Assignment II – CSC376 – Fall 2019

Fundamentals of Robot Design
Department of Mathematical and Computational Sciences
University of Toronto Mississauga

Abstract

This assignment covers various topics associated with forward and velocity kinematics (Module 4 & 5). It consists of three parts. In the first part, the forward kinematics of a serial manipulator have to be derived and implemented. In the second part, the velocity kinematics of the serial manipulator have to be determined. In the third part, different robot configurations are evaluated.

Remember to write your full names and student numbers prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted.

For Assignment II and Assignment III you are allowed to work in groups of 2 students or by yourself. Please note that the group cannot be changes between those two assignments. For assignment handling, group forming, and submission use MarkUs:

`https://mcsmark.utm.utoronto.ca/csc376f19`.

Please upload your submission as a single `.pdf` file alongside with an updated git-repository containing your code on MarkUs. By uploading, you confirm that this is your original work without using any references other than the listed ones.

Answer each question completely, always justifying your claims and reasoning and explaining your calculation/derivation. Furthermore, use meaningful and explanatory comments to go along your submitted code. Your solution will be graded not only on correctness, but also on clarity. Answers that are technically correct that are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

Name 1	Name 2 (if applicable)	Date

Due Date: Nov 8, 2019 (11:59pm)

Prerequisites

In order to investigate the forward and velocity kinematics of robot manipulators, the V-REP simulation environment is used. A predefined V-REP scene together with a Python code framework to interact with the simulation environment are provided on MarkUs. Copy the V-REP scene to '`/{V-REP-Installation-Destination}/scenes/`' and the folder containing the code to '`/{V-REP-Installation-Destination}/programming/`'. Throughout this assignment, you will derive the forward and velocity kinematics of a specific robot. Afterwards, using these derivations the functionality of predeclared functions in the provided file `assignmentII.py` need to be implemented:

- `forwardKinematicsPEF`
- `forwardKinematicsDH`
- `velocityKinematics`
- `calcRCond`
- `calcSingularValues`
- `drawManEllipsoid`

This code framework then acts as an interface to the provided scene, that should be loaded in V-REP. It does the following:

- Specifies a set of joint angles as desired configuration
- Calculates the robot end-effector pose for a defined configuration based on the Product of Exponentials Formula (needs to be implemented in `forwardKinematicsPEF`)
- Calculates the robot end-effector pose for a defined configuration based on Denavit-Hartenberg parameters (needs to be implemented `forwardKinematicsDH`)
- Moves the simulated robot in V-REP to the desired configuration
- Draws a blue sphere at the calculated end-effector posed to compare it to the simulated robot's end-effector
- Calculates the Jacobian matrix of the robot in a defined configuration (needs to be implemented in `velocityKinematics`)
- Calculates the reciprocal condition number of the obtained Jacobian matrix (needs to be implemented in `calcRCond`)
- Calculates the singular vectors and values of the end-effector position Jacobian matrix (needs to be implemented in `calcSingularValues`)
- Uses the singular values and vectors to draw a manipulability ellipsoid (needs to be implemented in `drawManEllipsoid`)
- Resets the scene and moves the robot to its initial configuration

Note: While the provided code framework is using Python, you can also choose to transfer the functionalities to a C++ framework and do the assignment in C++.

Part I: Forward Kinematics

1. Figure 1 shows a Kinova Robotics Mico manipulator together with its geometric parameters. This robot is a 6R serial chain. The revolute joints realize rotations in the direction depicted by the green arrows. You can find additional information on the geometric parameters in Appendix I at the end of this document. Draw the kinematic skeleton of the manipulator and assign frames for each joint accordingly. The home position of the robot is depicted in Figure 4 of Appendix I.

   (5 points)

2. Using the stated parameters, derive the forward kinematics using the Product of Exponentials Formula in the space frame. The derivation should be submitted in writing. Implement your derived forward kinematic calculations in the function `forwardKinematicsPEF` of the provided code framework. This function takes a set of angles as an input and outputs the end-effector pose as a 4×4 matrix $\in SE(3)$.

   (10 points)

3. Afterwards, derive the forward kinematics using the Denavit-Hartenberg method. The derivation should be submitted in writing. Implement your derived forward kinematic calculations in the function `forwardKinematicsDH` of the provided code framework. This function takes a set of angles as an input and outputs the end-effector pose as a 4×4 matrix $\in SE(3)$.

   (10 points)

4. Discuss the implementation differences you encountered for the PoE formula and D-H parameter function.

   (5 points)

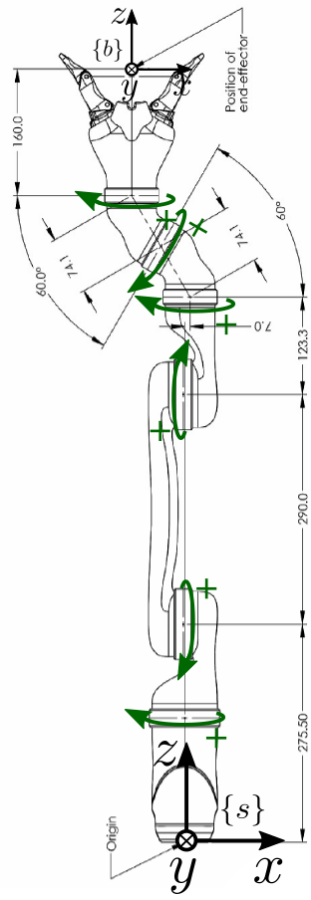


Figure 1: Right: Kinova Robotics Mico manipulator [1] with six revolute joints. Left: Dimensions and geometry of the manipulator together with its base and end-effector frames as well as the positive rotation direction of its joints (green).

Part II: Velocity Kinematics

1. Derive the velocity kinematics of the robot manipulator described in Part I on paper. The outcome should be a formulation of the 6×6 space Jacobian matrix $J_s(\theta)$ that relates joint velocities to the end-effector twist.

   (15 points)

2. Implement your derived velocity kinematic calculation in the function `velocityKinematics` of the provided code framework. This function takes a set of angles as an input and outputs the space Jacobian as a 6×6 matrix.

   (15 points)

3. Implement the calculation of the reciprocal condition number in the function `calcRCond` of the provided code framework. This function takes a 6×6 space Jacobian matrix as an input and outputs the reciprocal condition number as a scalar value.

   (10 points)

4. Implement the calculation of the Jacobian's singular vectors and values with respect to the translational velocities (Note: We do not need to consider the whole matrix here!) in the function `calcSingularValues` of the provided code framework. In order to do so, perform a singular value decomposition of the 3×6 space Jacobian matrix relating joint velocities to the end-effector's translational velocities.

   (6 points)

5. Using the singular vectors and values write a function that draws the manipulability ellipsoid (`drawManEllipsoid` in the provided code framework). Drawing the three main axes of this ellipsoid with their corresponding lengths is sufficient for this task. This function should return the handles of the drawn objects. Those will be used by V-Rep later to remove the drawn objects and to reset the scene.

   *You can get up to **5 extra points** on this task if you visualize a proper ellipsoid in the scene rather than just its main axes.*

   (4 points)

Part III: Evaluation

1. In order to verify your calculations and implementations, load the provided V-REP scene and run the provided code framework with the five configurations stated in Table 1 and five additional but distinct configurations of your choice (so 10 in total). These additional configurations should cover different areas of the robot's work space and at least one of them should be singular.

   For each configuration state the following:

   - The joint angles θ
   - The calculated end-effector pose based on your implementations (both for the Product of Exponential Formula and Denavit-Hartenberg)
   - The end-effector pose obtained from the simulated robot
   - The calculated reciprocal condition number
   - The calculated singular vectors and values

   Furthermore, take screenshots of the resulting V-REP scene for each configuration and discuss you results based on what you see.

   (20 points)

Joint	1	2	3	4	5	6
θ in °	0	0	0	0	0	0
θ in °	0	90	90	0	90	0
θ in °	45	-75	-90	90	45	90
θ in °	-90	-30	90	90	0	45
θ in °	-60	30	-45	35	-100	-120

Table 1: Parameters of five different robot configurations

Appendix I: Mico Robotic Arm - Geometric Parameters [1]

Basic geometric parameters - MICO 6 DOF curved wrist

This section describes the basic geometric parameters of the MICO 6 DOF curved wrist.

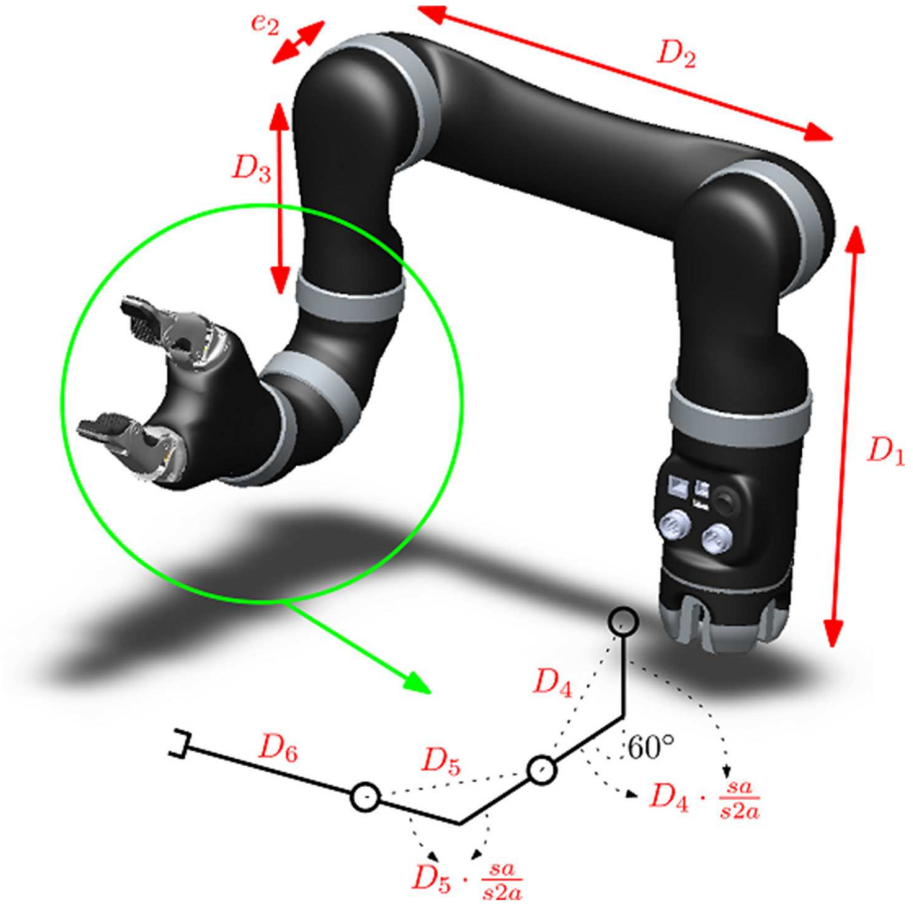


Table 16: MICO 6 DOF basic geometric parameters

Parameter	Description	Length (m)
D1	Base to shoulder	0.2755
D2	Upper arm length (shoulder to elbow)	0.2900
D3	Forearm length (elbow to wrist)	0.1233
D4	First wrist length (center of actuator 4 to center of actuator 5)	0.0741
D5	Second wrist length (center of actuator 5 to center of actuator 6)	0.0741
D6	Wrist to center of hand	0.1600
e2	Joint 3-4 lateral offset	0.0070

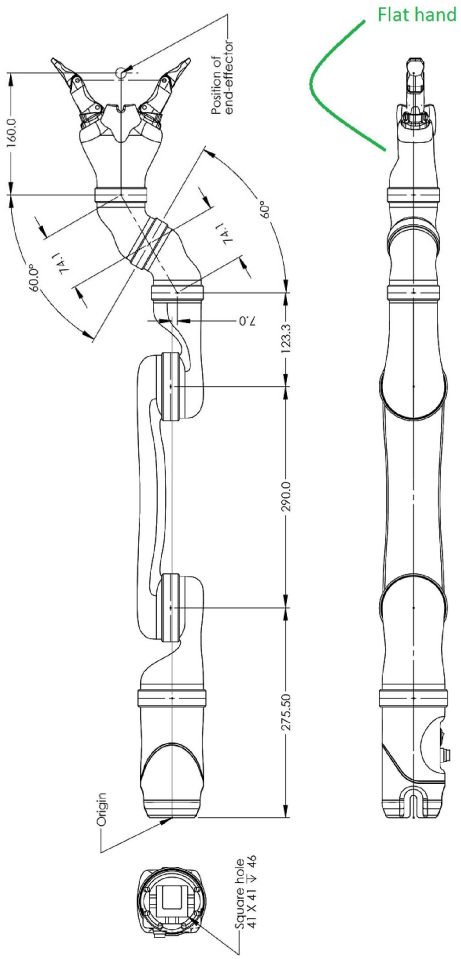
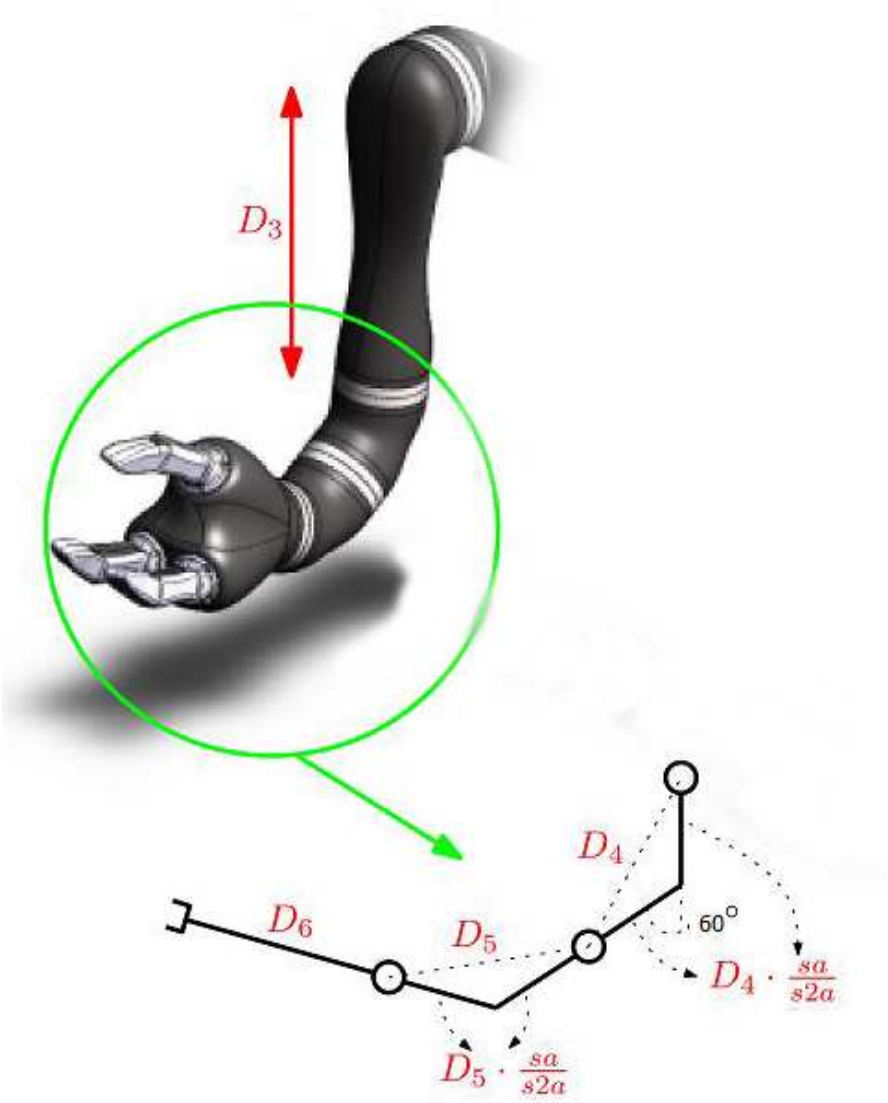


Figure 4: Detailed MICO 6 DOF curved wrist robot length values (units in mm)

Alternate geometric parameters - 6 DOF curved wrist

This section describes alternate parameters that are useful for describing the geometry for kinematics of the 6 DOF curved wrist configuration.

The kinematics of the 6 DOF curved wrist configuration are more complicated than for a spherical wrist due to the more complicated geometry. To simplify the analysis, it is useful to break down each of the two curved wrist segments into two component straight-line sub-segments of equal length, with the second sub-segment angled 60° from the first.



In this way, the arm from the elbow to the center of the hand can be analyzed as three straight-line segments:

- d4b - distance from elbow to end of first sub-segment of first wrist segment
- d5b - distance from end of first sub-segment of first wrist segment to end of first sub-segment of second wrist segment
- d6b - distance from end of second sub-segment of second wrist segment to center of hand

Table 17: Alternate parameters

Parameter	Description	Value
aa	Half of the angle of curvature of each wrist segment (60°), measured in radians	(30.0 * PI) / 180.0
sa	Sine of half the angle of curvature of wrist segment	sin(aa)
s2a	Sine of angle of curvature of wrist segment	sin(2*aa)
d4b	Length of straight-line segment from elbow to end of first sub-segment of first wrist segment.	D3 + (sa / s2a) * D4
d5b	Length of straight-line segment consisting of second sub-segment of first wrist segment and first sub-segment of second wrist segment	(sa / s2a) * D4 + (sa / s2a) * D5
d6b	Length of straight-line segment consisting of second sub-segment of second wrist segment and distance from wrist to the center of the hand	(sa / s2a) * D5 + D6

The DH parameters for the lower part of the robotic arm are most naturally expressed in terms of these alternate parameters.

References

[1] Kinova Robotics: Mico Manipulator User Guide,
https://www.kinovarobotics.com/sites/default/files/ULWS-RA-MIC-UG-INT-EN%20201804-1.0%20%28KINOVA%20MICO%E2%84%A2%20Robotic%20arm%20user%20guide%29_0.pdf