

Robotics Toolbox

using Matlab



Team members

- Shubham Gharat(GUI's, Integration of the GUI's with program functions, homogeneous transformation matrix)
- Laukika Sankpal(GUI's, Integration of the GUI's with program functions, Euler angles)
- Abhijith Thottumadayil Jagadeesh(Forward kinematics, Workspace)
- Rohit Vaddi (Differential Kinematics, Inverse Differential Kinematics, and their respective GUIs)
- Paarvendhan Puviyarasu (Inverse Kinematics, Inverse Kinematics using Jacobians, Integration and their respective GUIs)

Table of Contents

Various options in GUI

1. Homogeneous Transformation matrix
2. Euler angles
3. Forward kinematics
4. Workspace
5. Inverse kinematics
6. Inverse kinematics jacobian
7. Differential kinematic
8. Inverse differential kinematics
9. Manipulator dynamics
10. Manipulator Control

Running the code

Working of GUI

1. Start
2. Select DH or Custom
3. Number of links
4. DH parameters
5. Number of Joints
6. Custom
7. select operation
8. Homogeneous Transformation Matrix
9. Forward Kinematics
10. Euler Angles
11. Workspace
12. Inverse Kinematics
13. Inverse Kinematics using Jacobians
14. Differential Kinematics
15. Manipulator Dynamics
16. Inverse Differential Kinematics
17. Manipulator Control

Description:

GUI:

This gui in matlab is a robotics toolbox that can provide users with different results based on the users selection.

Homogeneous matrix transformation:

Homogeneous transformation matrix consists of a 3x3 rotation matrix, 3x1 displacement vector and the fourth row as [0 0 0 1]. The displacement vector tells us about the location of the end effector with respect to the previous frame, whereas the rotation matrix gives us the orientation of the end effector with respect to the previous frame.

For computing the homogeneous transformation matrix, we are taking the DH parameters (theta, a, alpha, d) from the user and displaying the transformation matrix to the user according to the following representation.[1]

$$A_i^{i-1}(q_i) = A_i^{i-1} A_i^{i'} = \begin{bmatrix} c_{\vartheta_i} & -s_{\vartheta_i} c_{\alpha_i} & s_{\vartheta_i} s_{\alpha_i} & a_i c_{\vartheta_i} \\ s_{\vartheta_i} & c_{\vartheta_i} c_{\alpha_i} & -c_{\vartheta_i} s_{\alpha_i} & a_i s_{\vartheta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Euler angles:

The representation in terms of Euler angles implies that the three angles are sufficient to describe the orientation of a rigid body in space. A minimal representation of orientation can be obtained by using a set of three angles $\phi = [\phi \ \vartheta \ \psi]^T$.

In this part, we are taking the three angles as an input from the user and computing the rotation matrix from these angles.

For the inverse problem, we are taking the computed rotation matrix and computing the Euler angles. [1]

Forward Kinematics:

Forward kinematics is the process of finding out the position of the end effector given the DH parameters. We use the homogeneous matrix transformation for the same. [1]

Workspace:

Workspace is defined as the volumetric range that is accessible by the robot manipulator. The workspace is plotted by retrieving the displacement vector from the transformation matrix in the form of symbolic dynamic variables and then passing on the values that are plotted in the form of a grid. The workspace makes use of the minimum and maximum values of theta (for revolute joints) and minimum and maximum values of d (for prismatic joints) and plots a number of points between the limits. Later

the plotted points are converted into a grid and a 3D plot function is used to plot the workspace.[1]

Inverse kinematics:

Inverse Kinematics is a branch of robotics where we find the Joint parameters given the end effector position.

Here, we are using the Transformation matrix from the DH parameters of the robot to calculate the final transformation of the matrix and that transformation matrix is used in the robotics toolbox to calculate the inverse kinematics of the robot and display it to the user.[1]

Inverse Kinematics using Jacobian:

Inverse Kinematics using jacobians is a different way of finding the joint parameters given the end effector positions. As the name suggests, the jacobian for the given end effector is calculated at first and then we have to find the inverse of the jacobian so that when we multiply it with the end effector position, we can get the joint parameters.

$$\dot{p}_e = \frac{\partial p_e}{\partial q} \dot{q} = J_P(q) \dot{q}, \quad \dot{\phi}_e = \frac{\partial \phi_e}{\partial q} \dot{q} = J_\phi(q) \dot{q},$$

so from this we can take the inverse of the jacobian and find the joint parameters using the following formula:

$$q(t_{k+1}) = q(t_k) + J^{-1}(q(t_k)) v_e(t_k) \Delta t.$$

but mostly, we will not get the perfect square jacobian matrices, so in that case, we have to use a pseudo-inverse jacobian and then calculate the joint parameters from that.

If the pseudo inverse is used then the following formula is used,

$$\dot{q} = J_A^{-1}(q) \begin{bmatrix} \dot{p}_d + K_P e_P \\ \dot{\phi}_d + K_O e_O \end{bmatrix}$$

In this program, we are giving qd(final position) and q0(initial position) of the end effector and the joint parameters are calculated for that.[1]

Differential Kinematics

Differential Kinematics shows the relation between velocity in the joint space and linear/angular velocity in the task space. Instantaneous velocity can be found from the time derivation of the direct kinematics. The standard approach is to relate the end-effector velocities to the velocity of the joints with the help of a Jacobian matrix. Also an important aspect of differential kinematics is that it can do singularity analysis.[4]

Inverse Differential Kinematics

Inverse differential kinematics is used for finding the end effector position when the velocities and the inverse of the jacobian matrix is known. If in case the jacobian matrix does not have an inverse, the pseudo inverse of the jacobian is taken and multiplied with the given joint velocities to find the end effector positions.[4]

Running the code:

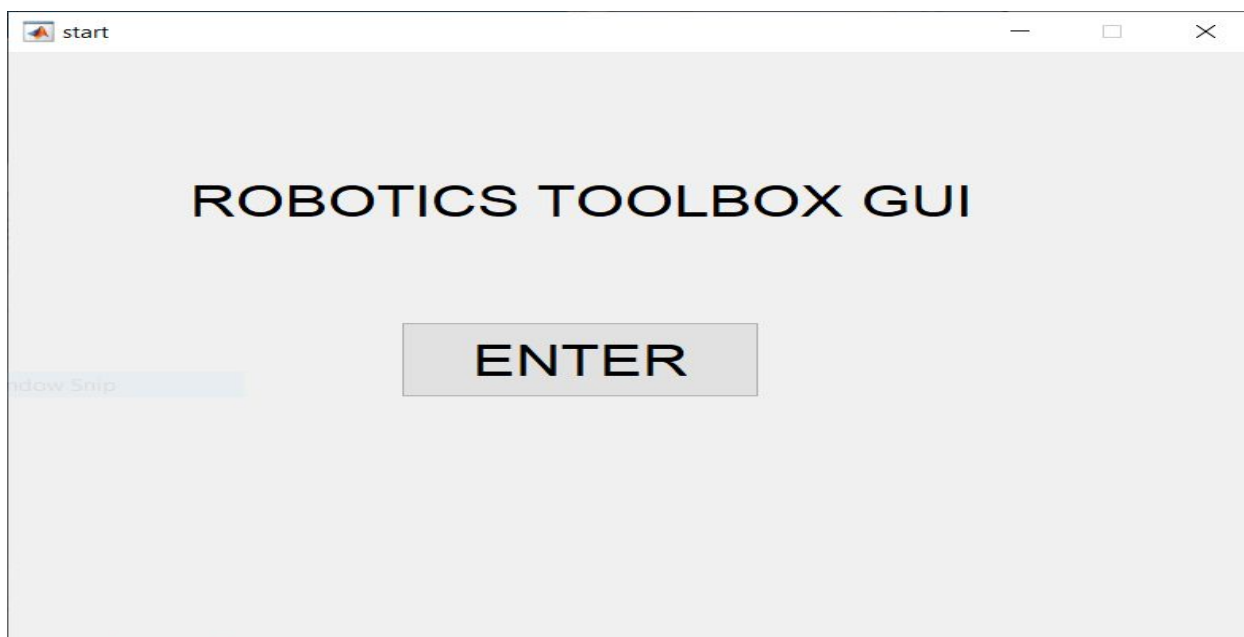
- Add path and Run the rvc_toolbox before starting the program.
- the entire program can be started by running start.m in the custom_robotics_toolbox.
- the program and the individual parts can be found in each of the respective directories.
- all the programs are written as functions so that if something is wrong we can easily inspect it. the modular structure especially helps when we want to expand the code or edit the code without other parts being altered.

Running the GUI:

Once the start.m is started, the GUI runs and what each window does is explained below,

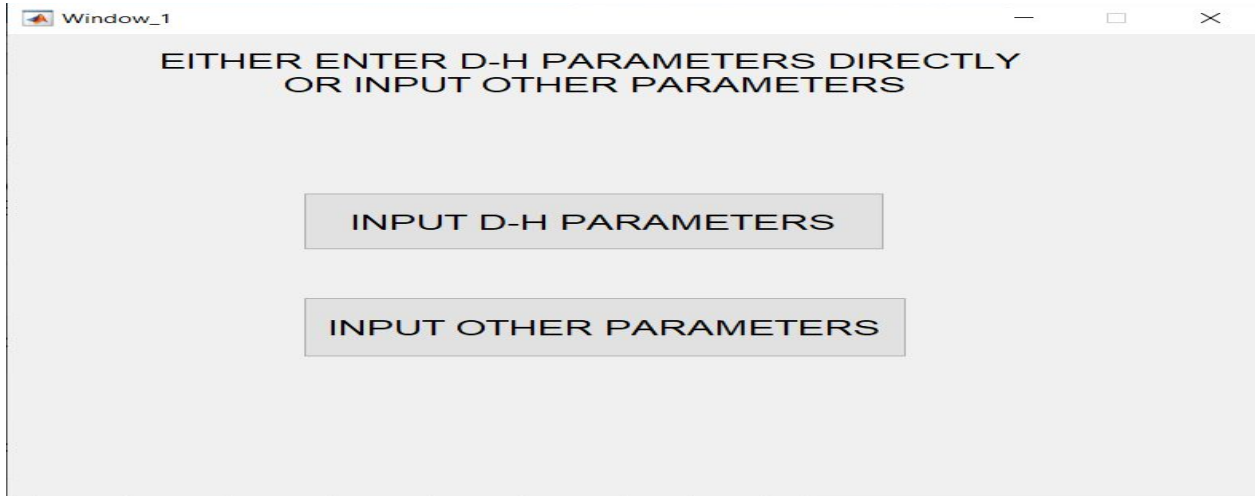
Start:

This window will act as a start window for the user to access the Robotics toolbox.



Select DH or Custom:

The user can either directly enter the D-H parameters or if the user is unaware of the D-H parameters then he can enter the other parameters like the joint parameters and Link parameters from which the D-H parameters will be calculated.



Window_1

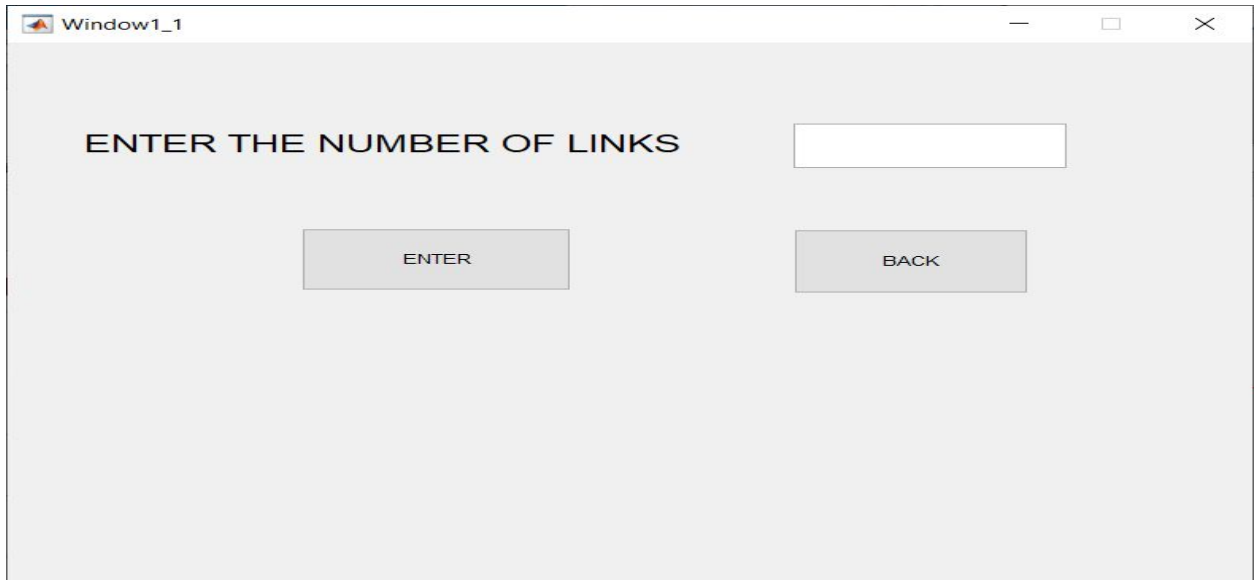
EITHER ENTER D-H PARAMETERS DIRECTLY
OR INPUT OTHER PARAMETERS

INPUT D-H PARAMETERS

INPUT OTHER PARAMETERS

Number of links:

When the user selects the option of entering the D-H parameters he is prompted to enter the no. of Links in the Robot following which the user can type in the respective D-H parameters.



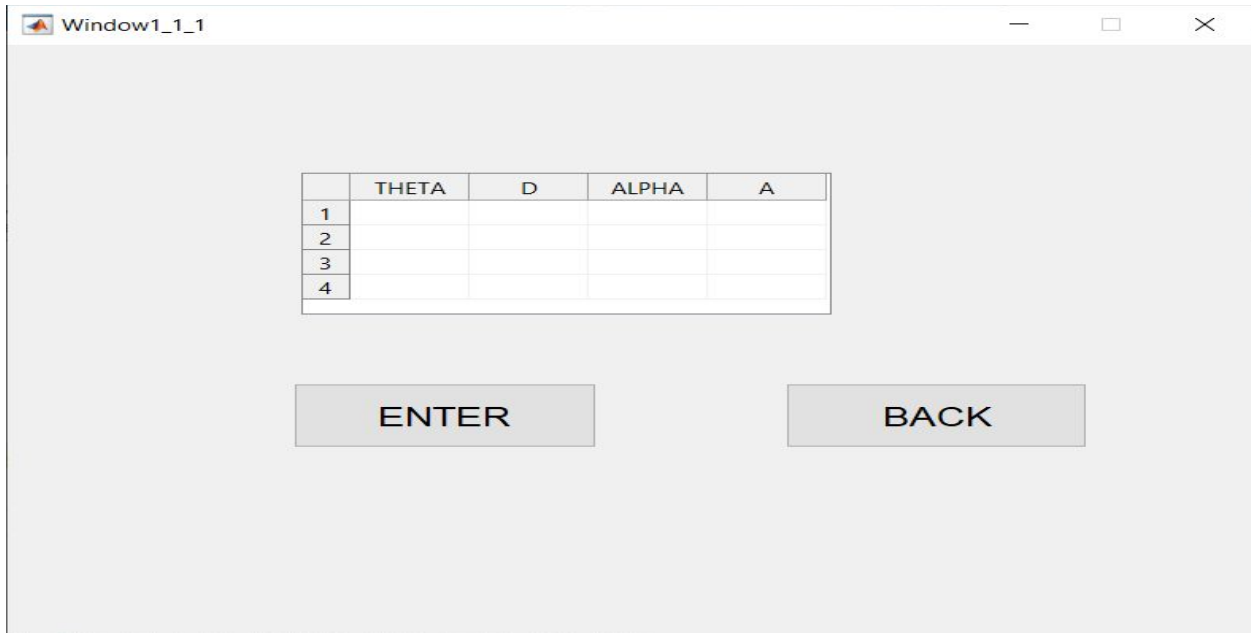
Window1_1

ENTER THE NUMBER OF LINKS

ENTER BACK

DH:

Enter the DH parameters.



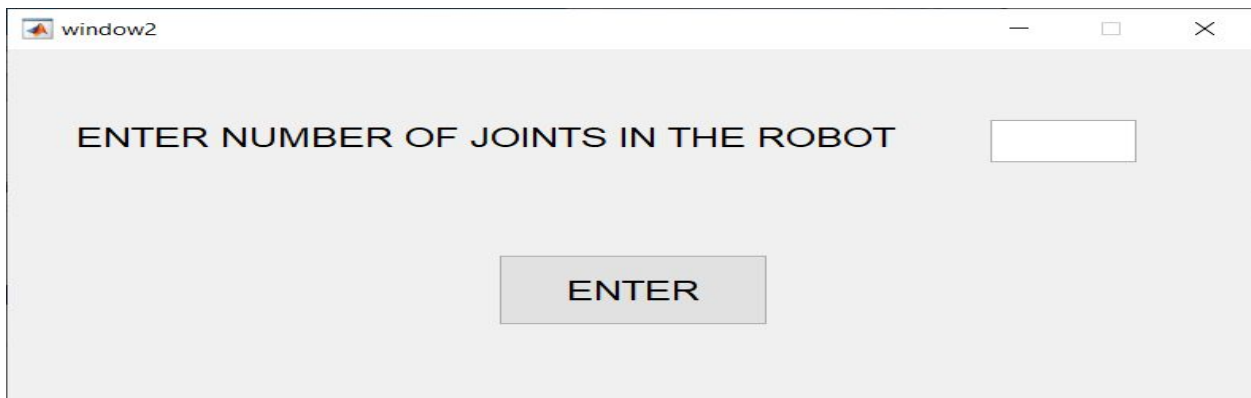
The screenshot shows a window titled "Window1_1_1" with a table for entering DH parameters. The table has four columns: THETA, D, ALPHA, and A. There are four rows, numbered 1 to 4. Below the table are two buttons: "ENTER" and "BACK".

	THETA	D	ALPHA	A
1				
2				
3				
4				

ENTER BACK

Number of Joints:

When the user selects the option to input the other parameters the user needs to enter the no. of joints present in the Robot followed by all the parameters related to the links and the joints like the type of joint, the Z-axis with respect to the base frame.



The screenshot shows a window titled "window2" with the text "ENTER NUMBER OF JOINTS IN THE ROBOT" and a text input field. Below the input field is an "ENTER" button.

ENTER NUMBER OF JOINTS IN THE ROBOT

ENTER

Custom:

The screenshot shows a window titled 'newWindow2_1' with two main sections: 'ENTER JOINT PARAMETERS' and 'ENTER LINK PARAMETERS'.

ENTER JOINT PARAMETERS: This section contains a table with three columns: 'JOINT NUMBER', 'TYPE OF JOINT', and 'Z-AXIS'. The 'TYPE OF JOINT' column has a dropdown menu with options: PRISMATIC, REVOLUTE, REVOLUTE, and PRISMATIC (highlighted in blue). Below the table is a 'Z-AXIS OF END EFFECTOR' input field and a 'SUBMIT' button.

ENTER LINK PARAMETERS: This section contains a table with two columns: 'LINK NUMBER' and 'LINK LENGTH'.

Select operation:

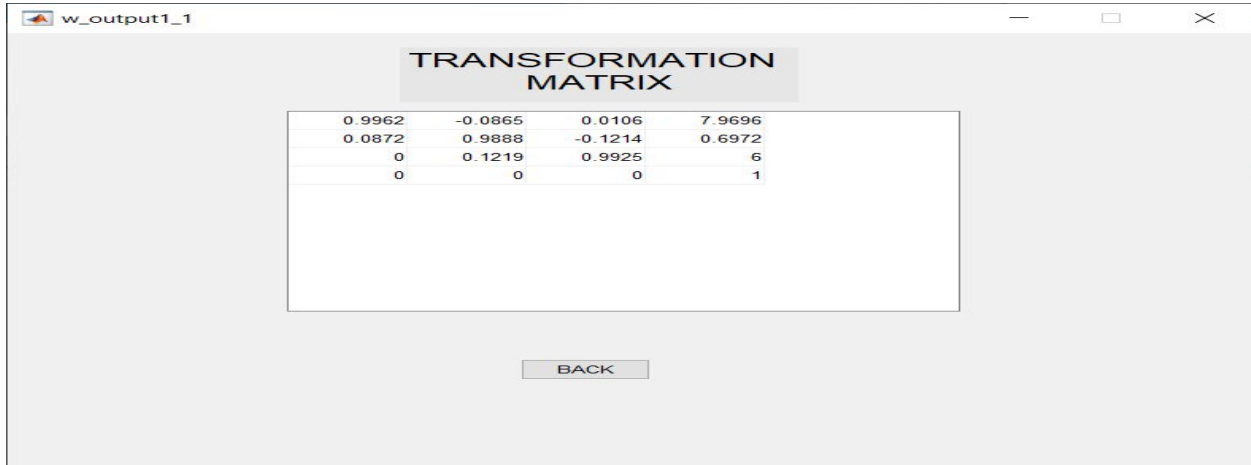
In this window the user can select between various options provided in the Robotics toolbox.

The screenshot shows a window titled 'topics_gui' with a grid of buttons for selecting different operations:

- HOMOGENEOUS TRANSFORMATION MATRIX
- EULER ANGLES
- FORWARD KINEMATICS
- WORKSPACE
- INVERSE KINEMATICS
- INVERSE KINEMATICS USING JACOBIANS
- DIFFERENTIAL KINEMATICS
- MANIPULATOR DYNAMICS
- INVERSE DIFFERENTIAL KINEMATICS
- MANIPULATOR CONTROL

Homogeneous Transformation matrix:

The transformation matrix calculated from the D-H parameters is displayed on this window.



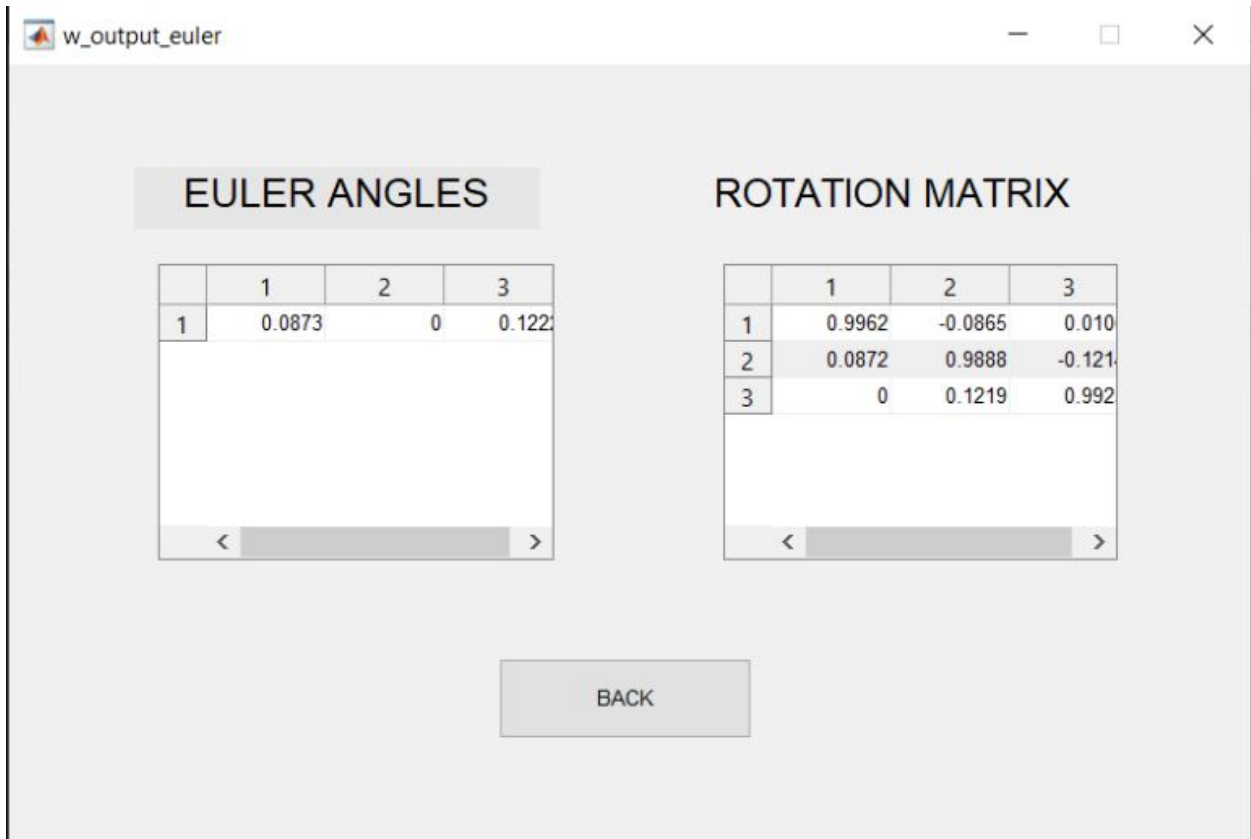
TRANSFORMATION MATRIX

0.9962	-0.0865	0.0106	7.9696
0.0872	0.9888	-0.1214	0.6972
0	0.1219	0.9925	6
0	0	0	1

BACK

Euler angles:

The euler angles and the Rotation matrix is displayed on this GUI.



EULER ANGLES

	1	2	3
1	0.0873	0	0.122

ROTATION MATRIX

	1	2	3
1	0.9962	-0.0865	0.010
2	0.0872	0.9888	-0.121
3	0	0.1219	0.992

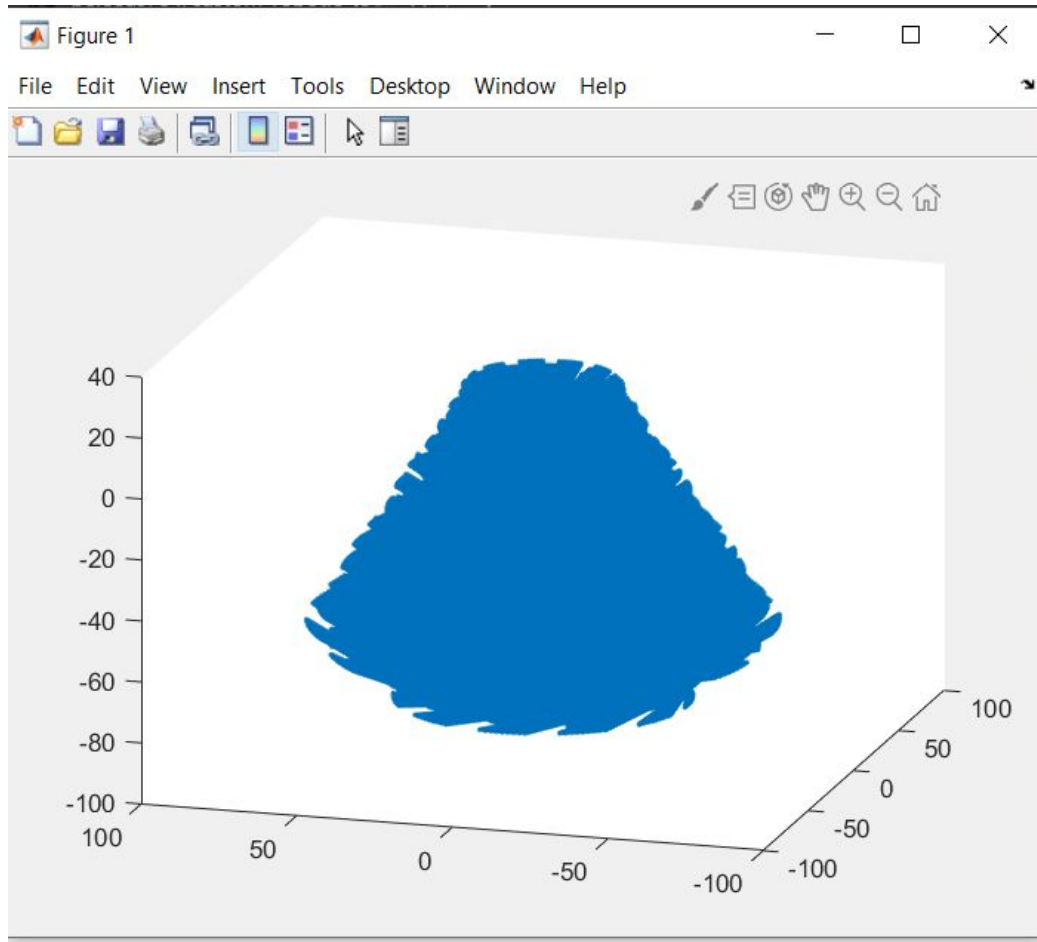
BACK

Forward Kinematics:

The transformation matrix calculated for the forward kinematics is displayed.

```
T_F =  
-0.3742  -0.7281  -0.5743  -4.819  
 0.5774   0.3017  -0.7587   9.019  
 0.7257  -0.6155   0.3075   3.158  
      0      0      0      1
```

Workspace:



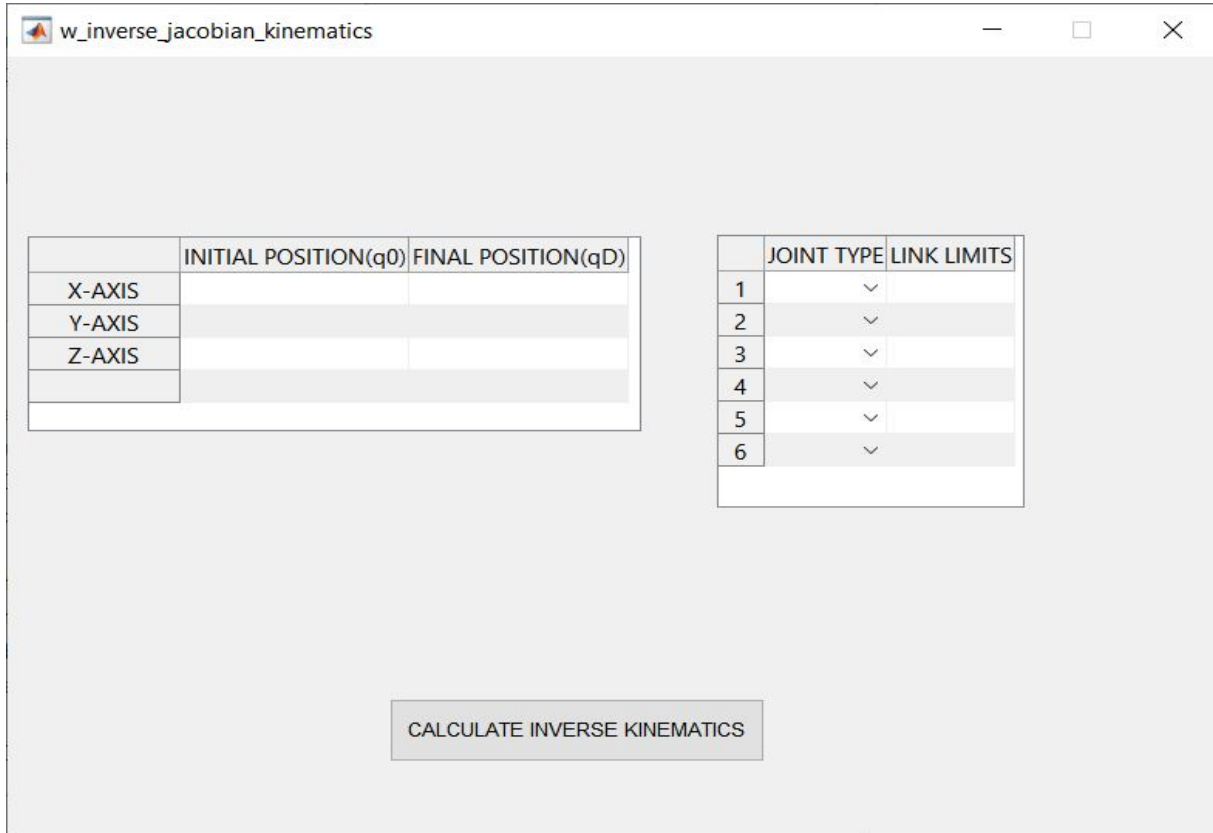
Inverse Kinematics:

Once the inverse kinematics button is clicked, the inverse kinematics is calculated and the output is shown.



Inverse Kinematics using Jacobians:

We have to enter the values for q_0 and q_d and also mention which are the prismatic joints and then the outputs are shown in the next page.



	INITIAL POSITION(q_0)	FINAL POSITION(q_d)
X-AXIS		
Y-AXIS		
Z-AXIS		

	JOINT TYPE	LINK LIMITS
1	▼	
2	▼	
3	▼	
4	▼	
5	▼	
6	▼	

CALCULATE INVERSE KINEMATICS

Differential Kinematics:

We have to enter the values for q_0 and also mention which are the prismatic joints and then the outputs are shown in the next page.

W_differential_kinematics

	Initial Position (q0)	
X-AXIS		
Y-AXIS		
Z-AXIS		

	Joint Type	Link Limits
1	∨	
2	∨	
3	∨	
4	∨	
5	∨	
6	∨	

CALCULATE DIFFERENTIAL KINEMATICS

Inverse Differential Kinematics:

We have to enter the values for q_0 and q_d and also mention which are the prismatic joints and then the outputs are shown in the next page.

w_inv_differential_kinematics

	Initial Position (q0)	Final Position (qd)
X-AXIS		
Y-AXIS		
Z-AXIS		

	Joint Type	Link Limit
1	▼	
2	▼	
3	▼	
4	▼	
	▼	
	▼	

CALCULATE INVERSE DIFFERENTIAL KINEMATICS

Citations:

- [1] Bruno Siciliano, Lorenzo SciaviccoLuigi, VillaniGiuseppe, Oriolo, Robotics: Modelling, Planning and Control, Springer Publishing Company, 2008
- [2] Sample codes from the Dr. Marvi's Lecture codes in Canvas.
- [3] P.I. Corke, "Robotics, Vision & Control", Springer 2017, ISBN 978-3-319-54413-7.
- [4]A. Campos, R. Guenther and D. Martins, "Differential kinematics of serial manipulators using virtual chains", 2005. .