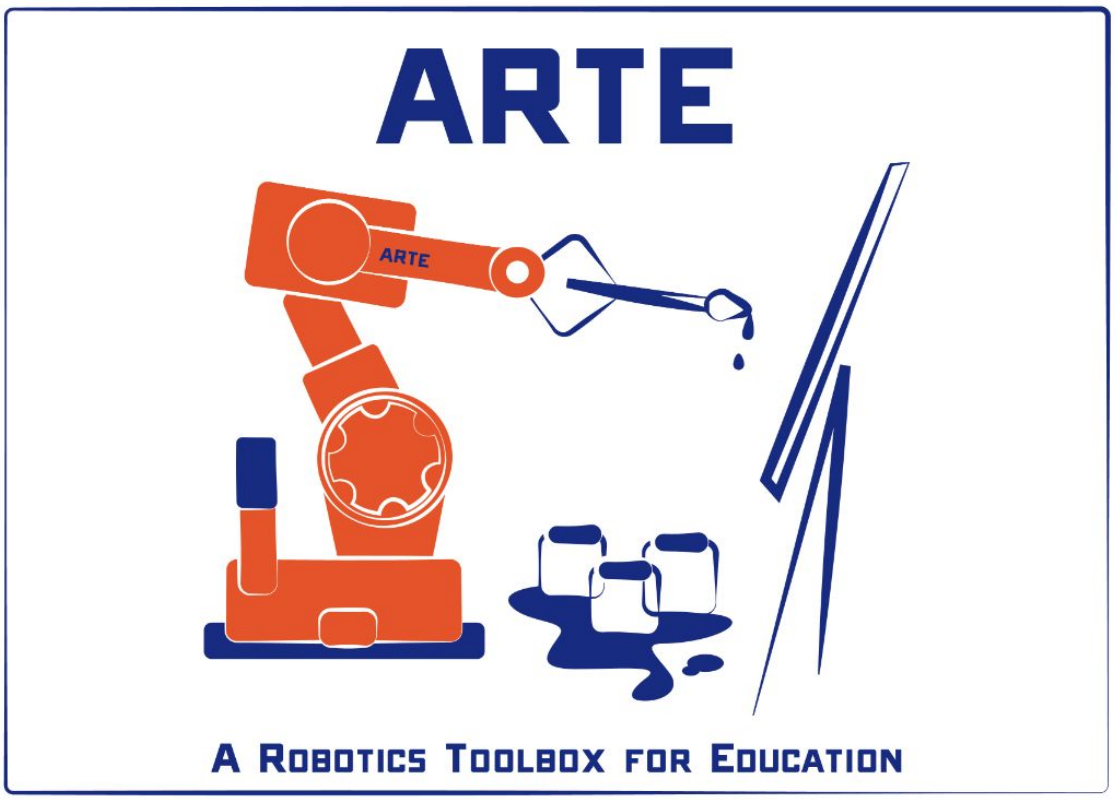
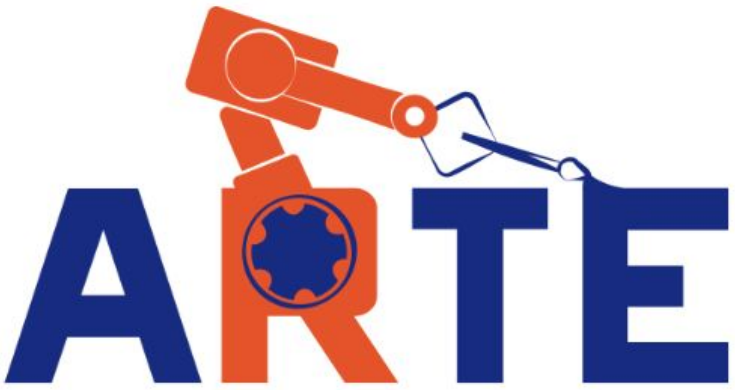


SESIÓN PRÁCTICA 1: CINEMÁTICA DIRECTA

Arturo Gil Aparicio

arturo.gil@umh.es



OBJETIVOS

El objetivo principal de esta práctica es ayudar al estudiante en la compresión de los siguientes conceptos:

- **Matrices de transformación homogénea: representación conjunta de la translación y orientación.**
- **Parámetros Denavit-Hartenberg de un manipulador robótico. Solución de la cinemática directa de un robot manipulador.**

Índice

1 Primeros pasos

Comienza por inicializar la librería.

```
>> pwd
ans =
/Users/arturogilaparicio/Desktop/arte
>> 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/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...
```

Si no lo has hecho antes, visualiza la siguiente introducción a la librería ARTE.

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

Seguidamente analizaremos y entenderemos la colocación de los sistemas de coordenadas D-H. Para hacer esto emplearemos las siguientes funciones de la librería:

- **`init_lib`**: inicializa la librería. Inicializa variables importantes para que la librería funcione correctamente.
- **`load_robot`:** Carga un robot en una variable.
- **`directkinematic(robot, q)`:** Calcula la cinemática directa para las coordenadas q del robot.
- **`drawrobot3d(robot, q)`:** Realiza una representación 3D del robot.

You can type the following commands at the Matlab prompt:

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

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

>> ans =

/Users/arturogilaparicio/Desktop/arte/robots/example/3dofplanar


[…]
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])
```

El primer comando **`init_lib`** inicializa el lugar del path donde se encuentra instalada la librería. Inicializa también otras variables de conrfiguración. Seguidamente, será necesario cargar un o varios brazos robóticos usando el comando **`load_robot`**. La variable `robot` guarda en este caso los parámetros asociados con el robot deseado. En nuestro caso, después de llamar a `robot=load_robot('example','3dofplanar')`, vemos que la librería lee los parámetros guardado en el directorio: `robots/example/3dofplanar/parameters.m`. De forma alternativa, podemos llamar a la función load_robot como:

```
>> robot = load_robot
```

En este caso será necesario navegar por la estructura de directorios de tu PC y cargar cualquier robot existente en la librería. Finalmente, con la variable robot cargada con un robot de 3GDL, podremos llamar a la función:

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

El resultado de la función de cinemática directa es la matrix `T` que describe la posición y orientación del sistema de referencia 4 con respecto al sistema de referencia 0 (colocado en la base del robot). El vector de coordenadas articulares `[pi/4 pi/4 pi/4]` define el valor de las articulaciones. En este caso, las tres articulaciones son rotacionales. Finalmente, la función `drawrobot3d(robot, [pi/4 pi/4 pi/4])` realiza una representación 3D del robot junto con sus sistemas de D-H, según se muestra en la figura 3.

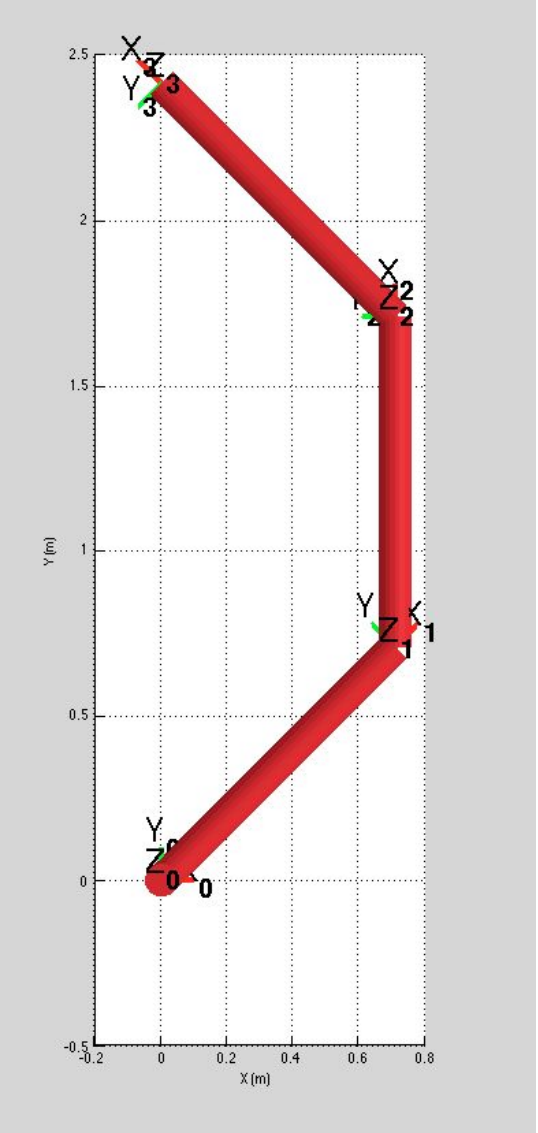


Figura 3

Una llamada general a la función `load_robot` es:

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

Por ejemplo:

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

Carga los parámetros del robot KR5 scara R350 Z200 fabricado por KUKA (KUKA roboter GmbH). La mayoría de robots incluidos en la librería poseen ficheros gráficos que nos permiten tener una visión más agradable y completa de ellos. Los siguientes comandos permiten variar la vista del robot:

```
>> 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]);
```

La propiedad `robot.graphical.draw_axes` controla la vista de los sistemas de D-H sobre el robot. Así, `robot.graphical.draw_axes=1;` dibuja los ejes del robot y `robot.graphical.draw_axes=1;` los esconde. Por otra parte, la propiedad `robot.graphical.draw_transparent` controla el factor de transparencia del robot. Deberías, ahora, probar con diferentes combinaciones de estos parámetros y observar los resultados. Se presentan resultados posibles a continuación (Figura 4 y Figura 5).

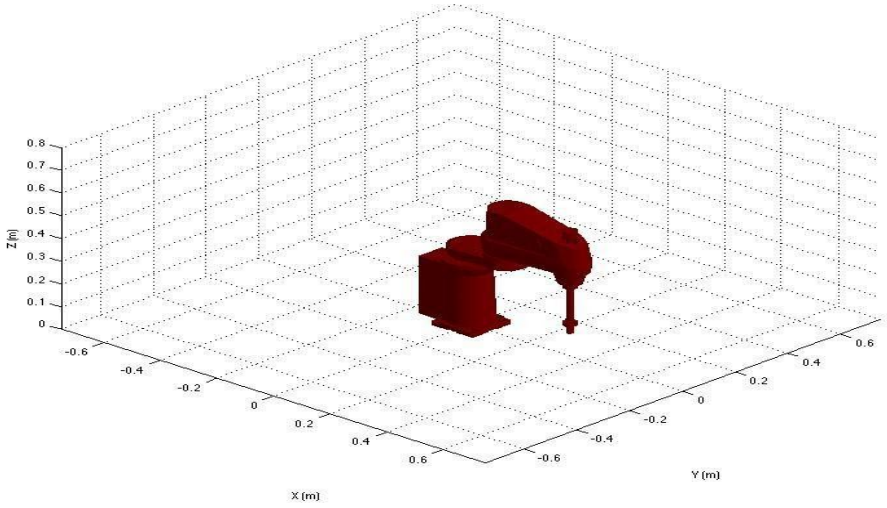


Figura 4

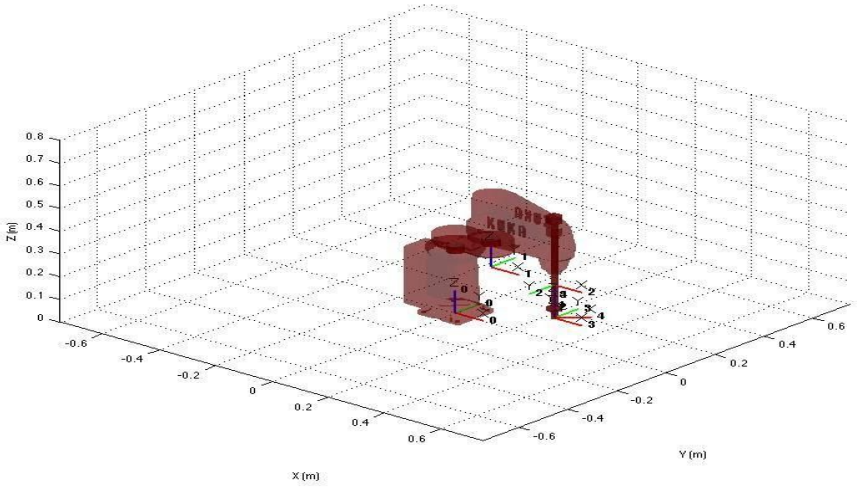


Figura 5

Ahora, intenta cargar otros robots, por ejemplo, el KUKA KR90 R2700 pro, el ABB IRB 140 y el ABB IRB 6620 (Figura 6).

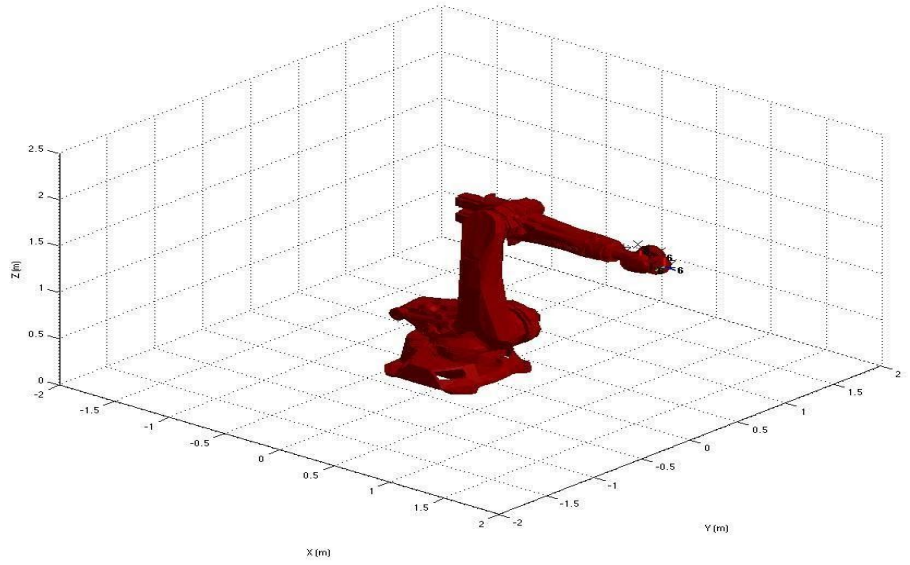


Figura 6

A continuación, analizaremos la función `directkinematic`. Puedes buscar la ayuda asociada con esta función si escribes:

```
>> 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
```

La función recibe dos parámetros: `robot` (que almacena los parámetros dinámicos, cinemáticos y gráficos del robot). La variable Q es un vector que almacena las coordenadas articulares del brazo. Como resultado, la matriz de transformación T se obtiene. Esta matriz T nos relaciona la posición/orientación del último sistema de referencia sobre el extremo del robot con el sistema 0 0 (X_0, Y_0, Z_0). Ahora deberías ver el código de la función `directkinematic` y de la función `denavit` ambas en el directorio `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 Hernandez 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*dh(theta(i), d(i), a(i), alfa(i));
end
```

2 Un análisis más detallado

En la Figura 7 se presenta el robot ABB IRB 140 (las unidades están en milímetros). El análisis de la cinemática directa de un robot sigue, generalmente, estos pasos.

i) Emplaza los sistemas de Denavit-Hartenberg de acuerdo con las reglas existentes.
ii) Escribe la tabla de parámetros D-H.
iii) Escribe las matrices ${}^{i-1}A_i$, i=1, ..., n, como función de las coordenadas articulares.

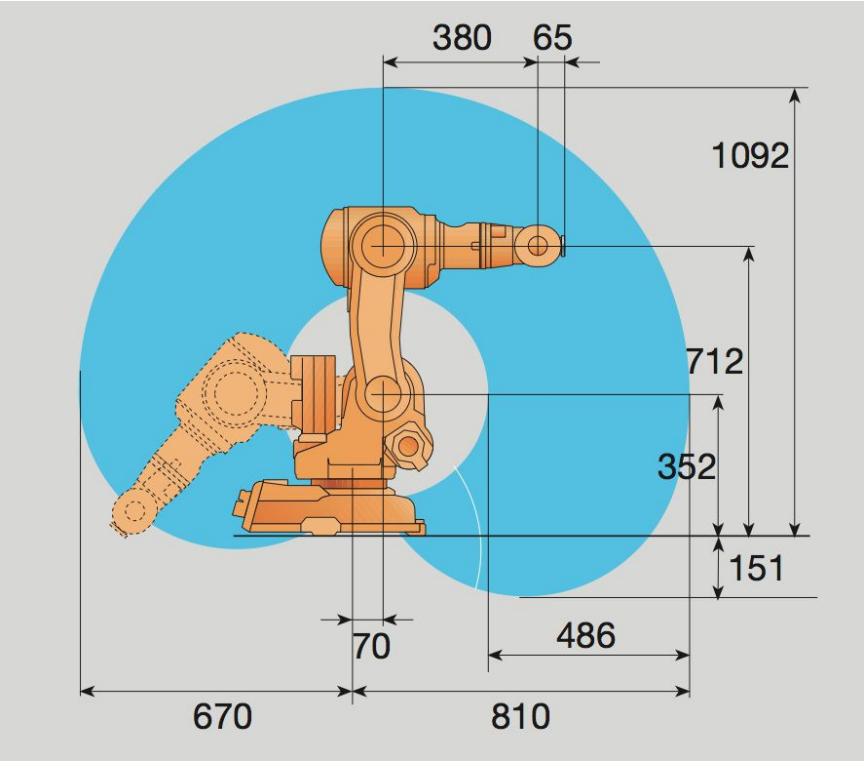


Figura 7

Ejercicio 1:

Coloca los sistemas de referencia en cada uno de los eslabones del robot IRB 140 (Figura 7).

Exercise 2:

Escribe las matrices D-H para el robot IRB 140. Calcula también las matrices D-H.

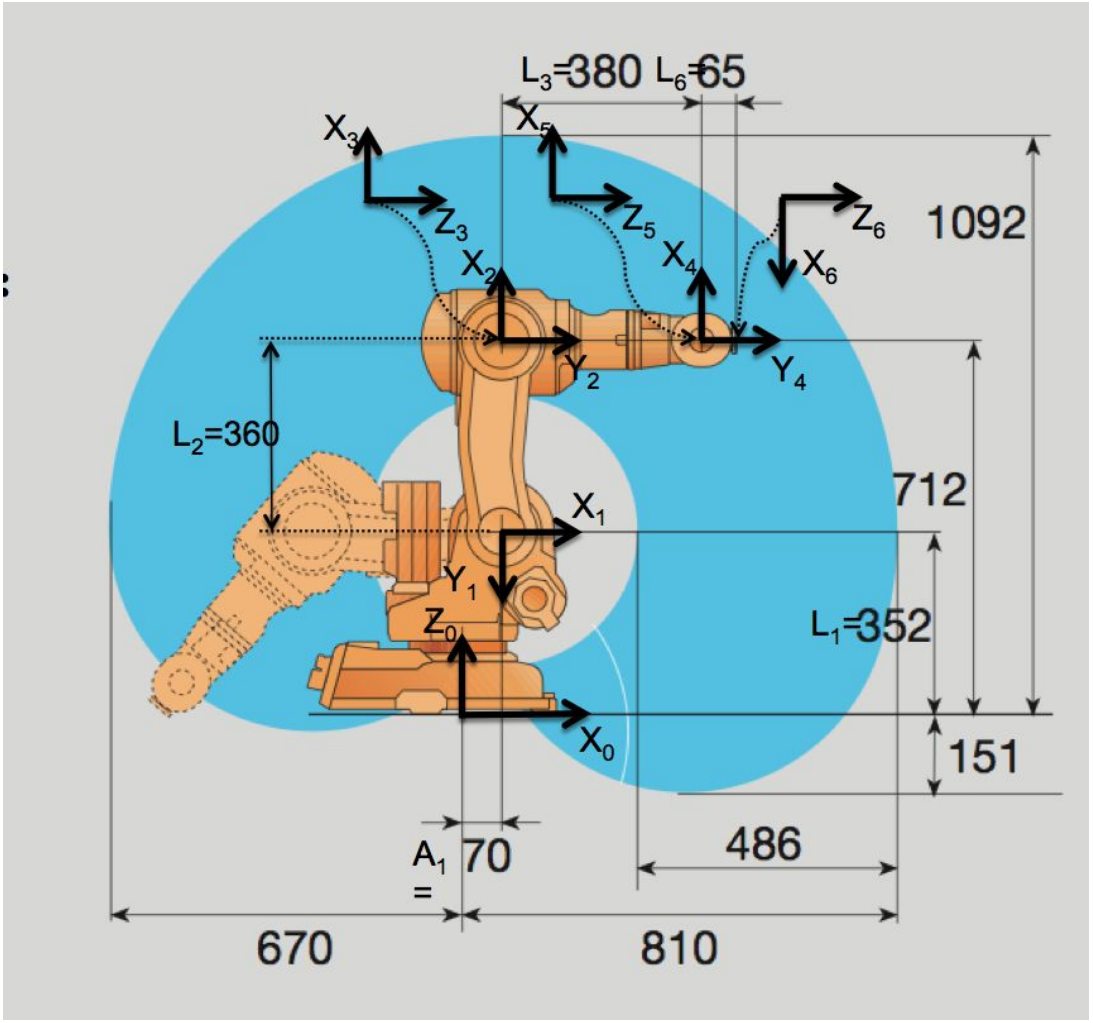


Figura 8

La Figura 8 presenta la colocación de los sistemas de D-H sobre el robot IRB 14. **Este debería ser el resultado que se te ha pedido en los ejercicios 1 y 2.** Ten en cuenta, no obstante, que este resultado es orientativo, ya que la colocación de los sistemas D-H no es única. La tabla de estos parámetros se puede ver en `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]';
```

Donde q(1), q(2)… etc son las coordenadas articulares. Ten en cuenta que las variables `robot.DH.theta`, `d`, `a` y `alpha` son cadenas de texto y, por tanto, se trata de funciones escritas en base a las variables articulares `q(i)`. Para poder utilizar estas variables es necesario evaluarlas con el valor de las coordenadas articulares. Por ejemplo, el código siguiente calcula la matriz ${}^0A_1$ para unos valores de las coordenadas articulares:

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

Finalmente, podemos comprobar el análisis cinemático directo del robot. Para ello, escribe:

```
>> 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])
```

Como resultado la matriz T representa la posición y orientación del sistema de referencia 6 con respecto al sistema de referencia de la base cuando todas las coordenadas son nulas. Después de llamar a `drawrobot3d(robot, [0 0 0 0 0 0])` deberías comprobar que la posición y orientación de T tiene sentido con lo que ves en la figura de Matlab.

Además, sabemos que la cinemática directa se basa en transformaciones sucesivas de manera que:

T = T01*T12*T23…*T56

se te propone ahora que veas que las transformaciones intermedias también son correctas. Para poder hacer esto, usaremos la función dh que permite calcular cada matriz D-H intermedia de la siguiente manera:

```
>> T01 = dh(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

>> T12 = dh(robot, [0 0 0 0 0 0], 2)

T12 =

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

>> T23 = dh(robot, [0 0 0 0 0 0], 3)

T23 =

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

Considere Ud. que la llamada `dh(robot, [0 0 0 0 0 0], i)` es una forma abreviada de la función dh en la que se especifica el índice de la transformación intermedia i. En el extracto anterior, T01 representa la transformación entre el sistema 0 y el 1, T12 representa al sistema 2 respecto del 1. Entrando en detalle, vemos que T01 tiene este valor:

```
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
```

En esta matriz, el vector (0.07, 0, 0.352) es la posición del sistema 1 en coordenadas del sistema 0 (X_0, Y_0, Z_0). La primera columna en T01 (1, 0, 0) representa la orientación de X_1 en coordenadas del sistema 0 (X_0, Y_0, Z_0), la segunda columna en T01 (0 0 -1) representa el vector Y_1 y, finalmente, la tercera columna en T01 (0 1 0) representa Z_1. En este momento deberías comprobar también que T12, T23 son correctas y entiendes claramente su significado. Finalmente, también será necesario comprobar el valor de T y entender cómo representa la posición y orientación del extremo del robot:

```
>> 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])
```

Además, el lector debería comprobar que:

T = T01*T12*T23*T34*T45*T56

Se presenta en la figura 9 la posición y orientación del robot.

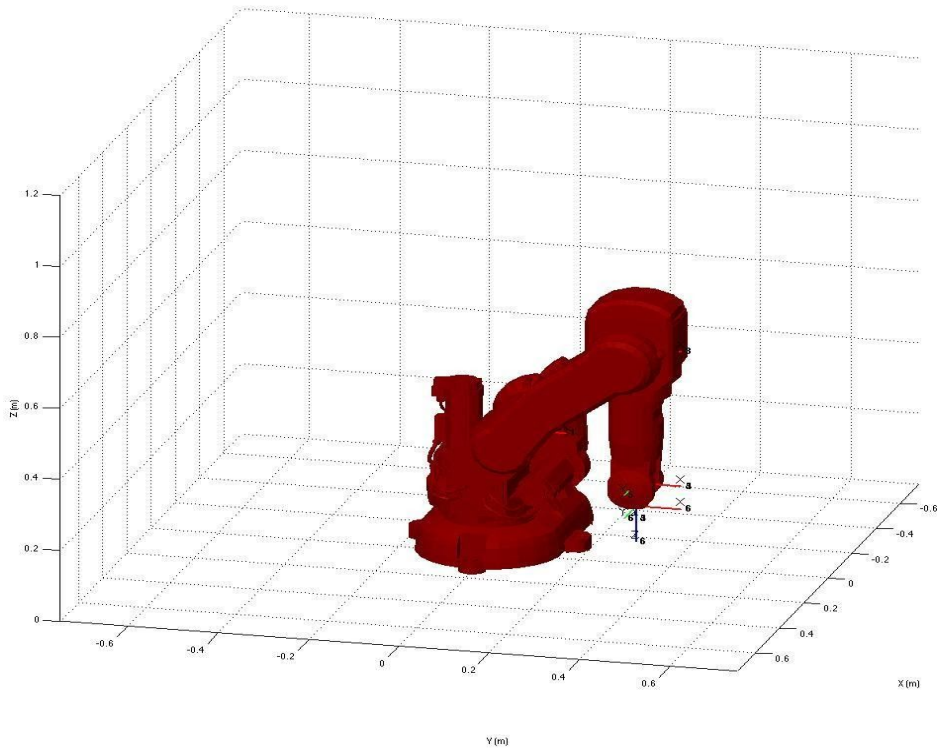


Figura 9

Ejercicio 3:

Mueve el robot a configuraciones diferentes y obtén una representación 3D usando la función `drawrobot3d`.

Ejercicio 4:

a) Carga 3 robots de 3 fabricantes diferentes (p. e. ABB, KUKA y ADEPT).

b) Calcula la matriz T para cada uno de ellos cuando q = [0 0 ... 0].

c) Comprueba el resultado visualizando el sistema de referencia del extremo mediante la función `drawrobot3d`.

3 Matriz Jacobiana

La velocidad lineal y angular del extremo se relacionan a través de la siguiente ecuación

$$\begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix} = J \cdot \dot{q}$$

Donde J es la Jacobiana del manipulador. Dada una posición articular q arbitraria, podemos calcular J usando la función `manipulator_jacobian`

```
>> q = [0 0 0 0 0 0]
>> J = manipulator_jacobian(robot, q)
```

y, a continuación, hallar su velocidad lineal/angular en el extremo

```
>> qd = [1 1 1 1 1 1]
>> vw = J*qd
```

Es importante que visualices en este momento las dimensiones de J: 6xn, donde n es el número de GDL del robot.

Ejercicio 5:

a) Carga un robot plano de 3GDL. Calcula J. Calcula vw para la pose q = [pi/4 pi/4 pi/4] y qd = [1 1 1]. ¿Qué puedes decir de J?

b) Carga el robot IRB140 de 6GDL. Calcula J. Calcula vw para la pose q = pi/4*[1 1 1 1 1 1] y qd = [1 1 1 1 1 1]. Repite lo mismo para q=[0 0 0 0 0 0] y qd = [1 1 1 1 1 1]. Compara ambas matrices J. ¿Qué conclusiones puedes extraer?

Ejercicio 6:

a) Carga el robot ABB IRB140. Ubícalo en la posición articular: q = pi/8*[1 1 1 1 1 1]. Evalúa la velocidad lineal y angular del extremo para la velocidad articular qd = pi/10*[1 1 1 1 1 1] (rad/s). Observe y comprenda el resultado.

b) Carga el robot plano de 3DOF. Ubícalo en la posición q = pi/4*[1 1 1]. Halle la velocidad lineal y angular del extremo cuando qd = [1 1 1] rad/s. Trate de entender el resultado.

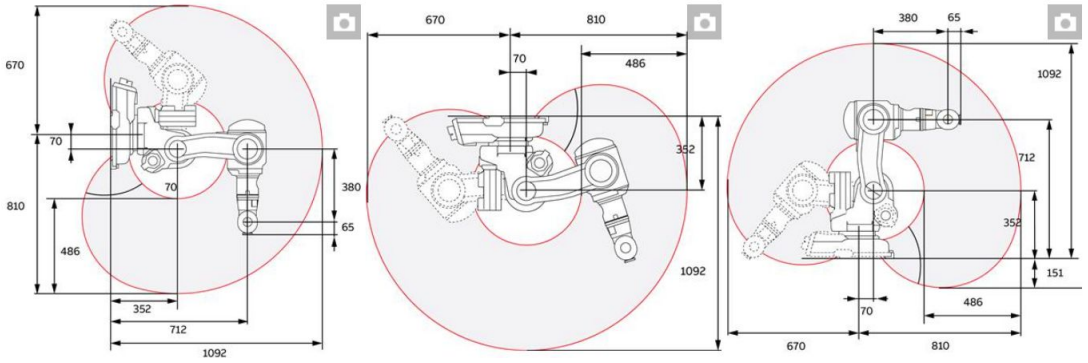
4 Ejercicios finales

Si has prestado atención hasta ahora, deberías ser capaz de resolver las siguientes cuestiones que se plantean a continuación.

4.1 Upside down

En ocasiones el robot industrial se monta de formas diferentes en su puesto de trabajo en su celda (denominadas, generalmente, *robot on the wall, robot on the floor, robot on the ceiling*). Modifica el fichero `parameters.m` del robot IRB140 para conseguir los siguientes modos de funcionamiento:

- sobre el suelo.
- en la pared.
- colgado del techo.
- a 45 grados sobre la horizontal..



4.2 Cinemática inversa

Carga el robot IRB140 y colócalo boca abajo a 2 metros de distancia sobre el suelo. A continuación, intenta averiguar las coordenadas articulares q que sitúan su extremo en (necesitarás hacer bastantes pruebas):

```
T =
   0   1   0   0.0
   1   0   0   0.3
   0   0  -1   0.16
   0   0   0   1.0000
```

4.3 Espacio de trabajo

Se presenta, a continuación, el espacio de trabajo del IRB140. Existen diferentes formas de definir el espacio de trabajo:

- a) como aquellos puntos de la muñeca del robot a los que el robot es capaz de alcanzar. Se supone que, si la muñeca del robot puede alcanzar ese punto, el robot podrá alcanzar todas las orientaciones posibles desde ese punto.
- b) Como aquellos puntos de alcance máximo del robot.

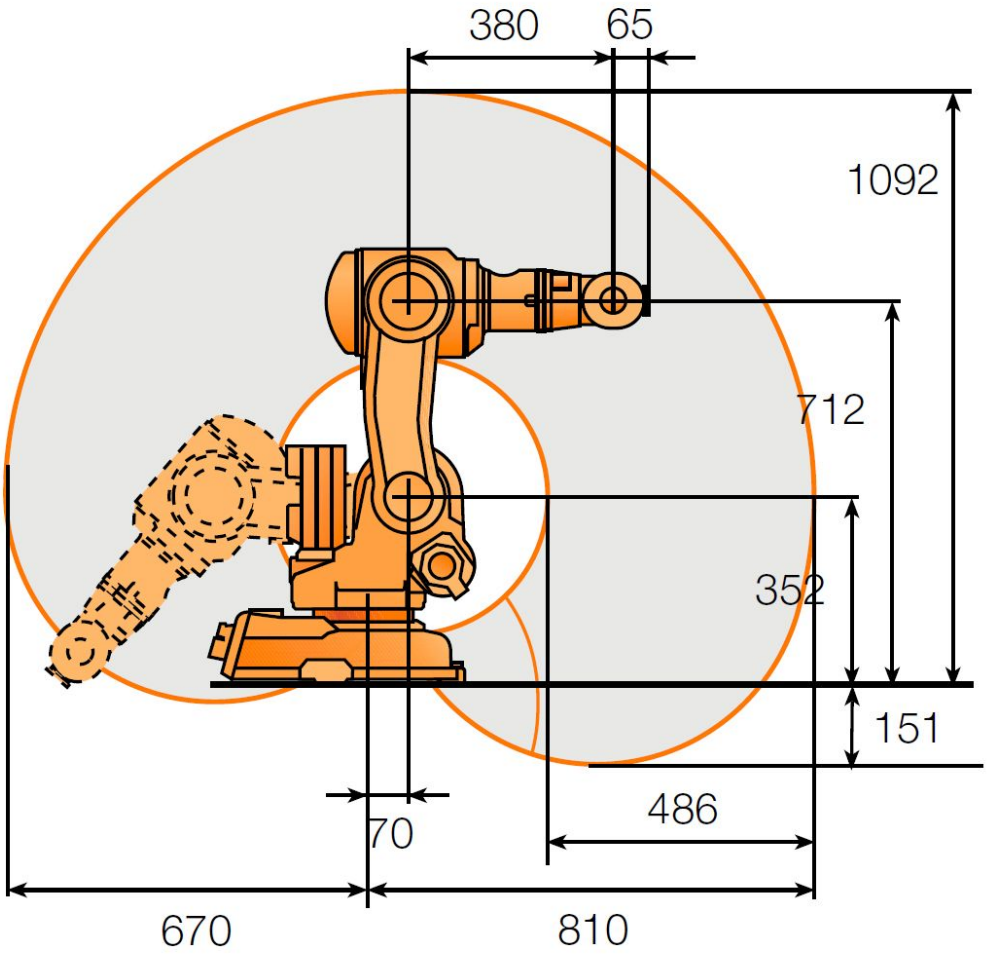


Figura 10

Las representaciones del espacio de trabajo, como las de la figura 10, no siempre resultan aclaradoras para el alumno. Por esta razón, se os plantea como ejercicio que obtengáis un espacio de trabajo 3D basado en puntos, como se muestra en la figura 11. Este espacio se obtiene calculando la posición de la muñeca para un número elevado y aleatorio de vectores articulares q. Un compañero vuestra ha escrito el código que tenéis a continuación. En este código, se calcula la posición y orientación del robot M veces (bucle for). Cada vez se calcula un valor aleatorio de las coordenadas articulares q (usando la función rand).

```
%base position of the robot
q = [0 0 0 0 0 0];
% Number of joint posisitions to be simulated
M = 10000;
robot=load_robot('ABB', 'IRB140');
adjust_view(robot)
drawrobot3d(robot, q)

%Now compute direct kinematics for this position q
pp = [];
for i=1:M
      q = [2*pi*rand-pi 2*pi*rand-pi 2*pi*rand-pi 2*pi*rand-pi 2*pi*rand-pi
2*pi*rand-pi];
    T = directkinematic(robot, q);
    %drawrobot3d(robot, q)
    %pause(0.5)
    p = T(1:3,4);
    pp = [pp p];
end
plot3(pp(1,:),pp(2,:),pp(3,:),'r.')
```

a) Ejecuta el código anterior.
b) Descomenta las líneas en negrita para poder visualizar las posiciones que alcanza el robot. **¿Qué puedes decir de estas posiciones?**
c) Comenta las líneas en negrita para poder visualizar todos los puntos del espacio de trabajo del robot. **¿Qué puedes decir de estos puntos?**
d) Modifica el código anterior para poder obtener un espacio de trabajo con mayor sentido y más fácil de interpretar. El resultado debería ser similar al siguiente: