QUANSER
INNOVATE · EDUCATE

STUDENT WORKBOOK

Quanser Robotics Package for Education for MATLAB®/Simulink® Users

Developed by:
Amir Haddadi, Ph.D., Quanser
Peter Martin, M.A.SC., Quanser

Quanser educational solutions are powered by:
MATLAB SIMULINK

CAPTIVATE. MOTIVATE. GRADUATE.

EXPERIMENT 1: MANIPULATOR KINEMATICS

The purpose of this experiment is to study the forward Kinematics of 4 degree-of-freedom (DOF) serial link robots. The following topics will be studied in this experiment.

Topics Covered

- The concept of kinematics
- Coordinate frame assignment
- DH parameters and DH table
- Manipulator kinematics derivation

Prerequisites

- The robot has been setup and tested. See the Quick Start Guide for details.
- You have access to the User Manual.
- You are familiar with the basics of MATLAB® and SIMULINK®.

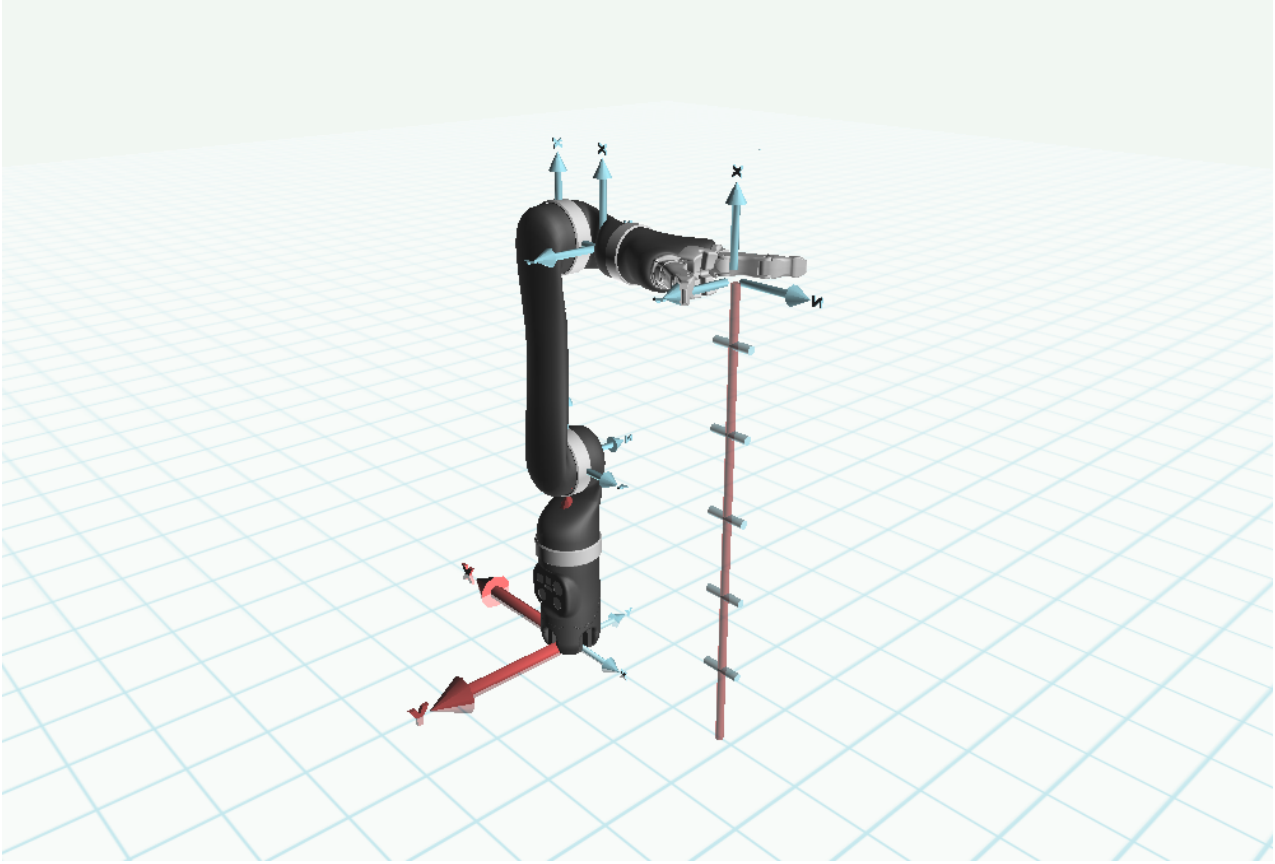


Figure 0.1: Kinematics is used to determine the position of the robot end-effector given the joint angles.

1 Pre-Lab Preparations

1.1 Background

Kinematics refers to the geometric and time-based properties of the motion of an object without considering the forces and moments that cause the motion. In this lab, we will study this relationship in the context of position and orientation of manipulator linkages and end-effector with respect to joint angles in static situations. This particular aspect of kinematics is called forward position kinematics.

In order to analyze geometrically complex manipulators, coordinate frames are attached to various parts of the manipulator including the base frame of the robot which is a fixed coordinate frame, and the end-effector frame of the robotic arm which is attached to the robot end-effector. The study of manipulator kinematics describes how the location and orientation of these frames vary in different configurations, based on the joints angles of the robot arm.

1.1.1 Coordinate Frame Assignment and DH parameters

Robotic manipulators are generally constructed from joints and links. Joints can be *revolute*, meaning they rotate, or *prismatic*, meaning they linearly slide. Each joint is considered to be a single degree-of-freedom (DOF). In this lab, we consider a 4DOF manipulator with four revolute joints. The links of the robot are numbered from the fixed base, L_0, all the way to the end-effector, L_4. Note that for the 4-DOF MICO arm, we have $L_0 + L_1 = 0.2755\,\text{m}$, $L_2 = 0.29\,\text{m}$, $L_3 = 0.1233\,\text{m}$, and $L_4 = 0.16\,\text{m}$.

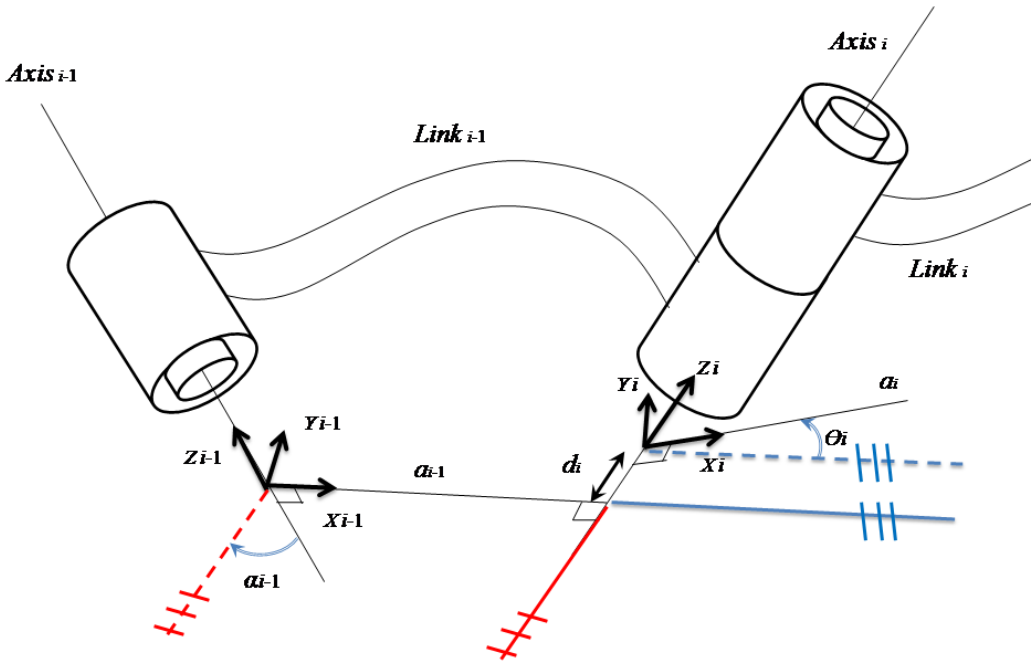


Figure 1.1: Link frame transformation.

Each joint of the manipulator has an axis of rotation (indicated by z_i where i is the joint number) called the *joint axis*. Each link of the manipulator is considered a rigid body that defines the relationship between the two neighbouring joint axis.

Consider the two neighbouring joint axes in Figure 1.1. To determine the relative position and orientation of two fixed axes (that do not move), only two parameters are required: *link distance* or *link length* (also called common

normal), and the *link twist*. Link distance, a_{i-1}, is measured along a line that is mutually perpendicular to the joint axes of neighbouring joints, $i-1$ and i. Link twist, α_{i-1}, is defined as the angle between the two joint axis, $i-1$ and i about the common normal (if the axes are parallel, then $\alpha_{i-1}=0$).

If additional axes are considered (three or more fixed axes), in general case the common normals do not intersect the common axes at the same time. Therefore, a new parameter is needed to describe the relationship between the neighbouring axes, which is called *link offset*. Link offset, d_i is the distance between the common normals a_{i-1} and a_i along the axis i.

If the axes are not fixed, a final parameter is required to describe the relationship between the joint axes, called *joint angle*. The joint angle, θ_i is the angle between the common normals a_{i-1} and a_i about axis i. These four parameters are the *Denavit-Hartenberg* or *DH* parameters of the robot.

As was mentioned earlier, a coordinate frame is attached to each link in order to describe the location and orientation of each link relative to its neighbours. Here, we assume that when all joints are zero, the x_0 axis is towards the front of the robot, y_0 is towards left, and z_0 is upward (to be aligned with the first joint axis of rotation).

Follow the steps below to attach frames to the robot links.

1. To each link i assign a frame i.
2. z_i axis is the joint axis (of rotation).
3. The origin of the frame i is chosen at the intersection of joint axis i and the common normal a_i (or at the intersection of joint axes i and $i+1$ if a_i = 0).
4. x_i is placed along the common normal a_i pointing to the joint $i+1$. If a_i = 0 the axis x_i is perpendicular to both z_i and z_{i+1} and direction is arbitrary; preferably select it along x_{i-1} when $\theta_i=0$.
5. y_i can be achieved with the right hand rule.

With the above frame assignment, a_{i-1} is the distance between z_{i-1} and z_i along x_{i-1}, α_{i-1} is the angle between z_{i-1} and z_i about x_{i-1}, d_i is the distance between x_{i-1} and x_i along z_i, and θ_i is the angle between x_{i-1} and x_i about z_i.

Note: There are multiple standards in assigning frames and DH parameter derivation. This lab follows the standard proposed by J.Craig[1].

1.1.2 Forward Kinematics Using DH Parameters

After multiplying the matrices of rotations and translations related to the coordinate frames attached to robot links, the transformation matrix for two consecutive link frames (frame i to frame $i-1$), with the defined DH parameters, is defined as follows:

$$ {}^{i-1}_{i}T = \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i}c_{\alpha_{i-1}} & c_{\theta_i}c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -s_{\alpha_{i-1}}d_i \\ s_{\theta_i}s_{\alpha_{i-1}} & c_{\theta_i}s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & s_{\alpha_{i-1}}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.1} $$

1.2 Pre-Lab Exercise

1. Using the procedure outlined in the background section for attaching coordinate frames to robotic manipulators, assign appropriate coordinate frames to the schematic of the MICO robotic manipulator in Figure 1.2. The base and end-effector of the arm are shown with red dots, and some axes are shown for simplicity.

[1] J. Craig: Introduction to Robotics, Addison-Wesley, 1986.

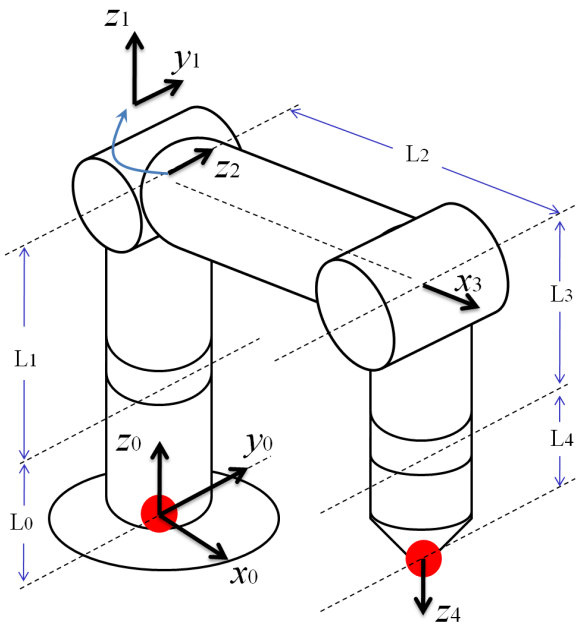


Figure 1.2: Schematic of the 4-DOF MICO robotic manipulator.

2. Fill in the DH parameters of the robot in Table 1.1 according to the frames you assigned in Question 1.

Table 1.1: DH table for the 4-DOF MICO robotic manipulator

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				

3. Derive individual link transformation matrices using Equation 1.1. Determine the portion of the matrices that represents the rotation matrices and translation vectors.

4. How would you derive the forward position kinematics of the MICO 4-DOF robotic manipulator?

2 In-Lab Exercise

2.1 Simulation

The QUARC model for this exercise is called ”MICO_Kinematics_Simulation.mdl” a snapshot of which shown in Figure 2.1.

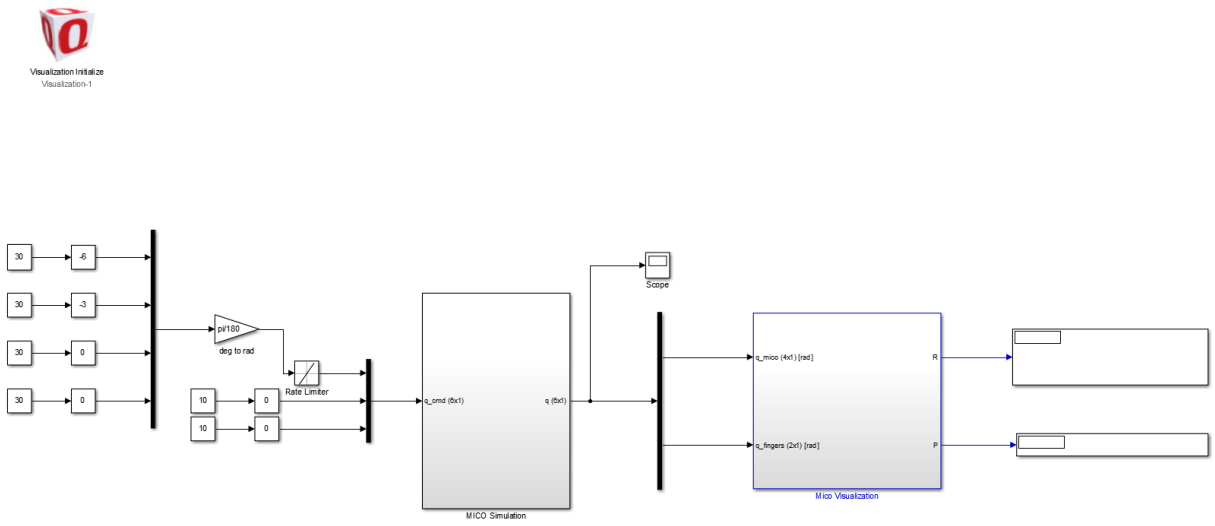


Figure 2.1: Snapshot of the controller model ”MICO_Kinematics_Simulation.mdl”

The ”Mico 4DOF FPK - Extended” block, inside the ”Mico Visualization” subsystem shown in blue, receives the joint angles as input and computes the position of the end-effector. Compile and run the model, then follow the procedure outlined below. Observe the motion of the robot, and answer the associated questions.

1. Describe the configuration of the robot when all joint angles are zero. You can use keyboard to view the robot from different angles. (W for zooming in, S for zooming out, A for moving to the left, D for moving to the right. Hold down the middle mouse button, and use the mouse to rotate. ESC will reset the view)
2. Press ESC key to reset the view. Using the slider gains 1, 2 and 3, move the first joint to +90 °, the second joint to -30 ° and joint 3 to zero. Looking at the virtual robot, determine the position of the robot end-effector (the distance between the grid lines is 10 cm).
3. Apply Equation 1.1 to all links using the angles specified in Question 2, and robot’s DH parameters, to verify the values for (x, y, z) (use MATLAB for matrix multiplications).
4. Modify joints 2 and 3 to raise the z value to 30 cm. What are the resultant joint angles? Can you achieve the same x, y and z position using a different joint configuration?
5. Modify the command to the last joint (θ_4) and observe its effect on the virtual robot. Does the last joint angle affect the position or orientation of the end-effector of the robot? Observe the motion of the robot as you change the angle of each joint, and outline how each joint angle changes the position and orientation of the end-effector?

2.2 Experiment

In this section, you will experimentally evaluate the forward kinematics of the 4DOF MICO. While working with the robot, we recommend that you always keep a safe distance from the robot, and test your position commands on the virtual robot first before applying them to the actual robot.

The QUARC model for this experiment is called ”MICO_Kinematics_Experiment.mdl” shown in Figure 2.2. Before running the model, manually move the robot arm into a pose where the joints have a reasonable range of motion and the robot end-effector is away from the table (preferably similar to Figure 0.1). You can move the arm easily with the power off. Turn on the robot power to hold the initial pose. Before running the model, make sure that the robot is disabled (the yellow manual switch, is set to ”Disabled”).

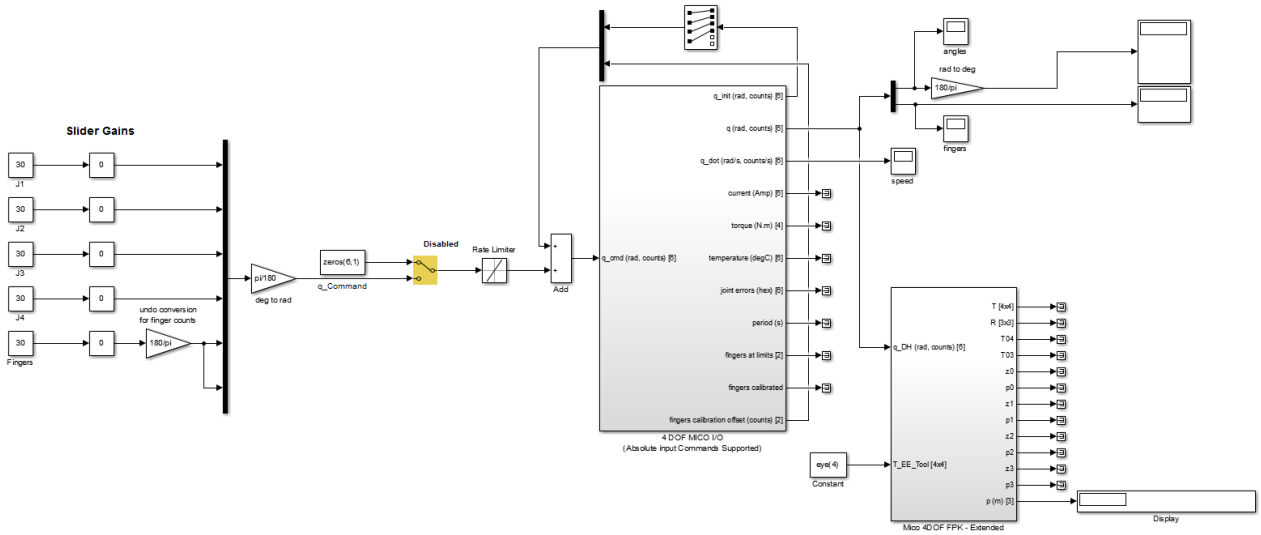


Figure 2.2: Snapshot of the controller model ”MICO_Kinematics_Experiment.mdl”

Compile and run the model. Follow the steps below and answer the questions.

Caution

Be sure to set the Ports in the 4-DOF MICO I/O block to the correct ports for your serial card. For more information, refer to the User Manual.

1. Move the first joint to +90 °, the second joint to -30 ° and joint 3 to zero by setting the command inputs. Looking at the virtual robot, determine the position of the robot end-effector (the distance between the grid lines is 10 cm).
2. Compare the position and orientation of the robot with the virtual robot from the simulation. Do they match?
3. With the last joint angle is set to zero, gradually change the first three joints of the robot so that the last joint axis of the robot (z_4 coming out of the hand parallel to the last joint axis) becomes parallel to the global x axis (If you forgot the frame axes directions, run the simulation model first). Derive the transformation matrix (${}^0_4T$) in this configuration using the joint angles, and discuss the rotation matrix (${}^0_4R = {}^0_4T(1:3, 1:3)$). Verify that the rotation matrix you derived with the output of the”MICO 4DOF FPK - Extended” block.

© 2016 Quanser Inc., All rights reserved.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. Quanser Inc. grants the following rights: a) The right to reproduce the work, to incorporate the work into one or more collections, and to reproduce the work as incorporated in the collections, b) to create and reproduce adaptations provided reasonable steps are taken to clearly identify the changes that were made to the original work, c) to distribute and publically perform the work including as incorporated in collections, and d) to distribute and publicly perform adaptations. The above rights may be exercised in all media and formats whether now known or hereafter devised. These rights are granted subject to and limited by the following restrictions: a) You may not exercise any of the rights granted to You in above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation, and b) You must keep intact all copyright notices for the Work and provide the name Quanser Inc. for attribution. These restrictions may not be waved without express prior written permission of Quanser Inc.

EXPERIMENT 2: MANIPULATOR INVERSE KINEMATICS

The purpose of this experiment is to investigate the Inverse Position Kinematics (IPK) of 4-DOF serial link robots. The following topics will be studied in this experiment.

Topics Covered

- The concept of inverse position kinematics (IPK)
- Geometric solution to IPK
- Existence of solution and multiple solutions to IPK

Prerequisites

- The robot has been setup and tested. See the Quick Start Guide for details.
- You have access to the User Manual.
- You are familiar with the basics of MATLAB® and SIMULINK®.

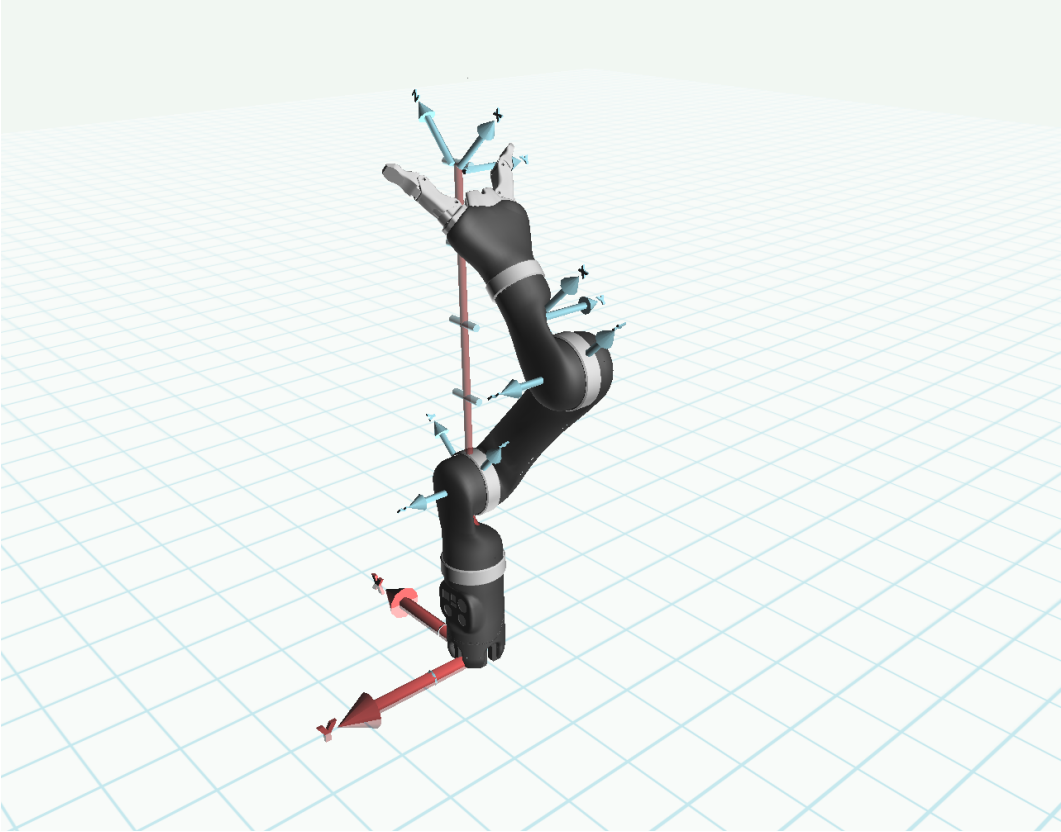


Figure 0.1: Inverse Kinematics is used to determine the joint angles of the robot given end-effector position.

1 Pre-Lab Preparations

1.1 Background

In most robotic applications, the robot end-effector needs to be able to reach a specific known position and/or orientation in space to achieve the desired task. Inverse kinematics solves the problem of finding the required joint angles to place the end-effector frame of the robot in a specific pose with respect to the base frame.

1.1.1 Solvability

In general, the problem of solving the kinematics equation of a manipulator arm is non-linear and non-trivial. The end-effector frame of a robot can be described by its transformation matrix, which is a 4$\times$ 4 matrix. Four of the elements are trivial (last row of the transformation matrix is [0 0 0 1]), and the remaining 12 elements of the transformation matrix are split into a 3$\times$1 vector position vector and a 3$\times$3 orientation matrix. However, the 9 parameters of the rotation matrix are not independent. The orientation of a frame with respect to another, can be described by 3 independent parameters called roll (rotation about x axis), pitch (rotation about y axis) and yaw (rotation about z axis).

1.1.2 Existence of Solutions

To achieve a desired pose in space, (x, y, z, roll, pitch and yaw parameters), six joint variables are necessary. Therefore, with a 4-DOF manipulator, the end-effector can not reach all positions in space with a desired orientation. As such, the 4DOF robot can be said to have no *dexterous workspace*, or volume of space where the robot end-effector can reach in any arbitrary orientation. The volume of space that the robot end-effector can reach with at least one orientation is called the *reachable workspace*.

For the 4-DOF MICO arm, since the first joint can freely move 360 °, the reachable workspace is when $(x^2 + y^2 + (z - (L_0 + L_1))^2 \leq L2 + L3 + L4$. Note that $x^2 + y^2 + (z - (L_0 + L_1))^2$ represents the distance of the origin of the end-effector from frame J_2. If this distance is greater than the sum of links 2,3 and 4 lengths, the robot cannot reach that point. Therefore, for any point in space that satisfies the above inequality, we will have a solution, provided that the resulting joint angles fall within the physical limitations of all joints. For the 4-DOF MICO arm, joints 2 and 3 are limited to -220° $< \theta_2 <$ 40° and -230° $< \theta_3 <$ 50°.

In this experiment, we solve for the possible joint positions for various positions of the robot end-effector, and study the possible orientations. The desired end-effector position for the robot has three elements: x, y, and z. Looking at the schematic of the robot in Figure 1.1 (side view), it is obvious that variations in the last joint of the arm (J_4) won't affect the position of the end-effector; the last joint can only change the orientation of the end-effector. As a result, we solve for the first three joint angles (θ_1, θ_2 and θ_3) given x, y and z.

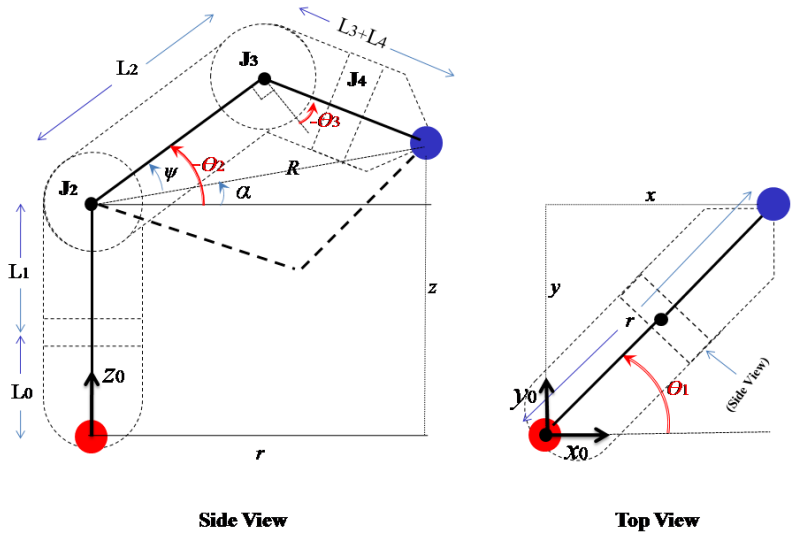


Figure 1.1: 2D top view, and side view schematics of the 4-DOF MICO robot arm.

The solutions to the inverse kinematics calculations can be either *numeric* or *closed-form*. Numeric solutions specify values, whereas closed-form solutions are expressions that are found using *algebraic* equations or *geometric* calculations. Here, we provide you with the closed-from solution using a geometric approach.

1.1.3 Geometric Solution

The geometric approach to inverse kinematics is an intuitive and usually simpler approach to solve for unknown joint angles given an end-effector position. In this method we try to decompose the spatial geometry of the arm into simple planar-geometry problems.

The motion of the 4-DOF MICO robotic arm can be decomposed in to a moving plane that rotates around the z axis when the first joint (J_1) changes. Figure 1.1 shows two different planar views of the robot arm. The left image shows the side view of the moving plane (perpendicular to the plane), and he right shows the top view (from the first joint axis looking down).

Looking at Figure 1.1 (side view), we can geometrically derive r and z as follows

$$r = (L_2 \cos(\theta_2) - (L_3 + L_4) \sin(\theta_2 + \theta_3)) \tag{1.1}$$
$$z = L_0 + L_1 - L_2 \sin(\theta_2) - (L_3 + L_4) \cos(\theta_2 + \theta_3) \tag{1.2}$$

The top view will result in x and y as follows

$$x = r \cos(\theta_1) \tag{1.3}$$
$$y = r \sin(\theta_1) \tag{1.4}$$

where r is shown in Equation 1.1. Therefore,

$$\theta_1 = \text{atan2}(y, x) \tag{1.5}$$

Also, from Figure 1.1 (side view) we can see that

$$R = \sqrt{r^2 + (z - (L_0 + L_1))^2}. \tag{1.6}$$

To calculate the joint angles θ_2 and θ_3, we need to first calculate α and ψ. The angle α can be calculated using the following equation:

$$\alpha = \arctan(z - (L_0 + L_1), r) = \arctan\frac{z - (L_0 + L_1)}{\sqrt{x^2 + y^2}} \tag{1.7}$$

where arctan is calculated using the atan2 command. To geometrically calculate ψ, given x, y and z, we will use the *law of cosines*. The law of cosines, also known as the cosine formula or cosine rule, describes the relationship between the lengths of the sides of a triangle and the cosine of one of the angles as shown in Figure 1.2 where $c^2 = a^2 + b^2 - 2ab\cos(\gamma)$.

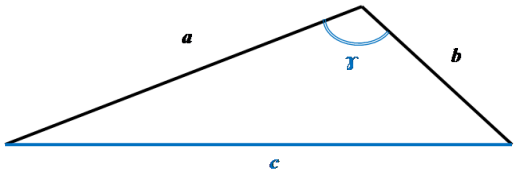


Figure 1.2: Law of cosines.

Applying the law of cosines to the triangle represented by R, L_2 and $L_3 + L_4$ and the angle ψ we have

$$(L_3 + L_4)^2 = R^2 + L_2^2 - 2L_2R\cos\psi \Rightarrow \psi = \arccos\left(\frac{R^2 + L_2^2 - (L_3 + L_4)^2}{2L_2R}\right) \tag{1.8}$$

where R is defined in Equation 1.6.

Also, applying the law of cosines to the same triangle with angle $\pi/2 - \theta_3$ we can derive θ_3 as follows

$$\theta_3 = \arcsin\left(\frac{L_2^2 + (L_3 + L_4)^2 - (x^2 + y^2 + (z - (L_0 + L_1))^2)}{2L_2(L_3 + L_4)}\right) \tag{1.9}$$

Now, looking at Figure 1.1, we can achieve the following two sets of solutions for the joint angles

Solution set 1:

$$\begin{aligned}\theta_1 &= \arctan({}^{y}/_{x}) \\ \theta_2 &= -(\alpha + \psi) \\ \theta_3 &= \arcsin\left(\frac{L_2^2 + (L_3 + L_4)^2 - (x^2 + y^2 + (z - (L_0 + L_1))^2)}{2L_2(L_3 + L_4)}\right)\end{aligned} \tag{1.10}$$

Solution set 2:

$$\begin{aligned}\theta_1 &= \arctan({}^{y}/_{x}) \\ \theta_2 &= -(\alpha - \psi) \\ \theta_3 &= -\pi - \arcsin\left(\frac{L_2^2 + (L_3 + L_4)^2 - (x^2 + y^2 + (z - (L_0 + L_1))^2)}{2L_2(L_3 + L_4)}\right)\end{aligned} \tag{1.11}$$

1.2 Pre-Lab Exercise

1. Discuss the two solution sets in Equation 1.10 and Equation 1.11. How should one chose the right solution?
2. Solve for the inverse kinematics of the robot where $x = 0$, $y = 0$ and $z = 0.7$ m

   Note: $L_0 + L_1 = 0.2755$, $L_2 = 0.29$, $L3 = 0.1233$ and $L_4 = 0.16$
3. Draw the schematic of the robot in the configuration from Question 2, and show the configurations related to the solutions. Are there other solutions for this position?
4. Does the $x = 0$, $y = 0$ and z change if you rotate the first joint?

2 In-Lab Exercise

2.1 Simulation

The QUARC model for this exercise is ”MICO_Inverse_Kinematics_Simulation.mdl” a snapshot of which shown in Figure 2.1.

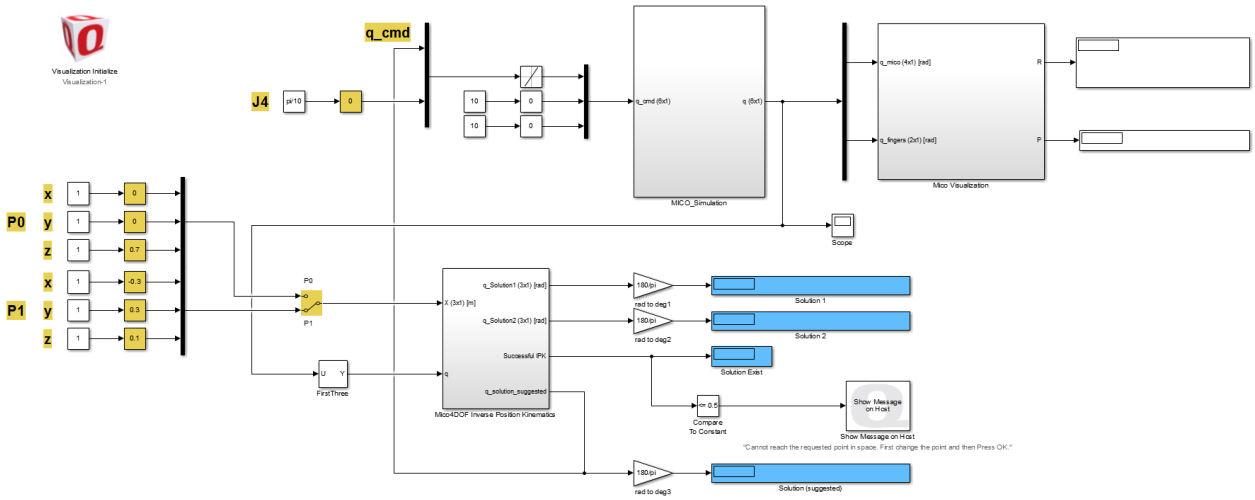


Figure 2.1: Snapshot of the controller model ”MICO_Inverse_Kinematics_Simulation.mdl”

Compile and run the model. A the Quanser 3D Viewer window should open, showing a visualization of the robot. Go through the following steps and answer the corresponding questions.

1. Set the x, y, and z values of the **P0** input command to 0, 0, and 0.7 m respectively (Use the slider gains highlighted in yellow). Observe the outputs ”Solution 1” and ”Solution 2” and verify your Pre-lab calculations. Compare the pose of the robot with the schematic from Question 3 in the Pre-Lab exercise.
2. How is the y axis of the end-effector frame oriented with respect to the global (or base frame)? Hold down the middle mouse button, and move the mouse to rotate the camera view of the 3D Viewer. Describe the orientation of the robot in relation to the rotation matrix?
3. Change the last joint angle, **J4**, so that the x axis aligns with the global y axis. What is J4’s joint angle?
4. Assume you want to program the robot to pick up an object, the centre of which located at $x = -0.3\,\text{m}$, $y = 0.3\,\text{m}$, and $z = 0.1\,\text{m}$. Using the simulation model, derive the required joint angles. Switch the input position command to **P1**, and set the x, y and z values to verify that the robot end-effector safely goes from one point to the other (The end-effector of the virtual robot does not hit the ground).
5. What is the path of the robot end-effector when you switch between **P1** and **P0**? How can one control this path?

Once you feel comfortable with the above exercise on the virtual robot, you are ready to proceed to the next section.

2.2 Experiment

The QUARC model for this exercise is ”MICO_Inverse_Kinematics_Experiment.mdl” the snapshot of which shown in Figure 2.2.

Before running the model, manually move the robot arm with the power turned off into a comfortable pose where the robot end-effector is away from the table (preferably a elbow-up pose), and turn on the robot to hold its pose. Make

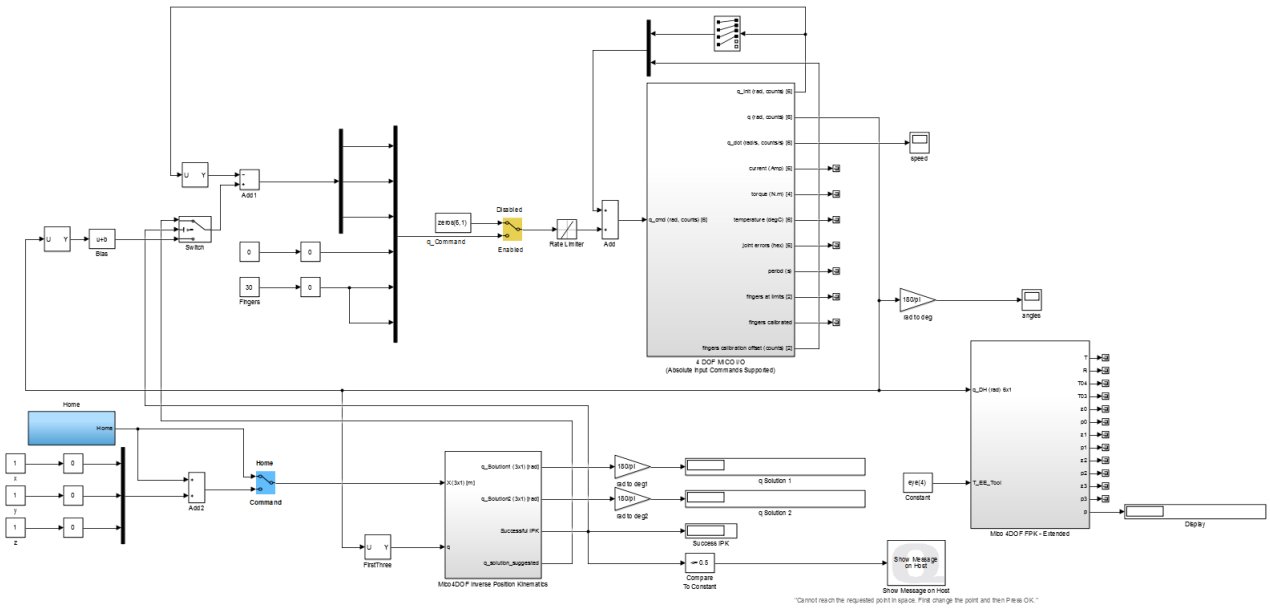


Figure 2.2: Snapshot of the controller model ”MICO_Inverse_Kinematics_Experiment.mdl”

sure that the robot is disabled (the yellow manual switch is set to ”Disabled”), and the blue manual switch is set to ”Home”.

Caution

Be sure to set the Ports in the 4-DOF MICO I/O block to the correct ports for your serial card. For more information, refer to the User Manual.

1. Set x, y, and z values of the **P0** input command to 0, 0, and 0.7 m respectively (Use the slider gains highlighted in yellow). Observe the outputs ”Solution 1” and ”Solution 2” and verify the values by comparing them to the ones you observed from the previous section. Compare the pose of the robot with the pose of the virtual robot you observed in the previous section. Enable the robot motion and observe the robot moving to **P0**.

2. What is the actual position of the robot end-effector (the output P from the forward Kinematics block highlighted in blue). What is the error between the commanded position and the output position? What do you think is the source of error?

3. Change the **P1** components to $x = -0.3$, $y = 0.3$ and $z = 0.1$ and switch the input position command to **P1**. Observe robot motion and compare it to the motion you observed on the virtual robot.

4. While the position command **P1** is selected, change **P0** to **P0** = [0.5 0 0.7] and then switch the input command to **P0**. Can the robot reach this point? Why?

© 2016 Quanser Inc., All rights reserved.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. Quanser Inc. grants the following rights: a) The right to reproduce the work, to incorporate the work into one or more collections, and to reproduce the work as incorporated in the collections, b) to create and reproduce adaptations provided reasonable steps are taken to clearly identify the changes that were made to the original work, c) to distribute and publically perform the work including as incorporated in collections, and d) to distribute and publicly perform adaptations. The above rights may be exercised in all media and formats whether now known or hereafter devised. These rights are granted subject to and limited by the following restrictions: a) You may not exercise any of the rights granted to You in above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation, and b) You must keep intact all copyright notices for the Work and provide the name Quanser Inc. for attribution. These restrictions may not be waved without express prior written permission of Quanser Inc.

EXPERIMENT 3: MANIPULATOR JACOBIAN

The purpose of this experiment is to study the relationship between the linear and angular velocity of the end-effector of a robot, and the velocity of the individual joints of a robot arm. The following topics will be studied in this experiment.

Topics Covered

- Linear and angular velocities of rigid bodies
- The notion of Jacobian
- Singularities
- Static forces and Jacobian

Prerequisites

- The robot has been setup and tested. See the Quick Start Guide for details.
- You have access to the User Manual.
- You are familiar with the basics of MATLAB® and SIMULINK®.

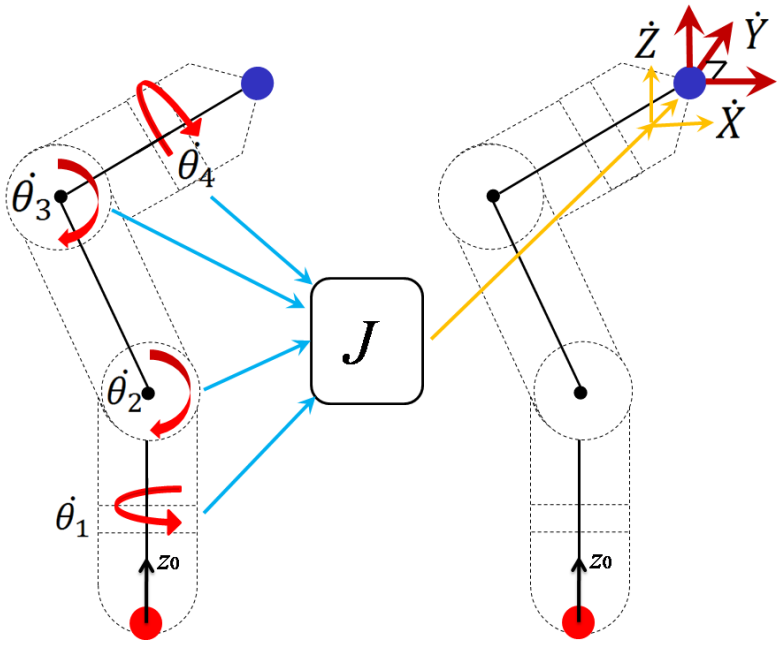


Figure 0.1: Manipulator Jacobian is used to determine the end-effector speed or forces (Cartesian Space) of the robot arm given individual joint speeds or torques (Joint Space).

1 Pre-Lab Preparations

1.1 Background

In this experiment, we study the linear and angular velocity of manipulators as well as the static forces and moments acting on manipulators. Both of these topics are closely related to a property of manipulators called *Jacobian* or *Velocity Kinematics*.

1.1.1 Linear and Angular Velocity

The linear and angular velocity of a rigid body can be achieved by differentiating position vectors. We use the following notation for the derivative of Q relative to frame B, expressed in frame B.

$$^{B}V_{Q} = \frac{d}{dt}{}^{B}Q = \lim_{\Delta t \to 0} \frac{^{B}Q(t+\Delta t) - {}^{B}Q(t)}{\Delta t} \tag{1.1}$$

To describe this velocity vector in frame A, we can write

$$^{A}(^{B}V_{Q}) = {}^{A}_{B}R\ {}^{B}V_{Q}. \tag{1.2}$$

If the origin of frame B is also linearly moving relative to frame A, we have

$$^{A}V_{Q} = {}^{A}\ V_{Borig} + {}^{A}_{B}R\ {}^{B}V_{Q}. \tag{1.3}$$

While linear velocity describes an attribute of a **point** in space, angular velocity describes an attribute of a rigid **body** (to which a frame is attached). Therefore, the angular velocity vector describes the rotational motion of a frame.

A notation similar to linear velocity is defined for angular velocity. Here $^{C}(^{A}\Omega_{B})$ indicates the rotation of frame B relative to frame A described in frame C.

Now, if the frame B rotates with the rotational velocity $^{A}\Omega_{B}$ relative to frame A, the motion of this vector as viewed in Frame A can be written as follows

$$^{A}V_{Q} = {}^{A}\ V_{Borig} + {}^{A}_{B}R\ {}^{B}V_{Q} + {}^{A}\Omega_{B} \times {}^{A}_{B}R\ {}^{B}Q \tag{1.4}$$

1.1.2 Velocity Propagation for Serial Link Manipulators and Manipulator Jacobian

The following equations can be used to propagate the linear and angular velocities for revolute-joint manipulators.

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}\ R\ {}^{i}\omega_{i} \tag{1.5}$$

$$^{i+1}v_{i+1} = {}^{i+1}_{i}R\ (^{i}v_{i} + {}^{i}\ \omega_{i} \times {}^{i}\ P_{i+1}) \tag{1.6}$$

Applying these equations successively from link to link, we can compute the linear and angular velocity of the end-effector frame with respect to the base frame. These equations can be written in a compact matrix format that relates the linear and angular velocities of the end-effector in Cartesian space to the joint angles of the robot arm

$$\dot{X} = \begin{bmatrix} v \\ \omega \end{bmatrix} = J\dot{q} \tag{1.7}$$

where v is the linear velocity of the end-effector frame, w is the angular velocity of the end-effector frame, $\dot{q}$ is the vector of the joint angles and matrix J is called the Jacobian matrix and can be written as follows (for revolute joint manipulators)

$$J = \begin{bmatrix} J_1 & \dots & J_N \end{bmatrix}, \quad J_i = J = \begin{bmatrix} J_{vi} \\ J_{\omega i} \end{bmatrix}, \quad J_{vi} = z_{i-1} \times (O_n - O_{i-1}), \quad J_{\omega i} = z_{i-1} \tag{1.8}$$

where O_j is the centre of frame j and z_i is the joint axis related to frame i.

Alternatively, in order to achieve the linear velocity components of the Jacobian, J_{vi}, we can calculate the partial derivative of the position vector of the end-effector of the robot with respect to the joint angles. The angular velocity component of the Jacobian is simply z_{i-1}.

1.1.3 Static Forces

The Jacobian matrix J also relates the forces and moments that a manipulator exerts at its end-effector, F, to the corresponding torques required at the joints, τ, as shown below.

$$\tau = J^T F \tag{1.9}$$

This theorem can be proved using the principle of virtual work.

1.1.4 Manipulator Singularities

As discussed, the Jacobian matrix J defines a mapping between the joint velocities ($\dot{q}$) and the end-effector velocities $\dot{X}$, which is a function of the arm pose determined by joint angles q. The Jacobian defines the linear transformation $dX = Jdq$ between the differentials or infinitesimal motions. To achieve a desired motion for the end-effector, the inverse solution is required. However, in some configurations, the rank of matrix J decreases; such configurations are called singularities. Therefore, singularities represent configurations or poses from which certain directions in the space may be unattainable by the robot arm.

At singularities, bounded end-effector velocities may correspond to unbounded or infinite joint velocities. Similarly, at singularities, bounded end-effector forces and torques may correspond to unbounded or infinite joint torques. These solutions result in undesired and unsafe motion of the robot.

1.2 Pre-Lab Exercise

1. Discuss the procedure you would follow to derive the Jacobian of the 4DOF MICO robot. Derive $J_{\omega i}$ for i = 1 and 2 at DH home (where all joint angles are zero).

   Note: you are not required to go through matrix calculations for the rest of the Jacobian matrix.

2. Write a MATLAB function to receive a transformation matrix of a tool and the joint angle and calculate the Jacobian matrix. What is the Jacobian matrix at zero DH when there is no tool attached?

   Hint: you can use the available DH() function in your code to derive ${}^{i-1}_{i}T$ and multiply these matrices to derive ${}^{0}_{i}T$.

3. Describe different possibilities for singularity of the 4DOF MICO arm. What would happen in case of singularities?

2 In-Lab Exercise

2.1 Simulation

The QUARC model for this exercise is ”MICO_Jacobian_Simulation.mdl” a snapshot of which shown in Figure 2.1.

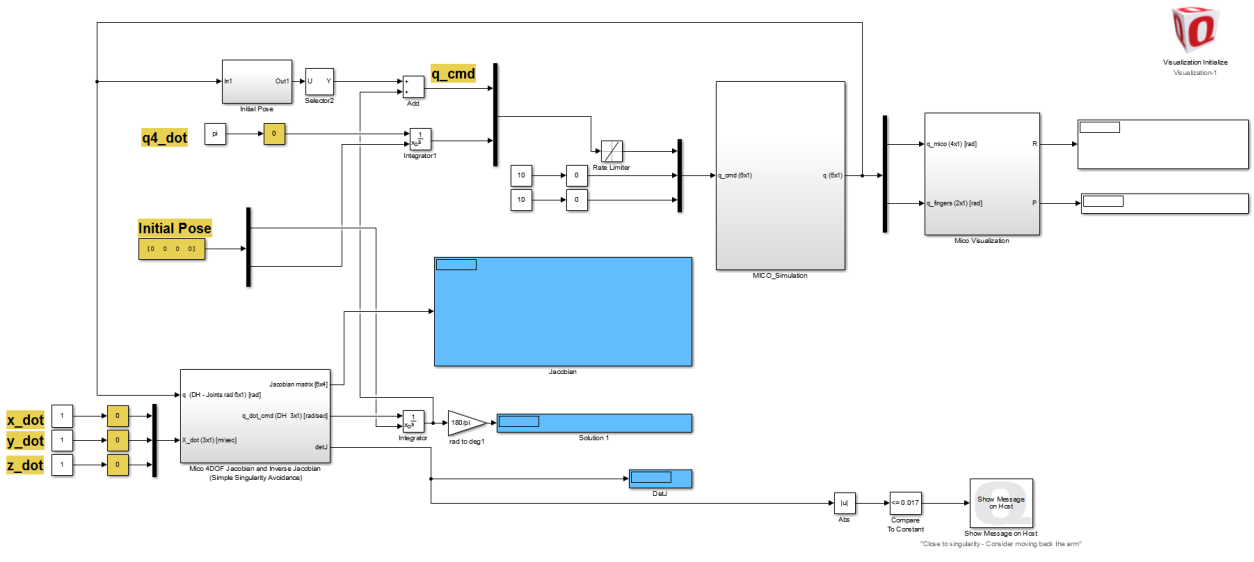


Figure 2.1: Snapshot of the controller model ”MICO_Jacobian_Simulation.mdl”

Make sure that the slider gains for x_dot, y_dot, z_dot and q4_dot inputs are set to zero and the initial pose is set to [0 0 0 0]. Compile and run the model. The Quanser 3D Viewer should open, showing a visualization of the robot. Go through the following steps and answer the corresponding questions.

1. What is the Jacobian matrix at DH zero? Compare this with the output of your function in the Pre-lab section.
2. What is the meaning of the last column of the J matrix? Discuss it using the virtual robot motion and configuration.
3. Stop the model and change the initial pose to [0 $-\pi/2$ 0 0]. Run the model again. Now slowly change the x_dot input command to +0.01 using the slider gain. What do you observe? Do the same thing with the y_dot and z_dot commands and discuss your observations.
4. Describe the model. In particular describe how the joint position commands are generated from the x_dot commands.
5. Using the Jacobian when the joint angles are $q_0 = [0, -\pi/2,\ 0,\ 0]$, what would be the required torque be for the four joints of the robot to apply -5 N at its end-effector along the z axis (with no moments)? Discuss your results.

Once you feel comfortable with the above exercises using the virtual robot, go to the next section.

2.2 Experiment

The QUARC model for this exercise is ”MICO_Jacobian_Experiment.mdl” the snapshot of which shown in Figure 2.2.

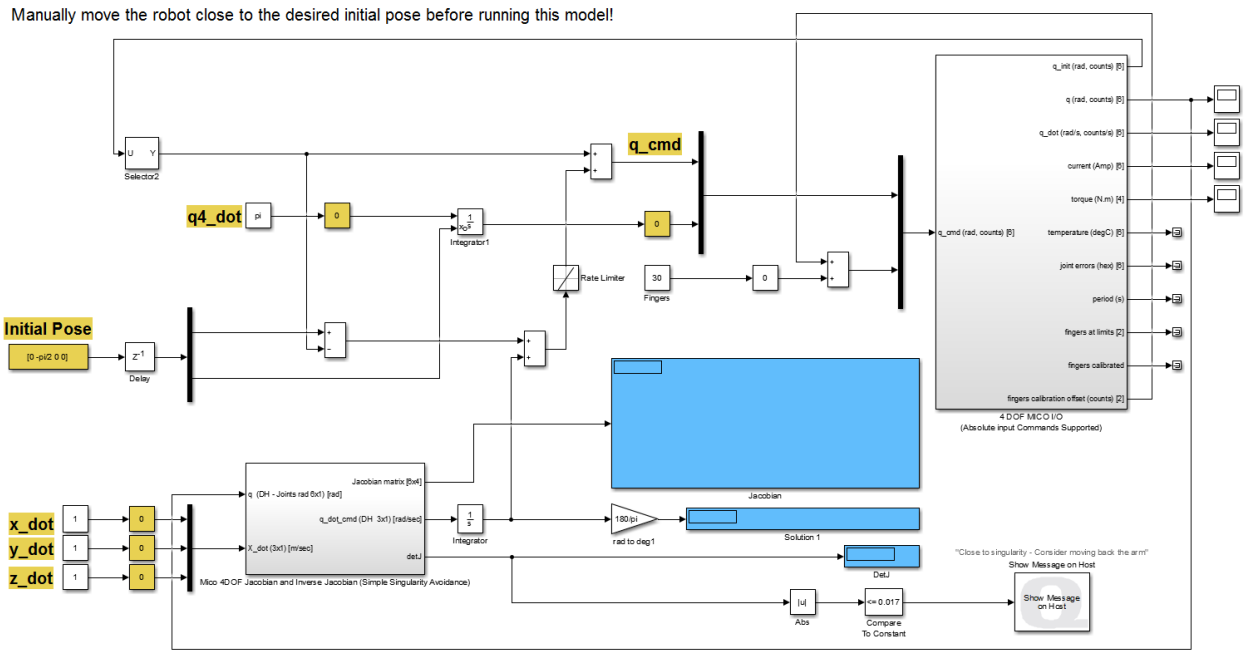


Figure 2.2: Snapshot of the controller model "MICO_Jacobian_Experiment.mdl"

Turn off the robot and manually pose it to roughly $q = [0, -\pi/2, 0, 0]$ as was studied in the simulation section. Turn on the robot. Also make sure that the slider gains for the x_dot, y_dot, z_dot and q4_dot inputs are set to zero. Compile and run the robot, and it will go to the home pose $q = [0, -\pi/2, 0, 0]$.

Caution

Be sure to set the Ports in the 4-DOF MICO I/O block to the correct ports for your serial card. For more information, refer to the User Manual.

1. What is the Jacobian matrix of the robot in this pose? Compare it with the values from your Pre-lab and the simulation section. Explain your observations.
2. Change the x_dot command to 0.01 (1 cm/sec) by clicking and changing the corresponding slider gain.
3. Change the slider gain value back to zero after the robot moves about 10-15 cm.
4. Move the robot along y and z axis the same way you did for the x axis and stop the robot.
5. Send the robot to the home position using the manual switch (switch to home pose $q = [0, -\pi/2, 0, 0]$).
6. Find an object of a known weight (around 0.5 Kg). Using the finger control slider gains to open and close the fingers, hold the object.
7. Read the torque values (output from MICO I/O block). Compare the results with the ones you calculated in the previous section.

© 2016 Quanser Inc., All rights reserved.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. Quanser Inc. grants the following rights: a) The right to reproduce the work, to incorporate the work into one or more collections, and to reproduce the work as incorporated in the collections, b) to create and reproduce adaptations provided reasonable steps are taken to clearly identify the changes that were made to the original work, c) to distribute and publically perform the work including as incorporated in collections, and d) to distribute and publicly perform adaptations. The above rights may be exercised in all media and formats whether now known or hereafter devised. These rights are granted subject to and limited by the following restrictions: a) You may not exercise any of the rights granted to You in above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation, and b) You must keep intact all copyright notices for the Work and provide the name Quanser Inc. for attribution. These restrictions may not be waved without express prior written permission of Quanser Inc.

EXPERIMENT 4: TRAJECTORY PLANNING

The purpose of this experiment is to learn methods of computing trajectories that describe the desired motion of manipulators in 3D space. The following topics will be studied in this experiment.

Topics Covered

- The notion of trajectory
- Cubic and higher-order polynomials
- Cartesian space trajectory planning

Prerequisites

- The robot has been setup and tested. See the Quick Start Guide for details.
- You have access to the User Manual.
- You are familiar with the basics of MATLAB® and SIMULINK®.

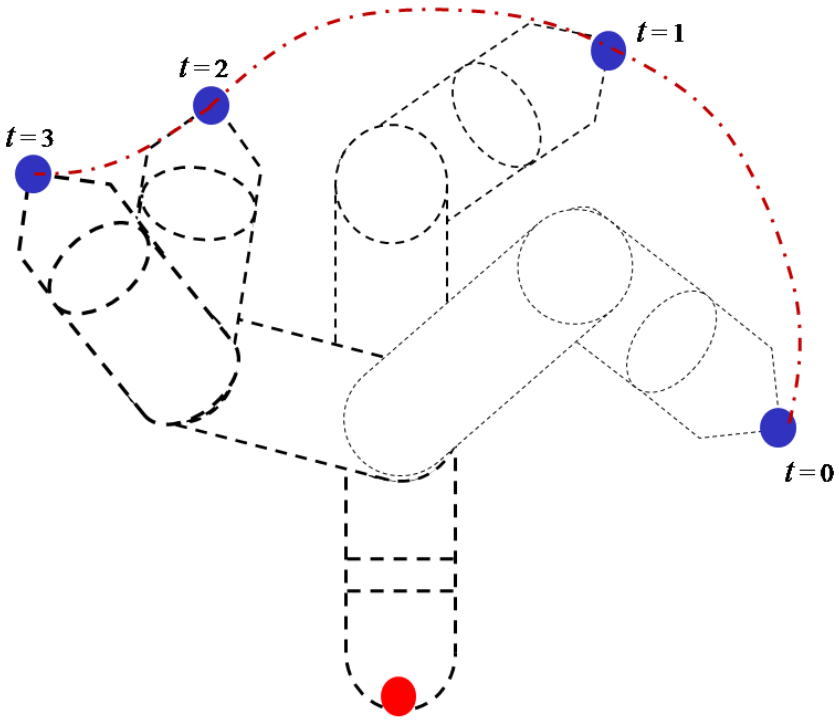


Figure 0.1: Robot trajectory is designed such that the manipulator moves from its initial position to the desired goal position in a smooth manner.

1 Background

In the context of robotics, *trajectory* refers to the time history of position, velocity and acceleration for each degree of freedom, in either the joint space or Cartesian space.

In various robotics applications, the end-user cannot be involved in determining complicated functions for the robot motion. Therefore, the robot program should be able to create the motion path of the robot using limited information such as the desired goal pose (position and orientation) of the robot end-effector, the attached tool, or a set of way points (via points) along the desired path.

In this laboratory you will learn how robot trajectories can be computed, represented, generated and applied to robots.

1.1 Path Description and Generation

As mentioned earlier, the end users of robot manipulators are mostly concerned about the motion of the tool frame of the robot attached to the end-effector, relative to the base or station frame. Therefore, in this lab we focus on the Cartesian-space trajectory planning, although these methods can be directly applied to joint-space schemes.

The motion of the robot includes both position and orientation. Since we are dealing with a 4-DOF robotic manipulator, if we want to control the position of the robot end-effector in space, we require three degrees of freedom. We use the rotation of the last joint (wrist) of the robot as the remaining degree of freedom of the robot that can be specified and controlled. For complete control of the 6 possible degrees of freedom of the end effector, the manipulator would need to be equipped with a complete 3-DOF wrist.

In most cases, it is necessary to specify the movement of the robot in a detailed way to achieve a specific task. Consider a factory in which a robotic arm has to reach a part without touching other obstacles around it. The user needs to provide a sequence of **way points** that represent intermediate points between the initial and final points. Depending on the design scheme, each way point can include different information. In a Cartesian scheme, each way point describes the position and orientation of the robot. In joint-space scheme, each way point represents all of the joint angles.

Along with the spatial constraints each way point usually contains some temporal attributes, such as the time elapsed between the way points. Having a smooth motion is also usually a constraint that is placed on the motion and controlled using various planning techniques. Usually, a trajectory function that is continuous and has a continuous first derivative will generate a ”smooth” trajectory. For a ”minimum jerk” trajectory, a continuous second order derivative is desirable. Minimum jerk motions minimize wear on the robot mechanisms and are desirable for many applications including accurate image analysis for robotic vision inspection.

1.2 Cubic vs. Quintic Polynomial Trajectory Planning

In order to achieve the desired smooth motion *between* any two consecutive way points, say $\mathbf{X}_0$ and $\mathbf{X}_1$, we need to satisfy the following constraints

1. $\mathbf{X}(t=0) = \mathbf{X}_0$
2. $\mathbf{X}(t=1) = \mathbf{X}_1$
3. $\dot{\mathbf{X}}(t=0) = \mathbf{V}_0$
4. $\dot{\mathbf{X}}(t=1) = \mathbf{V}_1$,

where the last two constraints are necessary for smooth operation between the way points and $\mathbf{V}_0$ and $\mathbf{V}_1$ should be zero for the initial and final way points. In order to satisfy the above four constraints, a cubic polynomial (that has

for parameters) is required. As mentioned earlier, for a minimum jerk trajectory, it is desirable to have a continuous second-order derivative which yields the following two constraints

5. $\ddot{\mathbf{X}}(t = 0) = 0$
6. $\ddot{\mathbf{X}}(t = 1) = 0.$

In order to satisfy the above six constraints a quintic polynomial is required.

2 Pre-Lab Questions

1. Open the MATLAB script called mico4DOF_generate_cubic.m and explain what the code is doing. Be sure to identify the key portions of the script.
2. Run the script with the default way points and plot the trajectory over time. Does the trajectory go through the way points? Is it smooth?
3. Describe how the trajectory is generated and plot the planar motion of the robot (y , z planar motion).
4. How can you show that the derivative of the trajectory is zero at the way points? What does this mean when you apply the trajectory to the robot?
5. Change the way points so that the robot moves from $x = -0.4$ to $x = 0.4$ while $y = +0.2$, $z = 0.4$ and $\theta_4 = 0$ (taking 0.1 steps for x; you will need 9 way points). Tune the variable speed (between 0 to 0.01) and plot the trajectories until you find a very smooth motion. Assuming that the robot begins to move at 2 s and completes the movement after 60 s, what is the ideal speed?
6. Do you think quintic polynomials are needed for the above way points? Discuss your conclusions. Open and run the mico4DOF_generate_cubic.m polynomials for the above way points with various speeds and discuss your findings.

3 Lab Experiments

3.1 Simulation

The QUARC model for this exercise is ”MICO_Trajectory_Planning_Simulation.mdl” a snapshot of which shown in Figure 3.1. This model works with the two trajectory generation MATLAB scripts, mico_generate_cubic.m and mico_generate_quintic.m.

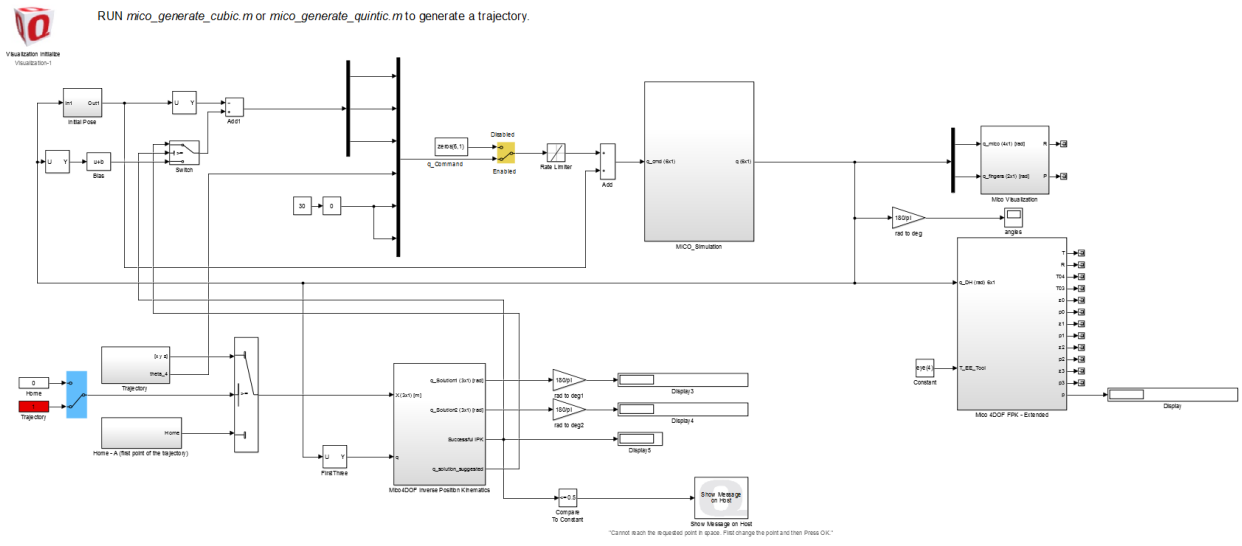


Figure 3.1: Snapshot of the controller model ”MICO_Trajectory_Planning_Simulation.mdl”

mico_generate_cubic.m generates a smooth trajectory based on cubic polynomials given a set of way points (or via points) and the corresponding velocities. **mico_generate_cubic.m** generates a smooth trajectory based on quintic polynomials given a set of way points and the corresponding velocities and zero accelerations. These two scripts generate a trajectory in the MATLAB workspace called ”Traj”. The QUARC model ”MICO_Trajectory_Planning_Simulation.mdl” applies this workspace variable to the model of the robot arm.

The MICO Simulation block, receives the joint angles and computes the position of the end-effector. Compile and run the model, then follow the procedure outlined below. Make observations on the motion of the robot, and answer the associated questions.

1. Use the default way points that were generated in the Pre-Lab for circular motion of the arm.
2. Open the ”MICO_Trajectory_Planning_Simulation.mdl” model, compile and run it making sure the yellow and blue switches are set to *trajectory*. Observe the motion of the virtual robot.
3. Use the set of way points you created from the Pre-Lab exercise section and run the robot. Compile the QUARC model ”MICO_Trajectory_Planning_Simulation.mdl” and run it again. Describe the motion of the virtual robot.

3.2 Experiment

In this section, you will experimentally evaluate the trajectory generation algorithms on the robot arm. During the experiment, always keep a safe distance from the robot and evaluate your results on the virtual robot first before applying your commands to the actual robot platform.

The QUARC model for this experiment is called ”MICO_Trajectory_Planning_Experiment.mdl” shown in Figure 3.2 which also works with the two trajectory generation MATLAB scripts that were studied before (cubic and quintic).

Before running the model, manually move the robot arm in a comfortable pose where the robot end-effector is away from the table (preferably a elbow-up pose), and turn on the robot to hold its pose. Make sure that the robot is disabled (the yellow manual switch is set to ”Disabled”) and the blue manual switch is set to ”Home”.

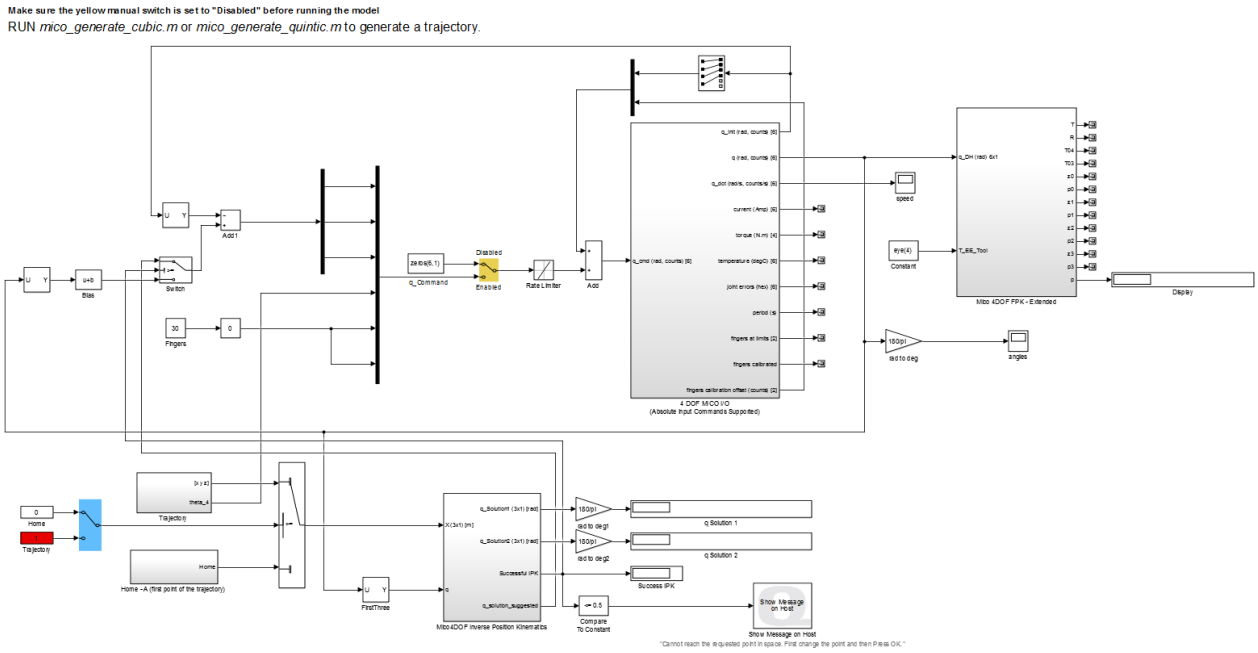


Figure 3.2: Snapshot of the controller model ”MICO_Trajectory_Planning_Experiment.mdl”

Go through the steps below and answer the questions.

1. Implement the way points that were created for a circular path. From the plots, confirm that the desired motion is achieved.
2. Open ”MICO_Trajectory_Planning_Experiment.mdl”, compile and run it making sure the blue switch is set to ”Home”. When the robot is in the home position, toggle the blue manual switch and observe the motion of the robot.

Caution

Be sure to set the Ports in the 4-DOF MICO I/O block to the correct ports for your serial card. For more information, refer to the User Manual.

3. Use the set of way points you created from the Pre-Lab exercise section, and evaluated with the virtual robot in simulation. Run the trajectory generation mathscript, and make sure that the desired motion along the x axis is achieved (look at the plots).
4. Compile the QUARC model ”MICO_Trajectory_Planning_Experiment.mdl”, connect to the target and run it. Describe the motion of the robot.
5. How would you use trajectory generation for assembly line automation in a factory?

© 2016 Quanser Inc., All rights reserved.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. Quanser Inc. grants the following rights: a) The right to reproduce the work, to incorporate the work into one or more collections, and to reproduce the work as incorporated in the collections, b) to create and reproduce adaptations provided reasonable steps are taken to clearly identify the changes that were made to the original work, c) to distribute and publically perform the work including as incorporated in collections, and d) to distribute and publicly perform adaptations. The above rights may be exercised in all media and formats whether now known or hereafter devised. These rights are granted subject to and limited by the following restrictions: a) You may not exercise any of the rights granted to You in above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation, and b) You must keep intact all copyright notices for the Work and provide the name Quanser Inc. for attribution. These restrictions may not be waved without express prior written permission of Quanser Inc.

EXPERIMENT 5: AUTOMATION CHALLENGE

The purpose of this challenge is to implement an example of manipulator control in a real-world factory automation context. The scenario that is presented involves many of the topics covered in the 4DOF MICO curriculum including kinematics and trajectory generation. In addition, the concept of state machines is introduced to bring the components of the overall system together.

Topics Covered

- Automation tasks
- State machines

Prerequisites

- The robot has been setup and tested. See the Quick Start Guide for details.
- You have access to the User Manual.
- You are familiar with the basics of Matlab® and Simulink®.
- You should have an understanding of the previous topics covered including kinematics and trajectory generation

1 Challenge

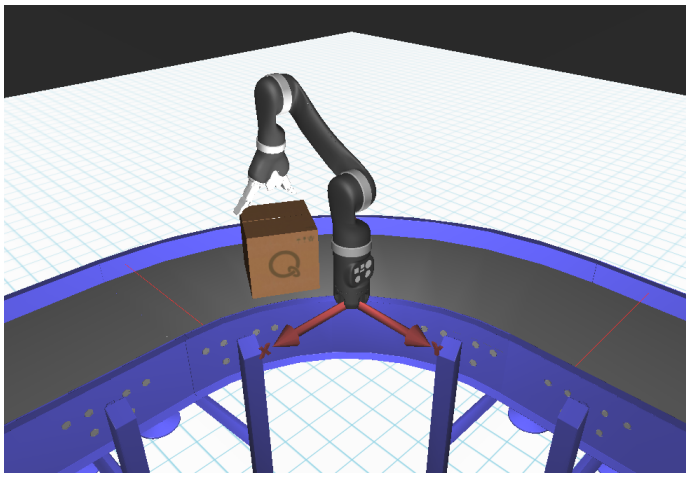


Figure 1.1: Automation Challenge.

The scenario that is presented in this challenge is that of a typical automation task. The 4DOF MICO has been placed in a factory, alongside a conveyor system. The workspace for the manipulator is defined as the reachable area between the two break beam photosensors on either side of the curved portion of the belt. Your task is to define a trajectory for the manipulator to follow in order to "inspect" the box when it is inside the workspace. The sequence of commands that bring the manipulator to the initial position of the box when it enters the workspace, then follow the box until it leaves the workspace, and finally return the manipulator to the home position, should be defined using a state machine. The easiest method for creating state machines in Matlab® is to use the Stateflow toolbox. The toolbox allows you to define states and transitions based on conditional statements and workspace variables. Each state can then call command arguments, or even custom functions. For more information, refer to the Stateflow documentation.

In the case of the automation challenge, the states that are defined will be determined using the break beam sensors. When the box breaks the first beam and enters the workspace, the `inspect` variable will become *true*. The box then moves along the curve shown in Figure 1.2, with velocity defined by the `vel` variable. As the box leaves the workspace, the `inspect` variable becomes *false* and the manipulator should return to the home position defined by the `home` variable.

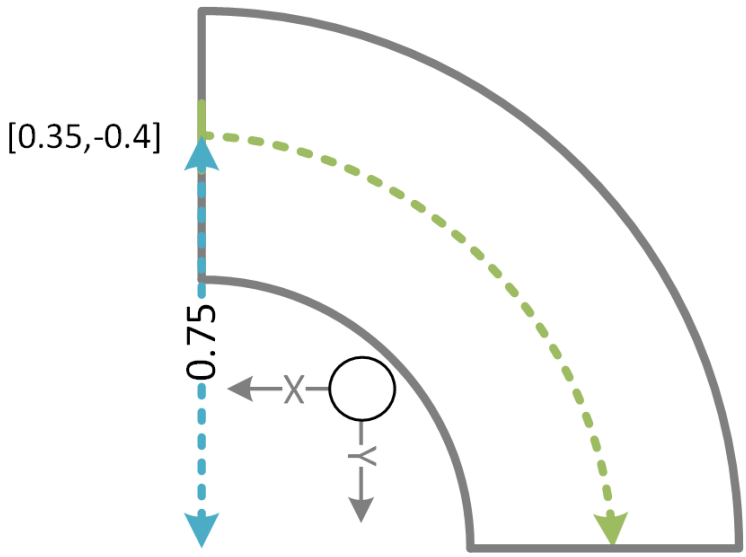


Figure 1.2: Workspace for the automation challenge.

© 2016 Quanser Inc., All rights reserved.

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada
info@quanser.com
Phone: 1-905-940-3575
Fax: 1-905-940-3576

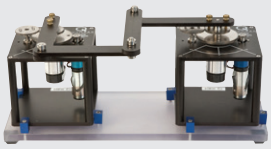
Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com

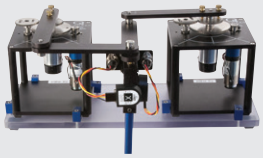
This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. Quanser Inc. grants the following rights: a) The right to reproduce the work, to incorporate the work into one or more collections, and to reproduce the work as incorporated in the collections, b) to create and reproduce adaptations provided reasonable steps are taken to clearly identify the changes that were made to the original work, c) to distribute and publically perform the work including as incorporated in collections, and d) to distribute and publicly perform adaptations. The above rights may be exercised in all media and formats whether now known or hereafter devised. These rights are granted subject to and limited by the following restrictions: a) You may not exercise any of the rights granted to You in above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation, and b) You must keep intact all copyright notices for the Work and provide the name Quanser Inc. for attribution. These restrictions may not be waved without express prior written permission of Quanser Inc.

Solutions for teaching and research in robotics and autonomous systems

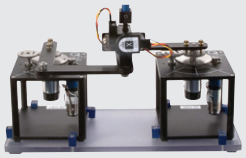
2 DOF Robot



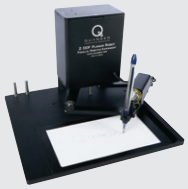
2 DOF Gantry



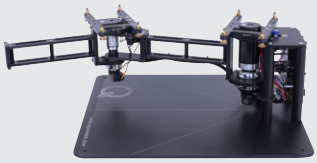
2 DOF Inverted Pendulum



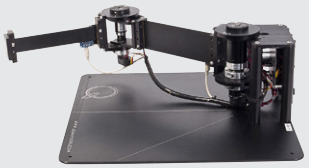
2 DOF Planar Robot



2 DOF Serial Flexible Joint

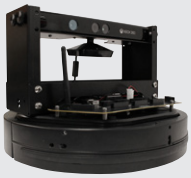


2 DOF Serial Flexible Link



6 DOF Denso Open Architecture Robot

QBot 2



QBall 2

Unmanned Vehicle Systems Lab

With Quanser robotic systems, you can introduce control concepts related to stationary and mobile robotics, from vibration analysis, resonance and planar position control to sensors, computer, vision-guided control to unmanned systems control. All of the experiments/platforms are compatible with MATLAB®/Simulink®.

©2015 Quanser Inc. All rights reserved.

QUANSER
INNOVATE · EDUCATE

INFO@QUANSER.COM +1-905-940-3575 QUANSER.COM

Solutions for teaching and research. Made in Canada.