Practical Assignment 2 (PA2)
Advanced Manipulation
RDC - RO47001 - 2024/2025

Instructors

J. Micah Prendergast (j.m.prendergast@tudelft.nl)

Learning Objectives

This assignment will guide you towards (1) applying the Denavit-Hartenberg method to a serial robotic manipulator such that you can perform forward and inverse kinematics, derive the Jacobian for the manipulator. And (2), this assignment will guide you in designing and coding simple trajectories for robotic manipulators.

LO1: Apply the Denavit-Hartenberg method to a manipulator to determine the frames at each joint and to build the complete DH-Table.
LO2: Use the DH-Table to determine the transforms between the robots base frame and each of its joints
LO3: Create a function that can generate the correct transformation matrices for any DH-Table
LO4: Derive the Jacobian for a robotic manipulator by hand
LO5: Create a script that can automatically generate the Jacobian at any time instance using the Velocity Propagation approach.
LO6: Apply cubic and quintic polynomials in creating robotic trajectories
LO7: Evaluate these trajectories with regard to their velocities and accelerations
LO8: Create functions for conversion between rotation matrices, quaternions and axis and angle representations

Software

We provide a template for coding in Python and you need to fill in the missing parts related to the learning objectives. The following software is required or recommended:

Python 3.x needs to be installed on your computer (Windows, Mac, or Linux)
We recommend using the Anaconda Navigator and its integrated development environment (IDE) called Spyder. Alternatively, Pycharm is also recommended.

The assignment template requires numpy and matplotlib libraries

DH-Frame Assignment Tips:

Tips for applying the DH method and for assigning frames:

(1) start at the base frame

(2) the next adjacent frame must be reachable through:

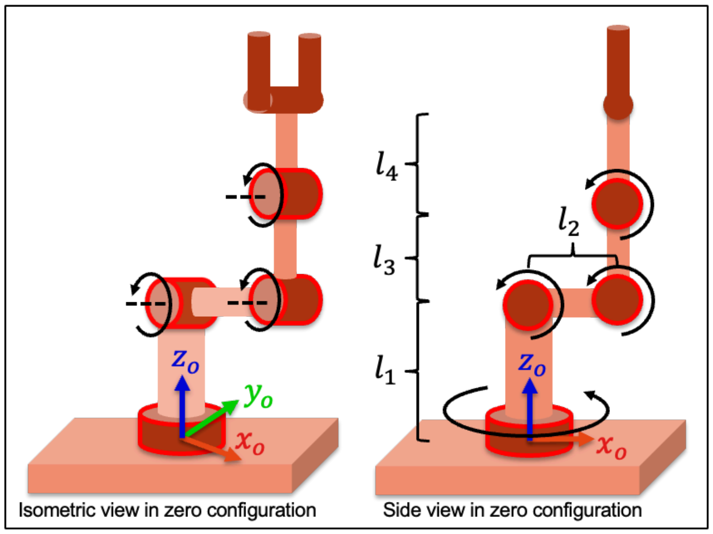
- (i) a translation along the current z-frame
- (ii) followed by a rotation about the z-frame to align the x-axis with the new x-axis
- (iii) followed by a translation along the new x-axis to align the origin with the new axis of rotation
- (iv) followed by a rotation about the new x-axis to align the z-axis with the new axis of rotation.

(3) remember to use the classical DH-method which makes use of the following formula to generate the necessary transformations:

$$Z \cdot X = {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & cos\theta_i \cdot a_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & sin\theta_i \cdot a_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Part 1: Denavit-Hartenberg Parameters

(a) Given the following robotic manipulator, label each joint and link, then add the xyz frames at each joint and at the end-effector/endpoint.

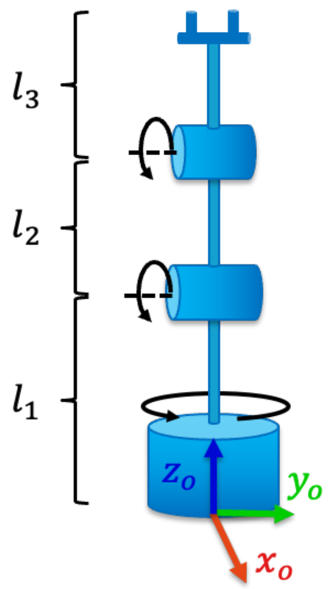

z_o
y_o
x_o
l_4
l_3
l_2
l_1
z_o
x_o
Isometric view in zero configuration
Side view in zero configuration

- (i) The assigned frames must be right-handed coordinate systems
- (ii) When assigning the frame orientation keep in mind this must allow for you to apply the denavit-hartenberg method
- (iii) Use the tips above to help guide you in the frame assignment
- (iv) Positive joint rotation direction is shown in the image and so is the correct base frame orientation (assume it is centered at the base of the robot).

(b) With the correctly assigned frames, complete the Denvit-Hartenberg table for the robot in part (a).

(c) Given the following robotic manipulator, label each joint and link, then add the xyz frames at each joint and at the end-effector/endpoint.

- (i) The assigned frames must be right-handed coordinate systems
- (ii) When assigning the frame orientation keep in mind this must allow for you to apply the denavit-hartenberg method
- (iii) Use the tips above to help guide you in the frame assignment
- (iv) Positive joint rotation direction is shown in the image and so is the correct base frame orientation (assume it is centered at the base of the robot).



(d) With the correctly assigned frames, complete the Denvit-Hartenberg table for the robot in part (c).

(e) Derive the 4x4 homogenous transformation matrices between the baseframe and each of the joints using the formula shown above and in the lecture.

Part 2: Applying Denavit-Hartenberg

(a) Complete the python function *dh()* provided in the code to automate the generation of your transformation matrices from DH-Parameters.

	θ	**d**	**a**	**alpha**
J_1	q_1	h_1	0	$\pi/2$
J_2	q_2	0	l_2	0
J_3	q_3	0	l_3	0

(b) Complete the python function *fk_calc()* to compute the final transformation from the baseframe to the end-effector frame. (hint, use your *dh()* function to do most of the work). Using h_1 = 0.5, l_2 = 1.4 and l_3 = 1.2, give the final transform matrix for the robot when all joint angles are set to 0.

(c) Using the functions you created, compute the end-effector position as all three joints move simultaneously from 0 to pi/4. Create a plot showing this end-effector position as the joint angles increase at small increments (step size of your choice).

Part 3: The Jacobian

(a) Given the same DH-table as **Part 2** above, compute the jacobian by hand using the velocity propagation method taught in class. Compute each transformation matrix and then setup the matrix multiplication steps you would use to compute each all the other transforms you would need. **Note:** you do not need to compute and simplify these by hand. Instead, show the setup and then use the final transforms as derived and shown in the slides. From these, **indicate your z and t vectors.**

(b) In the Python script provided, complete the function *jacobian_fromVP(),* to compute the jacobian of the 3 degree of freedom robotic manipulator using velocity propagation. You should call functions from **Part 2** to make this easier.

(c) In the Python script provided, complete the function *jacobianMoreJoints(),* to compete the jacobian for a robotic manipulator with any number of rotational joints. Confirm that this provides the same result as part (b) for the previous DH-table (it should!).

(d) Using your function from **(c)** compute the jacobian for a universal robot manipulator with the given DH-table and link lengths. Provide the final xyz position and the Jacobian of the robot end-point for its 0-position (all joints set to zero) AND position $q = [0,\ \pi/3,\ \pi/3,\ 0,\ \pi/4,\ 0]$.

$l_1 = 0.1519,\ l_2 = 0.24365,\ l_3 = 0.21325,\ l_4 = 0.11235,\ l_5 = 0.08535,\ l_6 = 0.0819$

	θ	**d**	**a**	**alpha**
J_1	q_1	l_1	0	$\pi/2$
J_2	q_2	0	$-l_2$	0
J_3	q_3	0	$-l_3$	0
J_4	q_4	l_4	0	$\pi/2$
J_5	q_5	l_5	0	$-\pi/2$
J_6	q_6	l_6	0	0

Part 4: Cubic and Quintic Trajectories

(a) In the Python script provided, complete the *cubicTraj()* function to generate the coefficients needed to define a cubic trajectory for any two start and end points for any given time duration. This function will return just the coefficients of the trajectory.

(b) In the Python script provided complete the *trajPoints()* function to generate a trajectory using the coefficient provided by *cubicTraj().* This function will take the coefficients of the *cubicTraj()* function along with the time duration of the trajectory and the number of points the total time should be broken into. In addition to returning the position at each time step, this function should also return the velocity and acceleration.

(c) Generate a plot that shows position, velocity and acceleration for a cubic trajectory starting at start = pi/3 and ending at end = pi over 5 seconds.

(d) In the Python script provided, complete the *quinticTraj()* function to generate the coefficients needed to define a quintic trajectory (similar to part (a)).

(e) Modify the *trajPoints()* function to allow the function to generate both cubic and quintic trajectories as selected by the user.

(f) Generate a plot that shows position, velocity and acceleration for a quintic trajectory starting at start = pi/3 and ending at end = pi over 5 seconds.

Part 5: 3D Trajectories

(a) In the Python script provided, write a function that can generate position trajectories for a robot with 3 degrees of freedom in both the joint or cartesian space (depending on what the user prefers). Use your functions from **Part 4** to help you here.

(b) Generate a plot that shows the 3D positional trajectory from a robot end-effector moving from start = (0, 0, 0) to end = (5.5, 3, 7.5).

Part 6: Coordinate Frames

(a) I have provided a small python library *Fame.py* that will allow you to manipulate and plot reference frames in the cartesian space. This library still needs several functions to be more useful for our robotics applications including functions for converting to/from axis and angle representation and quaternion representation. Complete these methods/functions within the code:

  (i) *to_axis_angle()*
  (ii) *to_quaternion()*
  (iii) *set_rotation_from_axis_angle()*
  (iv) *set_rotation_from_quaternion()*

(b) Create a frame that is rotated 20 degrees about the origin in x and 25 degrees in z and is then translated from the origin to point (1,3,5). Compute the homogenous transform for this frame and then convert it's rotation matrix to quaternion and axis and angle. Confirm that you can also convert back to your original homogenous transform.

(c) You would like to create a function that can compute the rotational error between two frames. Use the axis and angle approach we discussed in class to write the function *calc_rot_error().* Test this function with a range of different rotations from 2 degrees to 100 and plot the results (compare your rotated frame to a static frame and calculate the difference as your error). How accurate is your function for larger angles?

Assessment and Grading

For all parts, please provide the handwritten work (if there was a handwritten part), requested figures and the completed python code needed to generate any figure or answer (if there was a coding part). If it is unclear what you need to provide, please ask me!

A total of 10 points are available for this assignment. To assess the assignment with respect to the learning objectives and to determine the final grade we will run each individual code to determine proper functioning however I should not NEED to run your code if your outputs are clear in the PDF document you submit! With this in mind, make sure the PDF is complete and clear as it is good to show me that you had a working version so I can give partial credit if mistakes happen in the code.

Part 1:
Axis Labels/Correct DH Table (0.5 point) Labeled image of first robot (copied or sketched) with joints, links and axis frames labeled, a typed or hand drawn DH-table.
Axis Labels/Correct DH Table (0.5 point) Labeled image of robot (copied or sketched) with joints, links and axis frames labeled, a typed or hand drawn DH-table.
Transformation Matrices for second robot (1 point): All necessary homogenous transformation matrices between each adjacent joint and between the base frame and all other frames (typed).

Part 2:
Functioning Code and forward kinematics (1 point): Completed code for each of the functions requested, final transformation matrix (forward kinematics) for the robot
Plot (1 point): the plot showing the robot end-effector motion.

Part 3:
Jacobian by hand (1 points): Handwritten work showing how the jacobian was computed,
Functioning code for all parts (2 points): Completed code computes jacobian as expected, final jacobian computed for the Universal robot at the two requested joint positions.

Part 4:
Cubic trajectory (0.5 point): Completed code, one plot showing position, velocity and acceleration for the requested trajectory.
Quintic trajectory(0.5 point): Completed code one plot showing the position, velocity and acceleration for the requested trajectory.

Part 5:
3D trajectory (1 point): Completed code, one plot showing the 3D trajectory.
Part 6:
Coordinate frames functions (0.5 point): Completed code and printouts showing correct conversion between the given frame and axis and angle and the given frame and quaternion.
Rotational Error (0.5 point): Completed code and a plot that shows the estimated rotational difference from 2 to 100 degrees.

Notes For submission:

(1) **Please submit all code as working python files and all requested plots, printouts, derivations etc. as a single PDF. (I should not have to run or check your code to grade this assignment as the plots and printouts should tell me if it works.)**
(2) **If I have asked you to test with certain values, please provide the printout showing me the results from your python code (it is always better for you to run your code than to rely on me to do it properly).**
(3) **Carefully label each Part so that it is clear what question is being graded.**