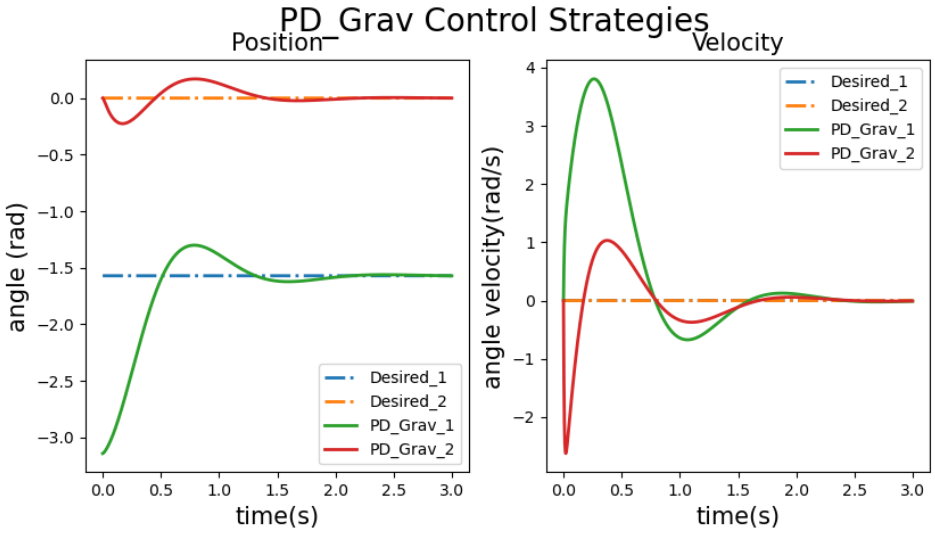

Abbildung 9: PD Control Strategy with gravity compensation

PD Control Strategy with gravity compensation has a better stability than PID, which means a shorter convergence peroid. Between PID and PD with compensation Controls, there's no obvious difference in overshoot and Settling time, hence they has a similar dynamic characteristics.

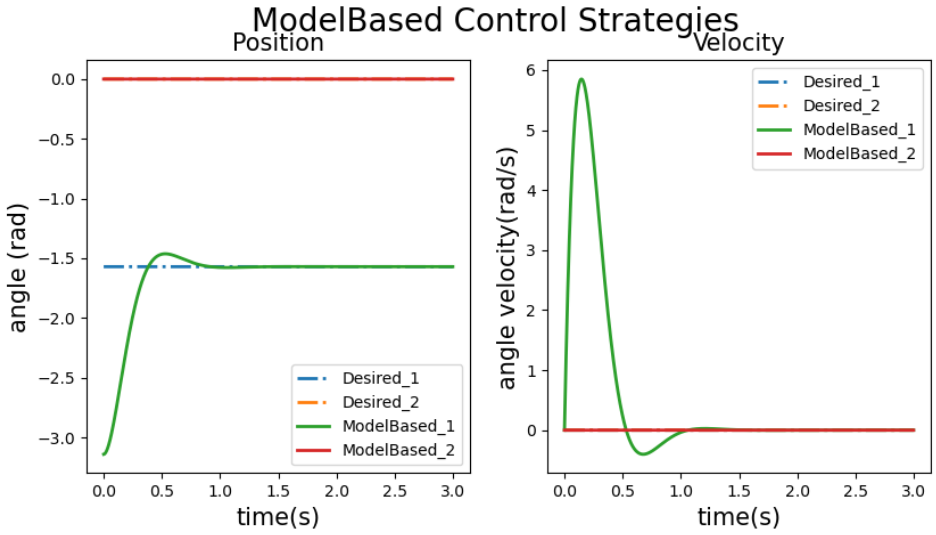

Abbildung 10: modelbased full inverse dynamics

The position and velocity of robot follow completely the desired position. Under an exact inverse model, it has the best performance (stability and dynamic character) of all above control Strategies.

In summary, if an accurate inverse model is available (such as in this Task), I will choose the modelbased full inverse dynamic strategy, otherwise the PD Control Strategy with graviy compensation is a better choice.

```python
import numpy as np

def my_ctl(ctl, q, qd, q_des, qd_des, qdd_des, q_hist, q_deshist, gravity, coriolis, M):
    KP = np.diag([60, 30])
    KD = np.diag([10, 6])
    KI = np.diag([0.1, 0.1])
    if ctl == 'P':
        u = KP @ (q_des - q)
    elif ctl == 'PD':
        u = KP @ (q_des - q) + KD @ (qd_des - qd)
    elif ctl == 'PID':
        u = KP @ (q_des - q) + KD @ (qd_des - qd) + \
            KI @ (np.sum(q_deshist, axis=0) - np.sum(q_hist, axis=0))
    elif ctl == 'PD_Grav':
        u = KP @ (q_des - q) + KD @ (qd_des - qd) + gravity
    elif ctl == 'ModelBased':
        qdd_ref = qdd_des + KP @ (q_des - q) + KD @ (qd_des - qd)
        u = M @ qdd_ref + coriolis + gravity
    return u.reshape(-1, 1)
```

1.2d) Tracking Trajectories (4 Punkte)

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.

Lösungsvorschlag:

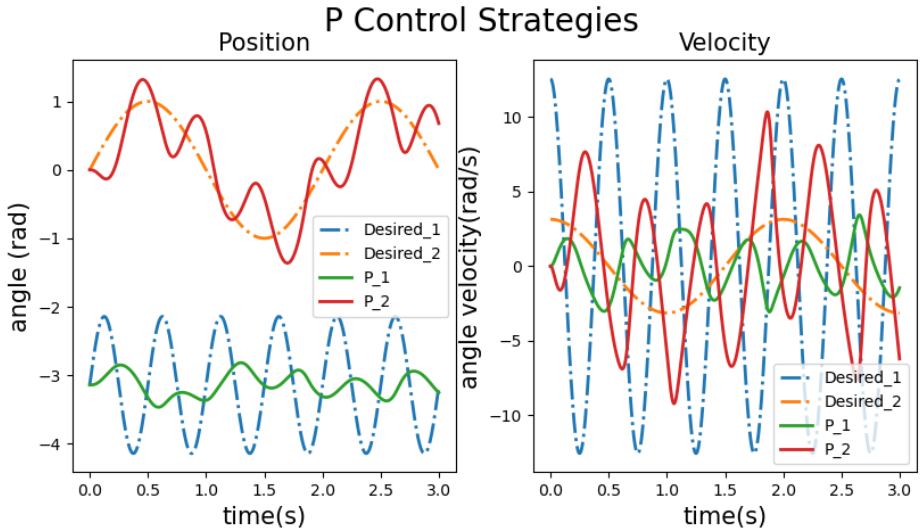

Abbildung 11: P Control Strategy

The position of second joint swings around the desired trajectory under P Control Strategy. In first joint it performs a significant undershoot.

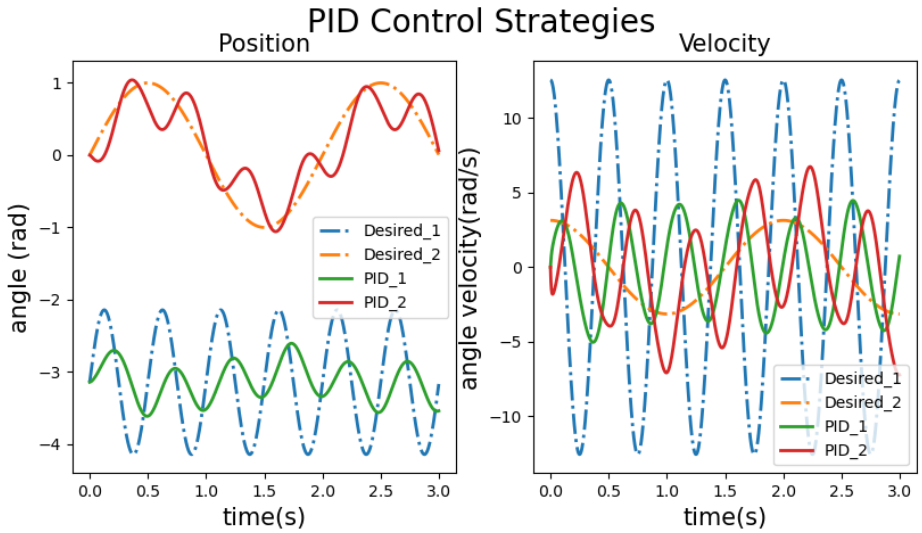

Abbildung 12: PID Control Strategy

PID Control strategy works better than P Control in second joint, bacause of the relative smaller overshoot between desired trajectory and actual trajectory. In first joint, it has a similar undershoot result as P Control.

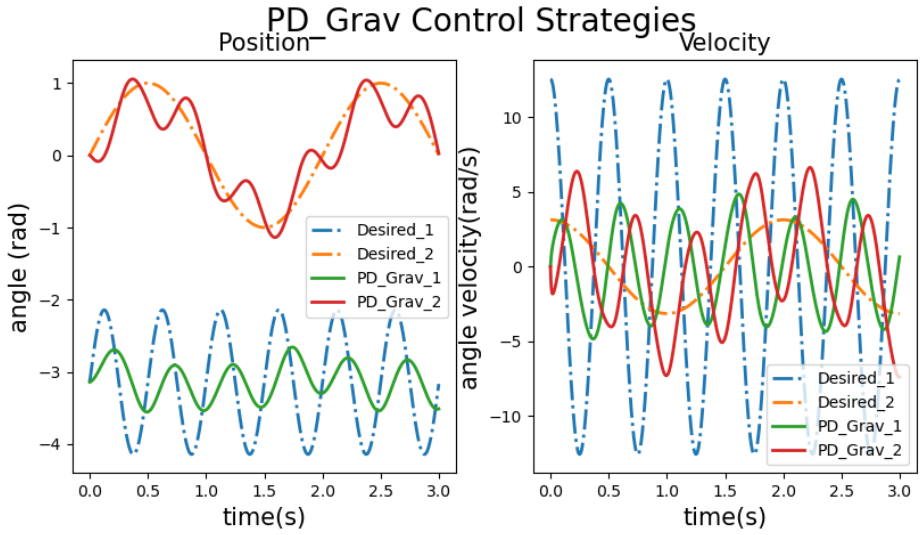

Abbildung 13: PD Control Strategy with gravity compensation

PD Control with gravity compensation performs better than PID Control. In tracking trajectory it has less Steady-state error of second joint.

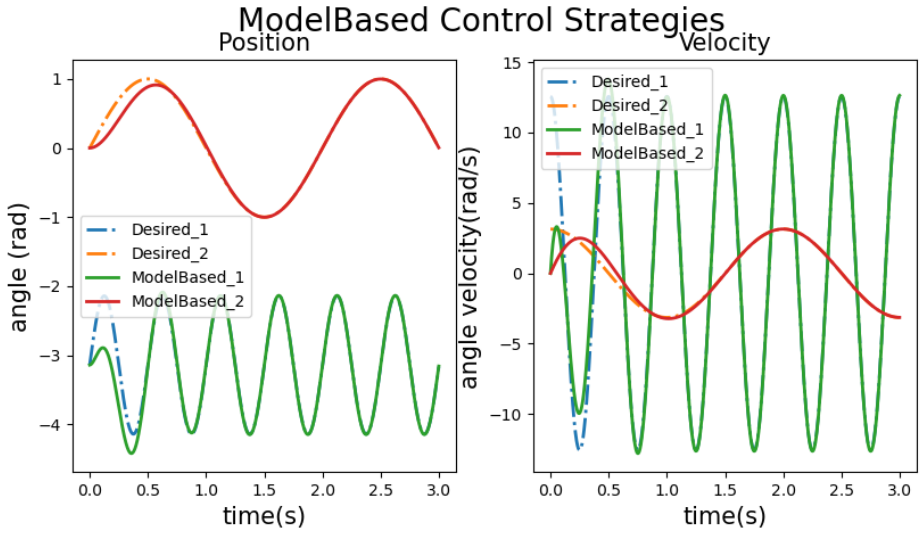

Abbildung 14: modelbased full inverse dynamics

Modelbased strategy has the best performance in comparing. On the beginning of tracking it performs a little Steady-state error. But after nearly 0.5s the actual trajecoty coincide exactly with the desired trajectory.

In summary, I will choose the Modelbased full inverse dynamic strategy in this task, because of better performance in Stability and tracking behavior

The performance of each strategy as below:

	P Control	PID Control	PD with Grav	ModelBased
Stability	- - -	- -	-	+++
Dynamics	- - -	- -	+	++

1.2e) Tracking Trajectories — High Gains (4 Punkte)

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?

Lösungsvorschlag:

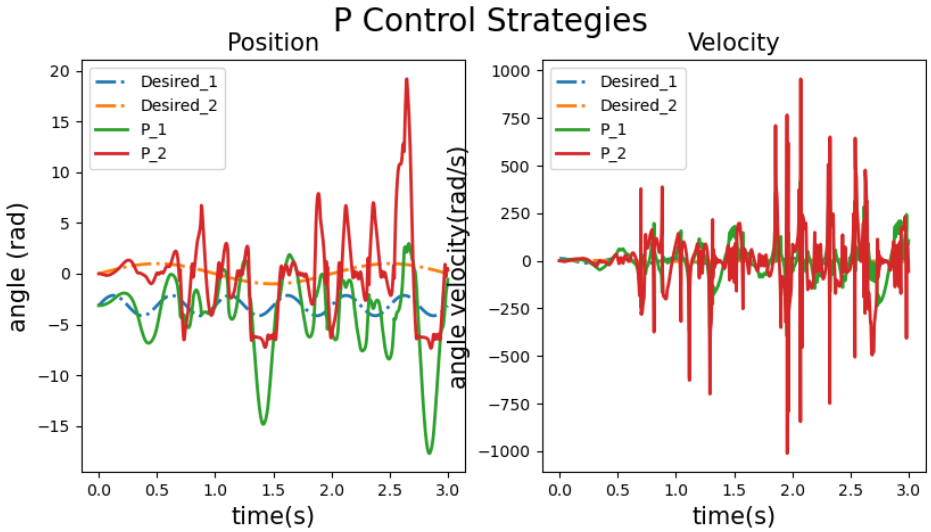

Abbildung 15: P Control Strategy

With gain of control parameters, the performance of P Control Strategy becomes unstable in both joints, which means a divergent oscillation appears at the trajectory tracking.

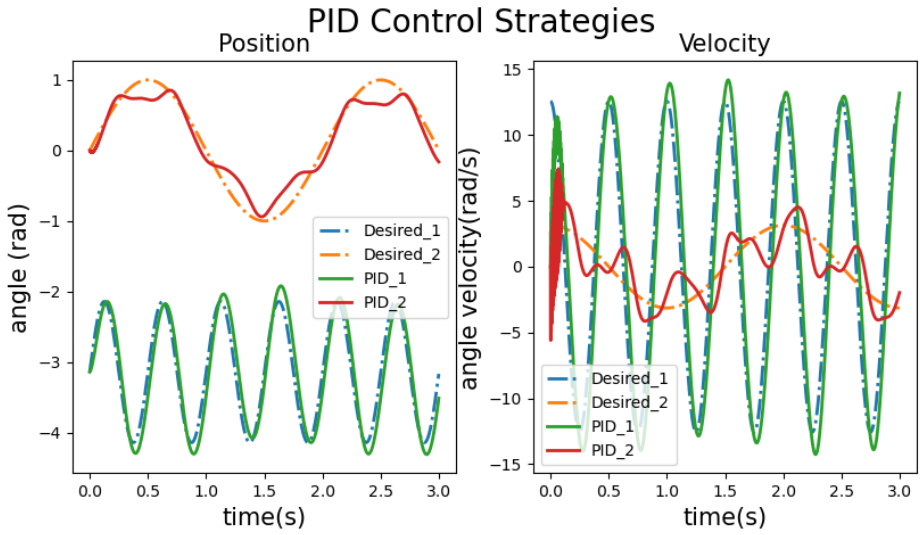

Abbildung 16: PID Control Strategy

With gain of control parameters, the PID Control Strategy performs more stability and faster response than before, especially in first Joint.

At the beginning of tracking, a oscillation appears in velocity of second joint. Possible reason is the numerical instability in simulation.
The overshoot in position of second joint is smaller than before, but still significant. It's similar to steady state error.

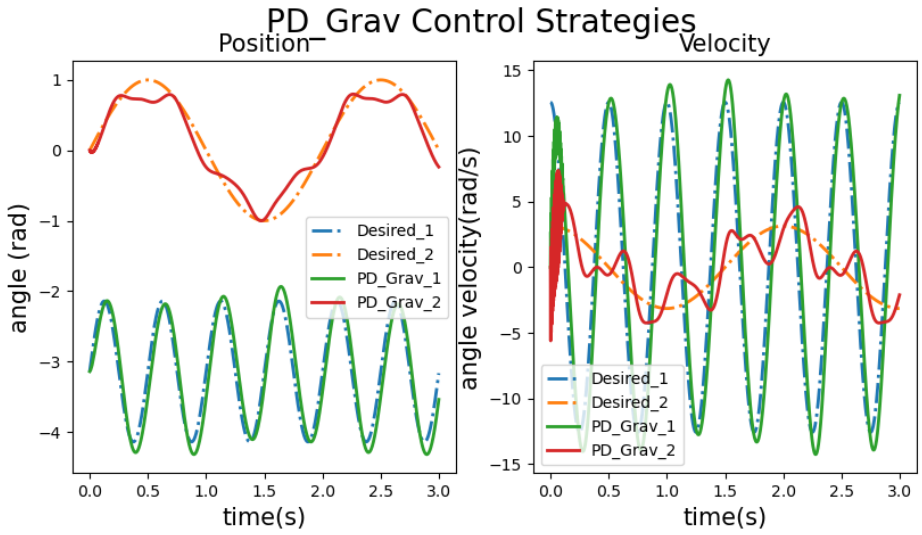

Abbildung 17: PD Control Strategy with gravity compensation

Compared to PID Strategy, PD with gravity compensation has a lower overshoot and less steady state error, which means a better tracking behavior.

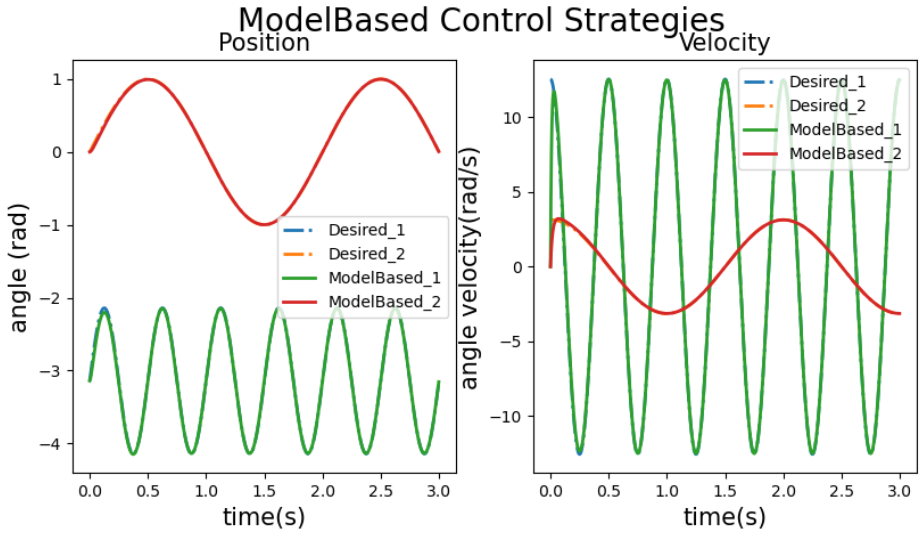

Abbildung 18: modelbased full inverse dynamics

The performance of Modelbased strategy is ideal. The actual trajectory (both joints) coincide exactly with desired trajectory.
Compared to previous case, it overcomes the difference at the begining of tracking.

The performance of each strategy as below:

Robot Learning Homework 1

Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

	P Control	PID Control	PD with Grav	ModelBased
Stability	unstable	+	++	perfect
Dynamics	-	+	+	perfect

The only drawbacks of using high gains is P Control (instability).

Robot Learning Homework 1 Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

1.2f) Task Space Control (5 Punkte)

The robot must now reach a desired position in task space $x_{\text{end}} = [-0.35, 1.5]$. 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 $q = [0, \pi]$. Run the simulation and plot the initial and final configuration of the robot. Then, change the resting posture to $q = [0, -\pi]$ 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.

Lösungsvorschlag:

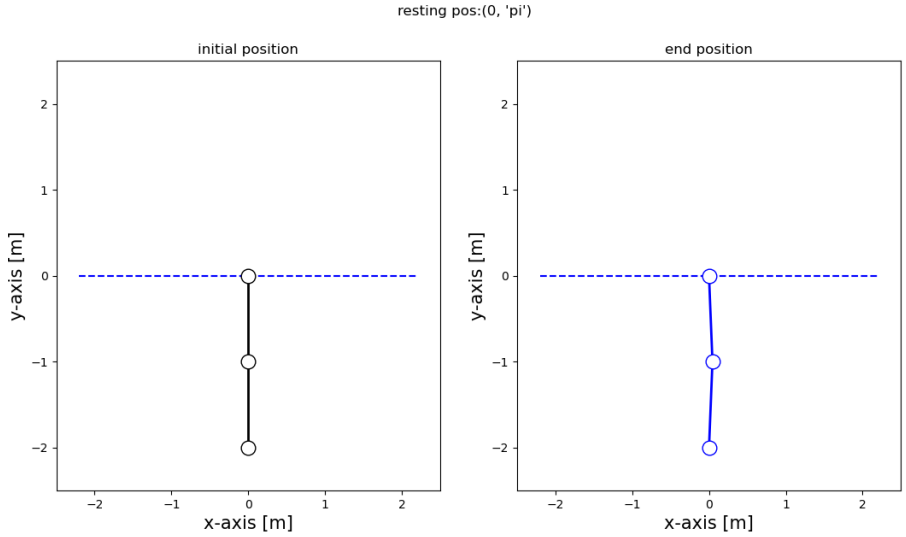

Abbildung 19: null-space resting posture: $q = [0, \pi]$., when damping coefficient 10^{-6}

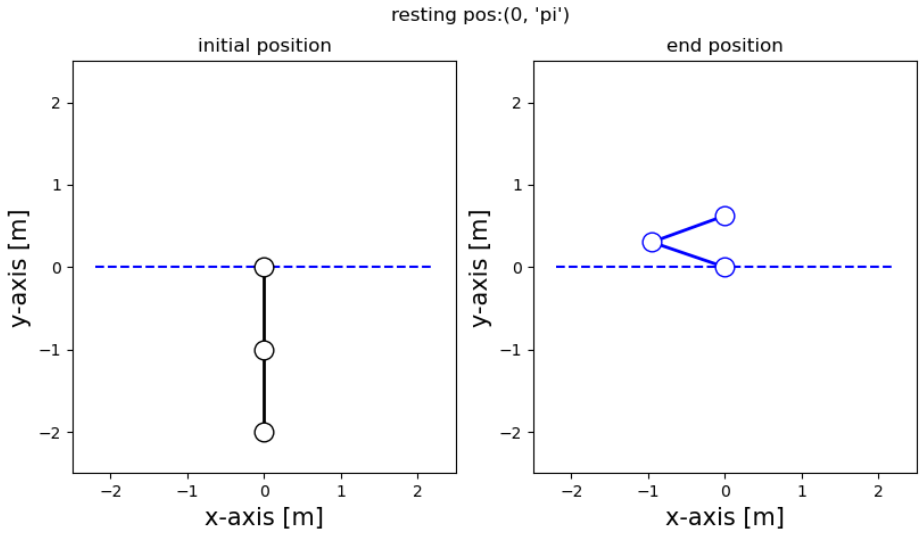

Abbildung 20: null-space resting posture: $q = [0, \pi]$., when damping coefficient 10^{-4}

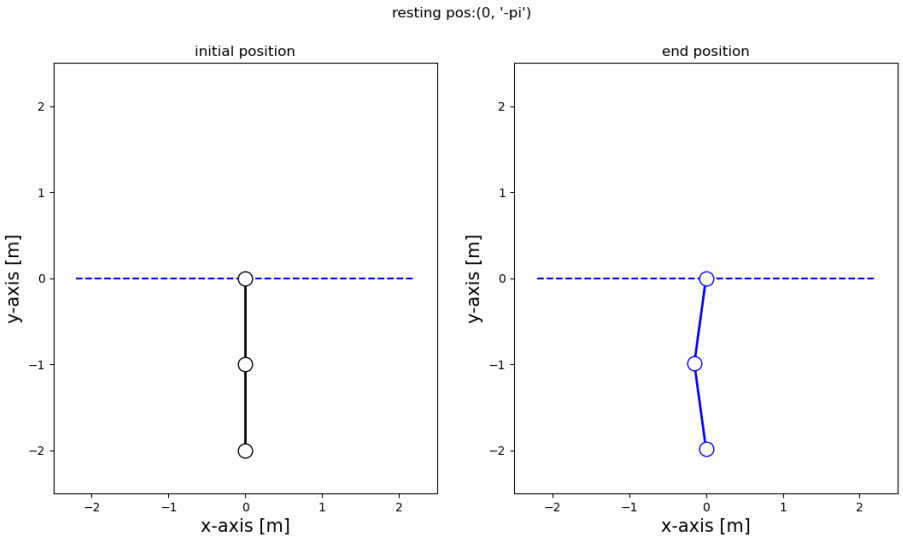

Abbildung 21: null-space resting posture: $q = [0, -\pi]$., when damping coefficient 10^{-6}

This final configuration is different. Possible reason is that the pseudo-inverse of Jacobian $J^{\dagger} = J^{\top}\left(JJ^{\top}\right)^{-1}$ nearby singularity point are different.
When the results of pseuso-inverse of both resting posture are printed in second step of calculation, it shows as below:

$$\underline{J_{\pi}^{\dagger}} = \begin{bmatrix} -4.00000000e-01 & 2.67410552e-12 \\ -2.00000000e-01 & 1.33705276e-12 \end{bmatrix}$$

$$\underline{J_{-\pi}^{\dagger}} = \begin{bmatrix} -1.65016857e-01 & 2.73575476e+02 \\ -6.69967206e-01 & -5.47149727e+02 \end{bmatrix}$$

You can find code snippet of this task as following:

Robot Learning Homework 1

Group 200:

Vorname	Name	Matrikel-Nr.
Yi	Cui	2758172
Yuting	Li	2547040
Liaotian	Zhihao	2897965

```python
import numpy as np
from math import pi

def my_taskSpace_ctl(ctl, dt, q, qd, gravity, coriolis, M, J, cart, desCart, resting_pos=None):
    KP = np.diag([60, 30])
    KD = np.diag([10, 6])
    gamma = 0.6
    dFact = 1e-6

    if ctl == 'JacTrans':
        qd_des = gamma * J.T * (desCart - cart)
        error = q + qd_des * dt - q
        errord = qd_des - qd
        u = M * np.vstack(np.hstack([KP,KD])) * np.vstack([error,errord]) + coriolis + gravity
    elif ctl == 'JacPseudo':
        qd_des = gamma * J.T * np.linalg.pinv(J * J.T) * (desCart - cart)
        error = q + qd_des * dt - q
        errord = qd_des - qd
        u = M * np.vstack(np.hstack([KP,KD])) * np.vstack([error,errord]) + coriolis + gravity
    elif ctl == 'JacDPseudo':
        qd_des = J.T * np.linalg.pinv(J * J.T + dFact * np.eye(2)) * (desCart - cart)
        error = q + qd_des * dt - q
        errord = qd_des - qd
        u = M * np.vstack(np.hstack([KP,KD])) * np.vstack([error,errord]) + coriolis + gravity
    elif ctl == 'JacNullSpace':
        J_star = J.T @ np.linalg.pinv(J @ J.T)
        qd_0 = KP @ (resting_pos - q)
        qd_des = J_star @ (dFact * (desCart - cart)) + (np.eye(2) - J_star @ J) @ qd_0
        error = q + qd_des * dt - q
        errord = qd_des - qd
        u = M * np.vstack(np.hstack([KP,KD])) * np.vstack([error,errord]) + coriolis + gravity

    return u.reshape(-1, 1)
```

Literatur

[1] Wikipedia, PID controller, https://en.wikipedia.org/w/index.php?title=PID$_{c}ontrolleroldid = 991250396$