Robot Morphology

Remember:

Team graoup 11k: Antonio & Juan

Link: https://drive.matlab.com/sharing/.....your

Table of Contents

6R Robot. Puma 560

Before start the exercise see the videos:

https://youtu.be/ArzP7rh4_9Q

Call the Wired Robot object and plot it

```
close all
clear
mdl_puma560 % Invoque the puma object from the RTB
p560.plot(qs) % qz is the joint vector 1x6. Try qr, qn, any within the limits
```

Work with the wire model and change the point of view.

See: https://es.mathworks.com/help/matlab/creating_plots/setting-the-viewpoint-with-azimuth-and-elevation.html

```
close all
p560.plot(qz)
view([-42.61 46.25])
```

Play with the teach

Modify the joint angle [q1q2 q3 q4 q5 q6]). It is a kind of Joystick.

Pay attention to [x y z].

[ax ay az] are no relevant for the exercise.

```
p560.teach('approach')
```

Moving the Robot

```
clear all
close all
mdl_puma560
```

Declare a joint motion by adding rows

```
Q=zeros(100,6); % at the moment no motion
```

See the Joint 1 limits

```
q1_limits=p560.links(1, 1).qlim
```

```
q1_limits = 1×2
   -2.7925    2.7925
```

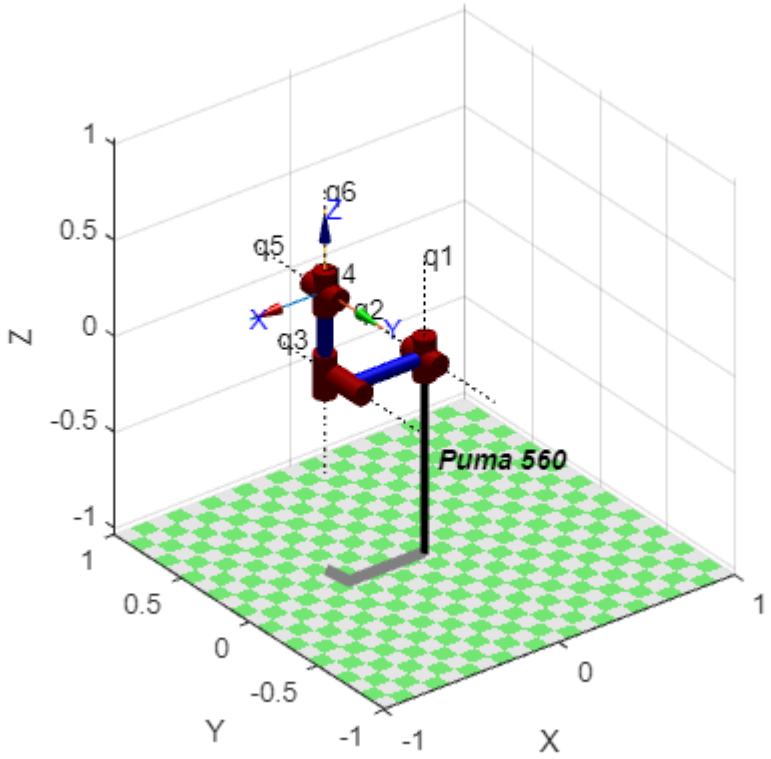
Build the joint's motion. Firts only Joint #1

```
q1=linspace(q1_limits(1),q1_limits(2),100)';
Q=[q1 Q(:,2:6)]
```

```
Q = 100×6
   -2.7925         0         0         0         0         0
   -2.7361         0         0         0         0         0
   -2.6797         0         0         0         0         0
   -2.6233         0         0         0         0         0
   -2.5669         0         0         0         0         0
   -2.5105         0         0         0         0         0
   -2.4540         0         0         0         0         0
   -2.3976         0         0         0         0         0
   -2.3412         0         0         0         0         0
   -2.2848         0         0         0         0         0
      ⋮
```

Plotting

```
p560.plot(Q,'jaxes')
```



Play with the plot options

Moving two joints. See above

```
q2_limits=p560.links(1, 2).qlim
```

```
q2_limits = 1×2
   -0.7854    3.9270
```

```
q2=linspace(q2_limits(1),q2_limits(2),100)';
Q12=[q1 q2 Q(:,3:6)];
```

Options: Add a trail to see the trajectory, display the join axis, make biger or smaller the robot

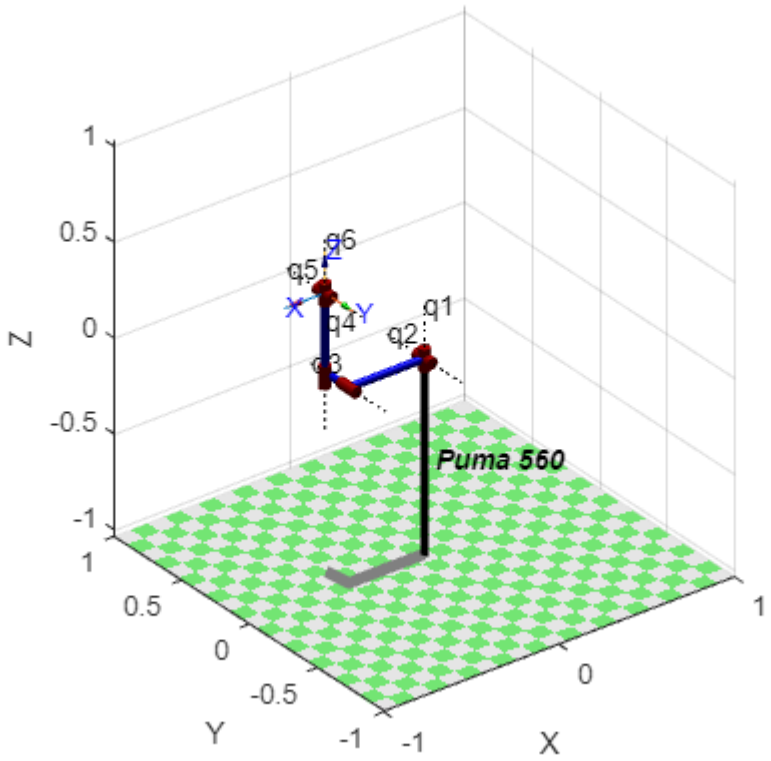
Visit the RTB manual.pdf at:

https://atenea.upc.edu/pluginfile.php/3871049/mod_resource/content/3/robot.pdf

or

https://petercorke.com/toolboxes/robotics-toolbox/

```
close all
mdl_puma560
p560.plot(Q12,'trail','--','jaxes','zoom',2) %% Play outside the mlx file to see it: copy the s
```



Play with other options to get familiar with. You must! becouse all along the course it will be necesary

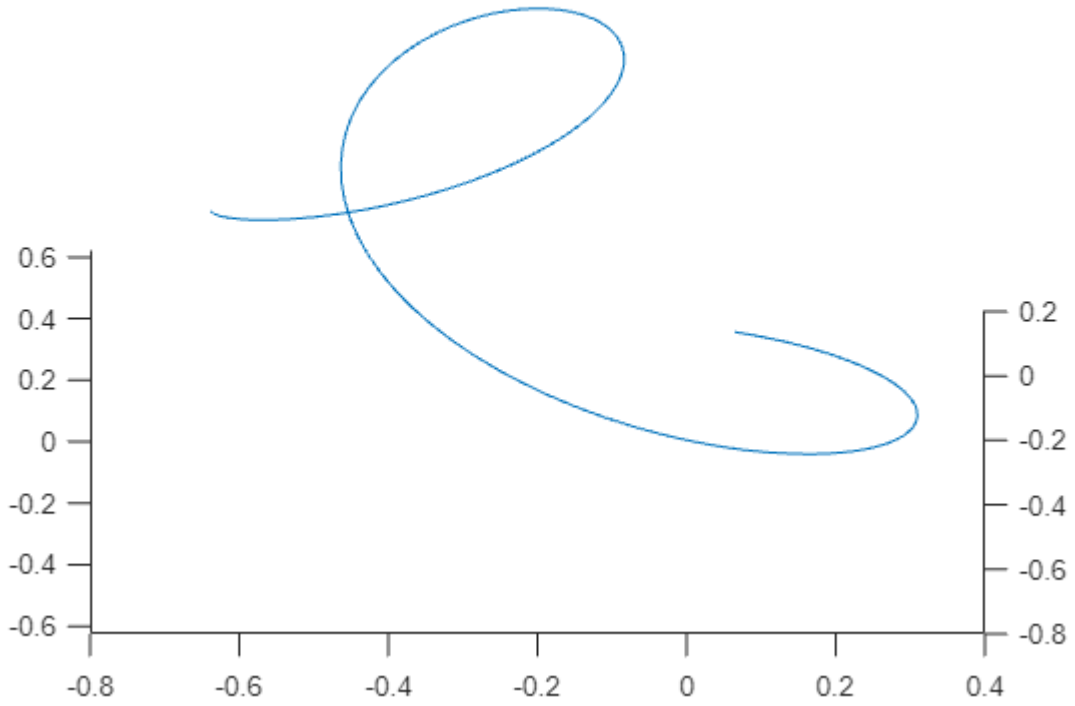
Recovering End efector position

Use function 'fkine' for recovering the finger tips of the robot

```
T=p560.fkine(Q12); % Forward Kinematic to be explained. Given Theta's (q's) obtain the robot EE
ft=[T.t] % to gert only the position
```

```
ft = 3×100
   -0.6386   -0.6335   -0.6251   -0.6135   -0.5990   -0.5817   -0.5618   -0.5397 ···
   -0.0728   -0.1086   -0.1436   -0.1772   -0.2092   -0.2393   -0.2672   -0.2928
   -0.0144    0.0154    0.0451    0.0747    0.1042    0.1334    0.1623    0.1909
```

```
figure
plot3(ft(1,:),ft(2,:), ft(3,:))
view(0,40)
```



Working area

```
clear all
close all
mdl_puma560
q2_limits=p560.links(1, 2).qlim
```

```
q2_limits = 1×2
   -0.7854    3.9270
```

```
q2=linspace(q2_limits(1),q2_limits(2),100)';
Q= [zeros(100,1) linspace(q2_limits(1),q2_limits(2),100)' zeros(100,4) ]
```

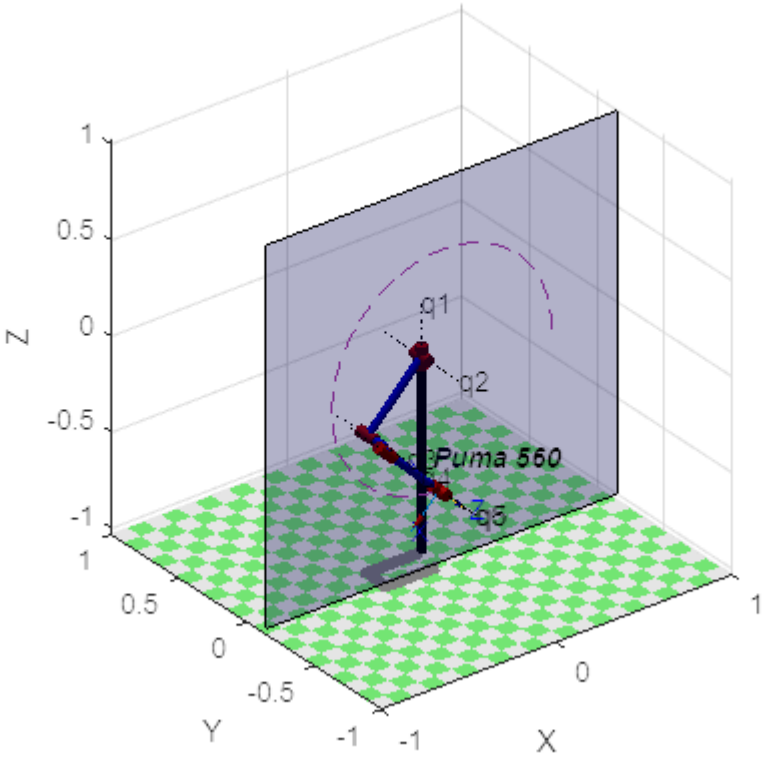
```
Q = 100×6
         0   -0.7854         0         0         0         0
         0   -0.7378         0         0         0         0
         0   -0.6902         0         0         0         0
         0   -0.6426         0         0         0         0
         0   -0.5950         0         0         0         0
         0   -0.5474         0         0         0         0
         0   -0.4998         0         0         0         0
         0   -0.4522         0         0         0         0
         0   -0.4046         0         0         0         0
         0   -0.3570         0         0         0         0
    ⋮
```

```
p560.plot(Q,'trail','--','jaxes','zoom',2)
T=p560.fkine(Q);
```

```
ft=[T.t]
```

```
ft = 3×100
    0.6250    0.6250    0.6235    0.6207    0.6164    0.6108    0.6037    0.5953 ···
   -0.1501   -0.1501   -0.1501   -0.1501   -0.1501   -0.1500   -0.1500   -0.1500
   -0.0144    0.0154    0.0451    0.0747    0.1042    0.1334    0.1623    0.1909
```

```
hold on
v = [-1 -0.1501 -1 ; 1 -0.1501 -1 ; 1 -0.1501 1; -1 -0.1501 1];
f = [1 2 3 4];
patch('Faces',f,'Vertices',v,'FaceColor','blue','FaceAlpha',.3)
```



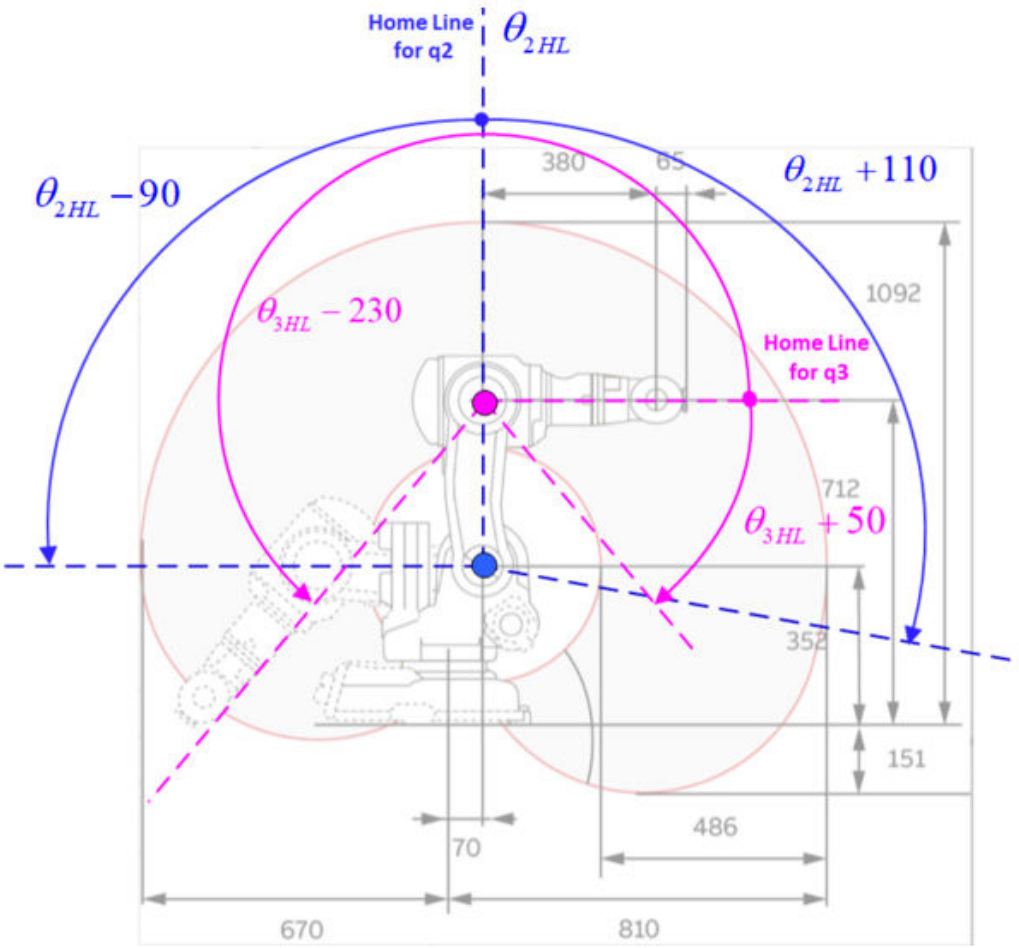
IRB140 exercise

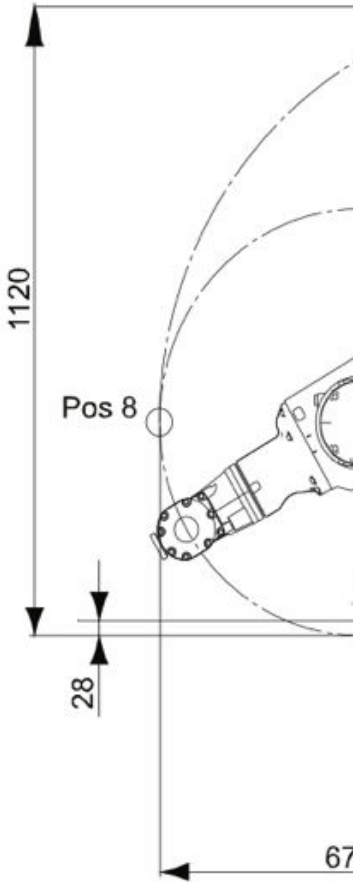
Fill the table

Understand the numbers that appears in the following table and fill/create a matrix with the irb140RTB angles.

Pay attention to home position of the ABB Drawing and the wired model of the RTB

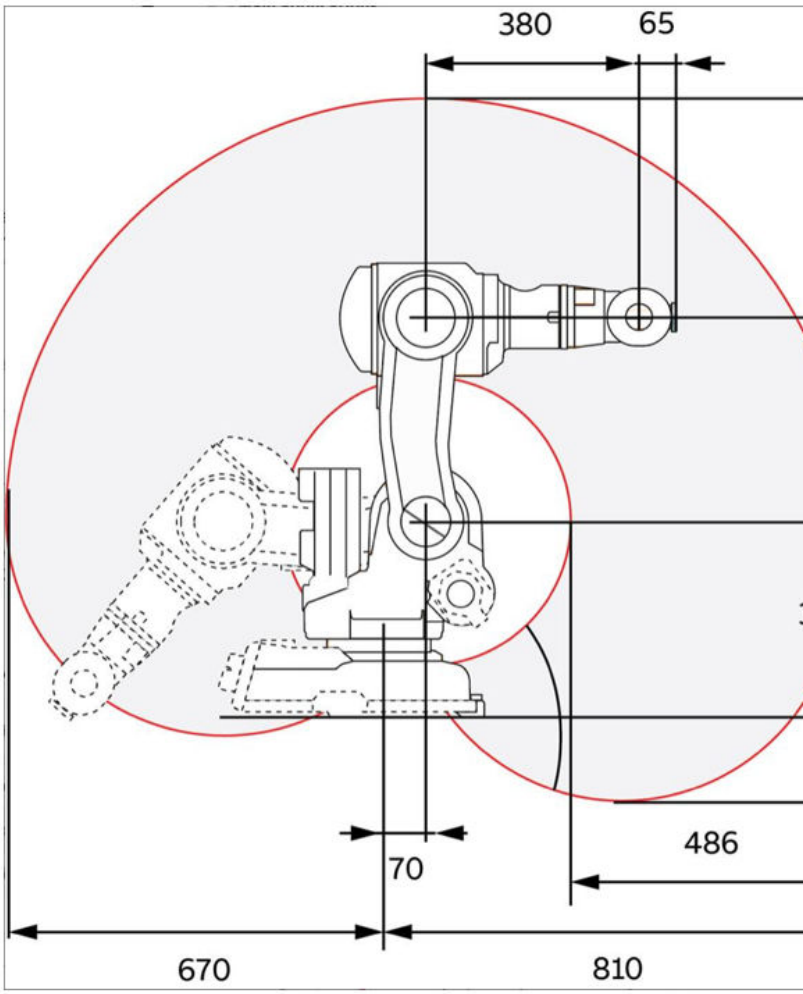
			ABB_Drawing		R
Pose	X posion	Z position	Axis-2	Axis-3	Axi
0	450	712	0	0	
1	70	1092	0	-90	
2	314	421	0	50	
3	765	99	110	-90	
6	1	596	-90	50	
7	218	558	110	-230	
8	-670	352	-90	-90	





Draw the work space

Get a joint sequence movement to recover the work space as shown in the figure. See video rb140_WS_Solution.mp4.



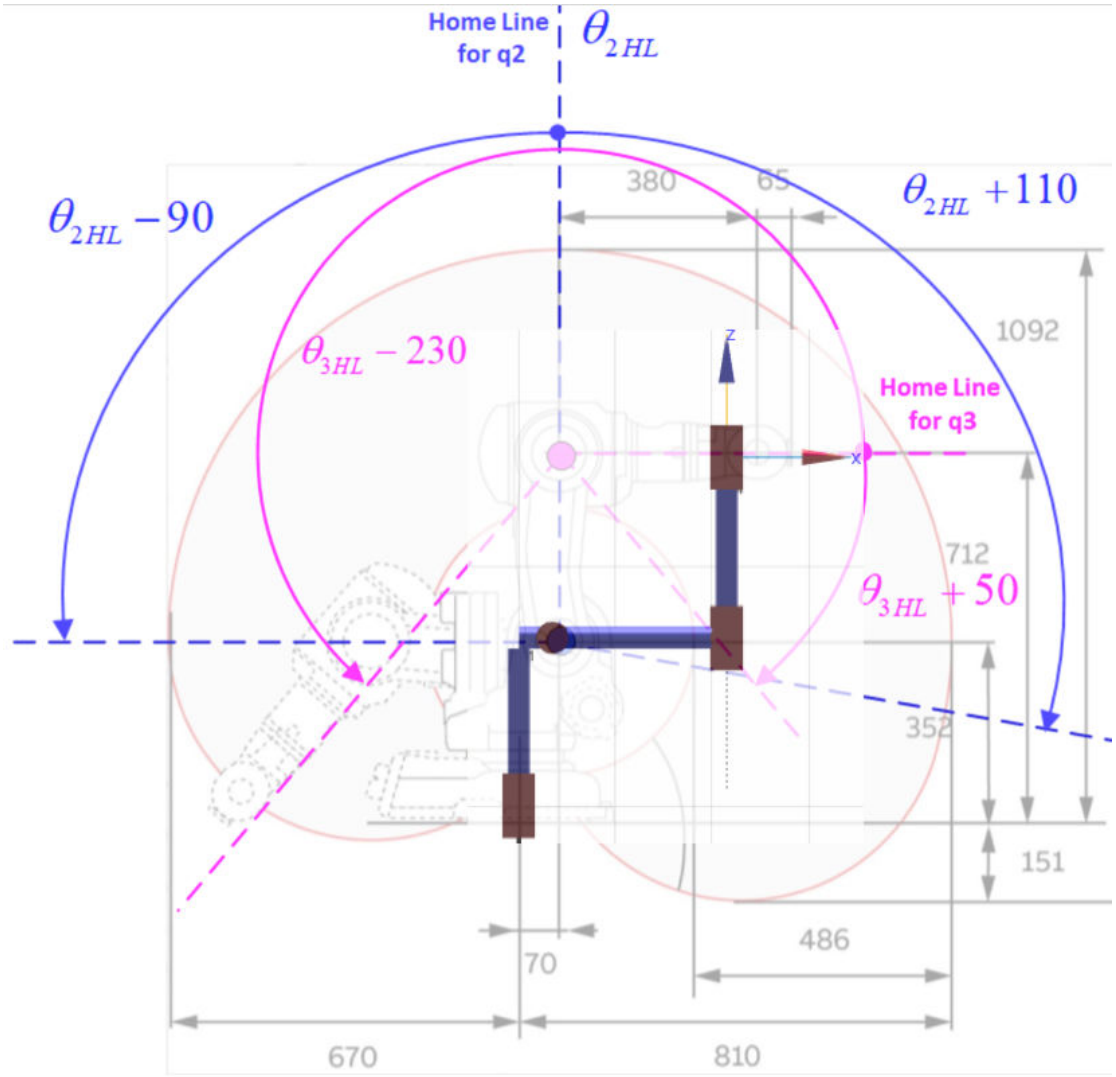
Invoque IRB140

```
clear
close all
mdl_irb140
```

Plot the IRB

```
irb140.plot(qz)
```

To think about




```
figure
irb140.plot(qz,'zoom',2, 'view',[0 0])
irb140.teach('approach')
```