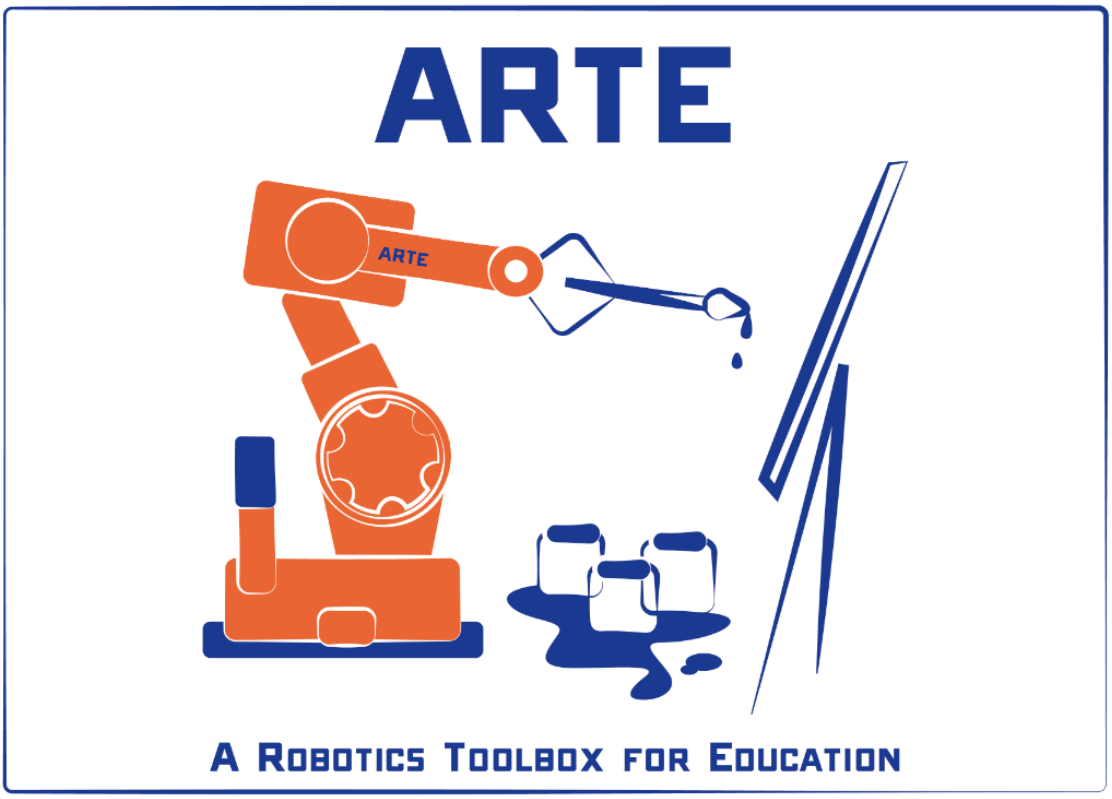


# PRACTICAL SESSION 2: INVERSE KINEMATICS

**Arturo Gil Aparicio**

**arturo.gil@umh.es**

# OBJECTIVES

After the practical session, the student should be able to:

- Solving the inverse kinematic problem of a serial manipulator using geometric methods.
- Implementing this code in Matlab for a chosen robotic arm.
- Generating joint trajectories so that the end effector follows a linear trajectory in the cartesian space.
- Analyzing singular points by means of the Jacobian matrix.

Objetivos generales: Después de esta sesión práctica, el estudiante debería ser capaz de:

- Resolver las ecuaciones de la cinemática inversa de cualquier manipulador de tipo serie hasta 6GDL y con muñeca esférica.
- Ser capaz de escribir estas ecuaciones en el entorno Matlab para un robot de su elección.
- Poder generar trayectorias articulares que hagan que el extremo siga un movimiento rectilíneo en el espacio cartesiano.
- Analizar puntos singulares en base a la matriz Jacobiana.

**Index**



## 1 First steps

We first start with a kinematic analysis of the SCARA robot arm represented in Figure 1. The table with the Denavit-Hartenberg parameters is specified next:

| | Theta | D | A | Alfa |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | $q_1$ | $l_1=0.5m$ | $l_2=0.5m$ | 0 |
| 2 | $q_2$ | 0 | $l_3=0.3m$ | π |
| 3 | 0 | $q_3$ | 0 | 0 |
| 4 | $q_4$ | 0 | 0 | 0 |

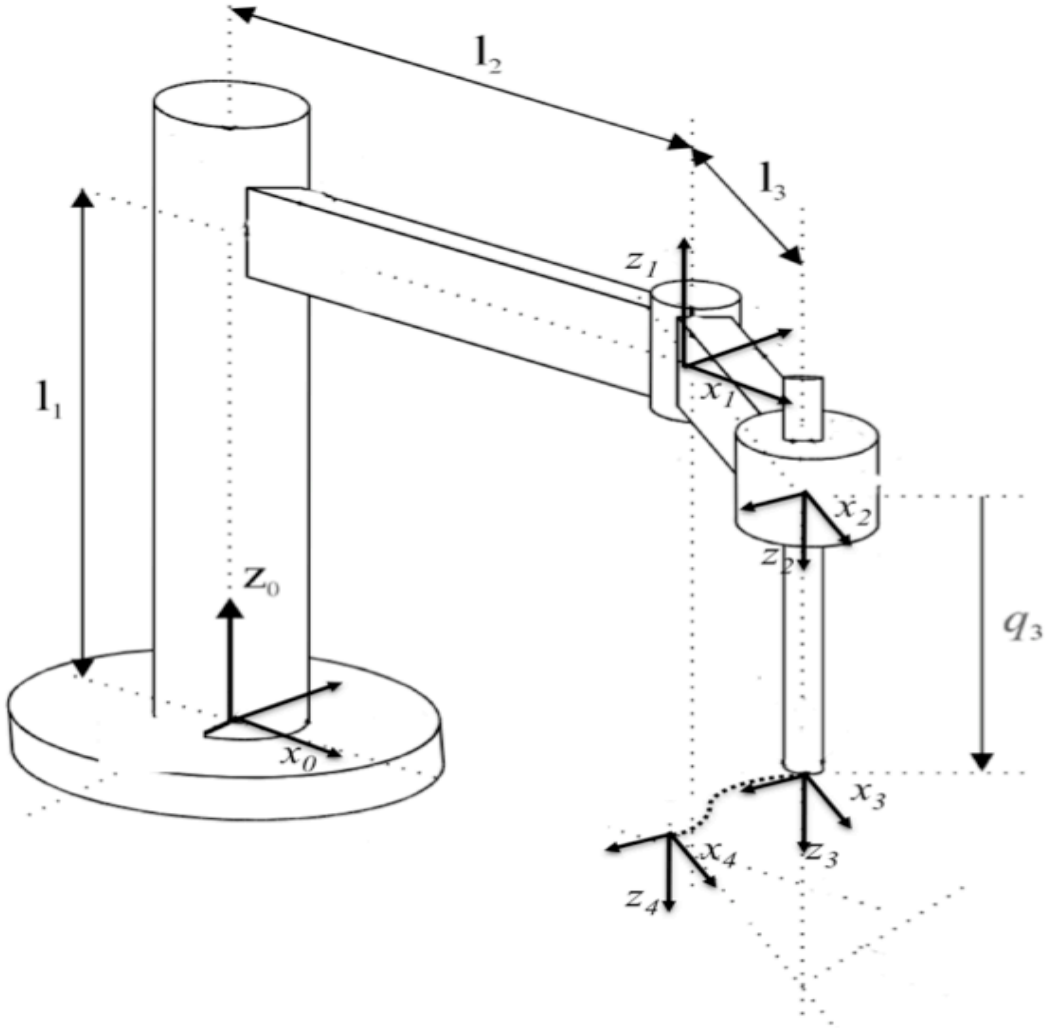


**Figure 1**

To begin with, we are going to compute the direct position and orientation of end effector as a function of the joint coordinates. You can type the following commands at the Matlab prompt:

```
>> init_lib
>> robot=load_robot('example','scara')
>> T=directkinematic(robot, [0 0 0.1 0])

T =

    1.0000         0         0    0.8000
         0   -1.0000   -0.0000   -0.0000
         0    0.0000   -1.0000    0.4000
         0         0         0    1.0000
>> drawrobot3d(robot, [0 0 0.1 pi/4])
```

Figure 2 presents the result of the `drawrobot3d` function.

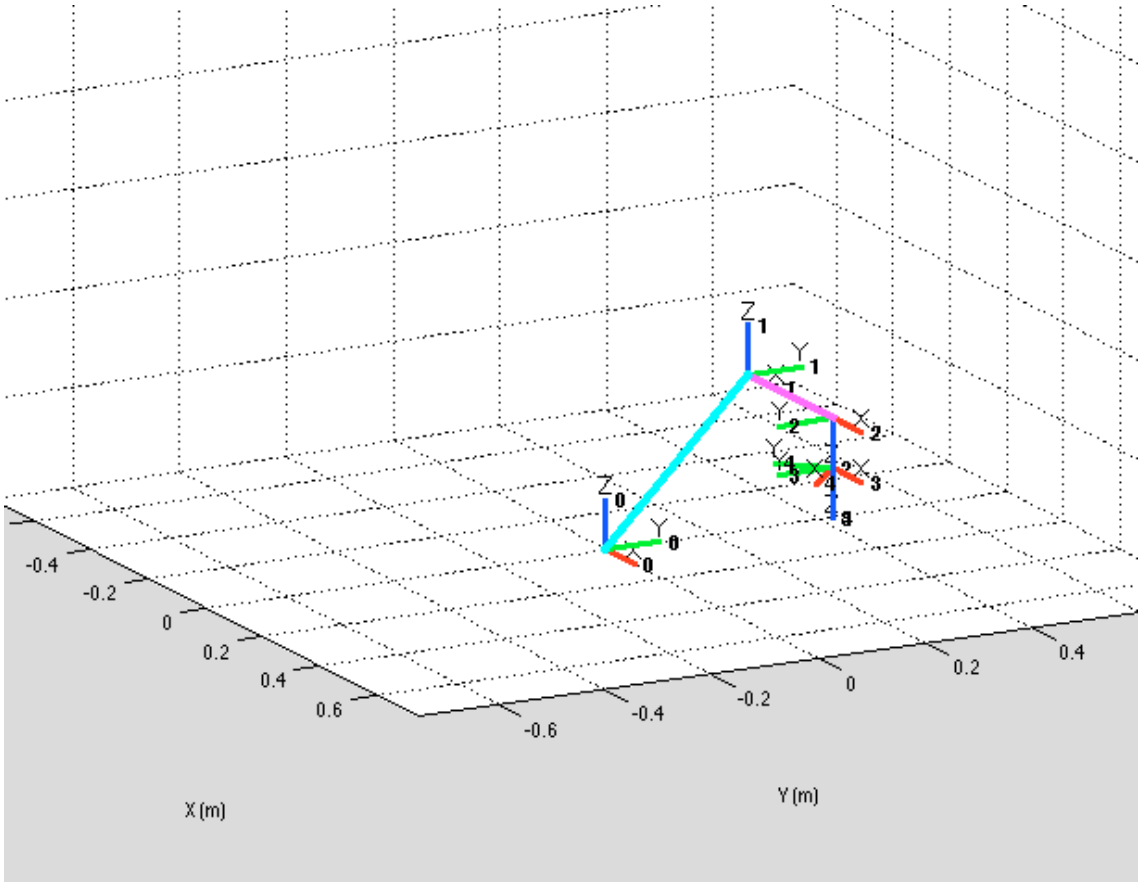


**Figure 2**

# 2 The inverse kinematic problem

Given a position and orientation of the end effector, we would like to compute the joint coordinates that bring the robot's end effector to it. In the case of the example SCARA we can achieve it with the following commands. Generally, there exist more than one solution to achieve it.

```
>> pwd

ans =
/Users/arturogilaparicio/Desktop/arte

>> init_lib
>> T=directkinematic(robot, [0 0 0 0])
>> qinv = inversekinematic(robot, T)
Computing inverse kinematics for the Scara example 4GDL arm robot
qinv =

     0     0
     0     0
     0     0
     0     0
>> T=directkinematic(robot, [pi/2 pi/4 0.1 pi/8])
>> qinv = inversekinematic(robot, T)
Computing inverse kinematics for the Scara example 4GDL arm robot
qinv =

    2.1498    1.5708
   -0.7854    0.7854
    0.1000    0.1000
   -0.5991    0.3927
```

Note that, after initializing the library, we compute the direct kinematics for the robot's initial position `q=[0 0 0 0]`. A call to the function `qinv = inversekinematic(robot, T)` takes two arguments. The first stores the robot parameters, whereas the second, T, specifies the position and orientation of the end effector. The function `qinv = inversekinematic(robot, T)` is common for all the robots in the library, however, the function calls internally a different function, named `inversekinematic_scara` that is placed at the robot's home directory `arte/robots/example/scara`. The name of the inverse kinematic function is specified in the parameters.m file, in the line `robot.inversekinematic_fn = 'inversekinematic_scara(robot, T)'`.

```
function robot = parameters()

%Path where everything is stored for this robot
robot.path = 'robots/example/scara';

%Kinematic parameters
robot.DH.theta= '[q(1) q(2) 0 q(4)]';
robot.DH.d='[0.5  0      q(3)  0]';
robot.DH.a='[0.5    0.3  0    0]';
robot.DH.alpha= '[ 0 pi  0 0]';

%Jacobian matrix. Variation of (X, Y, Z) as a function of (w1, w2, w3)
robot.J='[-a(2)*sin(q(1)+q(2))-a(1)*sin(q(1)) -a(2)*sin(q(1)+q(2)) 0;
a(2)*cos(q(1)+q(2))+a(1)*cos(q(1)) a(2)*cos(q(1)+q(2)) 0; 0 0 -1];';

robot.name='Scara example 4GDL arm';

robot.inversekinematic_fn = 'inversekinematic_scara(robot, T)';
```

It is important to note that the inversekinematic function returns two posible solutions for `qinv`, only one of them matches the initial q specified in the direct

kinematic function. However, both solutions for qinv match the original matrix T. You can test these ideas with:

```
>> T=directkinematic(robot, [pi/2 pi/4 0.1 pi/8])

T =
   -0.3827    0.9239    0.0000   -0.2121
    0.9239    0.3827   -0.0000    0.7121
   -0.0000    0.0000   -1.0000    0.4000
         0         0         0    1.0000
>>qinv = inversekinematic(robot, T)

>>T=directkinematic(robot, qinv(:,1))

T =
   -0.3827    0.9239    0.0000   -0.2121
    0.9239    0.3827   -0.0000    0.7121
   -0.0000    0.0000   -1.0000    0.4000
         0         0         0    1.0000

>> T=directkinematic(robot, qinv(:,2))

T =
   -0.3827    0.9239    0.0000   -0.2121
    0.9239    0.3827    0.0000    0.7121
    0.0000    0.0000   -1.0000    0.4000
         0         0         0    1.0000
```

You can observe this by calling `drawrobot3d` for both solutions to the inverse kinematics.

```
>> drawrobot3d(robot, qinv(:,1))
>> drawrobot3d(robot, qinv(:,2))
```

# 3 Solving the inverse kinematic problem for a 6 DOF robot

Next, Figure 3 presents the ABB IRB 140 robot. Next, its corresponding D-H table is presented.

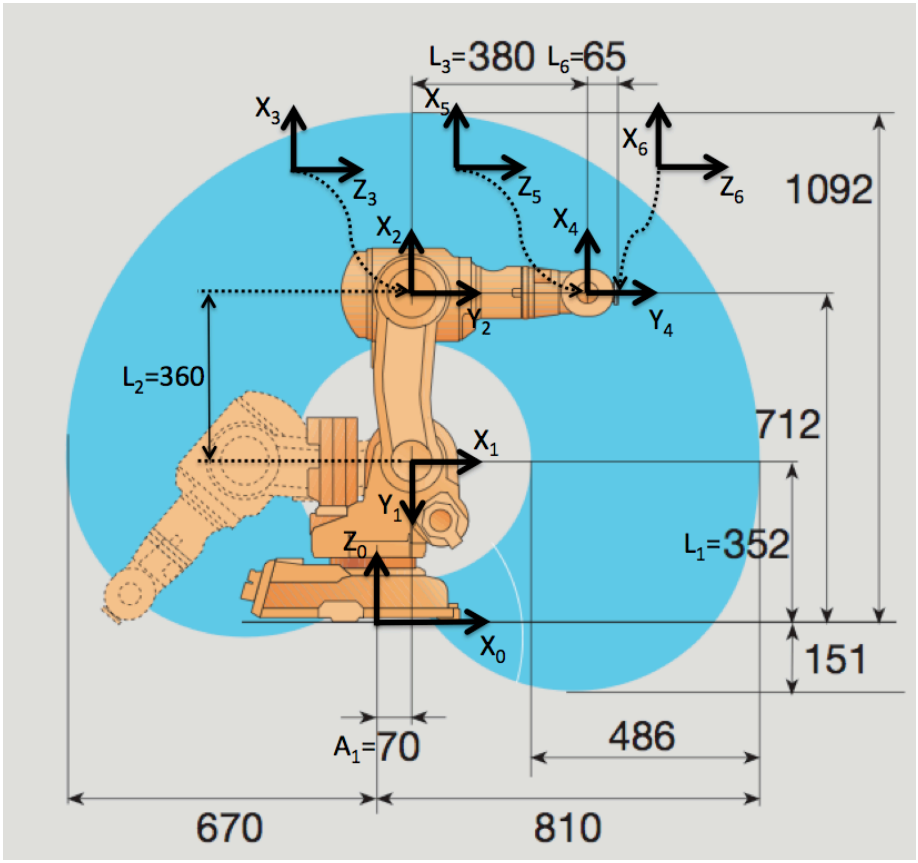


**Figure 3**

| θ | d | a | α |
|---|---|---|---|
| $q_1$ | 0.352 | 0.070 | - π/2 |
| $q_2$ - π/2 | 0 | 0.360 | 0 |
| $q_3$ | 0 | 0 | - π/2 |
| $q_4$ | 0.380 | 0 | π/2 |
| $q_5$ | 0 | 0 | - π/2 |
| $q_6$ | 0.065 | 0 | 0 |

You can observe these parameters in the `parameters.m` file, which resides in `arte/robots/ABB/IRB140`

```
robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)]';
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]';
robot.J=[];

robot.inversekinematic_fn = 'inversekinematic_irb140(robot, T)';
```

The inverse kinematic problem is solved by the function `inversekinematic_irb140(robot, T)` that is placed at `arte/robots/ABB/IRB140`. You can find some more information about solving this inverse kinematic problem in the ARTE reference manual.

## Exercise 1:

You should analyse the inverse kinematic problem for the IRB 140 robot. In this case, in general, there exist 8 different solutions to achieve a position/orientation of the end effector in space. You should analyse that any of the eight different solutions q returned by the function allows to achieve the same position/orientation in space (the same homogeneous matrix T).

a) Obtain T as a function of the forward kinematic equations for, for an arbitrary q = [pi/4 pi/8...].

b) Obtain n solutions to the inverse kinematic equations using the encoded functions for the IRB 140 robot.

c) Check that any of of the previous solutions allows to attain the same position/orientation defined by the initial T.

d) What other concepts/features should we check ino order to state that any solution $q_i$ is feasible, for i=1, 2, ...8?

--

Analiza el problema cinemático para el robot IRB140. Para este robot, en general, existen 8 soluciones diferentes de valores articulares que permiten llevar el extremo del robot a una determinada posición/orientación. Para conseguir esto deberías.

a) Obtener una matriz T con la cinemática directa y q = [pi/4 pi/8...] a su elección.

b) Obtener las soluciones qinv de la cinemática inversa.

c) Comprobar que todas las anteriores soluciones permiten llegar a la misma matriz T.

d) ¿Qué más deberíamos comprobar para que una solución $q_i$ sea válida, para i=1...8?

La función `test_kinematics_demo` allows to test automatically the forward and inverse kinematic functions for each robot.

## Exercise 2:

a) Test the `test_kinematics_demo`. In order to do so, load any of the 6DOF robot that exist in the library and run the script.

b) Next, edit `test_kinematics_demo`.m and read the code inside. Discuss the code and functions used with your team mates/teacher.

--

a) Utiliza el script `test_kinematics_demo`. APara ello, deberás cargar un robot de 6GDL de la librería y lanzar el script. Observa el resultado por pantalla.

b) A continuación, edita el script y lee el código. Deberás ser capaz de entender el código y relacionarlo con los conceptos de cinemática directa e inversa.

# 4 Direct and inverse jacobian

The functions that are associated to the Jacobian matrix allow to:

- Compute the speed of the end effector given the joint speeds.
- Compute the joint speeds, given a desired speed of the end effector in coordinates of the base reference frame.

Please try the functions `compute_end_velocity` and `compute_joint_velocity` and observe the results.

```
>> robot=load_robot --> load the 3DOF PLANAR EXAMPLE robot.
>> q = [0 0 0]
>> qd = [1 1 1]
>> V = compute_end_velocity(robot, q, qd)
V =
     0
     6
     0
     0
     0
     3
```

As a result, at the given position q of the robot, and assuming that , instantaneously it is moving at the given joint speeds of 1 rad/s, we get a linear and angular speed given by V.

Be aware that this is part of the concept of forward/direct kinematics. Given the joint speeds defined in qd (stands for q 'dot'), compute the end effector's speed in terms of linear and angular speed.

## Exercise 3:

Analyze the values of the vector V. Indicate which parts of V belong to the linear speed and which to the angular speed. Can you test other positions/speeds. Discuss your results with the teacher/student.

--

Analiza el resultado V del caso anterior. Indica qué partes de V definen la velocidad lineal y cuáles definen la velocidad angular. Prueba, por favor, otras posiciones q y velocidades articulares qd. Comenta los resultados con el profesor o tu compañero.

## Exercise 4:

In the case of the SCARA example arm, the expresion of the Jacobian matrix is provided as a function of the joint variables.

a) Compute the end effector's speed when the joint speeds are $\omega_q$ = (0.1 rad/s, 0.1 rad/s, 1 m/s)=( $\omega_{q1,}$ $\omega_{q2,}$ $\omega_{q3}$). Note that the third joint is translational and that the joint speed $\omega q_4$ does not contribute to the end effector's speed.

b) Compute the joint speeds necessary to obtain an end effector's speed of V=(1, 1, 1) m/s when the end effector is at P=(0.5, 0.5, 0) m. Please note that there are two possible solutions.

--

Para el caso del robot SCARA de ejemplo, la matriz jacobiana se encuentra codificada directamente (en velocidad lineal).

a) Calcula la velocidad del extremo cuando las velocidades articulares son $\omega_q$ = (0.1 rad/s, 0.1 rad/s, 1 m/s)=( $\omega_{q1,}$ $\omega_{q2,}$ $\omega_{q3}$). Nota que la tercera articulación es translacional y se mide en m/s. Además, la cuarta articulación $\omega q_4$ produce una rotación del extremo y no contribuye a la velocidad lineal.

b) Compute the joint speeds necessary to obtain an end effector's speed of V=(1, 1, 1) m/s when the end effector is at P=(0.5, 0.5, 0) m. Please note that there are two possible solutions.

# 5 Follow a line in space

Following a line in space is a typical task for an industrial robotic manipulator. Humans do find easy to move from a target point to the next one by finding a line that connects both positions/orientations. This line may not be optimal in

terms of energy consumption or time but one may find lots of applications where precisely following a line in space is required. Typical applications are:

- Line soldering. Soldering, for example, two pieces of steel along a line that connect both using a TIG welding tool.

- Point soldering. Or use point-to-point welding using an electrical tool. These soldering points are typical in the automotive industry.

## Exercise 5:

Implement a function:

**function q=followline_myrobot(robot, p1, p2, R, options)**

To follow a line in cartesian space. The function should interpolate a number of points along the line that connects the points p1 and p2. Orientation must be kept constant as defined by the rotation matrix R. Next, for each of these points, you should find a solution to the inverse kinematic problem. The parameter options should be used to choose one of the posible solutions. The orientation of the end effector should be constant and equal to R along the whole movement. Finally, you should:
a) Animate the trajectory of the robotm using the function drawrobot3d.
b) Represent the joint trajectories.
c) Plot the joint speeds, asuming that the movement between p1 and p2 is accomplished in 1 second. Consider that the final motion must be continuous, thus, no sudden change between different solutions is admitted. Check the Euclidian distance of the current solution for point pi with the next solution for the point p_i+1.

d) Enumerate the possible errors that may appear during this movement:
- Points outside the workspace of the robot.
- Singular points where det(J)=0 that indicate the need of infinite joint speeds.
- Joints out of range.

---

Implementa una función como la siguiente:

**function q=followline_myrobot(robot, p1, p2, R, options)**

La función debería ser capaz de hacer que el robot siga una línea recta en el espacio cartesiano. La función debe interpolar un conjunto de puntos pertenecientes a una recta que conecta los puntos p1 y p2. La orientación debe ser constante y definida por la matriz de rotación R. Tenga Ud. en cuenta que, para cada uno de esos puntos definidos deberá encontrar una solución de la cinemática inversa. El parámetro options puede usarse para seleccionar una de las posibles soluciones iniciales de la cinemática inversa. Finalmente, deberá:
a) Animar la trayectoria usando la función drawrobot3d.
b) Representar las trayectorias articulares a lo largo del movimiento.

c) Plotear las velocidades articulares asumiendo que el movimiento entre los puntos p1 and p2 se hace a velocidad constante (cartesiana) y termina en 1 s.
d) Enumera los posibles errores que pueden aparecer durante la consecución de la tarea como, por ejemplo:
- Puntos fuera del espacio de trabajo del robot.
- Puntos singulares donde det(J)=0. En estos puntos singulares pueden aparecer velocidades infinitas para intentar conseguir velocidades finitas del extremo.
- Valores articulares fuera de rango.

# 6 Use the teach pendant

Execute the graphical teach pendant (Figure 4). The teach pendant GUI application allows to move the robot 's end effector in the following ways:

- With an independent movement of each robot axis. Using the sliders at the top left área.
- Moving the end effector along along X, Y and Z in coordinates of:
    - The base reference system.
    - The end effector.
- Change the values of the T matrix and solve the inverse kinematic problem. The application calls internally the inversekinematic function. The application shows the joint configuration closest to the current one. Alternatively you can change the position and orientation in the Q controls.

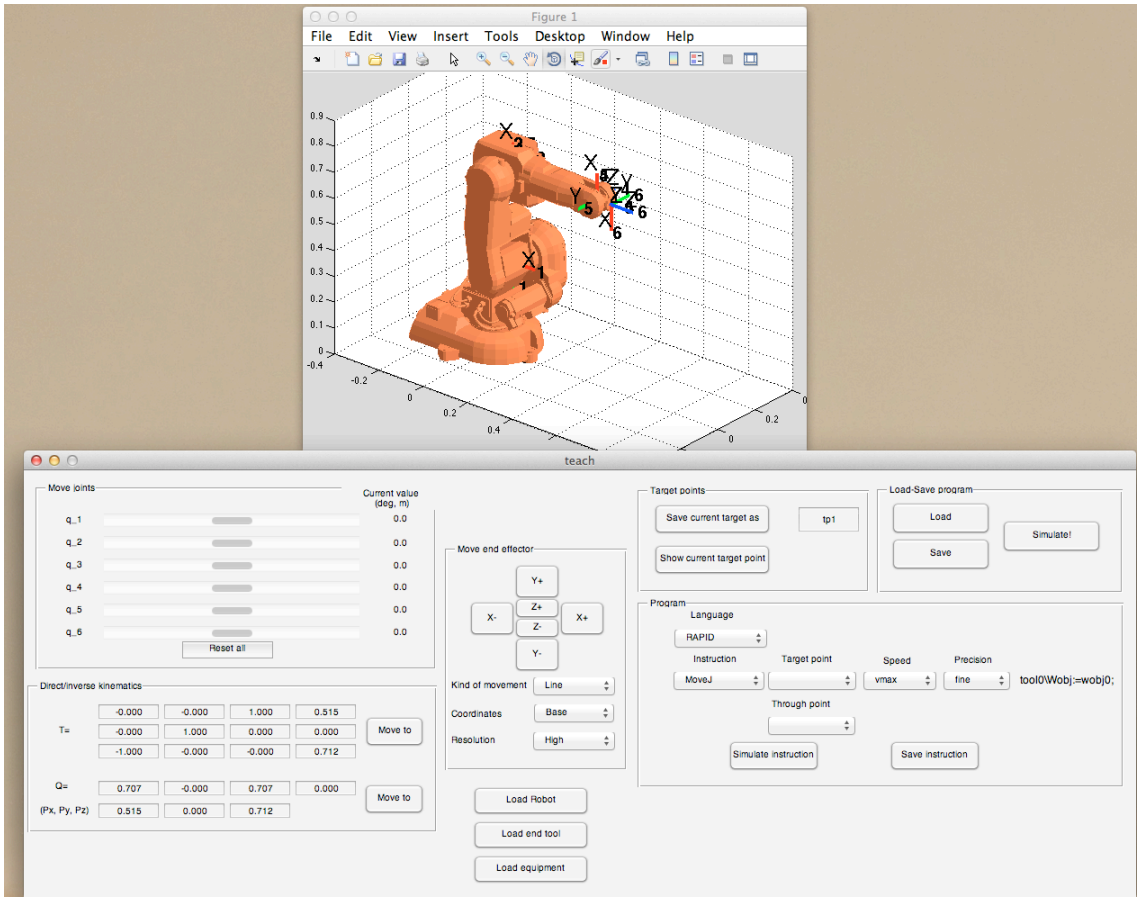


**Figure 4**

# 5 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?

# 6 Summary

Everything is summarized in the following videos:

- Inverse kinematic of a 6 DOF robot.

http://www.youtube.com/watch?v=CS2eGRSEGOo (spanish)

- Execute the teaching pendant application (teach)

http://www.youtube.com/watch?v=h1xnJw3-a7M#t=18m23s (spanish)