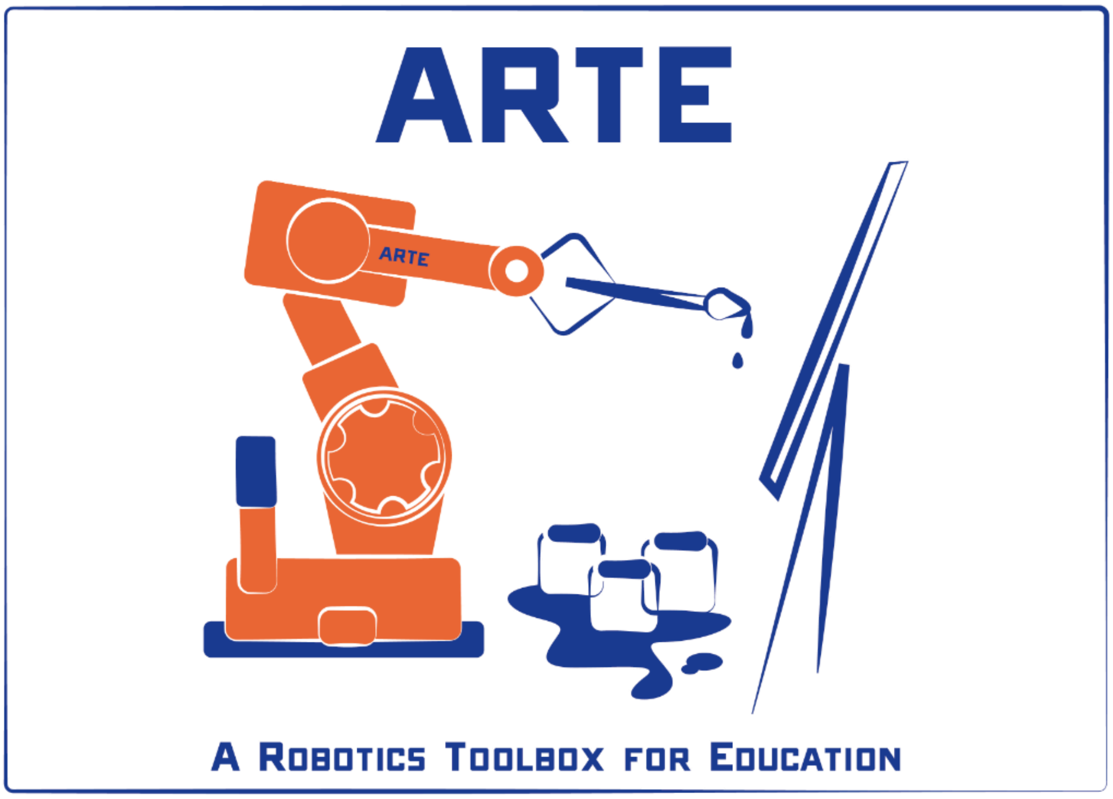


PRACTICAL SESSION 1: DIRECT KINEMATICS

Arturo Gil Aparicio

arturo.gil@umh.es

OBJECTIVES

The main objective is to reinforce the student in the following topics:

- **Homogeneous transformation matrices: joint representation of translation and orientation.**
- **Denavit-Hartenberg's parameters of a robotic manipulator. A solution for the direct kinematic problema for a serial manipulator.**
- **Kinematic analysis of the end effector's velocity.**

Index

1 First steps

You can start by initializing the library and running a demo:

```
>> pwd
ans =
/Users/arturogilaparicio/Desktop/arte3.1.4
>> init_lib
ARTE (A Robotics Toolbox for Education) (c) Arturo Gil 2012
http://www.arvc.umh.es/arte
>> demos
INVERSE AND DIRECT KINEMATICS DEMO
THE DEMO PRESENTS THE DIRECT AND INVERSE KINEMATIC PROBLEM
q =
   0.5000    0.2000   -0.2000    0.5000    0.2000   -0.8000

ans =
/Users/arturogilaparicio/Desktop/arte_lib2.7/robots/abb/IRB140

Reading link 0
EndOfFile found...
Reading link 1
EndOfFile found...
Reading link 2
EndOfFile found...
Reading link 3
EndOfFile found...
Reading link 4
EndOfFile found...
Reading link 5
EndOfFile found...
Reading link 6
EndOfFile found...

ADJUST YOUR VIEW AS DESIRED.
Press any key to continue...
```

Follow the steps to execute the demos. The following figures 1 and 2 present some of the results that you should obtain. To start with, we analyse a simple robotic manipulator. Figure 3 presents a 4 DOF robotic manipulator and its corresponding D-H table. A D-H reference system has been placed at each link. During the rest of the practical sessions you will be involved in the details concerning direct and inverse kinematics, dynamics, as well as robot programming.

An introduction to the library can be viewed here:

http://www.youtube.com/watch?v=s8QQydJ9PwI&list=PLClKgnzRFYe72qDYmj5CRpR9ICNnQehup&index=18

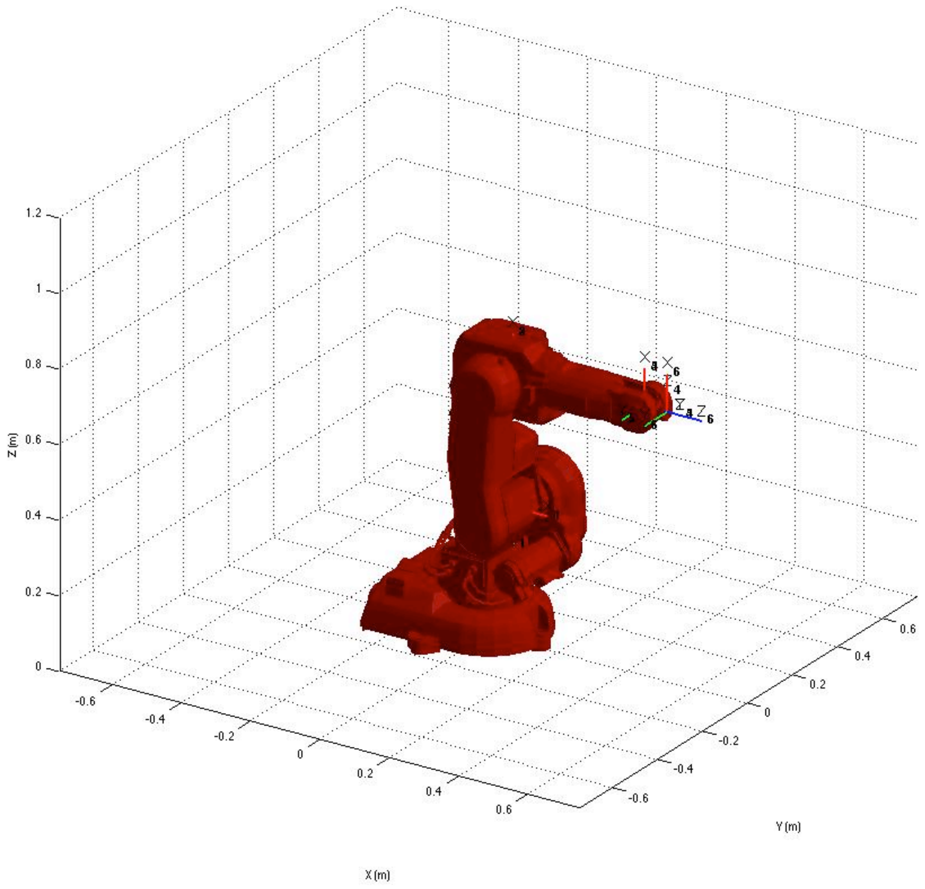


Figure 1

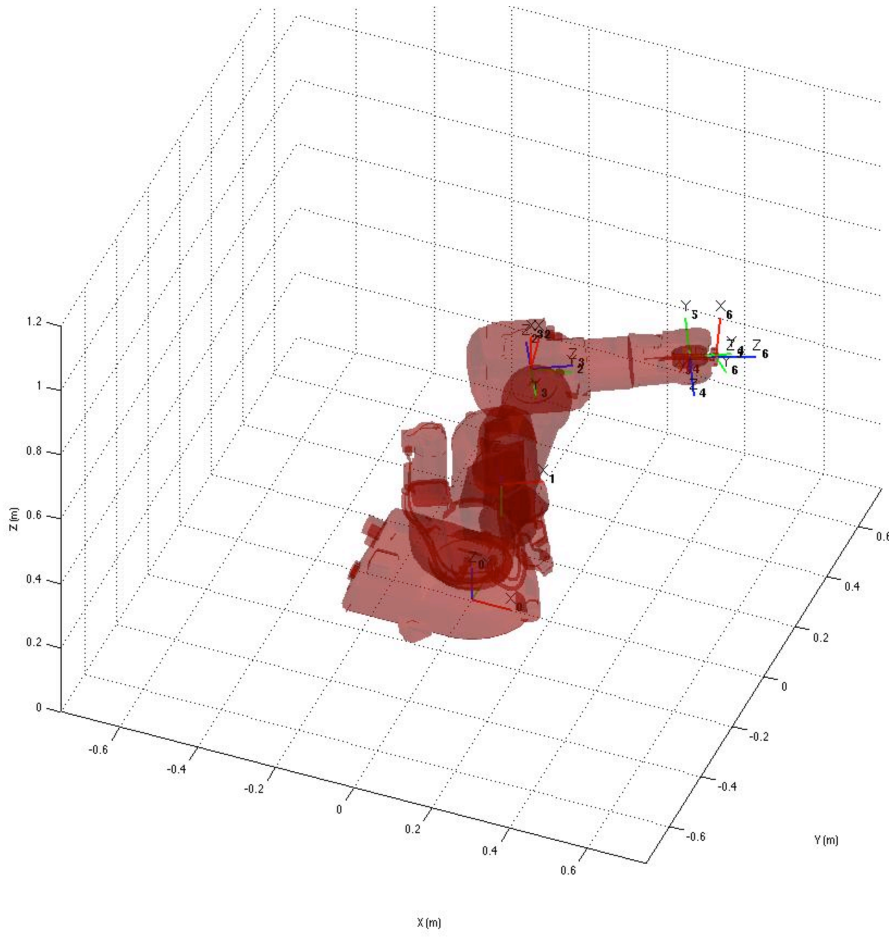


Figure 2

Next, we are going to test and understand the placement of the D-H reference systems. In order to do this, we will employ the following Matlab functions included in the library.

- **`init_lib`:** initialize the library. This command stores the current path as the base library path and initializes some configuration variables.
- **`load_robot`:** Load a robot in a variable.
- **`directkinematic(robot, q)`:** Compute the direct kinematics for a robot, given the joint coordinates q.
- **`drawrobot3d(robot, q)`:** Makes a 3D representation of the robot with joint coordinates q. The D-H reference systems are also drawn.

You can type the following commands at the Matlab prompt:

```
>> init_lib
ARTE   (A   Robotics   Toolbox   for   Education)   (c)   Arturo   Gil   2012
http://www.arvc.umh.es/arte

>> robot=load_robot('example','3dofplanar')

>> ans =
/Users/arturogilaparicio/Desktop/arte/arte3.1.4/robots/example/3dofpla
nar


[…]
EndOfFile found...
robot =
                   name: [1x23 char]
                     DH: [1x1 struct]
                    DOF: 3
                      J: []
                   kind: 'RRR'
   inversekinematic_fn: [1x37 char]
    directkinematic_fn: [1x25 char]
               maxangle: [3x2 double]
                 velmax: []
               accelmax: []
         linear_velmax: 0
                     T0: [4x4 double]
                  debug: 0
                      q: [3x1 double]
                     qd: [3x1 double]
                    qdd: [3x1 double]
                   time: []
               q_vector: []
              qd_vector: []
             qdd_vector: []
            last_target: [4x4 double]
         last_zone_data: 'fine'
                  tool0: [1x19 double]
                  wobj0: []
         tool_activated: 0
                   path: [1x73 char]
              graphical: [1x1 struct]
                   axis: [1x6 double]
          has_dynamics: 1
                      m: [1 1 1]
                      r: [3x3 double]
                      I: [3x6 double]
                     Jm: [0 0 0]
                      G: [0 0 0]
               friction: 1
                      B: [0 0 0]
                     Tc: [3x2 double]

T = directkinematic(robot, [pi/4 pi/4 pi/4])

T =

   -0.7071   -0.7071         0    0.0000
    0.7071   -0.7071         0    2.4142
         0         0    1.0000         0
         0         0         0    1.0000
>> drawrobot3d(robot, [pi/4 pi/4 pi/4])
```

First, the command **`init_lib`** initializes the library path and some variables. Next, a robotic arm has to be loaded. This is accomplished with the command **`load_robot`**. The variable `robot` stores the parameters associated with the desired robot. In our case, after calling `robot=load_robot('example','3dofplanar')`, the library reads the parameters stored in the file `robots/example/3dofplanar/parameters.m`. Alternatively, a short call to load_robot such as:

```
>> robot = load_robot
```

is also valid. You should navigate and select the parameters.m file of any robot in the library. You should now take a look at this `parameters.m` file. The meaning of each variable is described in the ARTE reference manual. Next, the call to the function:

```
>> T=directkinematic(robot, [pi/4 pi/4 pi/4])
```

Yields, as a result, the homogeneous matrix `T` describing the position and orientation of the Denavit-Hartenberg reference system 4 with respect to the reference system 0 (placed at the robot base). The vector `[pi/4 pi/4 pi/4]` corresponds to the joint coordinates. Please note that the joint coordinates 1, 2 and 3 are rotational. Finally, `drawrobot3d(robot, [pi/4 pi/4 pi/4])` makes a 3D representation of the D-H systems, as shown in Figure 3.

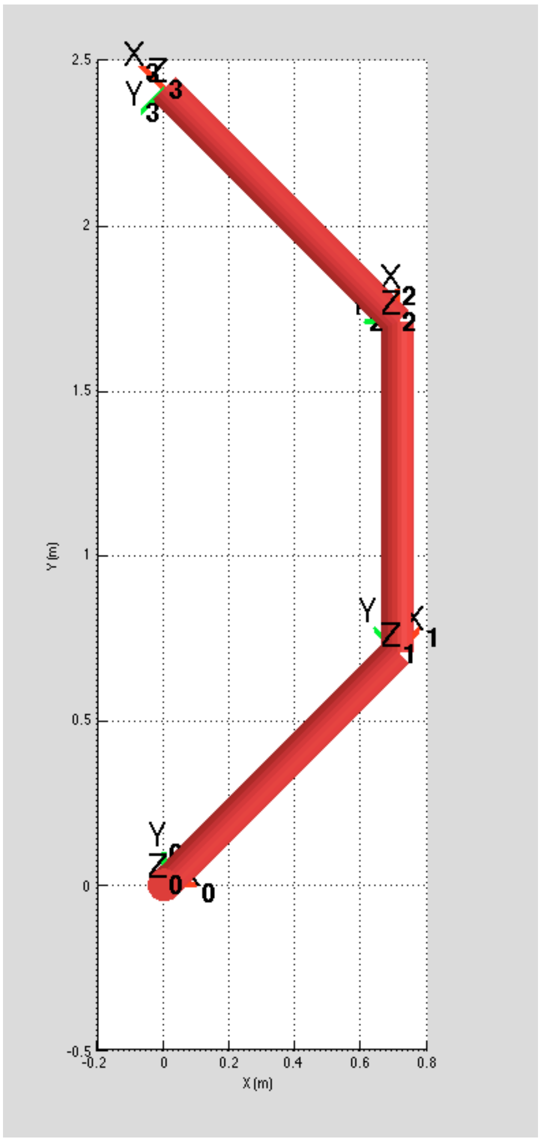


Figure 3

A general call to the function `load_robot` is:

```
robot=load_robot('manufacturer', 'robot_model');
```

For example:

```
robot=load_robot('kuka', 'KR5_scara_R350_Z200');
```

Loads the parameters of the KR5 scara R350 Z200 robot manufactured by KUKA roboter GmbH. Most of the robots already included in the library possess graphic files so that we can have a more pleasant view of the robot. For example, type the following commands:

```
>> robot=load_robot('kuka', 'KR5_scara_R350_Z200');
>> robot.graphical.draw_axes=0;
>> drawrobot3d(robot, [0 0 0.1 0]);
>> robot.graphical.draw_transparent=1
>> drawrobot3d(robot, [0 0 0.1 0]);
>> robot.graphical.draw_axes=1;
>> drawrobot3d(robot, [0 0 0.1 0]);
```

The property `robot.graphical.draw_axes` controls whether you want to draw the D-H axes on the robot. Typing `robot.graphical.draw_axes=1;` draws the axes on the robot (type 0 to hide them). On the other hand, the property `robot.graphical.draw_transparent` allows to draw the links with transparency. Typing `robot.graphical.draw_transparent=1;` draws the robot with transparency (0 is the default value for this property). You should now experiment with these parameters and observe the results. Here are some of the views that you should have obtained (Figure 4 and Figure 5).

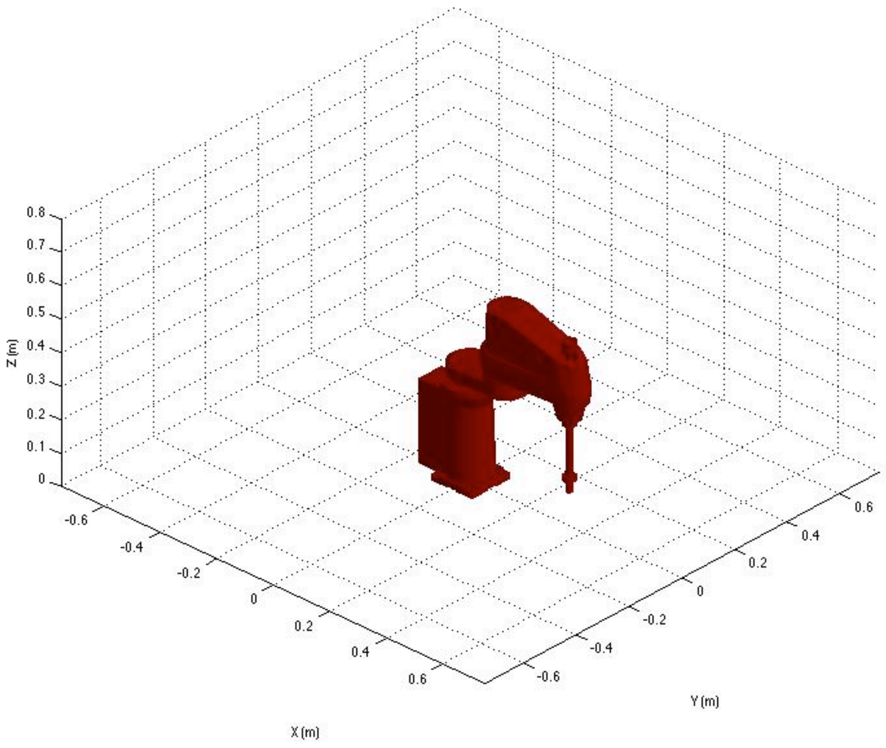


Figure 4

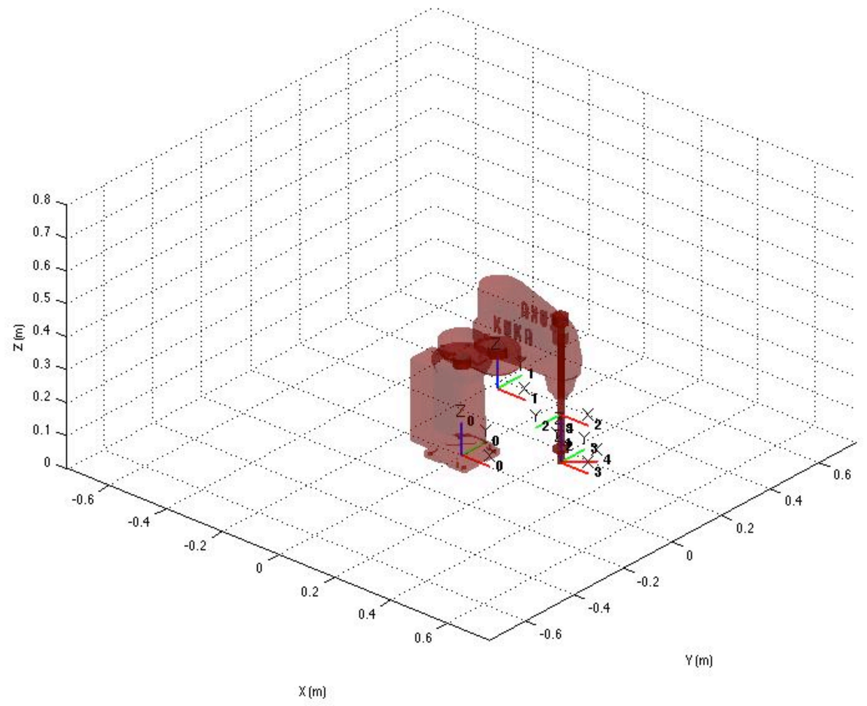


Figure 5

Now, try loading other robots. All the robots can be found at the robots directory. For example, try loading and drawing the KUKA KR90 R2700 pro, the ABB IRB 140 and the ABB IRB 6620. The KUKA robot is shown in Figure 6.

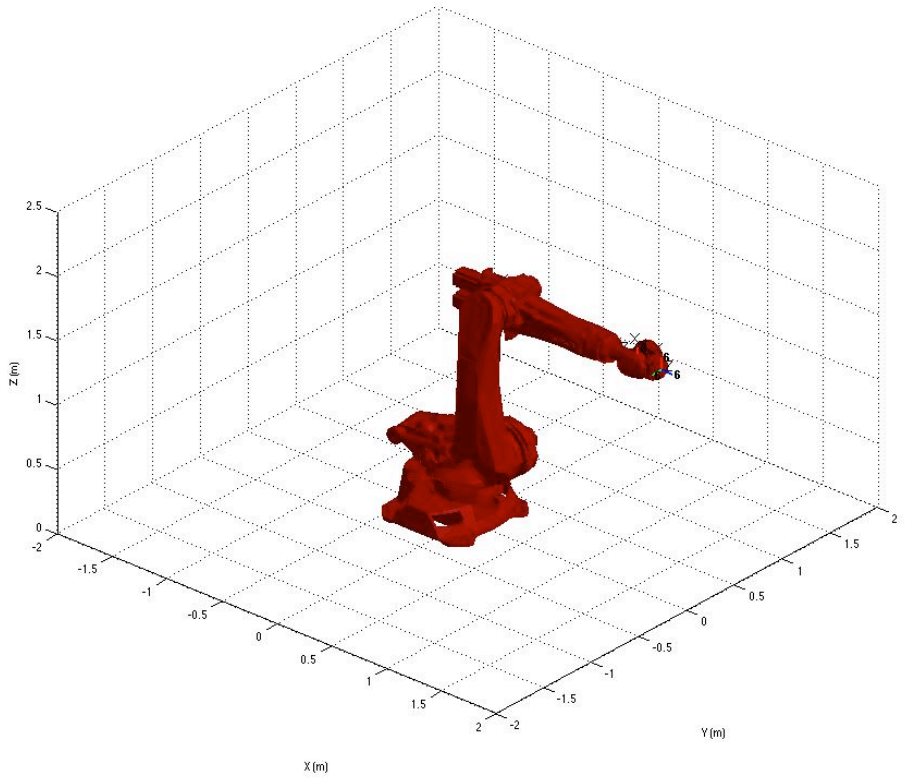


Figure 6

Next we analyze the function `directkinematic`. You can look up the help associated with this function with:

```
>> help directkinematic
DIRECTKINEMATIC         Direct Kinematic for serial robots.

        T = DIRECTKINEMATIC(robot, Q) returns the transformation matrix
   of the end effector according to the vector q of joint values.
        See also denavit.

   Author: Arturo Gil. Universidad Miguel Hernández de Elche.
   email: arturo.gil@umh.es date:   01/04/2012
```

The function receives two parameters: robot is a variable storing the parameters (kinematic, dynamic and graphical) of the robot. Q is a vector that stores the joint coordinates of the arm. As a result, the transformation matrix T is obtained. The matrix T relates the position and orientation of the last reference system in coordinates of the system 0 (X_0, Y_0, Z_0). You should have a look at the functions `directkinematic` and the function `denavit`. You can find these functions at the directory `lib/kinematics`

```
%   DIRECTKINEMATIC     Direct Kinematic for serial robots.
%
%   T = DIRECTKINEMATIC(robot, Q) returns the transformation matrix T
%   of the end effector according to the vector q of joint
%   coordinates.
%
%   See also DENAVIT.
%
%   Author: Arturo Gil. Universidad Miguel Hern·ndez de Elche.
%   email: arturo.gil@umh.es date:   01/04/2012
function T = directkinematic(robot, q)

theta = eval(robot.DH.theta);
d = eval(robot.DH.d);
a = eval(robot.DH.a);
alfa = eval(robot.DH.alpha);

n=length(theta); %number of DOFs

if robot.debug
    fprintf('\nComputing direct kinematics for the %s robot with %d
DOFs\n',robot.name, n);
end
%load the position/orientation of the robot's base
T = robot.T0;

for i=1:n,
    T=T*denavit(theta(i), d(i), a(i), alfa(i));
end
```

2 A more detailed analysis

Figure 7 shows an ABB IRB 140 robot (the units are millimeters). Generally, we should follow these steps to perform a kinematic analysis of a robotic arm.

i) Place the Dentavit-Hartenberg reference systems according to the rules.
ii) Write a D-H table.
iii) Write the D-H matrices ${}^{i-1}A_i$, i=1, ..., n, as a function of the joint coordinates.

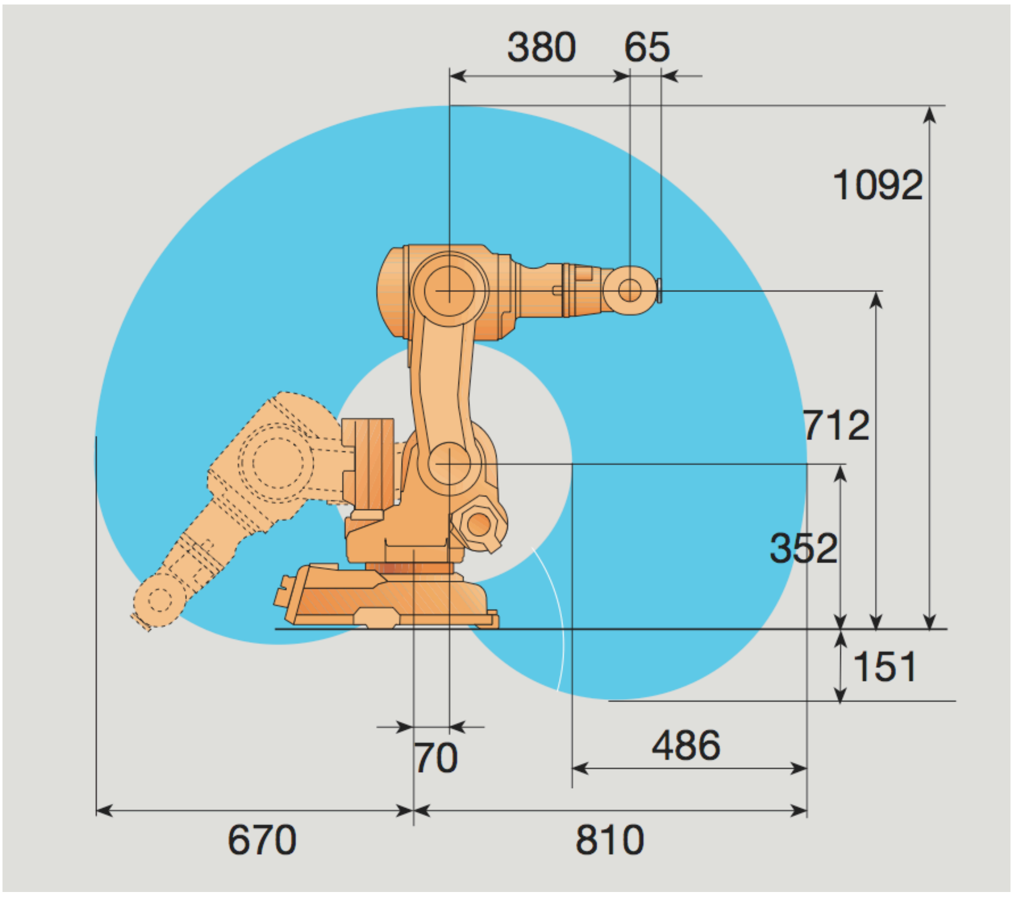


Figure 7

Exercise 1:

Place the D-H reference systems at each link of the IRB 140 robot.

Exercise 2:

Write the D-H table for the IRB 140 manipulator. Write the D-H matrices.

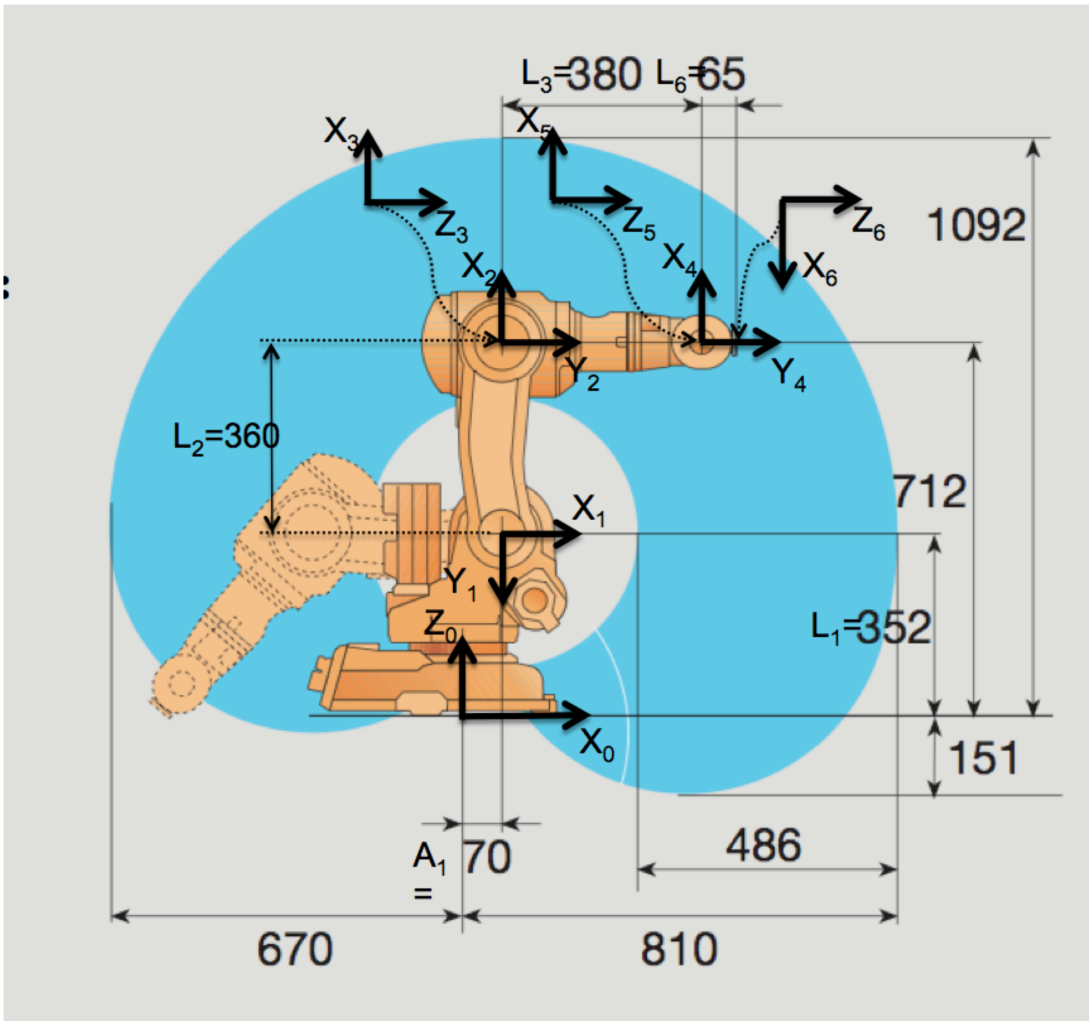


Figure 8

Figure 8 presents the location of the D-H reference systems on the IRB 140 robot. Please note that the placement of the D-H systems is not unique. The D-H table for this robot can be observed at `robots/abb/IRB140/parameters.m`

```
robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)+pi]';
robot.DH.d='[0.352 0 0 0.380 0 0.065]';
robot.DH.a='[0.070 0.360 0 0 0 0]';
robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';
```

Where q(1), q(2)… etc are the joint coordinates. Please note that the variables `robot.DH.theta`, `d`, `a` and `alpha` are text strings written as a function of the joint coordinates q(i). In order to use them we have to evaluate them at the current joint coordinates values. For example, the following code computes the D-H matrix ${}^0A_1$ using the function `denavit`:

```
q=[0 0 0 0 0 0]
theta = eval(robot.DH.theta);
d=eval(robot.DH.d);
a=eval(robot.DH.a);
alpha=(robot.DH.alpha);
A01 = denavit(theta(1), d(1), a(1), alpha(1))
```

Now, we can test the kinematic analysis of this robot. At the Matlab prompt, type:

```
>> robot=load_robot('abb', 'IRB140');
>> T=directkinematic(robot,[0 0 0 0 0 0])

T =

   -0.0000   -0.0000    1.0000    0.5150
   -0.0000    1.0000    0.0000    0.0000
   -1.0000   -0.0000   -0.0000    0.7120
         0         0         0    1.0000

>> drawrobot3d(robot, [0 0 0 0 0 0])
```

As a result, the homogeneous matrix T represents the position and orientation of the reference system 6 with respect to the base reference system when every joint position is null. After calling `drawrobot3d(robot, [0 0 0 0 0 0])` you should check that the position and orientation presented in the Matlab's prompt is correct.

We should also test that the relative transformation between the systems is correct. In order to do so, we can compute each D-H Matrix using the denavit function. Please type:

```
>> T01 = denavit(robot, [0 0 0 0 0 0], 1)

T01 =

    1.0000         0         0    0.0700
         0    0.0000    1.0000         0
         0   -1.0000    0.0000    0.3520
         0         0         0    1.0000

>> T02 = denavit(robot, [0 0 0 0 0 0], 2)

T02 =

    0.0000    1.0000         0    0.0000
   -1.0000    0.0000         0   -0.3600
         0         0    1.0000         0
         0         0         0    1.0000

>> T03 = denavit(robot, [0 0 0 0 0 0], 3)

T03 =

    1.0000         0         0         0
         0    0.0000    1.0000         0
         0   -1.0000    0.0000         0
         0         0         0    1.0000
```

Where T01 represents the position and orientation of the D-H system 1 with respect to the base system 0, T02 represents the system 2 and so on. We can, for example analyse T01:

```
T01 =

    1.0000         0         0    0.0700
         0    0.0000    1.0000         0
         0   -1.0000    0.0000    0.3520
         0         0         0    1.0000
```

The vector (0.07, 0, 0.352) is the position of the origin of the system 1 in coordinates of the system 0 (X_0, Y_0, Z_0) . The first column in T01 (1, 0, 0) represents the orientation of X_1 in coordinates of the system 0 (X_0, Y_0, Z_0), the second column in T01 (0 0 -1) represents Y_1 and, finally, the third column in T01 (0 1 0) represents Z_1. You should test that T02, T03 are also correct. Finally, we should now test that T represents adequately the position and orientation of the end effector with different coordinate values, for example:

```
>> T = directkinematic(robot, [pi/2 pi/4 pi/4 0 0 0])

T =
   0.0000    1.0000   -0.0000   -0.0000
   1.0000   -0.0000    0.0000    0.3246
        0   -0.0000   -1.0000    0.1616
        0         0         0    1.0000

>> drawrobot3d(robot, [pi/2 pi/4 pi/4 0 0 0])
```

The results are presented in Figure 9.

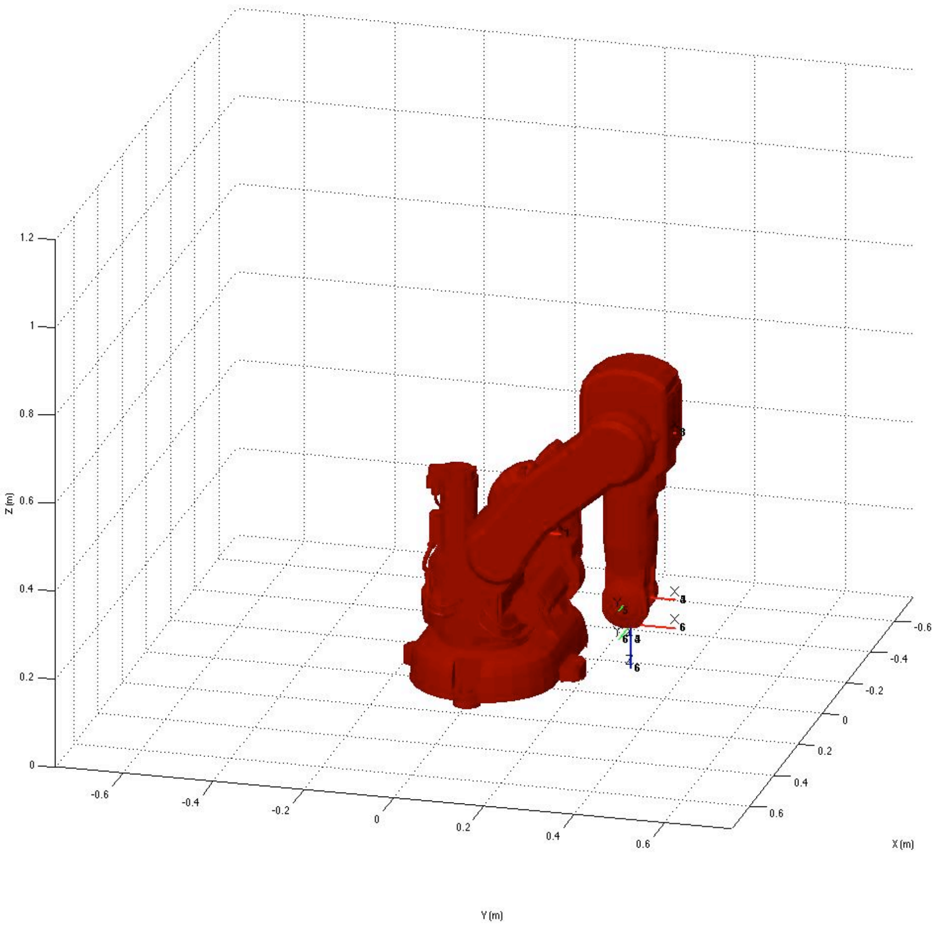


Figure 9

Exercise 3:

Try different configurations of the robot and draw it using `drawrobot3d`.

3 Using the teach pendant

You can use the `teach` function to move a robot interactively. Type:

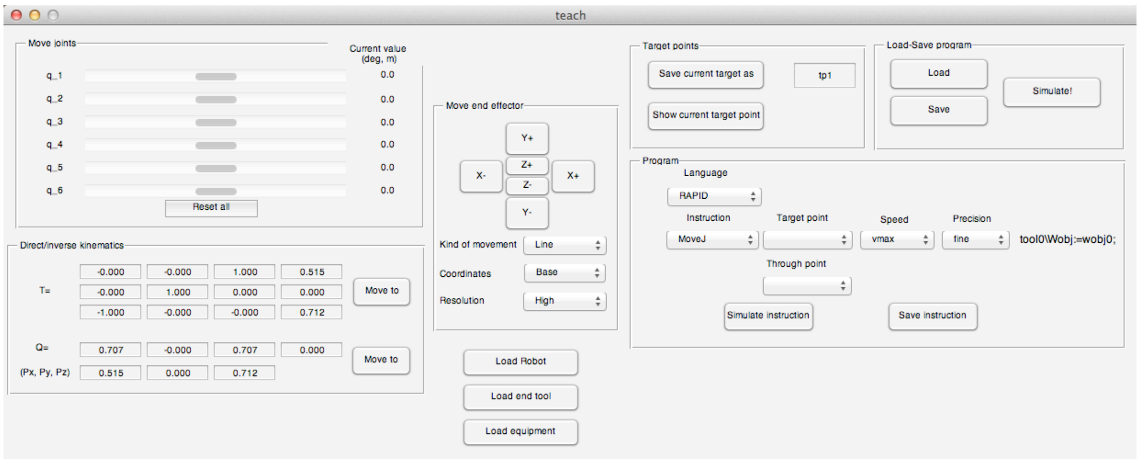
```
>> init_lib
ARTE (A Robotics Toolbox for Education) (c) Arturo Gil 2012
http://www.arvc.umh.es/arte
>> robot=load_robot('abb', 'IRB140');

ans =
/Users/arturogilaparicio/Desktop/arte_lib2.7/robots/abb/IRB140

Reading link 0
EndOfFile found...
Reading link 1
EndOfFile found...
Reading link 2
EndOfFile found...
Reading link 3
EndOfFile found...
Reading link 4
EndOfFile found...
Reading link 5
EndOfFile found...
Reading link 6
EndOfFile found...

>> teach
Select the desired view for your robot
>>
```

The graphical application appears (Figure 10):



		Current value (deg, m)
q_1		0.0
q_2		0.0
q_3		0.0
q_4		0.0
q_5		0.0
q_6		0.0

Direct/inverse kinematics

T=	-0.000	-0.000	1.000	0.515
	-0.000	1.000	0.000	0.000
	-1.000	-0.000	-0.000	0.712
Q=	0.707	-0.000	0.707	0.000
(Px, Py, Pz)	0.515	0.000	0.712	

Kind of movement: Line
Coordinates: Base
Resolution: High

Save current target as: tp1

Language: RAPID

Instruction	Target point	Speed	Precision	
MoveJ		vmax	fine	tool0\Wobj:=wobj0;

Through point:

Figure 4

This graphical application covers different topics, such as direct kinematics, inverse kinematics, quaternions and RAPID programming. For the moment we will concentrate on the direct kinematic problem. Use the slider controls to modify the joint coordinates of the robot. You can observe the position and orientation of the end effector by looking at the homgeneous matrix T and also with the orientation quaternion Q and (Px, Py, Pz).

5 Summary

Everything is summarized in the following videos:

- Initializing the library and running the demos:
http://www.youtube.com/watch?v=s8QQydJ9PwI

-Loading a robot, direct kinematics and obtaining a 3D representation of the robot: http://www.youtube.com/watch?v=Qg5HdVuTl3A

- Execute the teaching pendant application (teach)

http://www.youtube.com/watch?v=Qg5HdVuTl3A&t=29m21s

- Add a new robot. The videos show how to add a robot to the library. First, the CAD file (e.g. in STEP format) has to be imported to your CAD program. Next, each link has to be exported independently to STL format. Please note that the STL files represent the position of a series of points belonging to each file in coordinates of the robot base reference system (the units in the library are meters).

http://arvc.umh.es/arte/ videos/pr1_video4_divide_links_base.mp4

Next, the following video shows how to copy the STL files and edit the parameters.m file to create the links. After the call to the function transform_to_own, the files link0.stl, link1.stl… are referred to each of the D-H reference systems.

http://arvc.umh.es/arte/videos/pr1_video5_add_D-H_parameters_transform_test.mp4

6 Final exercises

Exercise 4:

You can now try a more advanced exercise using the robot that you are including in the library.

Asume that there exists a gaussian error in the first three joints (zero mean and standard deviation s_q=0.002 rad). This error models the lack of total precisión in the position sensors of the joints. Ask the following questions:

5.a) Asume that your robot is as the joint position q=(0, 0, 0, 0, 0, 0). Compute the error matrix in the position of the robot wrist center (X, Y, Z).

- 5.a.1) Asuming a linear error propagation.
  - $s_X=f(s_q)$
  - $s_Y=f(s_q)$
  - $s_z=f(s_q)$
- 5.a.2) Using a Monte-Carlo method.

5.b) ¿Is the error independent of the joint positions q?

5.c) ¿When is the error large/low?