# Foundations of Robotics

**Prof. Dr. Oskar von Stryk, Paul-Otto Müller**

**Winter Semester 2025 / 2026**

TECHNISCHE UNIVERSITÄT DARMSTADT

Exercise 3

## Notes for this Exercise

- Registration in the **Computer Science Learning Portal** is required to participate in the exercise. Additional information about the course and the regulations for crediting exercise performance towards the final grade can be found on the course page:

  `https://moodle.informatik.tu-darmstadt.de/course/view.php?id=1828`

- From this exercise, **Task 1** and **Task 2** will be graded. An average of 8 points are awarded for completing the programming task (the exact number of points is noted in the task description). Download the templates for the programming task according to the task description and work on the designated files.
- The **handwritten solutions** to the exercises must be submitted as a single PDF document via Moodle by **Tuesday, November 4, 2025, at 9:00 AM**. Permitted submissions include solutions written with pen on paper as scans or photographs, as well as digitally handwritten solutions.
- **Please observe the** regulations for the exercise specified on the course page in the Computer Science Learning Portal.
- **Submission of the programming task** by **Tuesday, November 11, 2025, at 9:00 AM** via file upload in the Computer Science Learning Portal.

## Task 1: Workspace (4.5 Points)

Consider the planar robot shown in Figure 1 with lengths $l_1$ and $l_2$. The first link is rotated by the angle $q_1$ via a revolute joint, the second link is adjusted to the length $q_2$ via a prismatic joint.

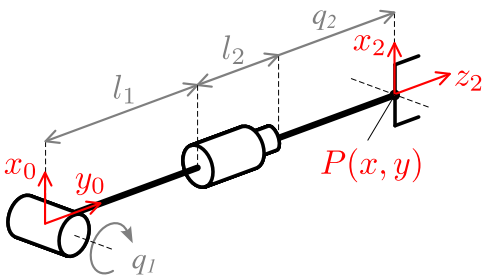



Figure 1: Arm with one revolute and one prismatic joint

a) What are the DH parameters of the depicted robot arm? Provide them in tabular form.

b) Now determine the forward kinematics model ${}^0\boldsymbol{T}_2$ from the DH parameters. Also provide ${}^0\boldsymbol{T}_1$ and ${}^1\boldsymbol{T}_2$.

c) Determine analytically the inverse kinematics solution for the robot using ${}^0\boldsymbol{T}_2$. Provide the functions $q_1(x_p, y_p)$ and $q_2(x_p, y_p)$. Note that the dependence should be exclusively on $x_p$ and $y_p$. You may assume the lengths $l_1$ and $l_2$ as given.
*Hint:* $\sin(x)^2 + \cos(x)^2 = 1$.

d) Based on the result of the inverse kinematics, provide criteria for point $P$ that must be satisfied for $P$ to lie in the workspace and for a solution of the inverse kinematics to exist. You may assume joint limits $q_1 \in [q_{1,min}, q_{1,max}]$ and $q_2 \in [0, q_{2,max}]$ in general and presuppose $l_1, l_2 > 0$.

e) Draw schematically a robot arm matching the workspace shown in Figure 2 with as few revolute and prismatic joints as possible. The origin of the coordinate system $S_0$ is located in the plot at position $(0, 0)$. Also provide the link lengths.

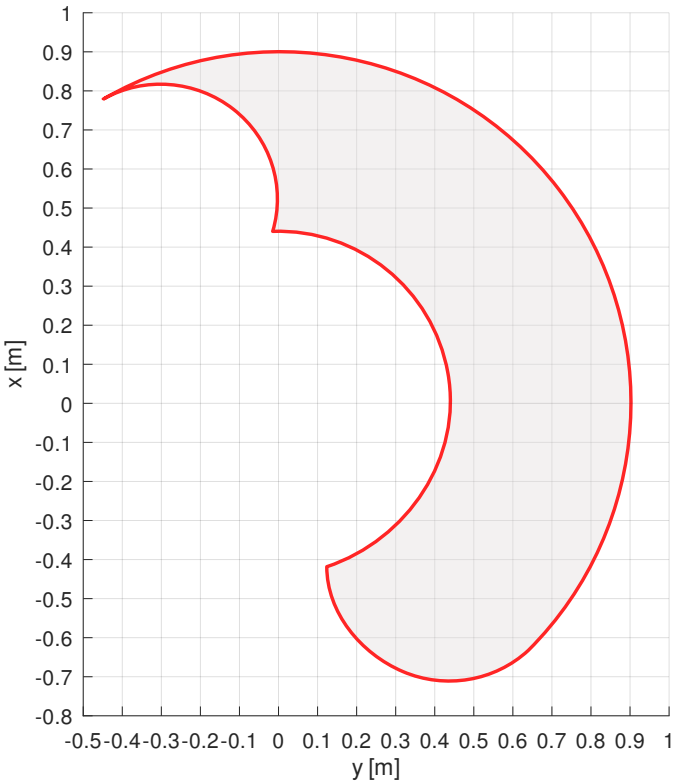



Figure 2: 2D workspace of the desired robot

## Task 2: Inverse Kinematics of a 3-DOF-Manipulator (5.5 Points)

A manipulator with three joints is described by the DH parameters

| $i$ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $\frac{\pi}{2}$ | $q_1$ | 0 | $\frac{\pi}{2}$ |
| 2 | $-\frac{\pi}{2}$ | $q_2$ | 0 | $\frac{\pi}{2}$ |
| 3 | $q_3$ | $-l_4$ | $l_3$ | 0 |
| 4 | $\frac{\pi}{2}$ | 0 | 0 | $-\frac{\pi}{2}$ |

where the first and second joints $q_1$, $q_2$ are prismatic joints and the third joint $q_3$ is a revolute joint. The lengths $l_3$ and $l_4$ are given by the mechanics of the manipulator and are $l_3 = 0.1\,\text{m}$ and $l_4 = 0.3\,\text{m}$. The fourth row of the table is only required to orient the coordinate system of the end-effector in the desired orientation and does not contain a joint variable.

a) Sketch a robot matching the DH parameters including the spatial arrangement of the coordinate systems $S_0$ to $S_4$. Also add the lengths $l_3$ and $l_4$ as well as the *variable* parameters $q_1$, $q_2$ and $q_3$ to the sketch.

b) Specify the transformation ${}^0\boldsymbol{T}_4$ that describes the pose of the end-effector coordinate system relative to the base coordinate system. Also specify all transformations ${}^{i-1}\boldsymbol{T}_i$.

c) Specify all possible tuples of joint positions for which the end-effector coordinate system $S_4$ has its origin at the point $\begin{pmatrix}0.5\,\text{m} & 0.35\,\text{m} & 1.2\,\text{m}\end{pmatrix}^T$ with respect to $S_0$ and is rotated in orientation by $-\frac{\pi}{2}\,\text{rad}$ relative to $S_0$ about the $y_0$-axis, i.e., ${}^0\boldsymbol{R}_4 = \text{rot}(y; -\frac{\pi}{2}\,\text{rad})$.

d) What are the possible tuples of joint positions if only the origin of the end-effector coordinate system is fixed at $\begin{pmatrix}0.5\,\text{m} & 0.35\,\text{m} & 1.2\,\text{m}\end{pmatrix}^T$ with respect to $S_0$, but the orientation is arbitrary? No joint angle limits need to be considered.

## Task 3: Inverse Kinematics of a 3-DOF-SCARA Manipulator

Consider the planar 3-DOF-SCARA manipulator shown in Figure 3 with arm lengths $l_1$, $l_2$ and $l_3$. The joints are rotated by the joint angles $q_1$, $q_2$ and $q_3$. The positive direction of rotation follows the $z_i$-axis orientation. The angle $p$ stands for pitch and describes the angle between the $y_0$-axis and the third link of the manipulator. The plotted arrow direction indicates the positive angle direction.

Calculate the geometric solution of the inverse kinematics for the depicted manipulator.

1. Determine the joint parameters $q_1$ and $q_2$ as a function of the position of the third revolute joint $(x_w, y_w)$. Provide the solution for both possible elbow positions.

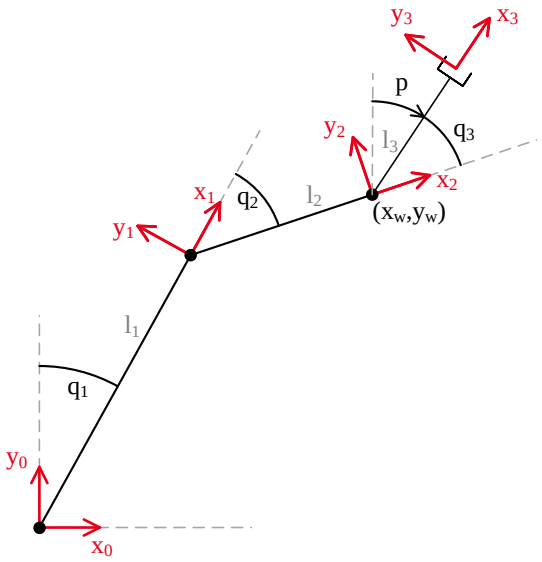



Figure 3: 3-DOF-SCARA manipulator with three revolute joints

2. The joint angle $q_3$ can be calculated using the computed joint angles $q_1$ and $q_2$ as well as the given pitch $p$.

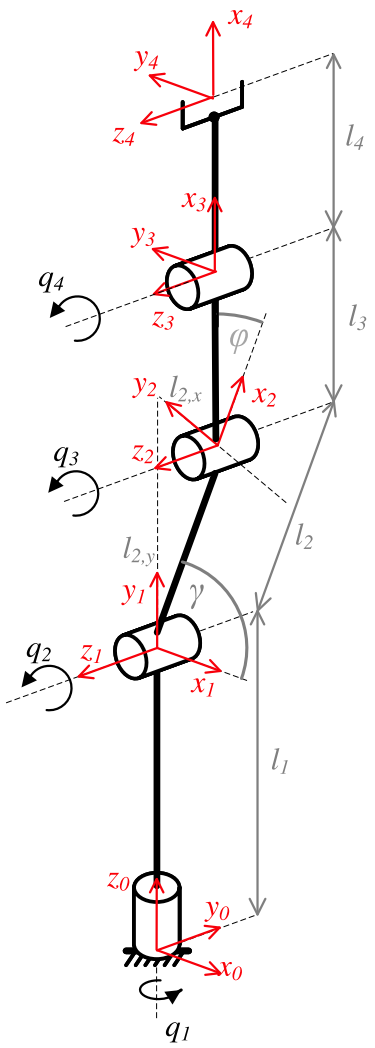
*Hint:*

- Note the positive direction of rotation of the joints. The positive direction of the pitch $p$ is given by the corresponding arrow and the others follow the right-hand rule around the respective $z_i$-axis. The reference is the respective dotted line.
- Consider that the function *atan* is not unique with respect to the quadrant. Therefore use the function *atan2* with two parameters.

## Programming Task P1 Inverse Kinematics of the TurtleBot3 Arm (8 Points)

In the context of the programming tasks, various methods are to be implemented that enable the use of the Turtlebot3. The goal of this task is to determine the forward and inverse solutions for the robot arm mounted on the Turtlebot3. Its kinematic structure with corresponding DH parameters is shown in Figure 4.

| Joint $i$ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $q_1$ | $l_1$ | 0 | $\frac{\pi}{2}$ |
| 2 | $q_2 + \gamma$ | 0 | $l_2$ | 0 |
| 3 | $q_3 + \varphi$ | 0 | $l_3$ | 0 |
| 4 | $q_4$ | 0 | $l_4$ | 0 |




$$
\begin{aligned}
l_1 &= 0.0765\,\text{m} \\
l_2 &= 0.1499\,\text{m} \\
l_{2,x} &= 0.024\,\text{m} \\
l_{2,y} &= 0.148\,\text{m} \\
l_3 &= 0.15\,\text{m} \\
l_4 &= 0.14\,\text{m} \\
\gamma &= 1.41\,\text{rad} \\
\varphi &= \pi/2 - \gamma \\
&= 0.1608\,\text{rad}
\end{aligned}
$$

Figure 4: Turtlebot3 Kinematics

The required kinematic calculations are to be implemented in Python functions, which you will find in the folder *turtlebot3_ik_solver_exercise* of the exercise package. The robot arm is described in a list of dictionaries. The list entries each provide all parameters for a link of the robot via a dictionary. The elements of a dictionary can be accessed via the corresponding key. For example, the expression `dh[0]['d']` accesses the value of the entry `'d'` in the first list element of the list `dh`. The dictionary for describing a link of the robot has the five elements

- `'theta'`, `'d'`, `'a'`, `'alpha'` (the DH parameters known from the lecture),
- `'rho'` (type of joint with 0 = prismatic joint and 1 = revolute joint).

The list of DH parameters of the robot arm can be retrieved via the method `turtle_arm_dh` in the file *turtle_arm.py*. In the first part of the task, the solution of the forward kinematics is to be implemented. To do this, implement the Python functions described below.

- The function `compute_transformation_dh` in the file *ik_helper.py* receives as input the dictionary with the DH parameters of a robot link as well as the joint variable $q_i$ and computes the corresponding homogeneous transformation matrix ${}^{i-1}\boldsymbol{T}_i$. In the case of a revolute joint, the DH parameter $\theta$ should

be used as an offset to the zero position of the joint configuration; in the case of a prismatic joint, the same applies to $d$.

- The function `forward_kinematics_dh` in the file *forward_kinematics.py* receives as input an array with the current joint variables as well as a list with the DH parameters of all robot links, where the $(i-1)$-th element of this list contains the DH parameters of link $i$ $(i = 1, ..., N)$. The return value of this function should be a list of transformation matrices. The $(i-1)$-th element of the list should contain the transformation ${}^0\boldsymbol{T}_i$ of the robot.
- The function `get_tcp_position` in the file *ik_helper.py* receives as input a list calculated with the function `forward_kinematics_dh` and returns the position of the end-effector relative to the base coordinate system as a return value.

To calculate the inverse kinematics of the robot arm, use the geometric solution of the planar 3-DOF-SCARA articulated robot determined in the written part of the task (Task 3). For simplification, assume $\gamma = \frac{\pi}{2}$ rad and $\varphi = 0.0$ rad (i.e., a robot arm extended straight along the $z_0$-axis). The deviations from this assumption are added to the inverse kinematics solution outside of the method you are implementing (see Python file *solve_ik.py* or *test_public.py*). Please note the following description during implementation.

- The function `inverse_kinematics_tb` in the file *inverse_kinematics.py* receives as input a vector of the end-effector position, a dictionary of the end-effector orientation consisting of the pitch (angle between $z_0$-axis and end-effector direction corresponding to the $x_4$-axis) as well as the yaw rotation about the $z_0$-axis, and a list of link lengths. The return value should be a list with the arrays of the determined joint angles. This should have two entries, which contain the solutions corresponding to the two different elbow configurations, or be empty if no solution could be found.
- For further simplification, use the passed pitch and yaw angles to determine the position of the wrist joint ($S_3$) with respect to the base coordinate system starting from the end-effector. Then rotate this point onto the $x_0$-$z_0$-plane to facilitate the determination of the joint angles $q_2$ and $q_3$.
- The joint angle $q_4$ can be calculated using the calculated joint angles $q_2$ and $q_3$ as well as the passed pitch. Note the positive direction of rotation of the joints here.
- Finally, calculate $q_1$.
- Before downloading the templates, first save your work. The updated or added packages are downloaded and built via the following commands:
  - `roscd`
  - `cd ..`
  - `git pull`
  - `turtle install tuda_exercises_student`
  - `turtle update`
  - `turtle make_clean`
- Starting the application:
  - The Python script *solve_ik.py* calls the method for inverse kinematics calculation and sends the result to the dynamics simulation.
  - By entering the command
    "`roslaunch turtlebot3_ik_solver_exercise start_sim.launch`"
    in a terminal, you start the simulation of the Turtlebot arm.
  - With the command
    "`roslaunch turtlebot3_ik_solver_exercise ik_solver.launch`"
    the node `solve_ik` is started, which calls the methods to be implemented by you and sends the solution of the inverse kinematics back to the simulation.

– You can also check your solution of the methods to be implemented via the included tests in the file *test_public.py* using the command
"`rostest turtlebot3_ik_solver_exercise test_public.test --text`"
without using the Turtlebot simulation. Additional, **not** public test cases will be used to evaluate your solution!

### Notes for the Programming Exercise

- Please note the information provided in Moodle on the introductory slides (especially "Introduction to ROS - Part II").
- The prepared virtual machine image can be downloaded via the following link: `hhttps://moodle.informatik.tu-darmstadt.de/mod/url/view.php?id=80362`. The user's password is "ubunturos". Links to descriptions for integrating the image in VMware or VirtualBox, as well as additional notes, can be found in the slides "Introduction to Programming Exercise". For the prepared image, the installation steps from the slides "Introduction to Programming Exercise" up to slide 21 have already been executed.
- You do not need to validate the input values of the functions.
- In the file `mat_nr.py` or `matNr.h`, provide the complete names and student ID numbers of all group members. You can work on the tasks in groups of up to three people. It is sufficient if one group member uploads the solution to Moodle, but the other group members must also be listed as co-authors.
- To submit the programming task, compress the files to be edited according to the task description into an unencrypted `.tar.gz` or `zip` archive. Other formats are not permitted. Name the file according to the pattern `P1_1234567_7654321.zip` with the programming task number and the student ID numbers of the group members separated by underscores. Then upload this file via the file upload for the programming task in the Computer Science Learning Portal.
- Your submitted code must be compilable for your submission to be graded.
- Up to 8 points will be awarded for the solution to the programming task.

## Note on Academic Integrity

The Department of Computer Science places great importance on adherence to the fundamental rules of scientific ethics. By submitting a solution for a written task or a programming task, you confirm that you/your group are the sole authors of the entire material. If the use of external material is permitted, sources must be cited correctly. Further information can be found on the Department of Computer Science website:

`https://www.informatik.tu-darmstadt.de/studium_fb20/im_studium/studienbuero/plagiarismus/index.de.jsp`