<u>Manual for Robotics User Interface</u>

This manual goes over how the GUI can be used to obtain various parameters, equations and joint/link quantities of a robotic manipulator. The topics covered are:

A. Homogenous Matrix Transformation
B. (A and B are clubbed together)
    1. Translation Animations
    2. Rotation Animations
        i. Euler (ZYZ) Rotations with Translation
        ii. RPY Rotations with Translation
        iii. Quaternions Rotations with Translation
        iv. Rotation about a vector with Translation
C. Forward Kinematics
    1. Creation of a Robotic Manipulator
        i. Using Direct DH parameters
        ii. Using descriptions given by a user
    2. Forward Kinematics from DH parameters
    3. Forward Kinematics for given joint positions (q)

D. Workspace
    1. Operational Workspace for given joint limits
    2. Singularities found within these joint limits

E. Inverse Kinematics
    1. Inverse kinematics for a given end effector pose using closed loop solution and Jacobian
    2. Inverse kinematics for a given end effector pose using convex optimization tools with and without given joint limits

F. Differential Kinematics
    1. Obtain the Jacobian matrix
        i. Obtain a symbolic Jacobian expression
        ii. Obtain a numeric Jacobian for given joint pose
    2. Get end effector velocity using computed Jacobian

G. Inverse Differential Kinematics
    1. Obtain joint velocities given the joint positions (q) and end effector velocity (Xe_dot) components

H. Dynamics
    1. Obtain Equation of Motion (Lagrangian)
    2. Plot Joint Pose and Velocity
I. Controls
    1. Force Control
    2. Motion Control

Functional overview

The following table gives an overview of the inputs and outputs that are taken by each of the sub components of the application.

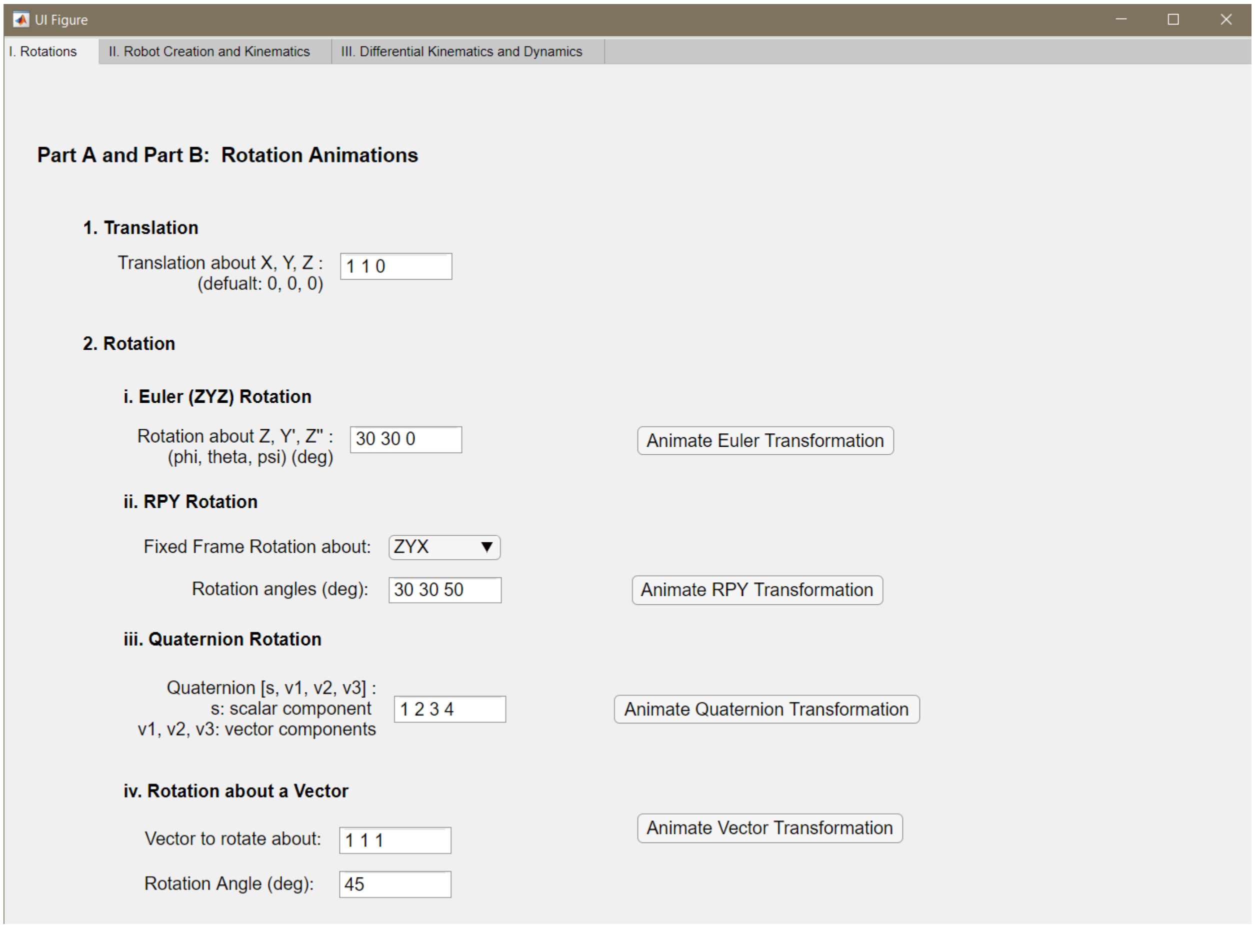
No	Question	Functionality	Owner	Input	Result
a	Homogenous matrix	Frame description	Aditya	Euler angles (ZYZ, XYZ), Quartenions, Angle Axis Position vector	T0H
		Transformation mapping	Rakshith	v_i , T0H	v_base
		Transformation operator	Rakshith	T01, T12, … , TnH	T0H
b	Euler Angles	Anlge -> Rotation matrix	Rakshith	alpha, beta, gamma	R
		Rotation matrix -> Anlge	Rakshith	R	alpha, beta, gamma
c	FW Kinematics	Robot config -> DH	Mitul	Manipulator description	DH table
		DH -> Serial Link	Mitul	DH table	T0H, Link obj
		Joint parameter -> EF pose	Mitul	Link Obj, q	EF Pose (T0H)
d	Workspace	Robot config -> Workspace	Ravi	Link Obj	
e	Inverse Kinematics (analytical)	EF Pose -> Joint Paramter	Osei	Link Obj, EF pose (T0H)	q
f	Differential Kinematics	Jacobian	Ravi	Link Obj, q	Jg, Ja
		Singularities	Ravi	Jg	q_singular
g	Inverse Diff Kinematics	EF velocity -> Joint Velocity	Ravi		
	Inverse Kinematics (Jacobian)	EF Pose -> Joint Paramter	Osei	Link Obj, EF pose (T0H), Jacobian	q
h	Dynamics	Equation of Motion (Lagrangian)	Ravi	Dynamic params	B,C,G
		Joint position and velocity -> plot	Ravi	Link Obj, q0, qd0	
i	Control (2 Planar robot)	Position Control	Ravi		
		Inverse Force control	Ravi		
j	GUI		Rakshith		
k	Manual		Rakshith, Aditya		

Usage of the Application:

The application has three main tabs that house different aspects of a robot. The GUI can be run using the **app1.mlapp** file.

Tab 1:

- Various rotation methods
  - Input: Translation and Rotation
  - Output: Effective Transformation Matrix and Frame Animation

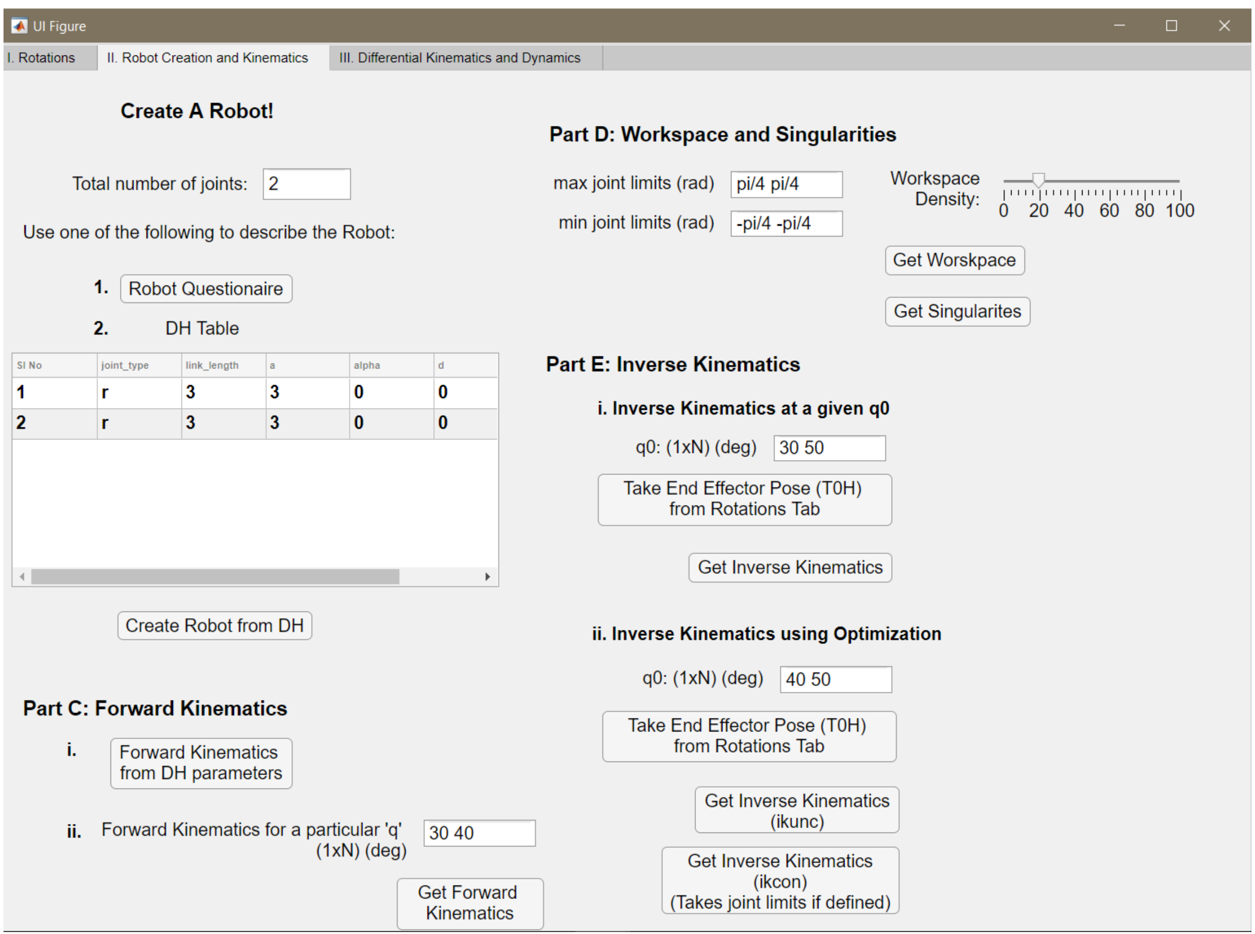


NOTE: Translation is common to all types of rotations. Pure rotation for quaternions and rotation about a vector can be obtained by keeping translation component to 0.

Tab 2:

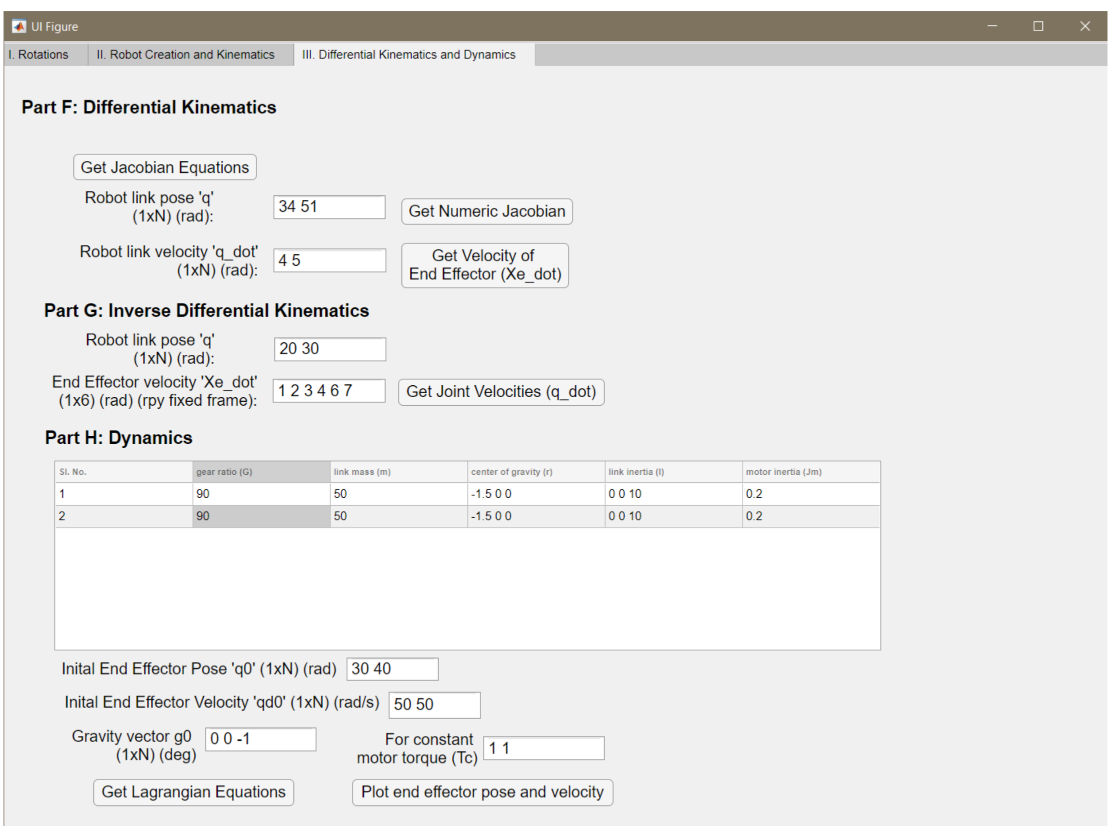
- Creation of Robot
  - Questionnaire
    - Inputs: Type of joints, Axis of rotation, Link Orientation, Link Length
    - Output: Created Robot and DH parameters
  - Direct DH Params
    - Inputs: DH parameters
    - Output: Created Robot

- Forward Kinematics (End Effector Pose)
  - Symbolic Forward Kinematics Equations
    - Inputs: Robot object (and its DH parameters)
    - Output: Symbolic forward kinematics equations
  - Numeric forward kinematics
    - Inputs: joint angles
    - Output: Numeric transformation matrix to end effector from base of robot
- Workspace and Singularities
  - Inputs: Joint Limits, density of workspace plot
  - Outputs: Workspace plot and Singularities
- Inverse Kinematics
  - Closed Loop Inverse Kinematics
    - Inputs: Initial joint positions
    - Outputs: Final Joint positions
  - Convex Optimization based Inverse Kinematics
    - Inputs: Initial joint positions
    - Outputs: Final Joint positions



Tab 3:

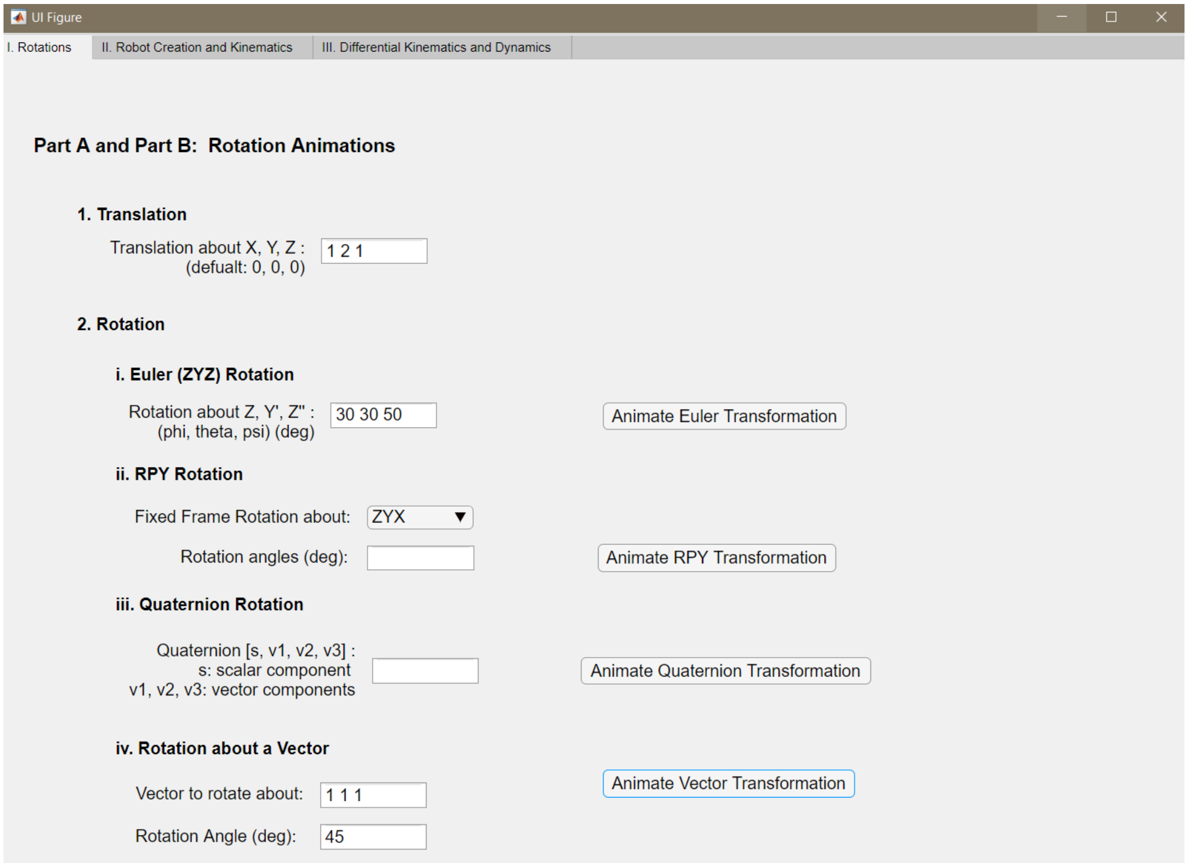
- Differential Kinematics
  - Symbolic Jacobian Equations
    - Inputs: Robot object
    - Outputs: Jacobian equations
  - Numeric Jacobian
    - Inputs: Initial joint positions
    - Outputs: Numeric Jacobian
  - End Effector Velocities
    - Inputs: Initial joint velocities
    - Outputs: end effector velocity
- Inverse Differential Kinematics
  - Inputs: Initial link locations, end effector velocity
  - Outputs: Joint veloicities (q_dot)
- Dynamics:
  - Symbolic lagrangian Dynamics equations
    - Inputs: Link parameters (mass, center of gravity, etc), initial joint locations, initial joint velocities, gravity vector,
    - Outputs: Symbolic lagrangian Dynamics equations
  - End Effector Plots
    - Inputs: Constant torque at each link
    - Outputs: Plots of joint positions and joint velocities



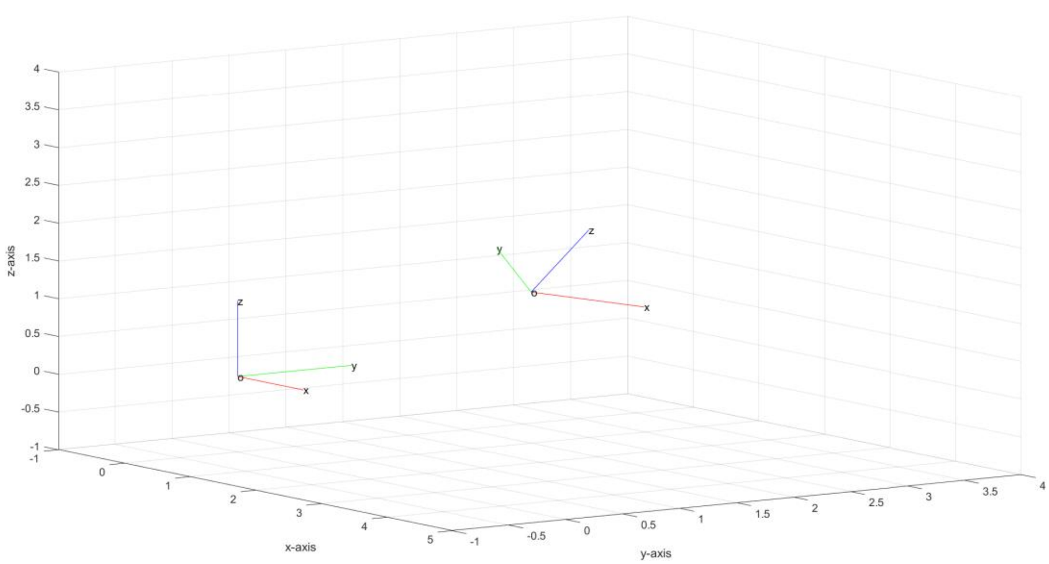
NOTE: The Toolbox has been tested on MATLAB 2019 (b). Some UI features may throw errors or warnings in 2019 (a) versions and before.

Example1: Rotations

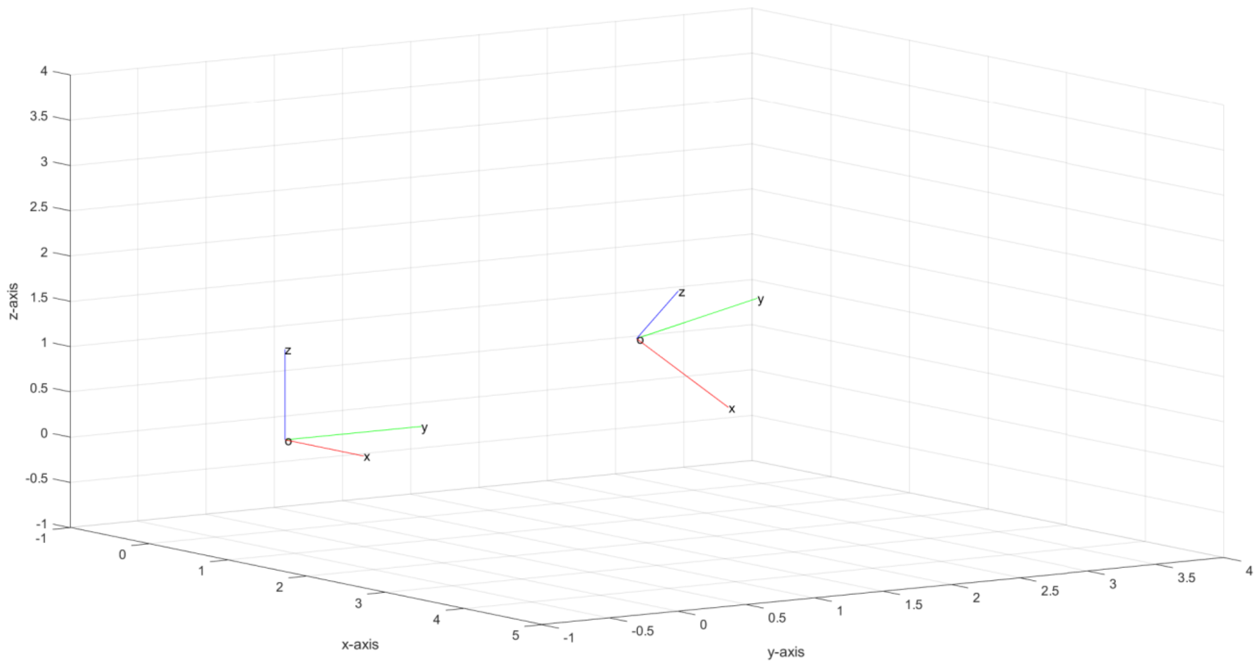
Here, examples of Euler (ZYZ) rotations and rotation about the vector [1,1,1] by 45 degrees have been provided. For both cases a translation of [1, 2, 1] was used.



Euler Angle Rotations:



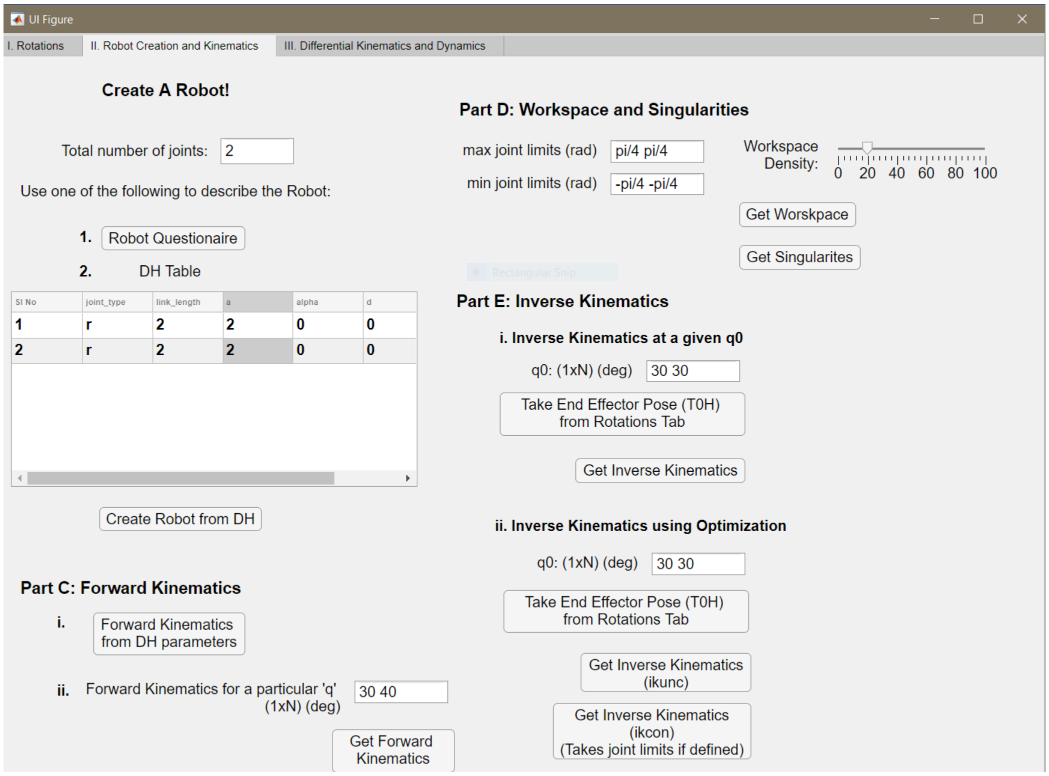
Rotation about a Vector:

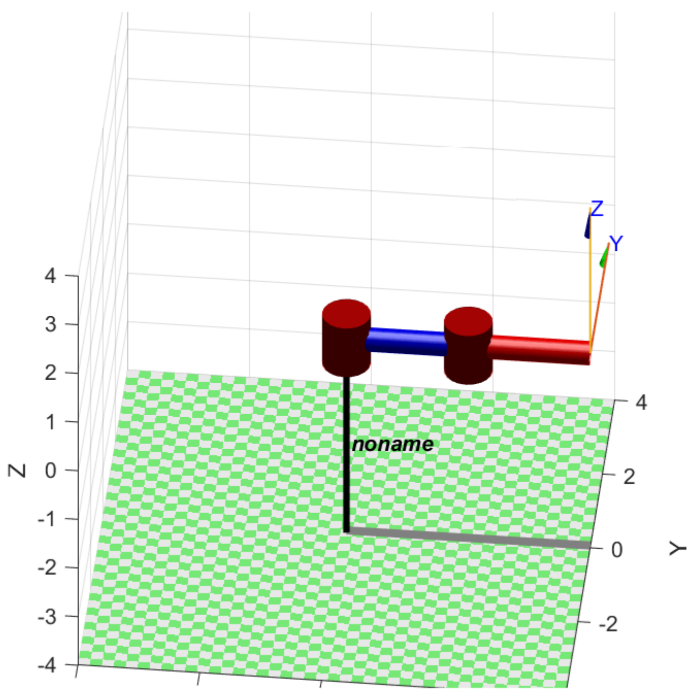


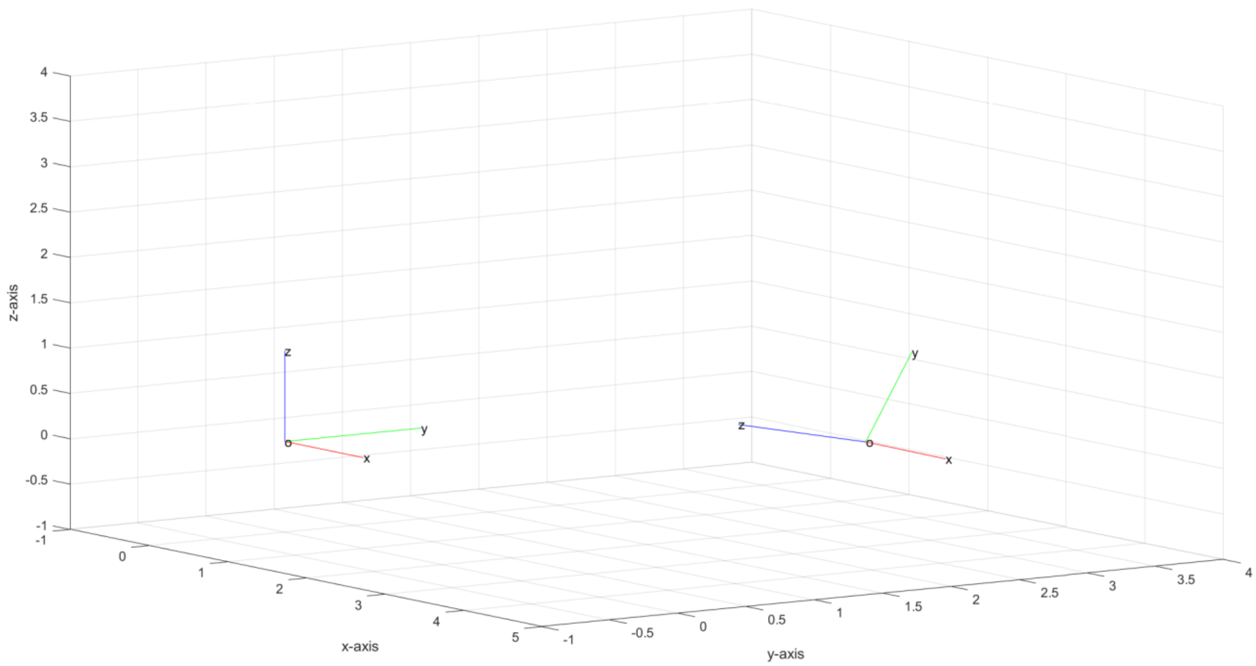
Similarly, other rotation operations can be performed.

Example2: Creation of 2 Link Planar Robot and Related Kinematics/Dynamics

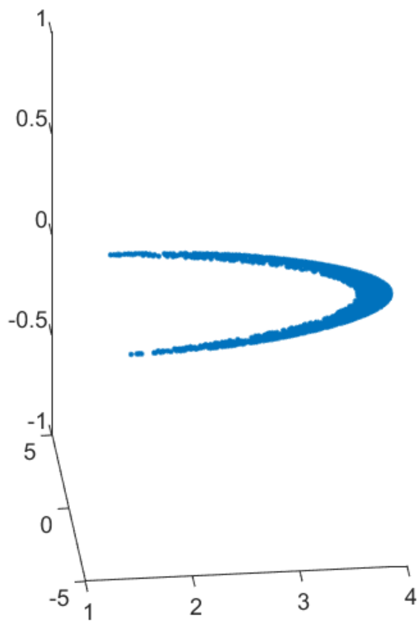
Here the direct DH table was used to create a two link planar robot. The screenshots for the inputs provided and the corresponding outputs have been provided below:



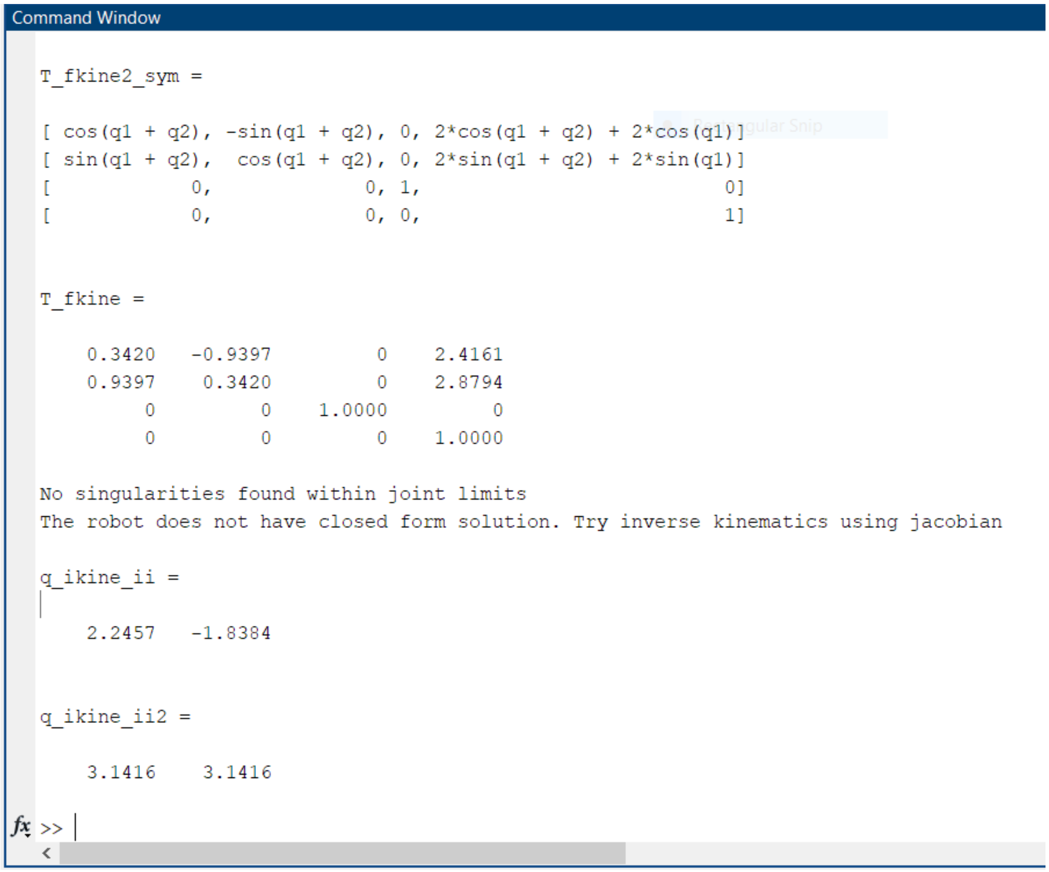




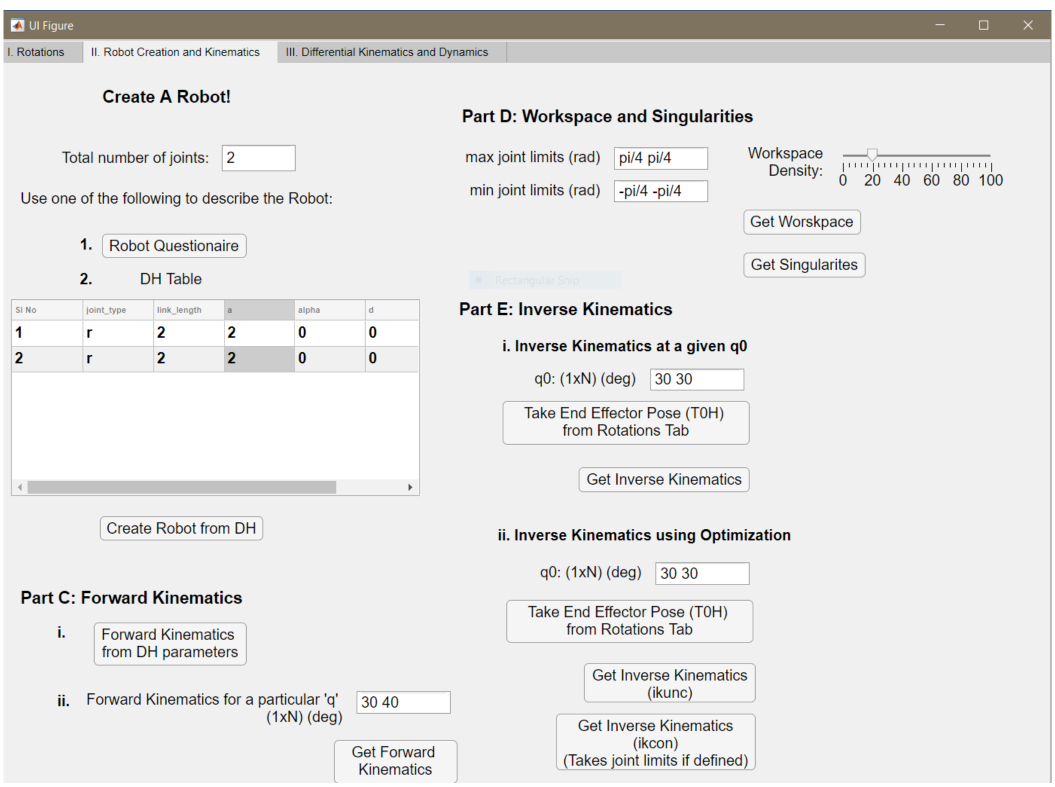
We see in this figure that the initial frame (representing the base of the robot) and the final frame (representing the end effector) are in the same plane.



Similarly, the workspace is also plotted in 2D, with a maximum value of L1 + L2 = 2+2 = 4


```
Command Window

T_fkine2_sym =

[ cos(q1 + q2), -sin(q1 + q2), 0, 2*cos(q1 + q2) + 2*cos(q1)]
[ sin(q1 + q2),  cos(q1 + q2), 0, 2*sin(q1 + q2) + 2*sin(q1)]
[            0,             0, 1,                          0]
[            0,             0, 0,                          1]


T_fkine =

    0.3420   -0.9397         0    2.4161
    0.9397    0.3420         0    2.8794
         0         0    1.0000         0
         0         0         0    1.0000

No singularities found within joint limits
The robot does not have closed form solution. Try inverse kinematics using jacobian

q_ikine_ii =

    2.2457   -1.8384


q_ikine_ii2 =

    3.1416    3.1416

fx >>
```

As seen in the T_fkine matrix, the z component is 0, since the robot is a planar one.



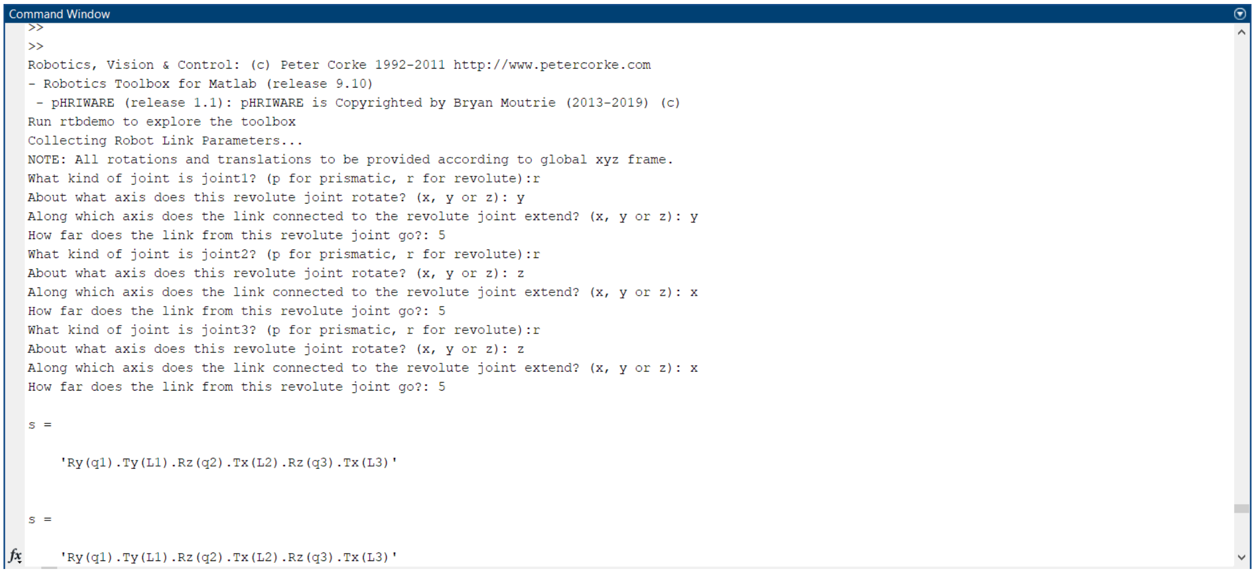
```
Command Window
J_sym =

[ - 2*sin(conj(q1)) - 2*sin(conj(q1) + conj(q2)), -2*sin(conj(q1) + conj(q2))]
[   2*cos(conj(q1)) + 2*cos(conj(q1) + conj(q2)),  2*cos(conj(q1) + conj(q2))]
[                                              0,                           0]
[                                              0,                           0]
[                                              0,                           0]
[                                              1,                           1]


J_num_0 =

   -3.4898   -1.7880
    0.1545   -0.8961
         0         0
         0         0
         0         0
    1.0000    1.0000


xe_dot =

   -7.0658
   -1.6378
         0
         0
         0
    3.0000

fx
```

```
xe_dot =

   -7.0658
   -1.6378
         0
         0
         0
    3.0000


q_dot_inv_diff =

   -0.2515
    0.4204

fx >>
```

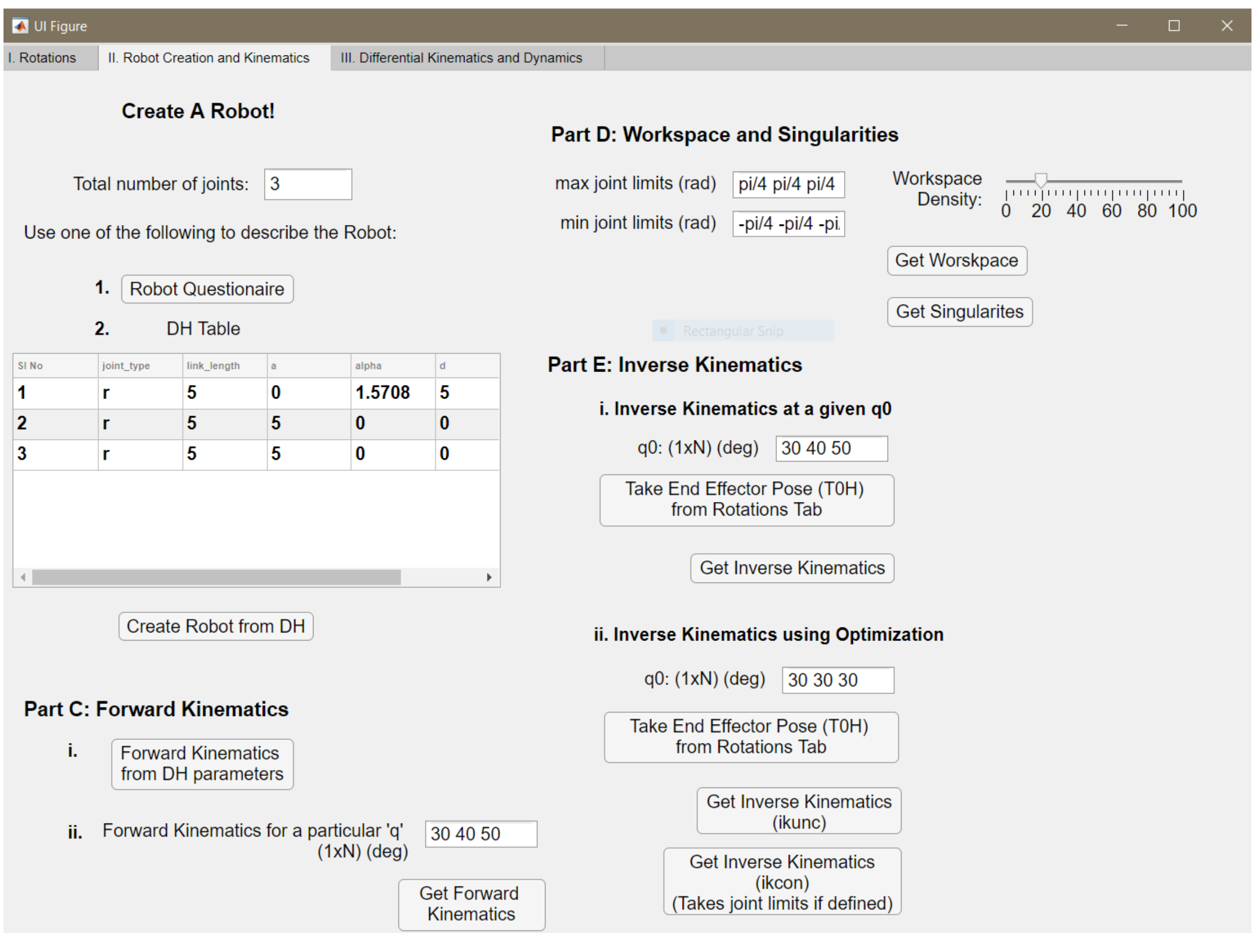
The jacobians and end effector poses also show the planarity of the robot.

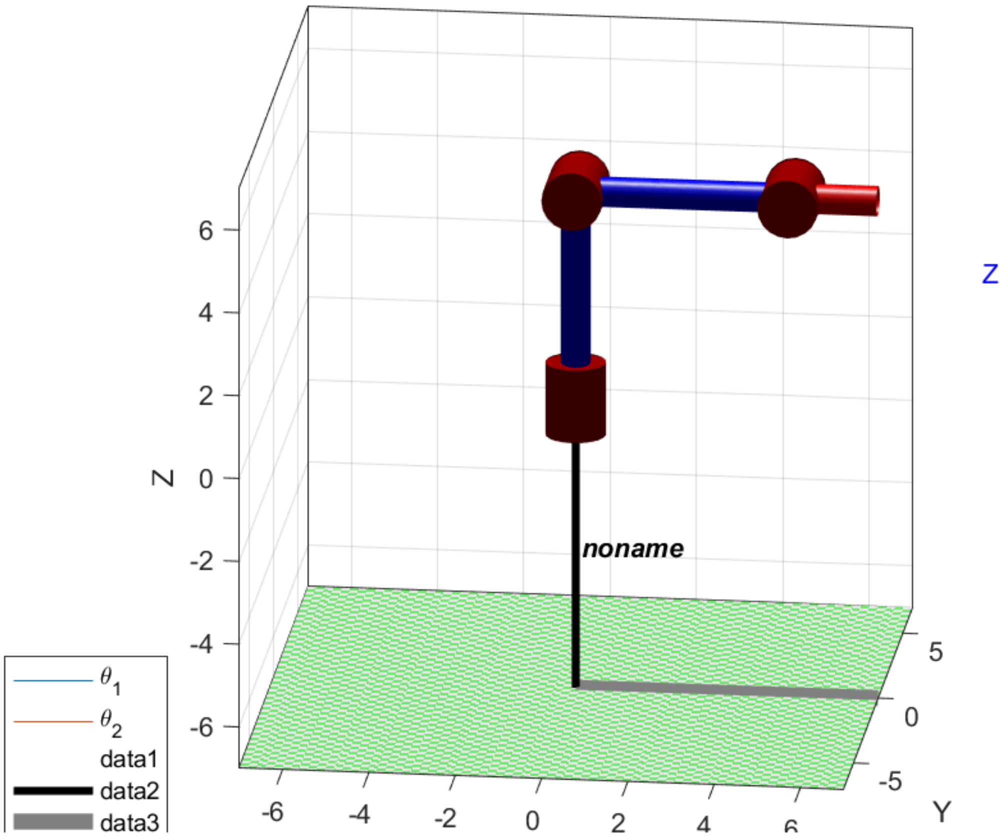
Example 3: Creation of Three Revolute 3D Robot and Statics/Dynamics

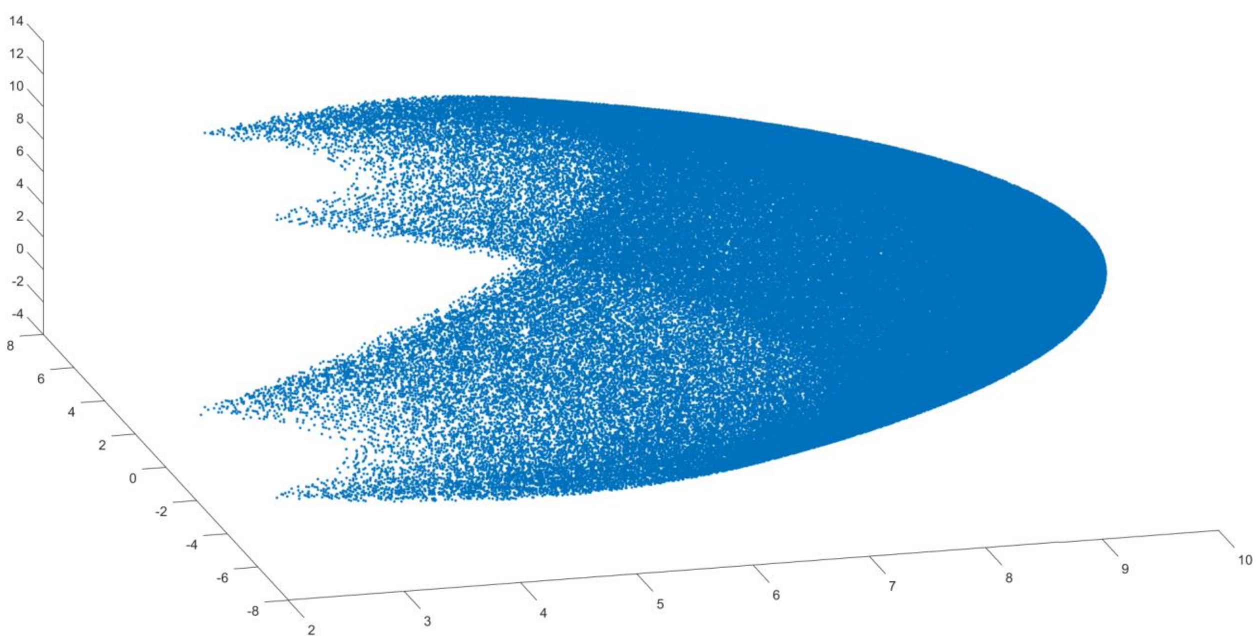
This time the robot was made using the questionnaire option provided. The following was input to the questionnaire.



```
Command Window
>>
>>
Robotics, Vision & Control: (c) Peter Corke 1992-2011 http://www.petercorke.com
- Robotics Toolbox for Matlab (release 9.10)
 - pHRIWARE (release 1.1): pHRIWARE is Copyrighted by Bryan Moutrie (2013-2019) (c)
Run rtbdemo to explore the toolbox
Collecting Robot Link Parameters...
NOTE: All rotations and translations to be provided according to global xyz frame.
What kind of joint is joint1? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): y
Along which axis does the link connected to the revolute joint extend? (x, y or z): y
How far does the link from this revolute joint go?: 5
What kind of joint is joint2? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y or z): x
How far does the link from this revolute joint go?: 5
What kind of joint is joint3? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y or z): x
How far does the link from this revolute joint go?: 5

s =

    'Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)'


s =

fx    'Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)'
```



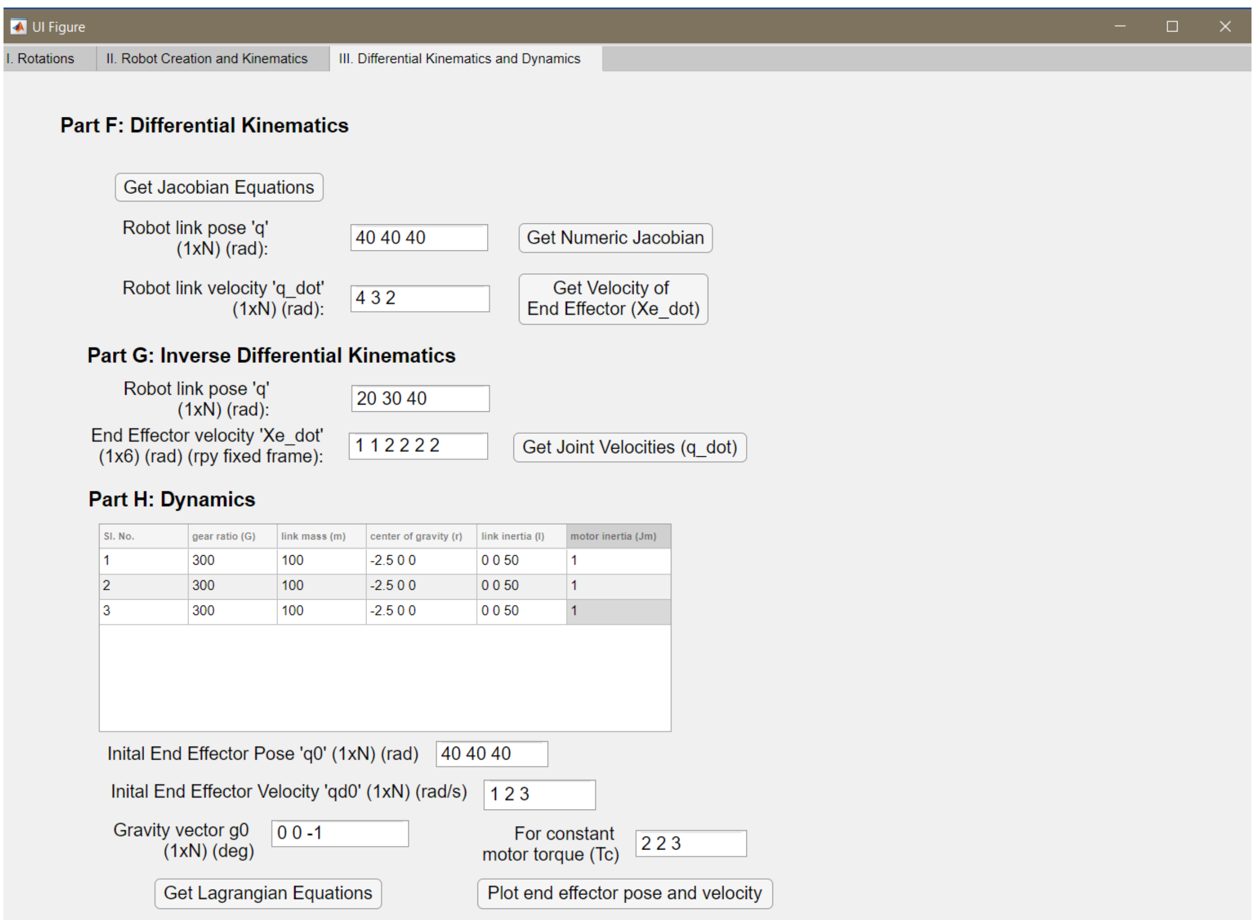




The workspace is seen to be 3 dimensional for the joint limits +/- [pi/4, pi/4, pi/4]

The following forward kinematics were obtained. Again showing three dimensional reach.

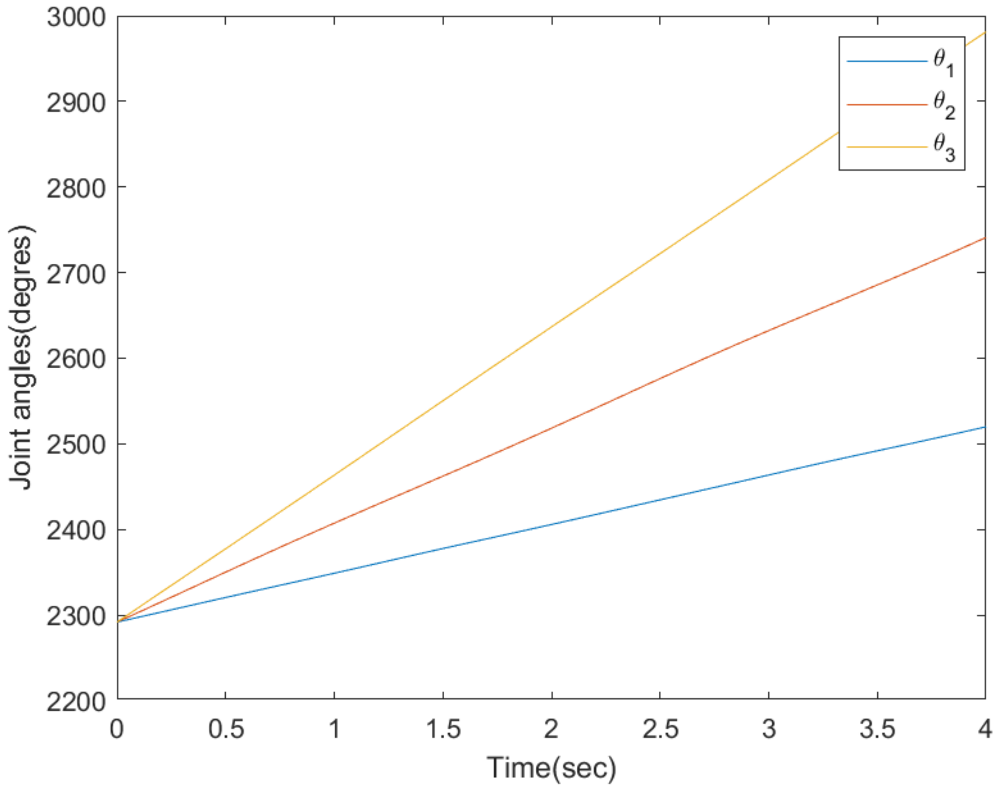
```
T_fkine2_sym =

[ cos(q3)*(cos(q1)*cos(q2) - cos(3927/2500)*sin(q1)*sin(q2)) - sin(q3)*(cos(q1)*sin(q2) + cos(3927/2500)*cos(q2)*sin(q1)), - cos(q3)*(cos(q1)*sin(q2) + c
[ cos(q3)*(cos(q2)*sin(q1) + cos(3927/2500)*cos(q1)*sin(q2)) - sin(q3)*(sin(q1)*sin(q2) - cos(3927/2500)*cos(q1)*cos(q2)), - cos(q3)*(sin(q1)*sin(q2) - c
[                                                                                              sin(3927/2500)*sin(q2 + q3),
[                                                                                                                          0,


T_fkine =

    0.0000   -0.8660    0.5000    3.3171
    0.0000   -0.5000   -0.8660    1.9151
    1.0000    0.0000    0.0000   13.2139
         0         0         0    1.0000


>>
>>
No singularities found within joint limits
The robot does not have closed form solution. Try inverse kinematics using jacobian

q_ikine_ii =

    2.6180   -0.5236   -2.0944


q_ikine_ii2 =

    2.6180   -0.5238   -2.0942
```



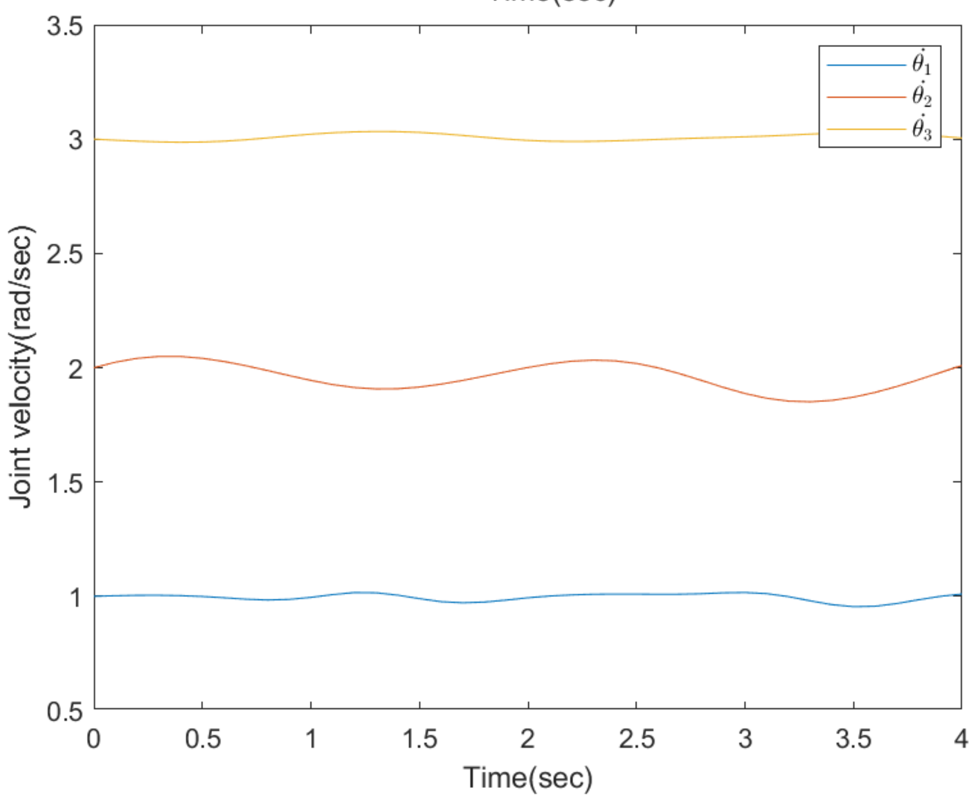
```
Command Window

J_sym =

[ 5*sin(conj(q3))*(sin(conj(q1))*sin(conj(q2)) - cos(conj(q1))*cos(conj(q2))*cos(3927/2500)) - 5*cos(conj(q3))*(cos(conj(q2))*sin(conj(q1)) + cos(conj(q1
[ 5*cos(conj(q3))*(cos(conj(q1))*cos(conj(q2)) - sin(conj(q1))*sin(conj(q2))*cos(3927/2500)) - 5*sin(conj(q3))*(cos(conj(q1))*sin(conj(q2)) + cos(conj(q2
[
[
[
[


J_num_0 =

    2.8960   -0.8296   -3.3143
    2.5921    0.9268    3.7028
   -0.0000   -3.8866   -0.5519
   -0.0000    0.7451    0.7451
   -0.0000    0.6669    0.6669
    1.0000    0.0000    0.0000


xe_dot =

    2.4665
   20.5546
  -12.7638
    3.7256
    3.3347
    4.0000
```

Dynamics Equations yielded the following results:

```
Command Window
    0.0007
    0.6639
   -0.1523


B =

[         Il1_1 + Il1_2 + Il2_1 + Im1_2 + Im2_1 + Im1_1*kr1^2 + mm1_2*(5*cos(conj(q1))*cos(conj(q2)) - 5*sin(conj(q1))*sin(conj(q2))*cos(3927/2500))*(5*c
[ Il1_2*cos(3927/2500) + Il2_1*cos(3927/2500) + Im1_2*cos(3927/2500) - mm1_2*(5*cos(conj(q1))*cos(conj(q2)) - 5*sin(conj(q1))*sin(conj(q2))*cos(3927/2500
[

>>
>>
>>

C =

[
[ q2_dot*(mm1_2*(5*cos(conj(q1))*sin(conj(q2)) + 5*cos(conj(q2))*sin(conj(q1))*cos(3927/2500))*(5*sin(q1)*sin(q2) - 5*cos(3927/2500)*cos(q1)*cos(q2)) - m
[

G =

                                                                                                                         0
 sin(3927/2500)*conj(g)*(5*ml1_2*cos(conj(q2)) + 5*mm1_2*cos(conj(q2)) + l2*ml2_1*cos(conj(q2)) + l3*ml1_2*cos(conj(q2) + conj(q3)))
                                                                               l3*ml1_2*cos(conj(q2) + conj(q3))*sin(3927/2500)*conj(g)

Fast RNE: (c) Peter Corke 2002-2012
>>
```



Joint angles(degres)
Time(sec)
θ_1
θ_2
θ_3



Joint velocity(rad/sec)
Time(sec)
$\dot{\theta}_1$
$\dot{\theta}_2$
$\dot{\theta}_3$

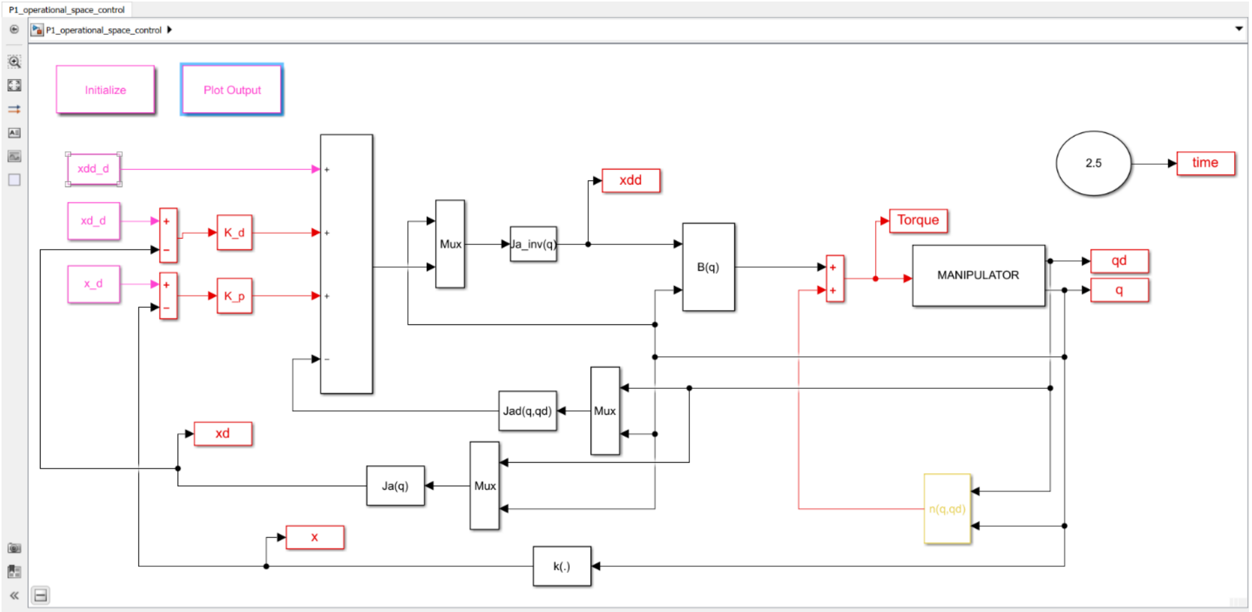
Example 3: Controls (Force and Motion Control)

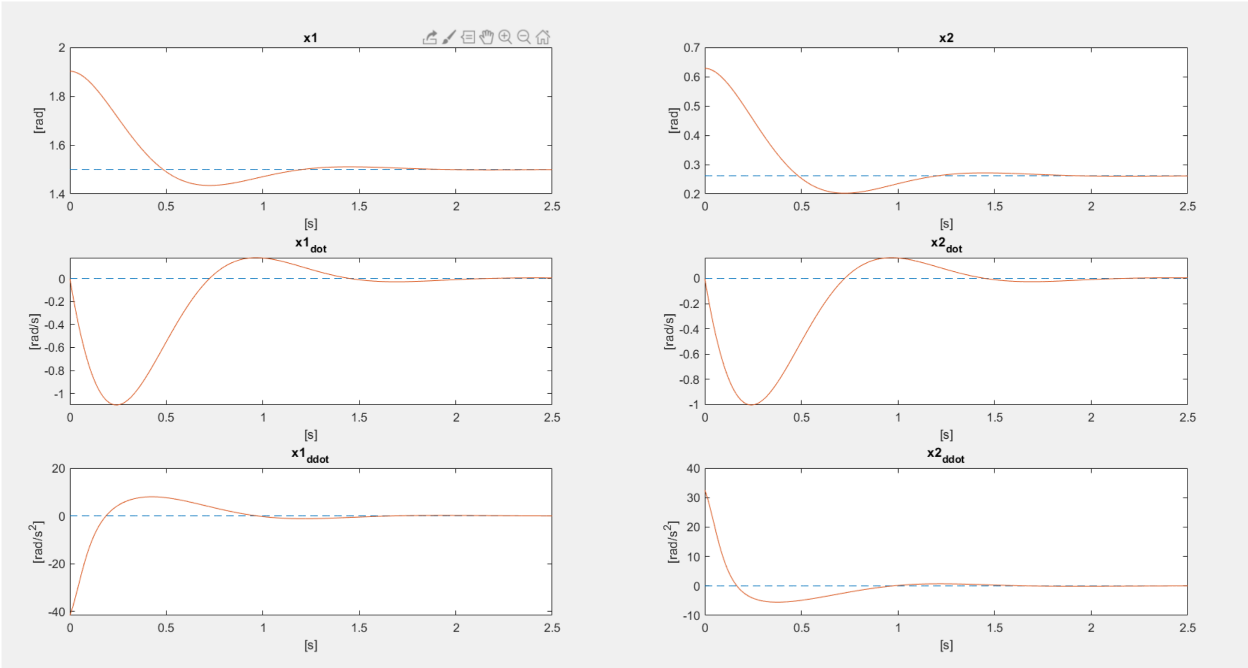
The control portion of the project has been implemented separate from the GUI. This can be run directly using matlab/Simulink using the two folders present in the controls sub-folder.

Steps to run controls:

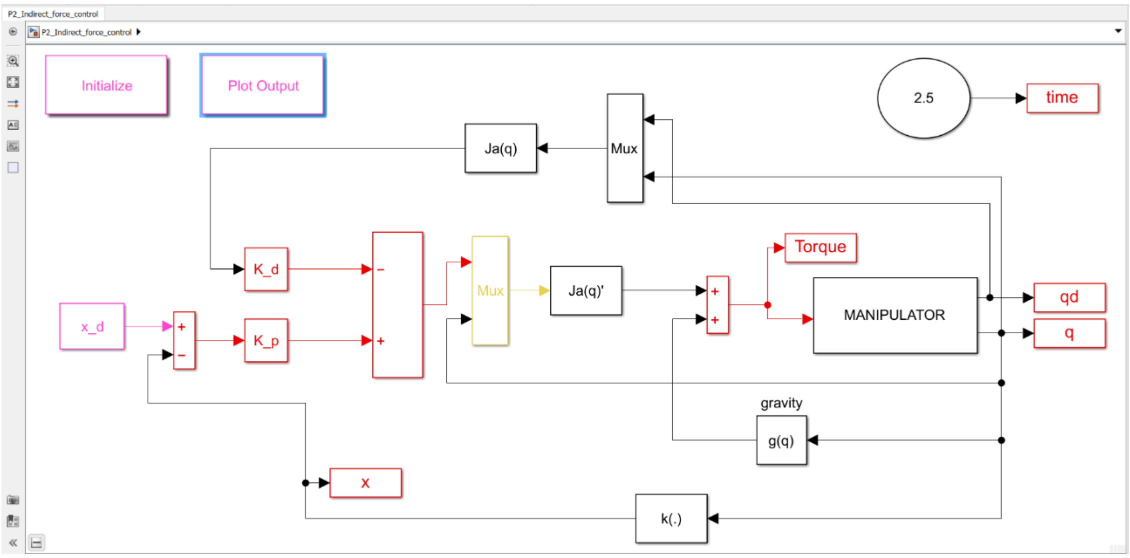
- Open Simulink file
- Press initialize
- Press Play button
- Press Plots

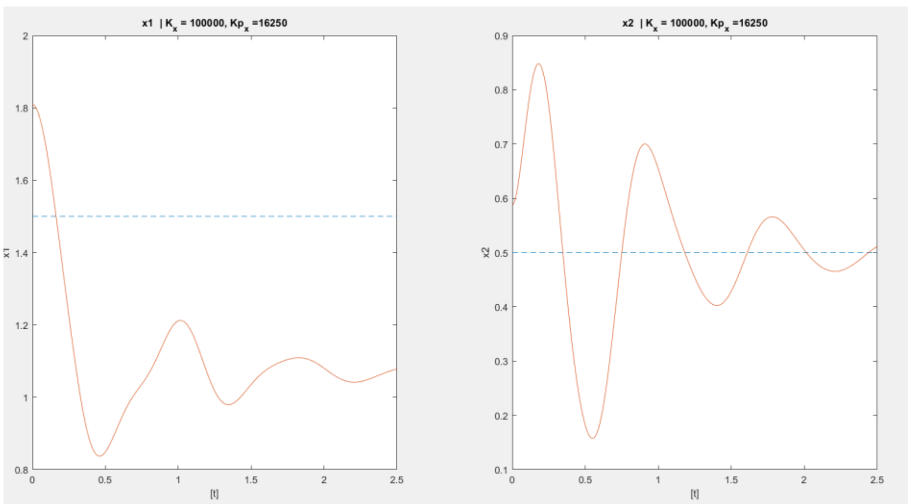
PART A: Operational space inverse dynamics control

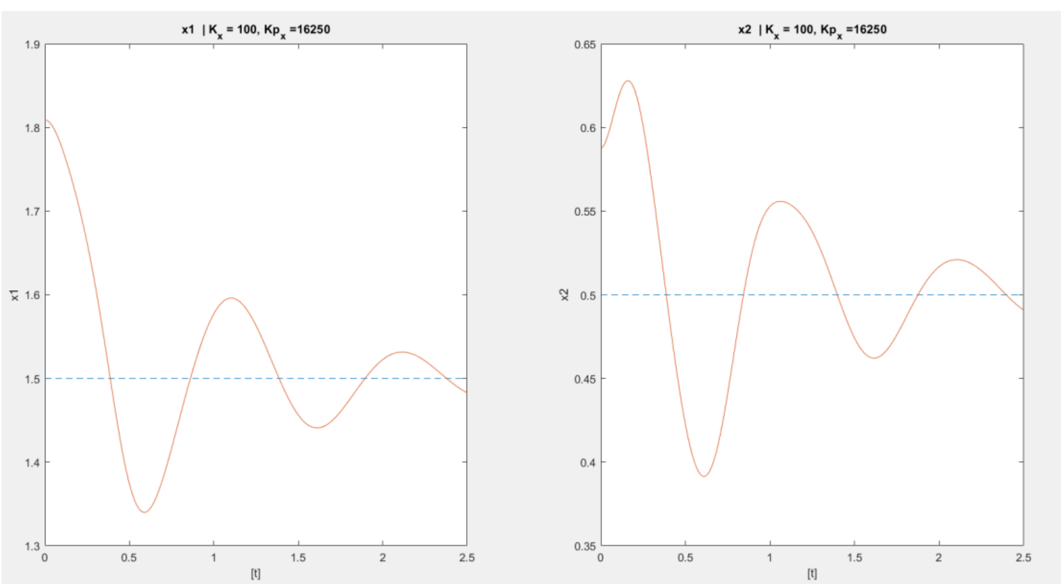




<u>PART B: Force control via compliance control</u>



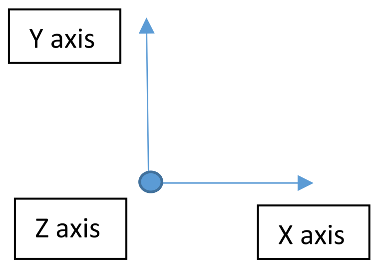




APPENDIX:

Conversion of information provided by the user to DH parameters of the robot

The program first asks for the number of links in your robot arm and then asks you which type of link the first link is. Remember, a prismatic link is a link that extends forward and a revolute link is a link that rotates about an axis. All rotations and translations are about a global frame xyz represented in the figure, which means x and y represent the plane of the paper and z axis is coming out of the paper if the robot manipulator is drawn on a sheet of paper.



Steps to generate the DH parameters –

1. The program asks for number of links and hence you must count the total number of links in your robotic arms and input that number.
2. It will then ask you for the type of link. Press ‘p’ for prismatic and ‘r’ for revolute. Depending on which type of link you chose, the program may ask you different questions.
3. First, it will ask you ‘‘How far does this revolute/prismatic link go?’ which simply means the length of your link.
4. It will then ask 'Which axis does this prismatic/revolute link extend towards?’ and you need to define it in terms of either x, y or z axis by typing ‘x’, ‘y’ or ‘z’.
5. It will then ask 'About what axis does this revolute joint rotate?’ for the case of revolute joints and you need to define the axis about which this link is rotating, again in terms of x, y and z.
6. Repeat steps 2 – 5 until all information of the links is gathered.

Examples of User Information to DH Parameter conversion –

Note: All figures have been taken from Robotics Modelling, Planning and Control - Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G

Example 1: Three link planar arm

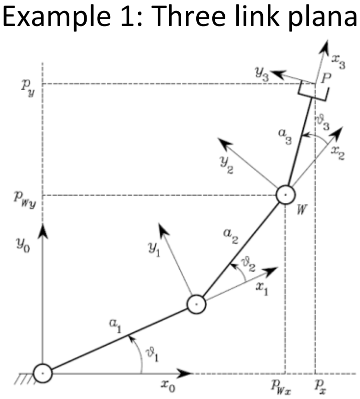
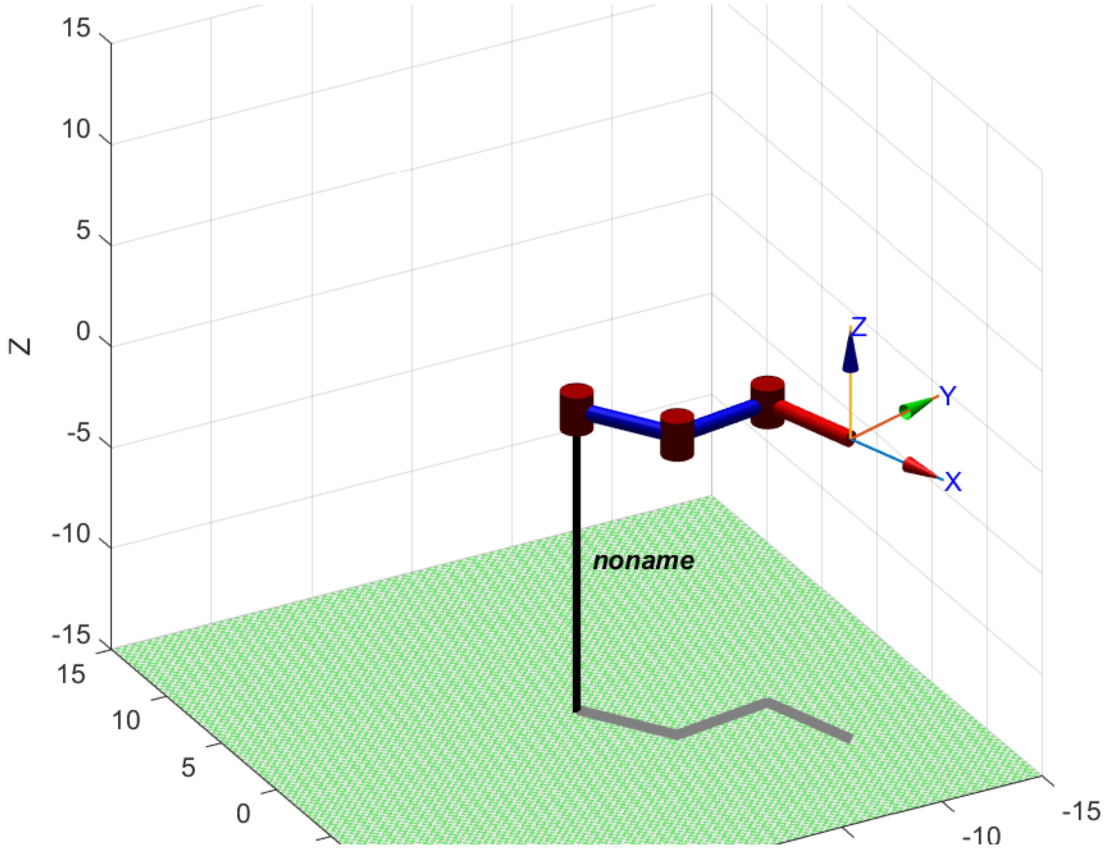


Fig. 2.20. Three-link planar arm




```
Collecting Robot Link Parameters...
NOTE: All rotations and translations to be provided according to global xyz
frame.
What kind of joint is joint1? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y
or z): x
How far does the link from this revolute joint go?: 5
What kind of joint is joint2? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y
or z): x
How far does the link from this revolute joint go?: 5
What kind of joint is joint3? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y
or z): x
How far does the link from this revolute joint go?: 5

s =

    'Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)'

>> dh = DHFactor(s);
Undefined function or variable 'DHFactor'.

Did you mean:
>> startup_rvc
Robotics, Vision & Control: (c) Peter Corke 1992-2011
http://www.petercorke.com
```

```
- Robotics Toolbox for Matlab (release 9.10)
 - pHRIWARE (release 1.1): pHRIWARE is Copyrighted by Bryan Moutrie (2013-
2019) (c)
Run rtbdemo to explore the toolbox
- Machine Vision Toolbox for Matlab (release 3.4)
>> dh = DHFactor(s);
In DHFactor
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
initial merge + swap
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
joint vars to Z
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
0--------------------------------------
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
1--------------------------------------
** deal with Ry/Ty
Rz(q1).Tx(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
  adding: DH(null, 0, 0, 0) += Rz(q1)
  adding: DH(q1, 0, 0, 0) += Tx(L1)
  adding: DH(null, 0, 0, 0) += Rz(q2)
  adding: DH(q2, 0, 0, 0) += Tx(L2)
  adding: DH(null, 0, 0, 0) += Rz(q3)
  adding: DH(q3, 0, 0, 0) += Tx(L3)
DH(q1, 0, L1, 0).DH(q2, 0, L2, 0).DH(q3, 0, L3, 0)
In DHFactor, parseString is done
>> r = eval(dh.command('myrobot'))
Error using evalin
Undefined function or variable 'L1'.

Error in opaque/eval (line 15)
    [varargout{1:nargout}] = evalin('caller', tryVal);

>> syms L1 L2 L3
r = eval(dh.command('myrobot'))

r =

myrobot (3 axis, RRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|         L1|          0|          0|
|  2|         q2|          0|         L2|          0|          0|
|  3|         q3|          0|         L3|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
Example 2: Anthropomorphic Arm
```

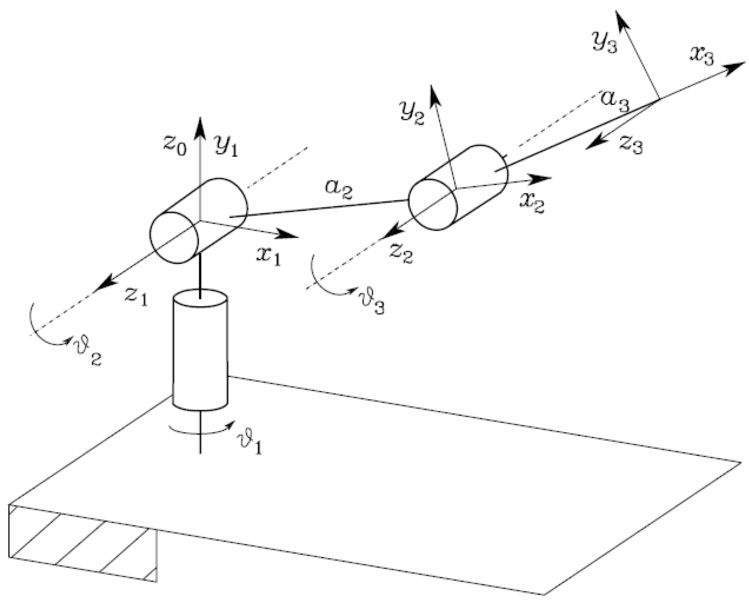
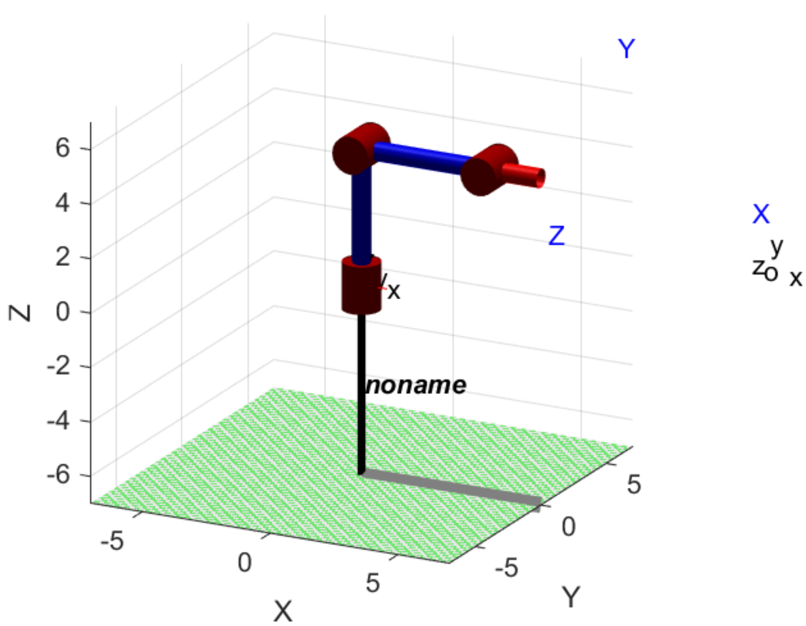


Fig. 2.23. Anthropomorphic arm




```
Collecting Robot Link Parameters...
NOTE: All rotations and translations to be provided according to global xyz
frame.
What kind of joint is joint1? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): y
Along which axis does the link connected to the revolute joint extend? (x, y
or z): y
How far does the link from this revolute joint go?: 5
What kind of joint is joint2? (p for prismatic, r for revolute):r
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y
or z): x
How far does the link from this revolute joint go?: 5
What kind of joint is joint3? (p for prismatic, r for revolute):r
```

```
About what axis does this revolute joint rotate? (x, y or z): z
Along which axis does the link connected to the revolute joint extend? (x, y
or z): x
How far does the link from this revolute joint go?: 5

s =

    'Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)'

In DHFactor
Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
initial merge + swap
Ry(q1).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
ReplaceToZ: Ry(q1) := Rx(-90)Rz(q1)Rx(+90)
joint vars to Z
Rx(-90).Rz(q1).Rx(+90).Ty(L1).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
0---------------------------------------
ReplaceY: Rx(+90)Ty(L1) := Tz(L1)Rx(+90)
Rx(-90).Rz(q1).Tz(L1).Rx(+90).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
1---------------------------------------
Rx(-90).Rz(q1).Tz(L1).Rx(+90).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
1---------------------------------------
** deal with Ry/Ty
Rx(-90).Rz(q1).Tz(L1).Rx(+90).Rz(q2).Tx(L2).Rz(q3).Tx(L3)
  adding: DH(null, 0, 0, 0) += Rz(q1)
  adding: DH(q1, 0, 0, 0) += Tz(L1)
  adding: DH(q1, L1, 0, 0) += Rx(+90)
  adding: DH(null, 0, 0, 0) += Rz(q2)
  adding: DH(q2, 0, 0, 0) += Tx(L2)
  adding: DH(null, 0, 0, 0) += Rz(q3)
  adding: DH(q3, 0, 0, 0) += Tx(L3)
Rx(-90).DH(q1, L1, 0, 90).DH(q2, 0, L2, 0).DH(q3, 0, L3, 0)
In DHFactor, parseString is done

r =

myrobot (3 axis, RRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|         L1|          0|       pi/2|          0|
|  2|         q2|          0|         L2|          0|          0|
|  3|         q3|          0|         L3|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  0  1  0           0  1  0  0
        9.81         0 -1  0  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```