

# Proyecto Final Robótica 1

## Robot IRB140 Soldadura por arco de metal y gas

### **Profesora**

Dra. Ing. Díaz Carolina

### **Jefe de Trabajos Prácticos**

Sánchez, Eric

### **Alumnos:**

Alfaro Ojeda, William.      Legajo 10128

García Girón Maximiliano      Legajo 9885

## INDICE

---

1. Resumen .....	4
2. Introducción.....	4
3. Etapas de desarrollo del proyecto .....	5
<b>3.1. Especificaciones técnicas .....</b>	<b>5</b>
<b>3.1.1 Robot IRB 140.....</b>	<b>5</b>
<b>3.1.2 Antorcha soldadora de la serie PKI.....</b>	<b>10</b>
<b>3.2. Librerías .....</b>	<b>17</b>
<b>3.3. Cinemática.....</b>	<b>18</b>
<b>3.4. Cinemática directa.....</b>	<b>19</b>
<b>3.5. Cinemática inversa .....</b>	<b>24</b>
4. Planificación de trayectorias.....	28
<b>4.1. Espacio de trabajo .....</b>	<b>29</b>
<b>4.2. Singularidades.....</b>	<b>31</b>
<b>4.3. Limites articulares .....</b>	<b>33</b>
5. Conclusiones.....	36
Bibliografía.....	37
Anexos.....	38
<b>A. Código para cinemática directa.....</b>	<b>38</b>
A.1. Cinemática Directa .....	38
<b>B. Matriz de Denavit Hartenberg .....</b>	<b>38</b>
<b>C. Código para cinemática inversa .....</b>	<b>39</b>
C.1. Cinemática Inversa.....	39
C.2. Theta2 .....	40
C.3. Theta3 .....	40
C.4. Pieper .....	41
C.5. Norma .....	41
<b>D. Código para planificación de trayectorias.....</b>	<b>42</b>

D.1. Trayectoria.....	42
D.2. AnimarRobot .....	44
D.3. Space.....	44
<b>E. PROYECTO_ROBOTICA1 .....</b>	<b>45</b>

## 1. Resumen

En el presente proyecto se procederá a evaluar el robot industrial IRB 140 para la tarea de soldadura por arco. Se analizará la cinemática, tanto directa como indirecta; el Jacobiano; su espacio de trabajo; y la planificación de trayectorias. Luego, se procederá a realizar su simulación para una tarea específica. El software utilizado para obtener los resultados será Matlab, donde trabajará especialmente con las librerías Robotics Toolbox de Peter Corke y ARTE. Además, se hablará de los tipos de soldadores que se pueden utilizar, y de cómo influyen al momento de realizar el control de movimiento para un determinado trabajo.

## 2. Introducción

Entre los robots considerados de más utilidad en la actualidad se encuentran los **robots industriales o manipuladores**, los cuales que permiten, entre otras cosas, el uso de nuevos mecanismos de producción y automatización.

Dado que existen ciertas dificultades a la hora de establecer una definición formal de lo que es un robot industrial, por algunas razones que pueden ser, por ejemplo, las diferencias conceptuales entre el mercado oriental y el occidental; o la evolución de la robótica en los últimos años, que obliga a nuevas actualizaciones, solo presentaremos la que ha sido adoptada por la Organización Internacional de Estándares (ISO), que lo define como:

***Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.***

En esta definición se debe entender que la reprogramabilidad y la multifunción se consiguen sin realizar modificaciones físicas.

Los manipuladores se basan en la unión de una estructura mecánica y otra de electrónica para el sistema de control, y genera una serie de movimientos que como tales se especifican y diseñan para una función establecida o conocida. Los recientes progresos en los campos de la inteligencia artificial, del aprendizaje y reconocimiento de formas permiten obtener información de su entorno y adaptar su comportamiento a las modificaciones del mismo, sin necesidad de intervención de un operario.

En forma general, un robot industrial está conformado por:

- **Articulaciones** o uniones entre los eslabones del brazo robótico, dependiendo del tipo de movimiento que realizan las articulaciones se pueden nombrar de tipo rotacional o prismática. Las unidades de las variables que manejan estas articulaciones son: radianes o ángulo para la articulación rotacional y metros para la prismática.
- **Actuadores** son los que suministran las señales para que en las articulaciones se produzca movimiento. Los actuadores usados más comunes son servomotores, elementos neumáticos, eléctricos o hidráulicos.

- **Sensores** que proporcionan información del estado de las variables del brazo robótico, como lo son la posición y velocidad articular además de las fuerzas y los torques aplicados a los eslabones, entre otros.

Este tipo de robot es el centro de estudio dentro del presente trabajo, donde se ha utilizado el modelo IRB 140 de 6 ejes perteneciente a la corporación multinacional ABB, el cual fue diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robots. En cuanto a la herramienta utilizada, se ha optado por una antorcha soldadora de la serie PKI, también perteneciente a ABB, para evitar cualquier problema de incompatibilidad al momento de montarlo.

Para comenzar, se mostrarán los detalles técnicos más relevantes para nuestro trabajo, tanto del robot como de los distintos tipos de soldador de la serie PKI, y se hablará de cómo las elecciones de estos pueden afectar el control del movimiento.

Luego, se procederá a desarrollar la cinemática directa, en donde se aplicará el método de Denavit Hartenberg; y la inversa, en donde se utiliza el método algebraico. Se comentarán los problemas obtenidos al usar las funciones de la librería Arte al momento de realizar el algoritmo para el cálculo cinemático, y de cómo fueron solucionados. También, se hablará del criterio utilizado para elegir la mejor solución entre las ocho que son obtenidas.

Además, se mostrará el espacio de trabajo total y las singularidades que se presentan. Teniendo en cuenta esto, y lo mencionado anteriormente se procede a desarrollar el algoritmo para realizar la planificación de trayectorias. Para evaluarlo, se comienza realizando una línea recta y luego se prueba con una curva. En cada caso, se observa que siga con la orientación deseada.

Finalmente, se comentan los resultados obtenidos al realizar el presente proyecto. Se hablará tanto de los problemas encontrados como de las soluciones implementadas.

### 3. Etapas de desarrollo del proyecto

#### 3.1. Especificaciones técnicas

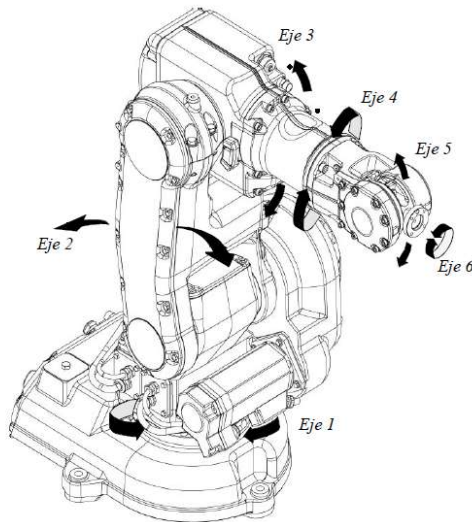
A continuación, se mostrarán las especificaciones tanto del robot como del soldador. Más adelante se darán informaciones de los sensores y actuadores con los que se trabaja y de sus limitaciones.

##### 3.1.1 Robot IRB 140

Tiene una estructura abierta especialmente adaptada para un uso flexible y presenta grandes posibilidades de comunicación con sistemas externos. Está disponible en distintas variantes, y todas ellas pueden montarse sobre el suelo, en posición invertida o en una pared con cualquier ángulo.

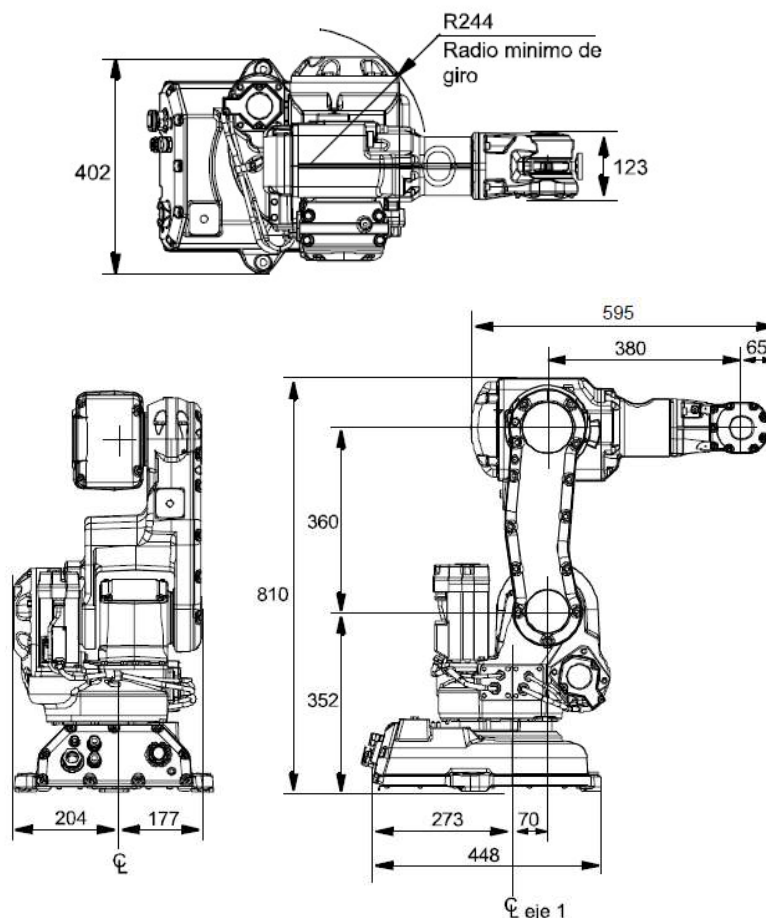
**Ejes del manipulador:** Son cada uno de los movimientos independientes que una articulación permite efectuar y le confiere un grado de libertad a la estructura. El número total está dado por la suma de los grados de libertad asociados a cada uno de estos ejes.

El IRB 140 cuenta con todos sus ejes de rotación. La siguiente figura muestra una vista isométrica.



Al momento de asignarle una herramienta, en el extremo final del último eslabón, se deben tener en cuenta los GDL que este pudiera tener, y se deben sumar al robot.

**Dimensiones:** estas medidas están dadas en mm y son de suma importancia al momento de armar la matriz de Denavit Hartenberg para obtener la cinemática. A continuación, se observa la vistas superior, posterior, lateral.



## **Carga soportada**

En vista de que una de las aplicaciones del IRB 140 es la soldadura por arco, y el fabricante ofrece los diversos tipos y tamaños de soldadores que puede utilizar, no se entrarán en mayores detalles. Nos limitaremos a utilizar alguno de serie PKI que están permitidos para este robot, y ya no tendremos que preocuparnos por el tema de los esfuerzos.

## **Seguridad**

Cuando se va a poner en marcha un determinado trabajo, es preciso especificar las medidas de seguridad para disminuir riesgos y gravedad en caso de accidentes. Algunas estrategias que normalmente son utilizadas para lograrlo son:

- Determinación de los límites del sistema: intención de uso, espacio y tiempos de trabajo, entre otros.
- Identificación y descripción de todos aquellos peligros que pueda generar la máquina durante las fases de trabajo. Se deben incluir los riesgos derivados de un trabajo conjunto entre la máquina y el ordenador y los riesgos derivados de un mal uso de la máquina.
- Definición del riesgo de que se produzca el accidente. Se definirá probabilísticamente en función del daño físico que pueda producir.
- Comprobar que las medidas de seguridad son adecuadas.

Dado que el presente manipulador es muy utilizado a nivel industrial, en todas sus etapas de diseño ya se han considerado muchos de los posibles inconvenientes que pudiera tener y se le ha dado un diseño final tal que pueda ofrecer una seguridad total en las distintas etapas de operación.

Las características que se tuvieron en cuenta para lograrlo son:

- El modo de control puede ser manual o automático.
- Sistema de seguridad dedicada, que se basa en un circuito de doble canal controlado continuamente. Si cualquiera de los componentes falla, se interrumpe la alimentación eléctrica de los motores y se aplican los frenos.
- El uso de un paro retardado proporciona un paro más suave. El robot se detiene de la misma forma que con un paro de programa, sin desviarse de la trayectoria programada. Después se corta la alimentación de los motores.
- Paro de emergencia en el controlador y en la unidad de programación.
- Se pueden limitar el movimiento de los distintos ejes por software.
- Se cuentan con límites de velocidad para todas las partes.
- Monitoreo de la velocidad por dos ordenadores independientes.
- Cuenta con detección de colisiones.
- Sistema de seguridad contra incendios.
- Función "Hold-to-run": es necesario presionar el botón de inicio para poder mover el robot. Al liberar el botón, el robot se detiene.

La importancia de lograr la seguridad en las industrias radica en proteger a los trabajadores, así como el de las empresas, ante cualquier posible fallo que pudiera dañar los equipos o producir un retraso en la producción. Para garantizarlo, las empresas deben cumplir numerosas normas al momento de desarrollar su actividad. A continuación, se nombrarán las que son cumplidas por el robot con el que se trabaja.

## Normas

En cuanto a la normativa legal relativa a la instalación y empleo de robots, ésta ha aumentado en los últimos años. Las normativas más relevantes que existen a nivel mundial son las siguientes:

- Normativa internacional ISO 10218: 1992
- Normativa Americana ANSI/RIA R15.06-1992
- Normativa Europea EN 775 y española UNE-EN 775

A continuación, solo se mencionarán las normas que son cumplidas, y no se entrará en mayores detalles porque eso excede los fines del presente proyecto.

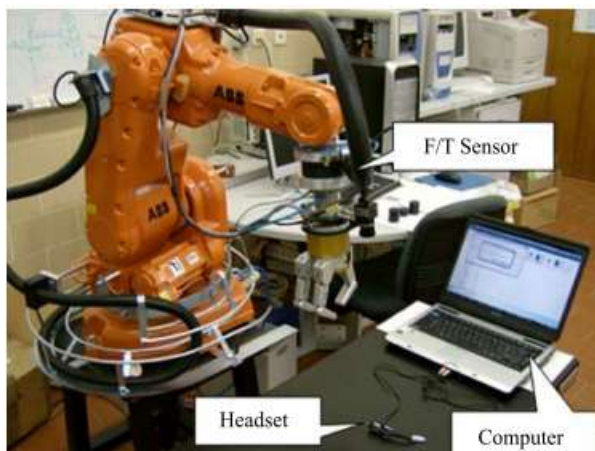
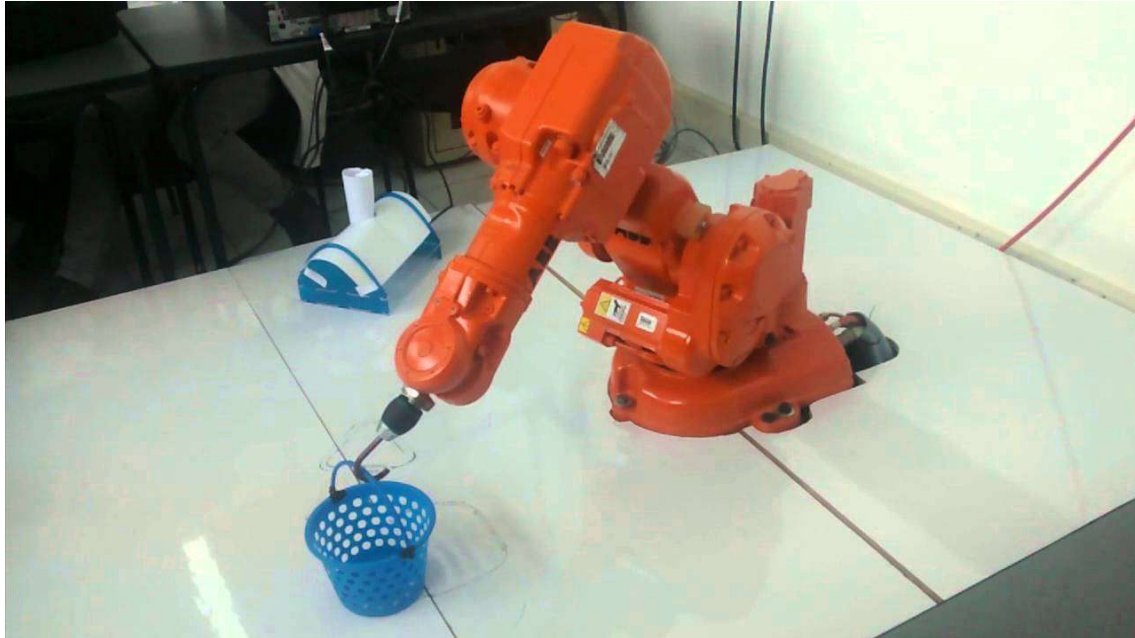
Estándar	Descripción
EN ISO 12100 -1	Seguridad de maquinaria, terminología básica
EN ISO 12100 -2	Seguridad de maquinaria, especificaciones técnicas
EN 954-1	Seguridad de maquinaria, partes de sistemas de control relacionadas a seguridad
EN 60204	Equipos eléctricos de máquinas industriales
EN 61000-6-4	Compatibilidad electromagnética, emisión genérica
EN 61000-6-2	Compatibilidad electromagnética, inmunidad genérica
EN 775	Robots industriales con manipulación, seguridad
IEC 204-1	Equipos eléctricos de máquinas industriales
IEC 529	Grados de protección proporcionados por los alojamientos
ISO 10218	Robots industriales con manipulación, seguridad
ISO 9787	Robots industriales con manipulación, sistemas de coordenadas y movimientos
ISO 9409-1	Robots industriales con manipulación, interfaz mecánica
ANSI/RIA R15.06/1999 (opción)	Requisitos de seguridad para robots industriales y sistemas robotizados
ANSI/UL 1740-1998 (opción)	Norma de seguridad para robots y equipo robotizado
CAN/CSA Z 434-03 (opción)	Robots industriales y sistemas robotizados - Requisitos generales de seguridad
Norma federal 209 de los EE.UU.	Clasificación de sala limpia

Como se puede observar, se cumplen con numerosas normas internacionales y, además, se cumplen todos los estándares de salud y seguridad especificados en las directivas de la CEE sobre la maquinaria.



### Robot ABB IRB140 uso industrial

Debido a que tiene una estructura muy flexible, se lo puede utilizar para realizar numerosas tareas, las siguientes imágenes muestran algunos ejemplos de su uso habitual en las industrias.



### 3.1.2 Antorcha soldadora de la serie PKI

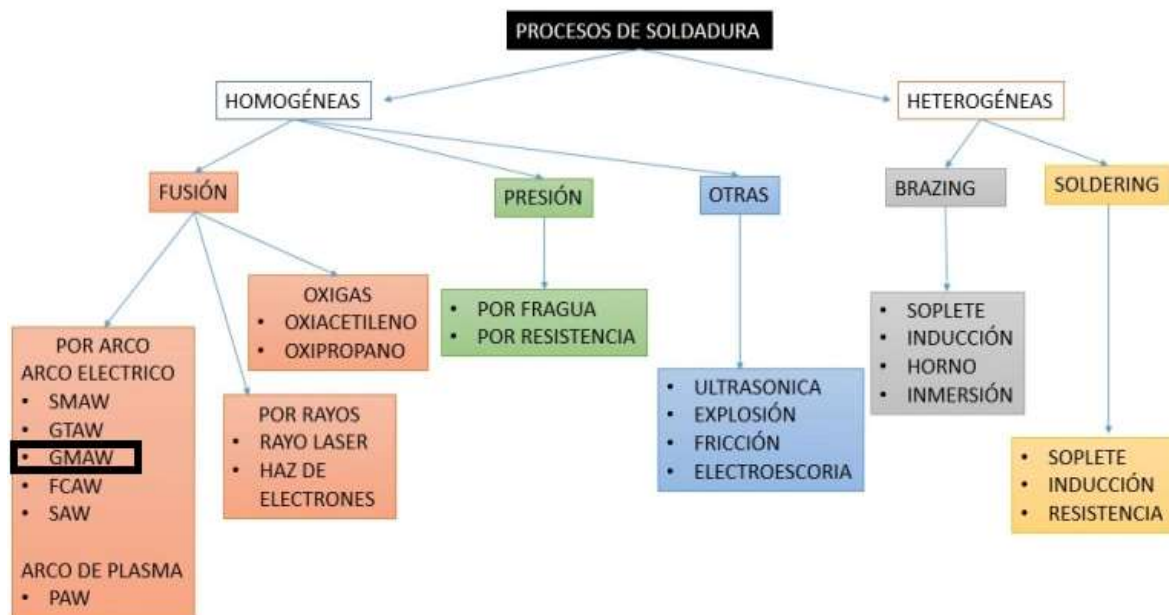
#### Soldadura

Es un proceso de union permanente de materiales en el cual se funden las superficies de contacto de dos, o mas, partes mediante la aplicación conveniente de calor, presion o ambas a la vez. A esta integracion de las partes se denomina “union soldada”.

Se pueden distinguir primeramente los siguientes tipos de soldadura:

- **Heterogénea:** Se efectúa entre materiales de distinta naturaleza, con o sin metal de aportación, o entre metales iguales, pero con distinto metal de aportación. Puede ser blanda, que se alcanza a temperaturas por debajo de 400°C, o fuerte, que se puede alcanzar a temperaturas de 800°C.
- **Homogénea:** Los materiales que se sueldan y el metal de aportación, si lo hay, son de la misma naturaleza. Si no hay metal de aportación, las soldaduras homogéneas se denominan autógenas.

A su vez, estas dos principales tipos presentan las siguientes subdivisiones:

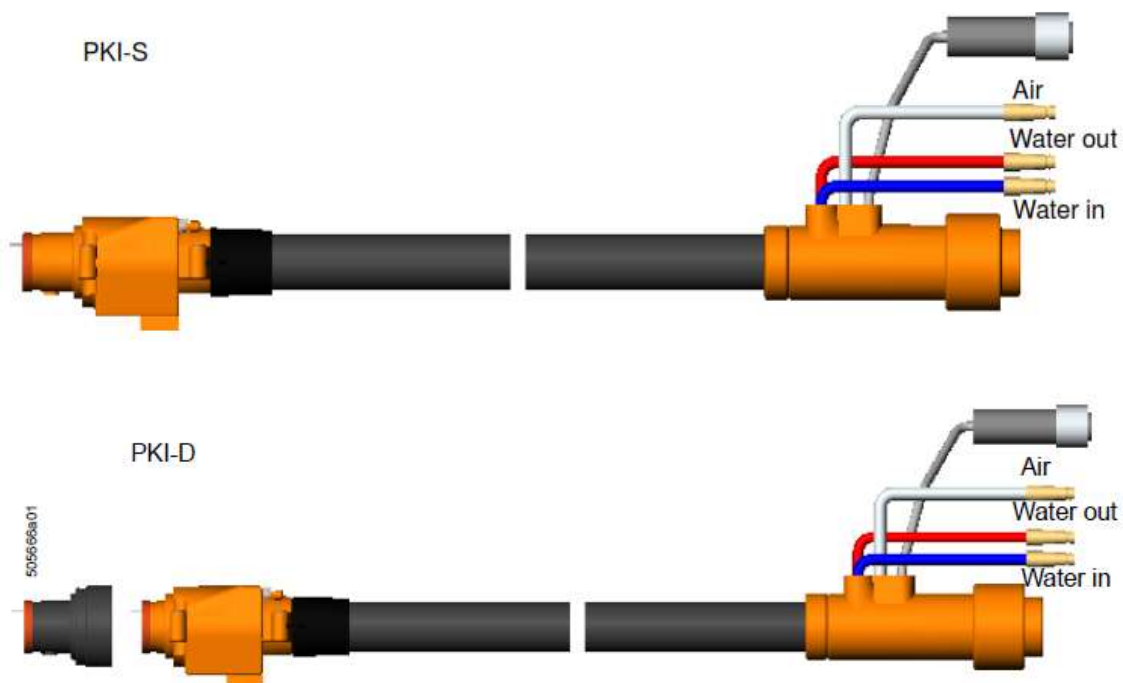


El proceso de trabajo seleccionado es la soldadura por arco de metal y gas (Gas Metal Arc Welding (GMAW o MIG)), en donde, el calor del arco generado entre el electrodo consumible y la pieza a ser soldada es utilizado para fundir las superficies del metal base y el extremo del electrodo. El metal fundido del electrodo es transferido hacia la pieza a través del arco, donde se convierte en metal de soldadura depositado. La protección es obtenida por una cubierta de gas, que puede ser un gas inerte, gas activo o una mezcla de ambos, con el fin de protegerlo de contaminantes de la atmósfera.

Con respecto a la herramienta utilizada para llevar a cabo este proceso, como ya se ha dicho anteriormente, se ha optado por la **antorcha de soldadura de la serie PKI** perteneciente a ABB, que está diseñado para encajar en el brazo del robot.

El sistema cuenta con una refrigerado por agua, incorpora conductos para la limpieza neumática de salpicaduras, tiene un acoplamiento de conexión rápida para el cambio manual de los cuellos de la antorcha. Además, los cables para la señal, el agua de refrigeración, el gas, el aire y el revestimiento de alambre están todos corridos en el paquete de manguera.

Existen dos tipos de diseños, el de desmontaje simple, y el de doble. En vista que estos tipos no nos influyen en los resultados que buscamos obtener, solo nos limitaremos a usar el primero.



Sin embargo, al decidir sobre el cuello de la antorcha que se utilizará, no se puede elegir cualquier, sino que se debe tener en cuenta, entre otras cosas, el tipo de robot que se utilizará y el tipo de soldadura que buscamos que realice.

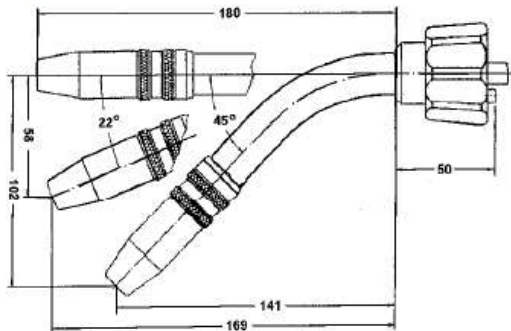
Durante este proceso la carga máxima se reduce en aproximadamente un 25% cuando se realiza la soldadura por impulsos. Los factores que determinan esto son la fuente de energía particular que se está utilizando, la pieza de trabajo que se está soldando y el gas de protección.

En vista de la suma importante que tienen los cuellos de antorcha, los diseñadores dan la posibilidad de contar con diseños especiales suministrados a pedido.

La serie PKI cuenta con muchos tipos de antorchas. A continuación, se mostrarán algunos ejemplos.



## PKI 250

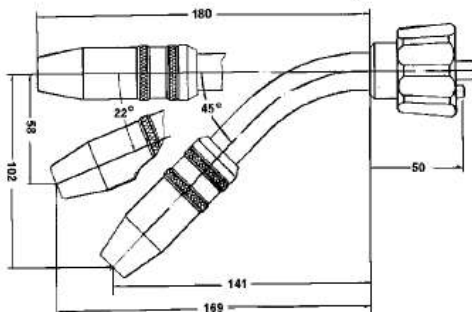


Maximum load 250 A - 100% duty cycle  
Wire diameter 0.8, 0.9, 1.0, 1.2 mm

The contact tip is secured by a clamp nut

Supplied with 12/23 gas nozzle, 68 mm long  
(part.no 438 533-003) and 1.0 mm contact tip  
(part.no 438 533-032)

## PKI 500/PKI 500A (Aluminium)



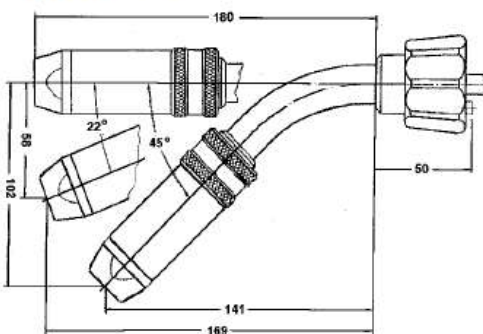
Maximum load 400 A - 100% duty cycle  
Wire diameter 0.8, 0.9, 1.0, 1.2, 1.4, 1.6 mm

The contact tip is secured by a clamp nut.

PKI 500A; the contact tip is screwed on.

Supplied with 15/28 gas nozzle, 76 mm long  
(part.no 441 528-001) and 1.0 mm contact tip  
(part.no 441 922-003)

## PKI 630

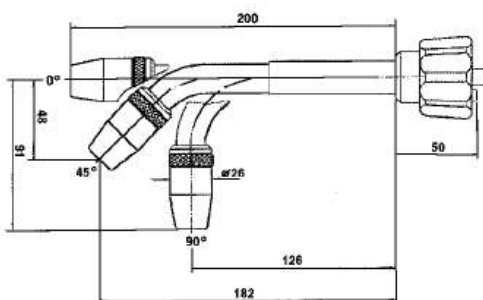


Maximum load 500 A - 100% duty cycle  
Wire diameter 1.0, 1.2, 1.4, 1.6, 2.0, 2.4 mm

The contact tip is screwed on.

Supplied with 18/34 gas nozzle, 32 mm long  
(part.no 438 633-002) and 1.2 mm contact tip  
(part.no 438 633-008)

## PKI 300



Maximum load 300 A - 100% duty cycle  
Wire diameter 0.8, 0.9, 1.0, 1.2 mm

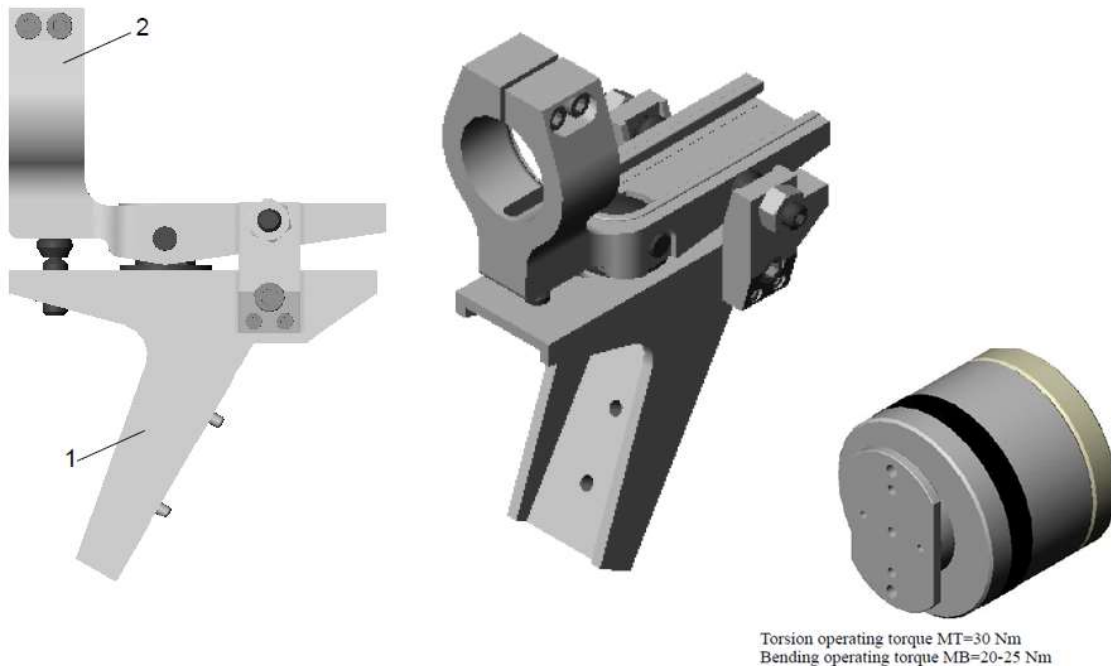
The contact tip is screwed on.

Supplied with 15/26 gas nozzle, 46 mm long  
(part.no 418 423-003) and 1.0 mm contact tip  
(part.no 418 424-003)

Como se puede observar, los cuellos de antorcha son similares entre sí en cuanto a la forma, pero cambian sus capacidades de trabajo y algunas dimensiones. En tanto que la forma de conexión al robot es el mismo para todos.

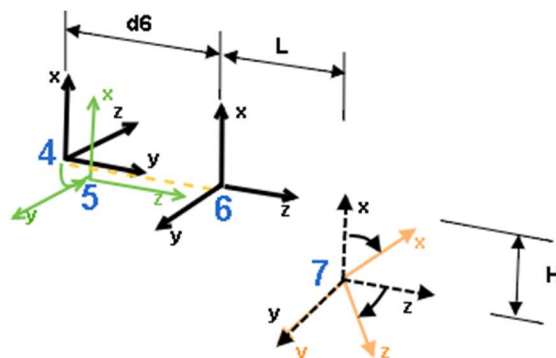
Debido a esto último, para el presente informe no implican mayores cambios al pasar de uno a otro dado que, sólo estamos interesados en las dimensiones.

Al momento de poner en marcha el robot, no solo se consideran las dimensiones de la herramienta, sino también el de su portadora que se muestra en la siguiente imagen.



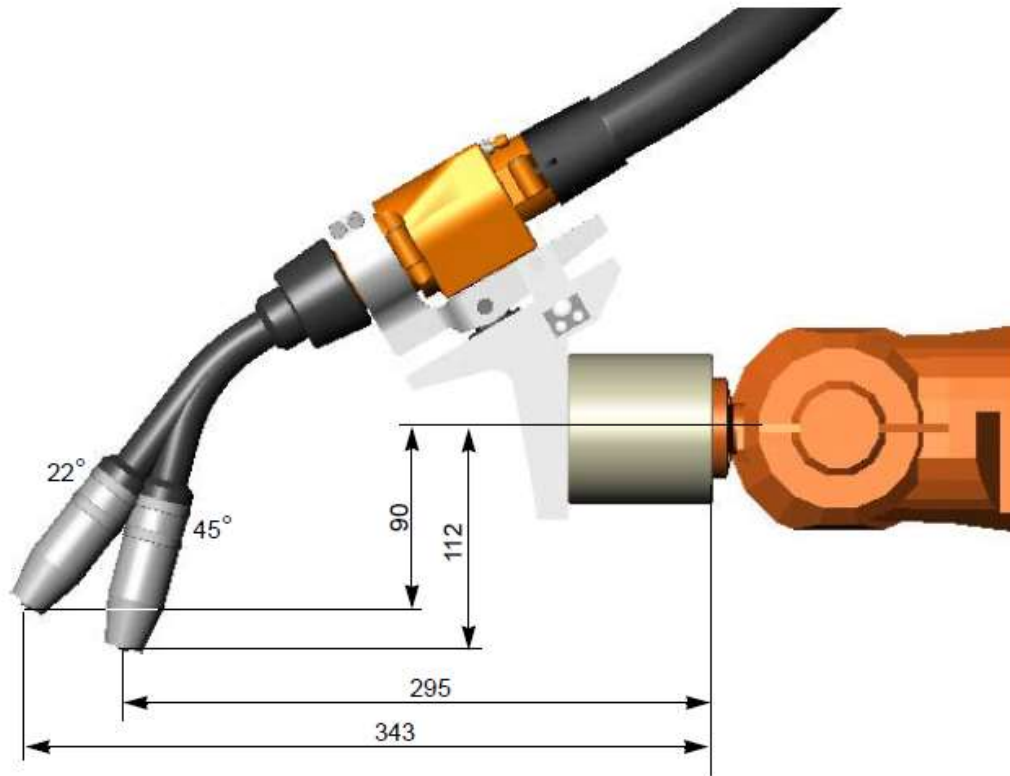
Luego de unirlos para ensamblarla al robot, se deben calcular la distancia total y los ángulos de giro de su extremo con respecto al sistema de referencia asociado al extremo del robot. Para hacerlo se hace uso del método de Denavit Hartenberg que se verá más adelante.

En la siguiente figura el sistema 6 se asocia al extremo del robot y el 7 hace referencia al extremo de la herramienta, en cambio 4 y 5 son los sistemas asociados a los eslabones anteriores. L y H son las distancias entre los orígenes de los sistemas 6 y 7, y se puede observar que el eje Y en ambos tiene la misma dirección, pero los ejes X y Z se encuentran girados.



Cuando se dispone de estos datos, se puede armar una matriz de transformación homogénea que se asociará a la herramienta. Esto nos permite conocer la posición y orientación del extremo final de la herramienta con respecto a la base para poder planificar correctamente el movimiento. Por lo que, si la cambiamos, solo se deberá modificar dicha matriz.

La siguiente grafica muestra el conjunto completo formado por el cuello de la antorcha PKI 300, el portaherramientas y la conexión al extremo del robot.



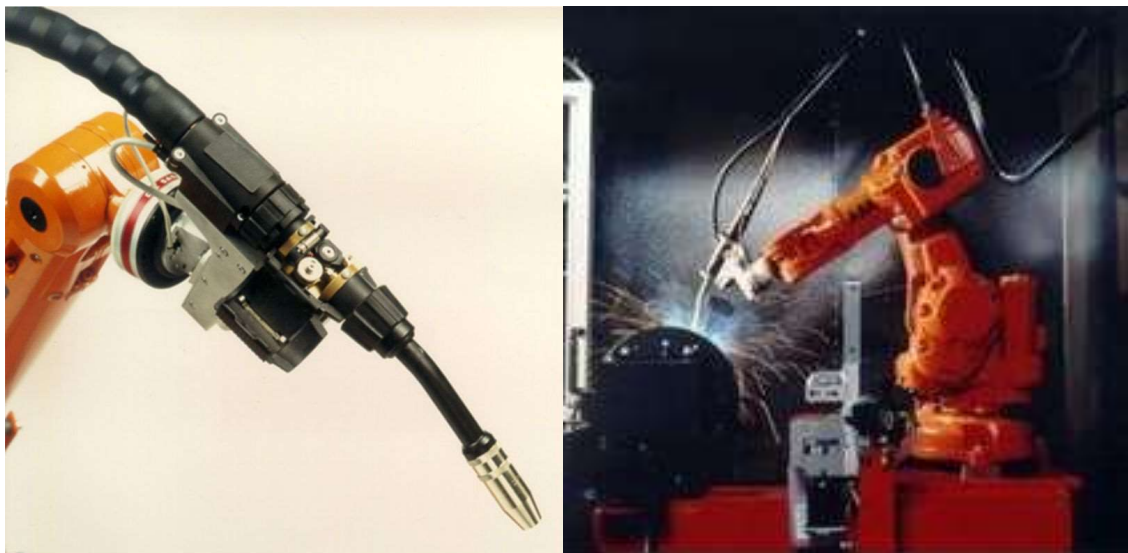
El mismo permite obtener la siguiente matriz. Los detalles de cómo se llega a esta y que significan los valores se darán más adelante cuando se explique la convención de Denavit Hartenberg.

$${}^6A_H = \begin{bmatrix} 0.7071 & 0 & 0.7071 & -0.112 \\ 0 & 1 & 0 & 0 \\ 0.7071 & 0 & 0.7071 & 0.295 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

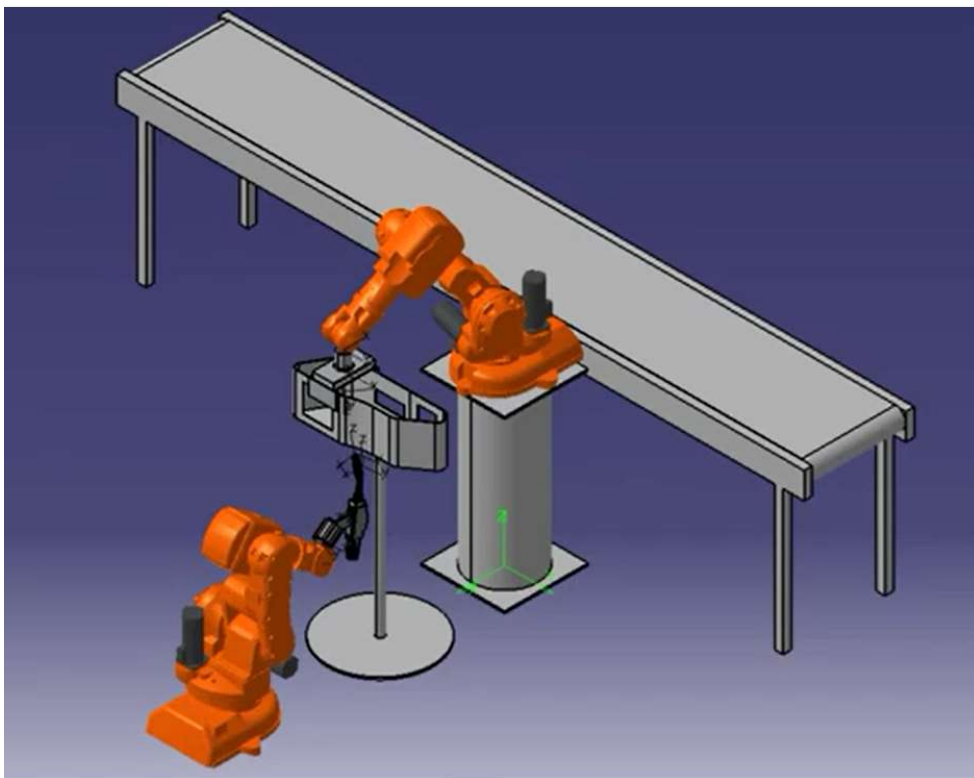
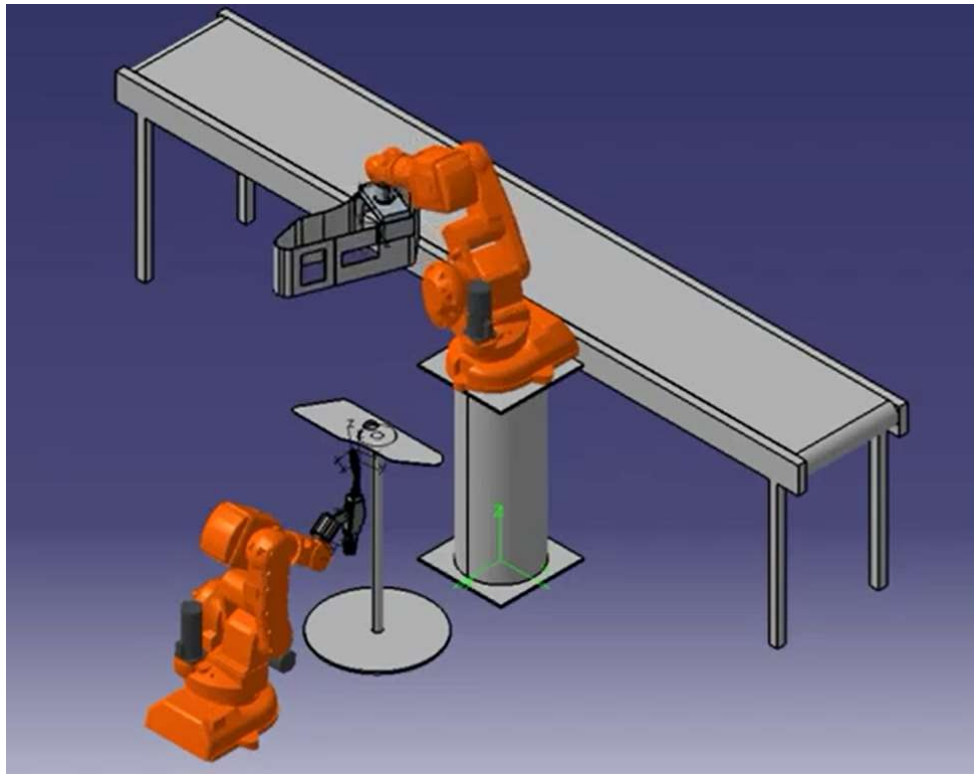
Los robots de ABB ofrecen una gran variedad tanto en potencia como en el alcance de la soldadura. Algunos, como el del presente trabajo, tienen un alcance cercano a un metro, mientras que otros como el IRB 4400, tienen un alcance de más de dos metros y medio sin sumar a la herramienta. Aunque, sin importar el alcance, siempre se establece un área de seguridad para evitar posibles contingencias. Esto se muestra en la siguiente gráfica.



A continuación, se puede apreciar la herramienta ensamblada al robot IRB140, y ambos realizando el trabajo de soldadura por arco.



Luego, las otras imágenes muestran a dos de estos robots trabajando juntos en un ambiente industrial, en donde uno acerca la pieza y el otro la suelda.





### 3.2. Librerías

Para poder llevar a cabo el proyecto planteado se hace uso de software Matlab, en donde se desarrolló el programa trabajando en conjunto con las librerías Arte y Robotics de Peter Corke.

**Librería Arte:** es una toolbox de Matlab centrada en manipuladores robóticos, se incluyen tanto los mecanismos paralelos como los de serie. Las principales características son las siguientes: cuenta con diseños 3D de un gran número de robots industriales y permite incluir nuevos modelos; permite graficar y observar la posición, la velocidad y la aceleración de las coordenadas conjuntas del robot cuando se realiza un movimiento; los torques y fuerzas en cada unión pueden ser graficadas; una guía de enseñanza GUI está disponible para mover y programar los robots; sesiones de prácticas; Representar una visualización del sistema de Denavit-Hartenberg en el robot; se incluye un intérprete Matlab a RAPID que permite programar el robot real utilizando su código Matlab; los robots se pueden programar y simular el lenguaje ABB RAPID y una simulación paso a paso de los programas puede llevarse a cabo dentro del editor y depurador de Matlab; entre otros complementos.

Se hacen uso de los diseños CAD en formato STL, de la base, los eslabones, como así también de la herramienta, y para poder orientarlos de forma correcta se tuvo que corregir algunos parámetros de Denavit Hartenberg que ya venían por defecto. En cuanto a la herramienta y a su portadora, no se disponía mucha variedad ni se aclaraba de que tipo eran, por lo que se tuvieron que elegir para el proyecto.

Se utilizaron como base algunas de sus funciones que se tuvieron que adaptar y modificar a los requerimientos, como pueden ser: **inversekinematic\_irb140(robot, T)** y **directkinematic (robot, q)**, para recién poder utilizarlas.

**Librería Robotics:** al igual que ARTE, también es un toolbox de Matlab centrada en manipuladores robóticos, que no se encuentra desarrollado para tanta variedad. Las ventajas que brinda son:

- El código es maduro y proporciona un punto de comparación para otras implementaciones de los mismos algoritmos;
- Las rutinas generalmente se escriben de una manera directa que permite una fácil comprensión, quizás a expensas de la eficiencia computacional. Si siente firmemente la eficiencia computacional, entonces puede volver a escribir la función para ser más eficiente, compilar el archivo M usando el compilador Matlab o crear una versión MEX;
- Dado que el código fuente está disponible, hay un beneficio para la comprensión y la enseñanza.

Dado que es más eficiente computacionalmente, se utiliza su entorno grafico para llevar a cabo la simulación, con lo que se consigue que no haya mayores retardos y que el movimiento se asemeje más a la realidad.

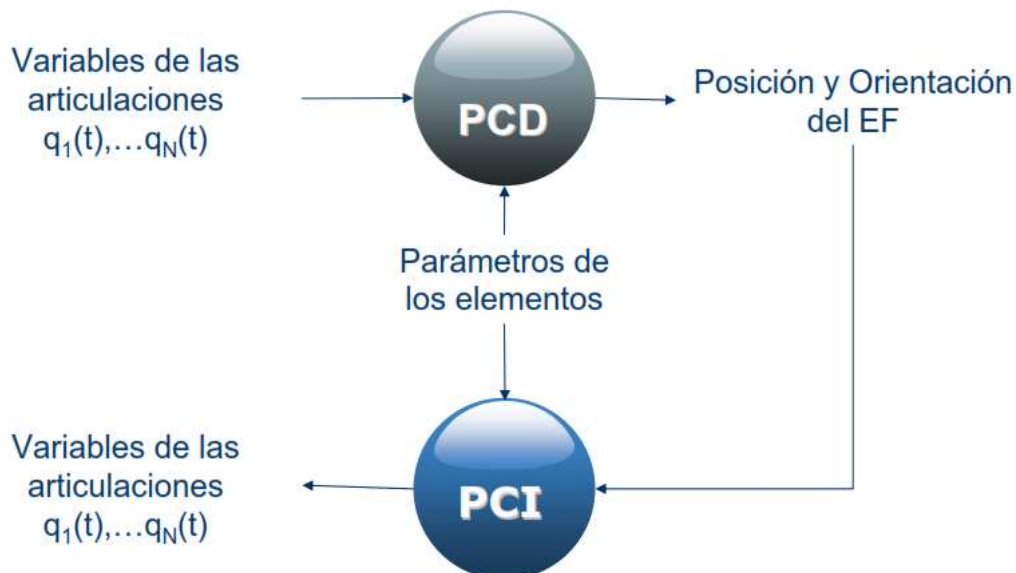
### 3.3. Cinemática

La cinemática de un robot estudia el movimiento del mismo con respecto a un sistema de referencia, y su área de interés es la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular las relaciones entre la posición y la orientación de su efector final con los valores de sus coordenadas articulares.

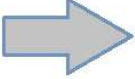

Existen dos problemas fundamentales a resolver al tratar con la cinemática:

- **Problema cinemático directo:** consiste en determinar cuál es la posición y orientación del efector final con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.
- **Problema cinemático inverso:** resuelve la configuración de los valores articulares que debe adoptarse para tomar una posición y orientación deseada.

En el siguiente diagrama se observa la relación que existe entre ambos problemas.



Ahora se observan las ecuaciones que las relacionan.

	Cinemática directa	$x_{extremo}^0 = f_x(q_1, q_2, \dots, q_n)$ $y_{extremo}^0 = f_y(q_1, q_2, \dots, q_n)$ $z_{extremo}^0 = f_z(q_1, q_2, \dots, q_n)$ $\alpha_{extremo}^0 = f_\alpha(q_1, q_2, \dots, q_n)$ $\beta_{extremo}^0 = f_\beta(q_1, q_2, \dots, q_n)$ $\gamma_{extremo}^0 = f_\gamma(q_1, q_2, \dots, q_n)$
$q_k = f_k(x_{ext}^0, y_{ext}^0, z_{ext}^0, \alpha_{ext}^0, \beta_{ext}^0, \gamma_{ext}^0)$ $k = 0 \dots n \text{ (GDL)}$	 	
	Cinemática inversa	

Se observa que el robot puede tener 'n' grados de libertad, o articulaciones, y para obtener la cinemática directa solo es necesario obtener 6 parámetros, que son 3 debido a la posición, y otros 3 debido a la orientación, ambos con respecto a la base.

### 3.4. Cinemática directa

Para hacer la resolución se utiliza una transformación homogénea total, con el que se puede relacionar los sistemas de referencias asociados a cada uno de los eslabones y expresar la posición y orientación del extremo del robot con respecto al sistema de la base en función de los parámetros de las articulaciones, los cuales se obtienen de aplicar la **Convención Estándar de Denavit Hartenberg**.

Una vez que se obtienen los 4 parámetros que caracterizan al sistema de referencia asociado a cada eslabón  $(\theta_i, d_i, \alpha_i)$  se puede obtener la matriz de transformación de este aplicando la siguiente matriz general:

$${}^{i-1}A_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i) \cdot \sin(\theta_i) & \sin(\alpha_i) \cdot \sin(\theta_i) & a_i \cdot \cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i) \cdot \cos(\theta_i) & -\sin(\alpha_i) \cdot \cos(\theta_i) & a_i \cdot \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Los parámetros mencionados se interpretan de la siguiente manera:

$\theta_j$ : Ángulo medido desde el eje  $X_{j-1}$  hasta el  $X_j$ , alrededor del eje  $Z_{j-1}$ .

$d_j$ : Distancia desde el origen del sistema  $j - 1$  hasta el eje  $X_j$ , a lo largo del eje  $Z_{j-1}$ .

$a_j$ : Distancia entre el eje  $Z_{j-1}$  y el eje  $Z_j$  a lo largo del eje  $X_j$ .

$\alpha_j$ : Ángulo medido desde el eje  $Z_{j-1}$  hasta el  $Z_j$ , alrededor del eje  $X_j$ .

Luego de obtener todas las matrices, se las multiplica en la secuencia dada y se obtiene la matriz de transformación homogénea total, que relaciona el extremo del robot con respecto a la base.

$${}^0T_{extremoR} = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6$$

Cuando se agrega la matriz de transformación asociado a la herramienta se obtiene la matriz de transformación total del sistema completo con el que ya se puede proceder a calcular la cinemática directa.

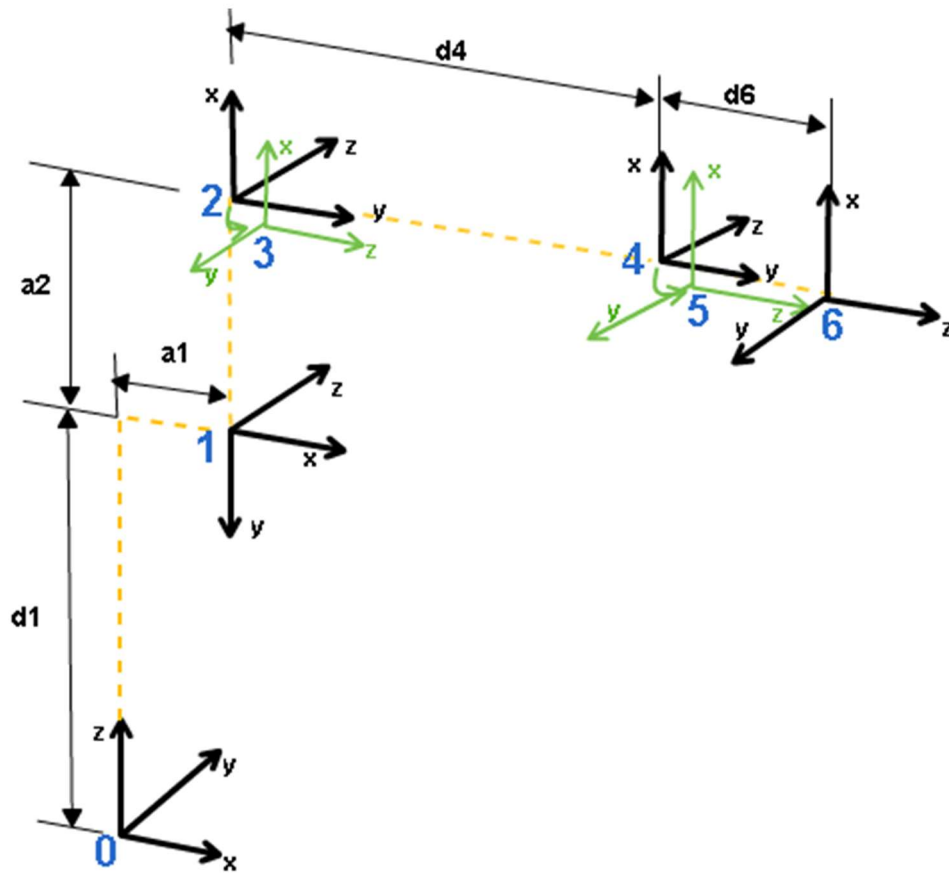
$${}^0T_{extremo} = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6 \cdot {}^6A_H$$

$${}^0T_{extremo} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_{extremo}^0 \\ r_{21} & r_{22} & r_{23} & y_{extremo}^0 \\ r_{31} & r_{32} & r_{33} & z_{extremo}^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A su vez, esta se compone de una submatriz de rotación y un vector de traslación, las cuales representan la orientación y posición deseadas.

$${}^0Rot_{extremo} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad {}^0Tras_{extremo} = \begin{bmatrix} x_{extremo}^0 \\ y_{extremo}^0 \\ z_{extremo}^0 \end{bmatrix}$$

La siguiente grafica muestra los sistemas de referencia asociados a cada uno de los eslabones del manipulador que se obtuvieron de aplicar las reglas DH.



Con estos sistemas de referencia podemos obtener los parámetros de DH.

### La tabla de parámetros de Denavit Hartenberg

Articulación	1	2	3	4	5	6
$\theta$	$\theta_1$	$\theta_2$	$\theta_3$	$\theta_4$	$\theta_5$	$\theta_6$
$d$	d1	0	0	d4	0	d6
$a$	a1	a2	0	0	0	0
$\alpha$	-90	0	-90	90	-90	0
O'(Tipo Rot/Tras)	0	0	0	0	0	0

El nuevo parámetro tenido en cuenta es  $\sigma_j$ , que vale 0 para articulaciones rotacionales ( $\theta$  variable), y 1 para articulaciones prismáticas ( $d$  variable).

Como resultado de aplicar estos parámetros en la matriz de transformación homogénea general podemos obtener las siguientes matrices, donde el subíndice indica el número del eslabón al que pertenecen y el superíndice el eslabón con respecto al cual se mide.

$${}^0A_1 = \begin{bmatrix} c1 & 0 & -s1 & a1.c1 \\ s1 & 0 & c1 & a1.s1 \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1A_2 = \begin{bmatrix} c2 & -s2 & 0 & a2.c2 \\ s2 & c2 & 0 & a2.s2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2A_3 = \begin{bmatrix} c3 & 0 & -s3 & 0 \\ s3 & 0 & c3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

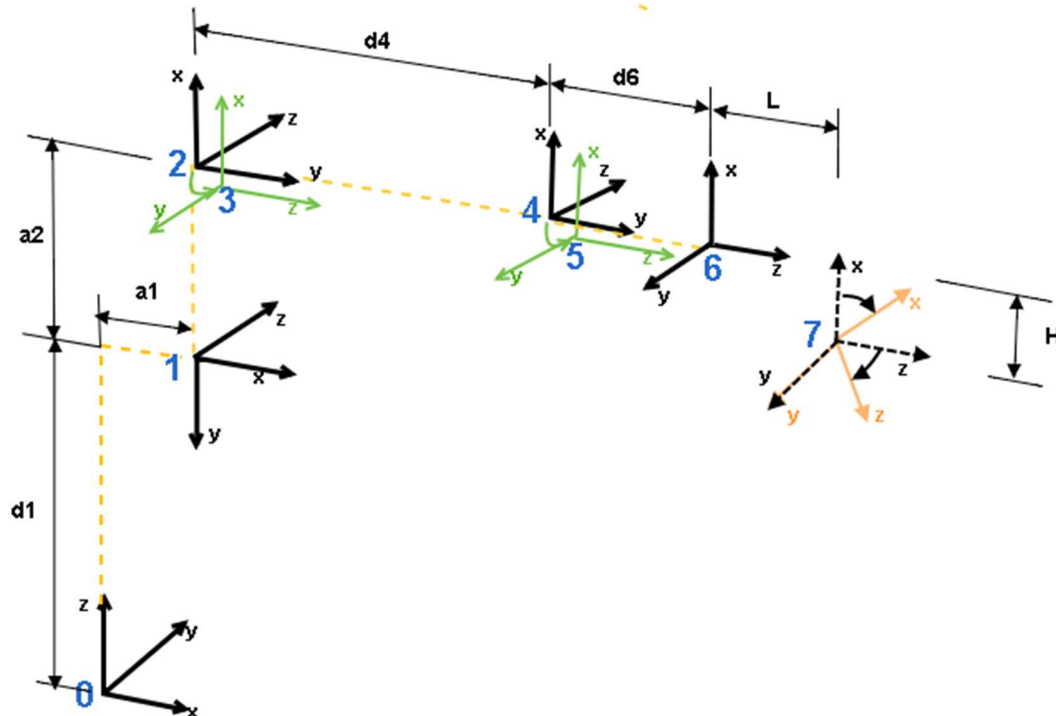
$${}^3A_4 = \begin{bmatrix} c4 & 0 & s4 & 0 \\ s4 & 0 & -c4 & 0 \\ 0 & 1 & 0 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^4A_5 = \begin{bmatrix} c5 & 0 & -s5 & 0 \\ s5 & 0 & c5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^5A_6 = \begin{bmatrix} c6 & -s6 & 0 & 0 \\ s6 & c6 & 0 & 0 \\ 0 & 0 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De igual modo, se procede a mostrar el mismo conjunto, pero con el agregado del sistema debido a la herramienta, el cual se encuentra trasladado una distancia  $-H$  con respecto al eje X, una distancia  $L$  con el eje Z, y rotado un ángulo de  $45^\circ$  con el eje Y. Esto se representa con la siguiente línea de código.

```
R.tool = transl([-H, 0, L])* troty(-pi/4); % SERIE PKI300 45°
%R.tool = transl([H, 0, L])* troty(-pi*22/180); % SERIE PKI300 22°
```



La siguiente es una matriz de transformación homogénea general, para cualquier traslación respecto al eje X y Z, y cualquier giro; y a la derecha se encuentra la que obtenemos para nuestro caso de aplicación.

$${}^6A_H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -H \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^6A_H = \begin{bmatrix} 0.7071 & 0 & 0.7071 & -0.112 \\ 0 & 1 & 0 & 0 \\ 0.7071 & 0 & 0.7071 & 0.295 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Como se mencionó anteriormente, la librería ARTE cuenta con funciones para obtener la cinemática directa, una de las cuales es **directkinematic ()**. Sin embargo, esta no considera los valores offset ni a la herramienta, por lo que se tuvo que modificar para adaptarlo a los requerimientos.

El programa creado para resolver este punto se llama **CinemáticaDirecta ()** y el código desarrollado está en el **Anexo A.1**.

En primer lugar, para tener en cuenta los valores offset, ángulos que fueron usados para indicar el estado inicial de cada articulación, al momento de usar los parámetros para obtener la matriz asociada a cada eslabón hace uso de la función **D\_H ()**, mostrada en el **Anexo B**, en donde se tuvieron que sumar a los valores articulares para poder obtener los valores reales tanto de posición como de orientación.

Luego, se hace agregado de la matriz asociada a la herramienta para poder obtener la posición y orientación su extremo con respecto a la base. Esto se hizo post multiplicándola con la que relacionaba el extremo del robot con la base del mismo.

Luego de tener en cuenta las consideraciones anteriores, podemos obtener la cinemática directa llamando a la función **CinemáticaDirecta (R, q)**, que tiene como argumentos el robot, R, y los valores articulares, q, para poder llegar a la posición y orientación final del extremo de la herramienta con respecto a la base.

### Prueba de la Cinemática Directa

Para probar el correcto funcionamiento de nuestro cálculo de la cinemática directa procedemos a definir un conjunto de valores articulares que deseamos que tome el robot, luego comparamos la matriz de transformación homogénea obtenida con los dados por la gráfica en la que se representa el robot con cilindros en color rojo y azul. Este procedimiento se aplica al sistema con y sin la herramienta.

Parámetros a tener en consideración:

qpos: representa la posición consigna

Ttotal: es la matriz de transformación obtenida.

[x, y, z]: en la gráfica representan la posición alcanzada por el robot al estar en los valores articulares qpos.

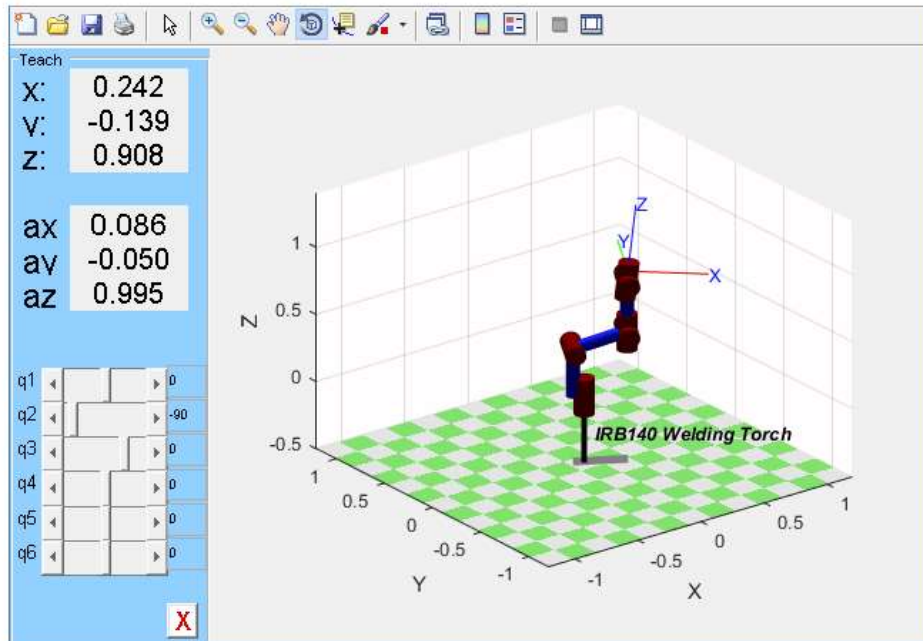
[ax, ay, az]: en la gráfica representan los valores en el eje Z.

Posiciones deseadas:

0 -1.5708 0 0 0 0

Ttotal:

0.8617	0.5000	0.0865	0.2415
-0.4975	0.8660	-0.0499	-0.1394
-0.0998	0.0000	0.9950	0.9079
0	0	0	1.0000

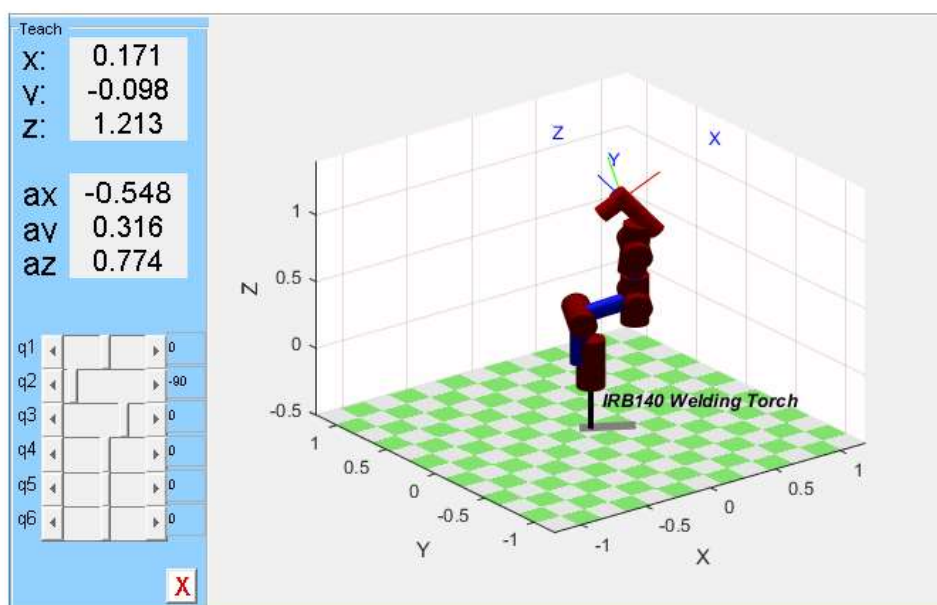


Posiciones deseadas:

0 -1.5708 0 0 0 0

Ttotal:

0.6704	0.5000	-0.5482	0.1705
-0.3871	0.8660	0.3165	-0.0984
0.6330	0.0000	0.7742	1.2126
0	0	0	1.0000





### 3.5. Cinemática inversa

Para analizar la cinemática inversa se hace uso de la matriz de transformación homogénea total, obtenida anteriormente, de donde se pueden obtener doce ecuaciones, nueve correspondientes a los elementos de la submatriz de rotación, y tres del vector de traslación. Sin embargo, solo tres de la submatriz son linealmente independientes, que junto con las tres del vector de traslación dan un total de seis. Representadas a continuación.

$$\begin{aligned}x_{extremo}^0 &= f_x(q_1, q_2, \dots, q_n) \\y_{extremo}^0 &= f_y(q_1, q_2, \dots, q_n) \\z_{extremo}^0 &= f_z(q_1, q_2, \dots, q_n) \\\alpha_{extremo}^0 &= f_\alpha(q_1, q_2, \dots, q_n) \\\beta_{extremo}^0 &= f_\beta(q_1, q_2, \dots, q_n) \\\gamma_{extremo}^0 &= f_\gamma(q_1, q_2, \dots, q_n)\end{aligned}$$

Se ha usado la función “**inversekinematic\_irb140()**”, perteneciente a la librería ARTE, como base de nuestro algoritmo para obtener la cinemática inversa, de donde se obtienen 8 de todas las soluciones posibles, y se elegirá la más óptima cuando se realice planificación de trayectorias. Al igual que en el caso anterior, se tuvo que adaptar para que considere los valores offset y la herramienta. A este programa modificado, que tiene en cuenta estas consideraciones se lo denominó

**CinemáticaInversa(R,T)**, y está detallado en el **Anexo C.1**. Recibe como parámetros al robot, R, y la matriz de transformación homogénea total.

Para poder obtener los valores articulares se usa el método de Pieper, que se explica a continuación.

#### Solución de Pieper

La esencia de esta solución subyace en que, una vez posicionada en un punto del espacio una muñeca en la que los tres ejes se cortan, el movimiento de las articulaciones de la misma en torno a sus ejes no altera la posición espacial del punto de corte, o lo que es igual de la muñeca. Por tanto, se puede solucionar el problema cinemático inverso tan solo para la posición del punto de corte de los ejes de la muñeca del robot, es decir, para las tres primeras articulaciones que son las que posicionan en el espacio a la muñeca, puesto que cuando se produzca la orientación correspondiente no se alterara la posición final. De esta forma se reduce notablemente la dificultad del método de resolución. Sin embargo, la posición de los puntos de corte de los ejes de la muñeca no es la posición deseada del extremo del robot, pues la muñeca tiene unas dimensiones físicas, por lo que es necesario determinar la posición del punto de corte de los ejes de la muñeca a partir de la posición deseada del extremo del robot.

Para poder aplicar este programa tenemos que usar la matriz que relaciona el extremo del robot con la base. Dado que tenemos:

$${}^0T_{extremo} = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6 \cdot {}^6A_H$$



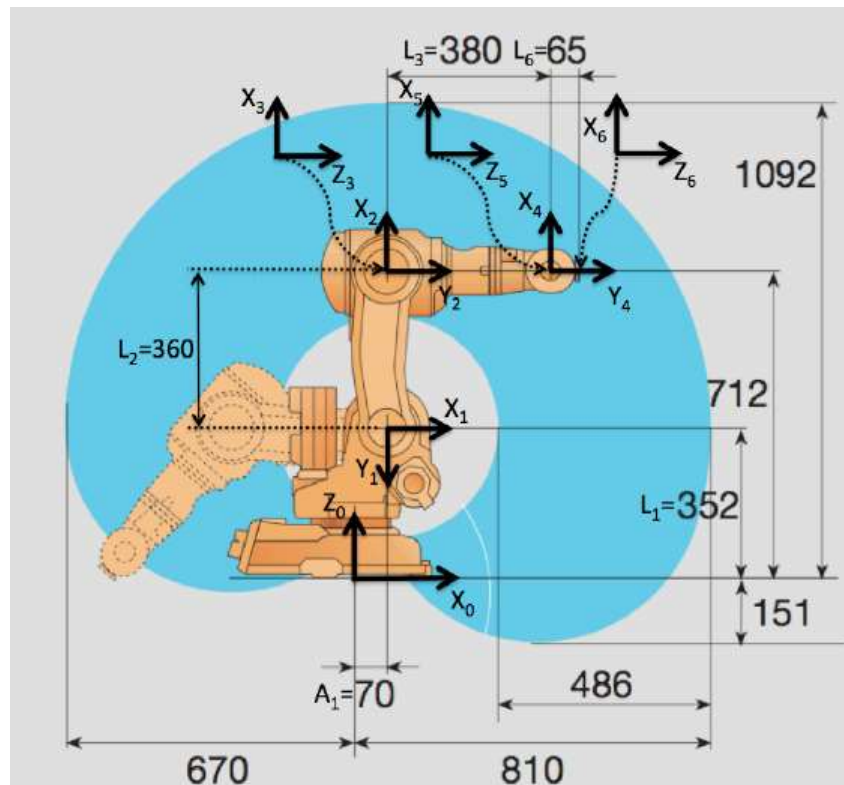
Debemos post multiplicar por la inversa de la matriz de transformación asociada a la herramienta en el lado derecho para obtener:

$${}^0T_{extremoR} = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \cdot {}^4A_5 \cdot {}^5A_6$$

Del mismo se utiliza el eje z, tres primeros valores de la tercera columna, denominado **Z6**, y la posición (Px, Py, Pz) para poder obtener la posición de la muñeca.

$$P_m = {}^0P_3 = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - L_6 \cdot Z_6$$

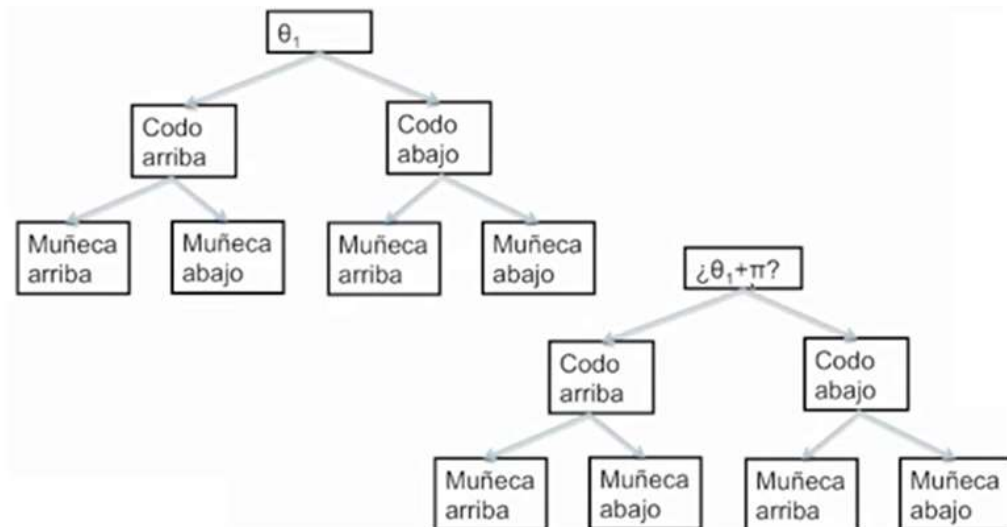
Esto se observa mas claramente en la siguiente grafica, donde los ejes Z del sistema 5 y 6 son coincidentes, por lo que se resta unicamente el desplazamiento L6 medida desde el eje Z del sistema 6.



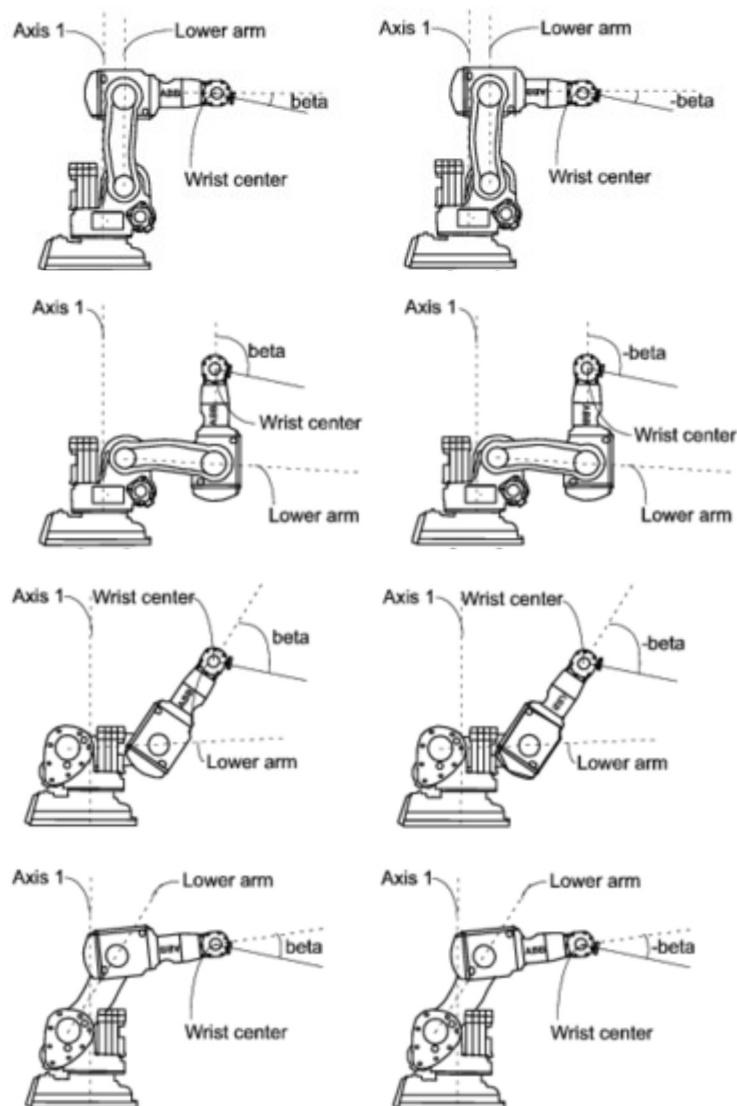
Para poder obtener los tres primeros valores articulares no se tuvieron inconvenientes, sin embargo, para los tres ultimos se tuvieron que cambiar algunos signos al usar la funcion atan2(), porque el metodo original tenia los sistemas de coordenadas distintos a los que se obtuvieron de aplicar la convencion DH. El desarrollo de este metodo se encuentra detallado en el **Anexo C.4**, en la función denominada **Pieper()**.

Cabe aclarar que en cada función usada de la librería ARTE se tuvieron que hacer algunas modificaciones porque muchos datos estaban expresados como estructuras.

Para resumir todo lo dicho anteriormente, si al programa **CinematicaInversa(R,T)**, le damos como parámetro el robot, R, y la matriz de transformación total, T, nos devuelve las ocho soluciones que se calculan. El esquema siguiente muestra de donde se consideran cada una de ellas.



Las cuatro primeras posiciones son para la base a  $0^\circ$ , mientras que las cuatro ultimas son para la base girada  $180^\circ$ .



Para poder comprobar que se obtienen los valores correctos se comienza indicando los valores consigna de las articulaciones,  $q_{pos}$ , para obtener la matriz de transformación homogénea total aplicando la función **CinemáticaDirecta(R,qpos)**. Luego, usamos este para hacer el proceso inverso y así obtener las consignas iniciales, que este debe coincidir con alguna de las soluciones.

```
% POSICION A MOSTRAR -----
qpos = [(pi/8) -pi/3 -(2*pi/3) pi pi/2 pi/4];

%=====
%      CINEMATICA
%=====

Ttotal = CinematicaDirecta(R,qpos);
qinvSoluciones = CinematicaInversa(R,Ttotal);
```

En los resultados mostrados a continuación se puede observar que la octava solución encontrada coincide con la posición deseada. Para terminar la verificación, se debería sacar la cinemática directa a cada solución y deberían ser iguales a Ttotal.

En esta verificación realizada consideramos al robot con la herramienta.

Mostramos como ejemplo la solución tres, en donde se observa que coinciden.

Posiciones deseadas:							
0.3927	-1.0472	-2.0944	3.1416	1.5708	0.7854		
Ttotal:							
0.6947	-0.7169	-0.0582	0.5139				
-0.5958	-0.6188	0.5120	0.0122				
-0.4030	-0.3210	-0.8570	0.4967				
0	0	0	1.0000				
qinvSoluciones:							
-2.7489	-2.7489	-2.7489	-2.7489	0.3927	0.3927	0.3927	0.3927
-0.6496	-0.6496	0.5984	0.5984	-1.7470	-1.7470	-1.0472	-1.0472
-0.2031	-0.2031	-2.6243	-2.6243	-0.7330	-0.7330	-2.0944	-2.0944
3.1416	-0.0000	3.1416	-0.0000	-0.0000	3.1416	-0.0000	3.1416
-1.4810	1.4810	-2.6543	2.6543	-2.2323	2.2323	-1.5708	1.5708
-2.3562	0.7854	-2.3562	0.7854	-2.3562	0.7854	-2.3562	0.7854
>> Tt = CinematicaDirecta(R,qinvSoluciones(:,3)')							
Tt =							
0.6947	-0.7169	-0.0582	0.5139				
-0.5958	-0.6188	0.5120	0.0122				
-0.4030	-0.3210	-0.8570	0.4967				
0	0	0	1.0000				

## 4. Planificación de trayectorias

Es la búsqueda de una sucesión de posiciones para un robot, que permitirán llevarlo desde un estado inicial a uno final, entendiéndose por estado a la descripción de la ubicación del robot referida a un sistema de referencia absoluto, que en nuestro caso se considera que está ubicado en la base.

La importancia radica en la encontrar estrategias de control para obtener trayectorias adecuadas, seguras y que posean la mayor calidad en su desplazamiento.

La configuración que adquiere una determinada trayectoria queda definida por el espacio de trabajo, por la geometría del robot, sus capacidades de movimiento de los actuadores, suavidad deseada, al igual que precisión, entre otros.

La topología del ambiente de trabajo restringirá el espacio libre de obstáculos en el cual se pueden expresar las posibles trayectorias para alcanzar el estado final deseado. Además, influye el modelo dinámico del robot, pero esto no se tiene en cuenta porque excede los fines del presente proyecto.

Para llegar a obtener un planificador que funcione correctamente, se deben seguir los siguientes pasos:

1. Estudiar las necesidades de movimiento especificadas por el usuario o por los sensores propios del sistema robotizado, evitando colisiones con el entorno etc., obteniendo una expresión analítica en coordenadas cartesianas de la trayectoria deseada en función del tiempo (libre de colisiones).
2. Muestrear la trayectoria anterior en una serie finita de puntos nudo de control que se utilizan como puntos inicial y final de cada segmento. Cada uno de estos puntos está especificado por sus componentes cartesianas de posición y orientación  $(x, y, z, \alpha, \beta, \gamma)$ .
3. Pasar cada uno de estos puntos a coordenadas articulares del robot, utilizando para ello la transformación homogénea inversa.
4. Realizar la interpolación entre los puntos de las coordenadas articulares y obtener para cada articulación una expresión del tipo  $q_i(t)$  para cada segmento de control.

Se observa que un planificador consiste en obtener una función de trayectoria  $q(t)$  que se modifica en cada intervalo de control. Hay que hacer notar que el paso 3 debe tratarse con cuidado, pues hay que tener en cuenta las posibles soluciones múltiples de la transformación inversa.

**La tarea que se va a realizar en el presente proyecto es una soldadura en línea recta, en donde el ángulo de soldadura, al igual que el punto inicial y final, son ingresados por el usuario.** Para ello se discretiza la trayectoria a seguir en una serie de puntos, y el robot avanza entre los puntos sucesivos.

A continuación, se describirán algunos de los puntos que influyen al momento de generar la trayectoria, y se explicará cómo se tuvieron en cuenta al momento de desarrollar el programa para realizar la tarea deseada.

## 4.1. Espacio de trabajo

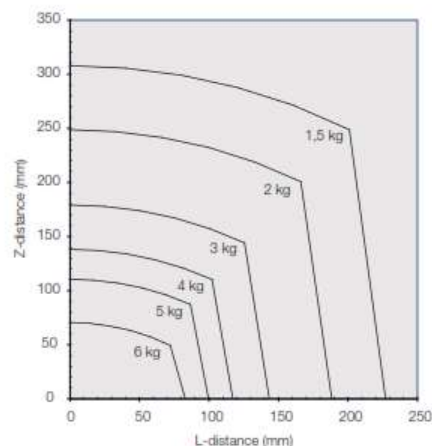
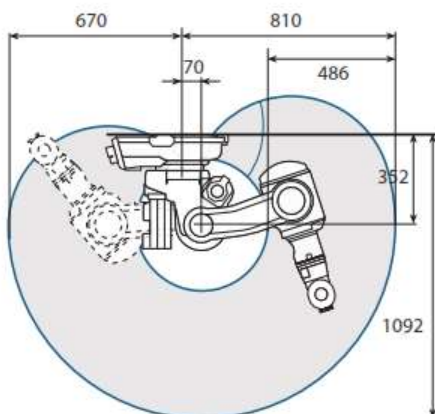
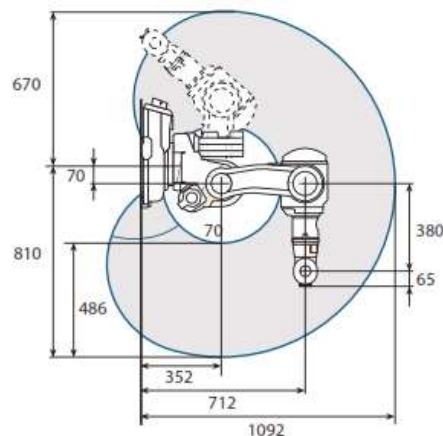
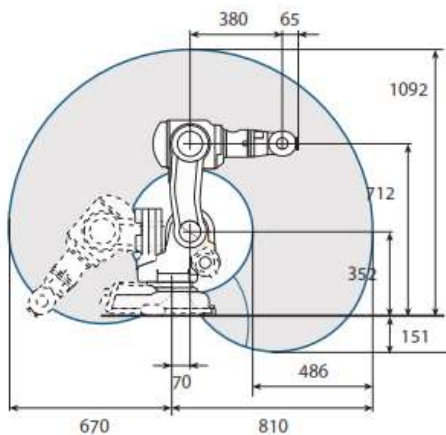
Se refiere únicamente al espacio dentro del cual puede desplazarse el extremo de la muñeca del robot. Para determinarlo, no se toma en cuenta el efecto final, dado que, a la muñeca del robot se le pueden adaptar herramientas de distintos tamaños y tipos.

Todos los puntos de este espacio no cuentan con la misma accesibilidad. Hay algunos a los que se puede llegar de varias formas, teniendo así múltiples soluciones, en cambio, hay otros con accesibilidad mínima a los que sólo se puede llegar con una única orientación. En cuanto a los puntos no son accesibles, estas darán soluciones imaginarias a la cinemática inversa del robot.

Para el caso de los robots industriales, los fabricantes brindan en sus hojas de especificaciones, o datasheet, planos del espacio de trabajo desde distintas perspectivas, a fin de dar la mejor representación posible. En estos se muestran al manipulador con toda su estructura totalmente extendida y plegada, para indicar los límites externos, y totalmente contraída, para indicar los internos.

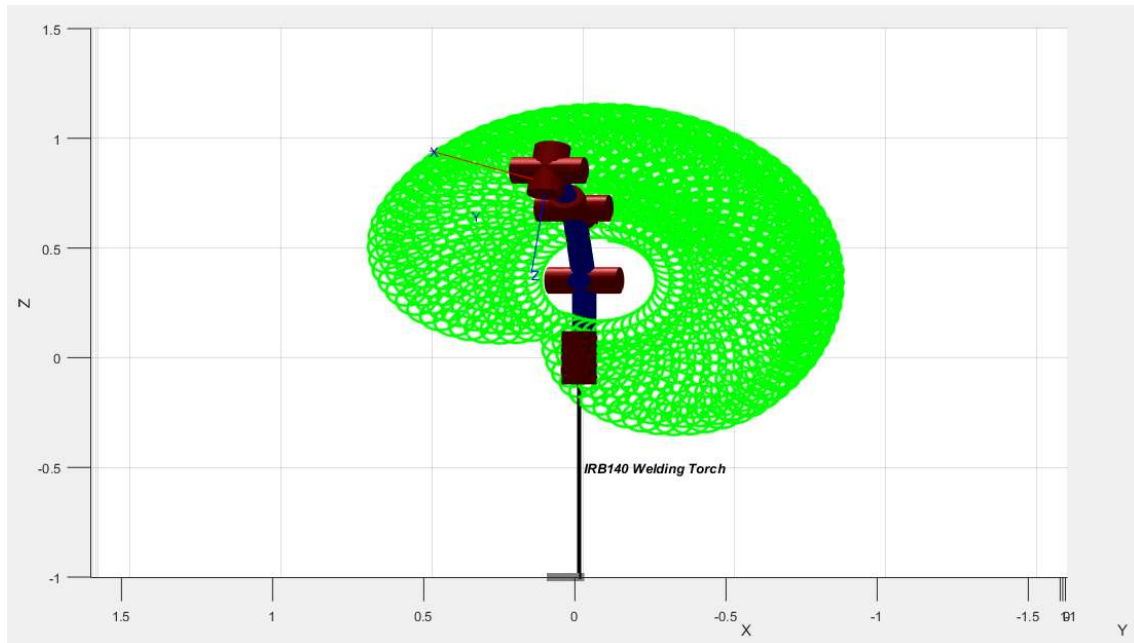
Los siguientes planos representan el espacio de trabajo del IRB 140. Dado que se puede colocar en distintas orientaciones, se tienen distintos planos de trabajo, aunque la forma siempre resulta la misma. En nuestro caso, lo tomamos como esta en la primera gráfica, en donde la base se encuentra conectado en la parte inferior.

La cuarta grafica indica el esfuerzo que se puede ejercer en función de la distancia medida en el eje Z, se observa que conforme aumenta la distancia a la carga, es menor la carga que se puede soportar.



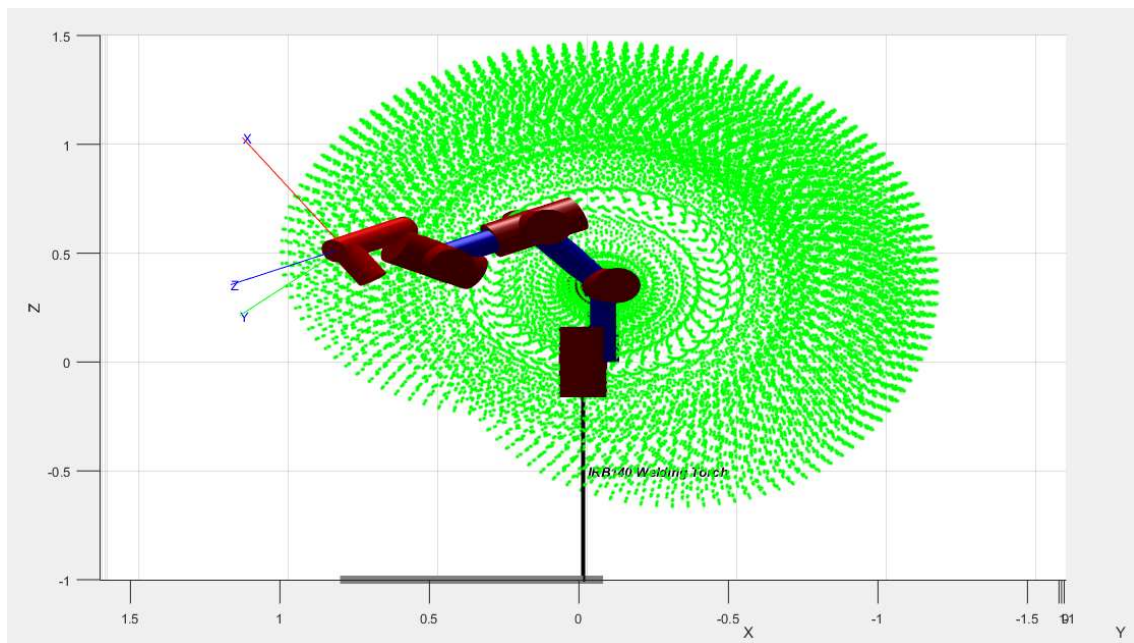


La siguiente grafica muestra el plano de trabajo que se ha obtenido en nuestro proyecto, como se puede observar es coincidente con el que da el fabricante. Aunque el plano resultante también depende del valor offset que se le otorgue. Para obtenerlo usamos la función **Space(R)**, que recibe como argumento el robot, detallado en el **Anexo D.3**, y se considera que la base no gira.



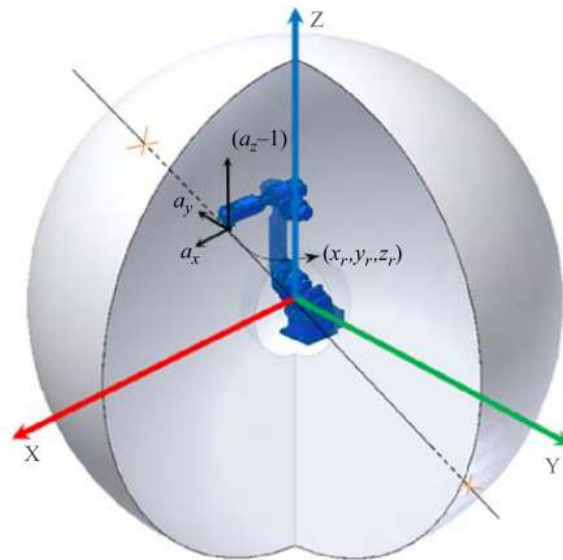
Al darle una aplicación al robot, en este caso es la soldadura arco, resulta de gran utilidad conocer el espacio de trabajo para la misma.

Para poder llegar a obtenerlo, comenzamos ingresando los parámetros de la herramienta, y luego se procede a usar la misma función utilizada para el anterior. Resultando la siguiente gráfica.



Para poder obtener el volumen de trabajo tenemos que hacer girar la base, para ello se elige un ángulo de giro (paso) que aumenta hasta producir un giro completo. De esta forma el plano inicial se repite y produce espacio de trabajo completo.

En este punto, al realizar la gráfica completa no se llega a apreciar muy bien el resultado obtenido. Por ello, se opta por incluir la siguiente imagen a modo de representarlo.



## 4.2. Singularidades

Las configuraciones singulares de un robot son aquellas en las que el determinante de la matriz Jacobiano es igual a cero, por lo que no se puede calcular su inversa. Aunque también puede darse que se tengan problemas cuando este determinante es muy cercano a cero, e incluso dentro del espacio de trabajo.

Al anularse el Jacobiano, un incremento infinitesimal de las coordenadas cartesianas supondría un incremento infinito de las coordenadas articulares, lo que en la práctica se traduce en que las inmediaciones de las configuraciones singulares, el pretender que el extremo del robot se mueva a velocidad constante, obligaría a movimientos de las articulaciones a velocidades inabordables por los actuadores. Por ello, en las inmediaciones de las configuraciones singulares se pierde alguno de los grados de libertad del robot, siendo imposible que el extremo se mueva en una determinada dirección cartesiana.

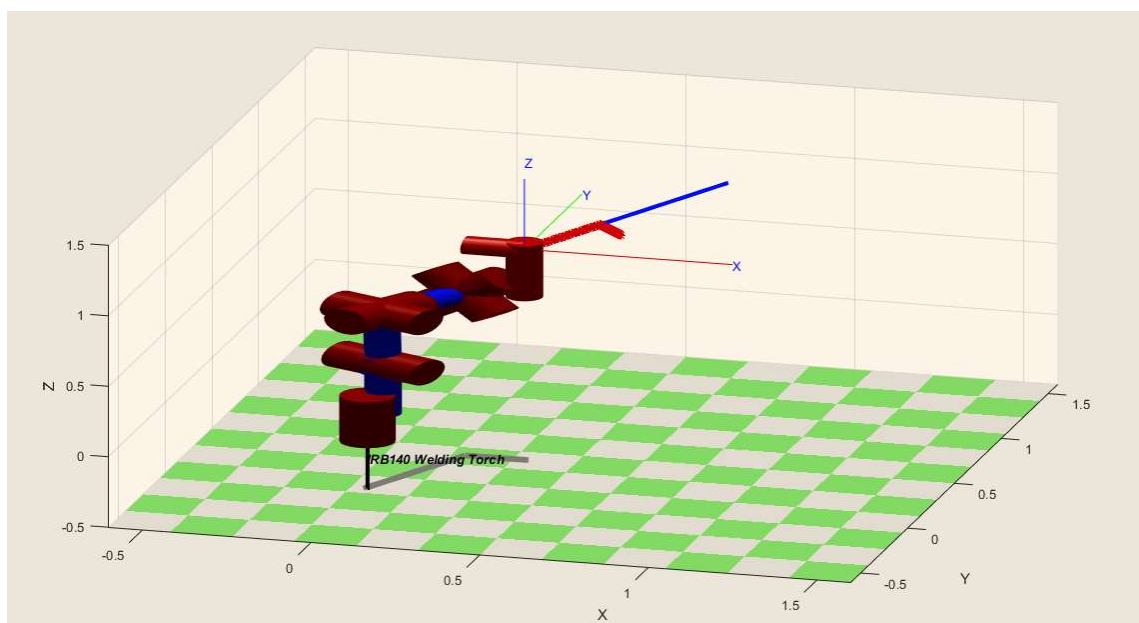
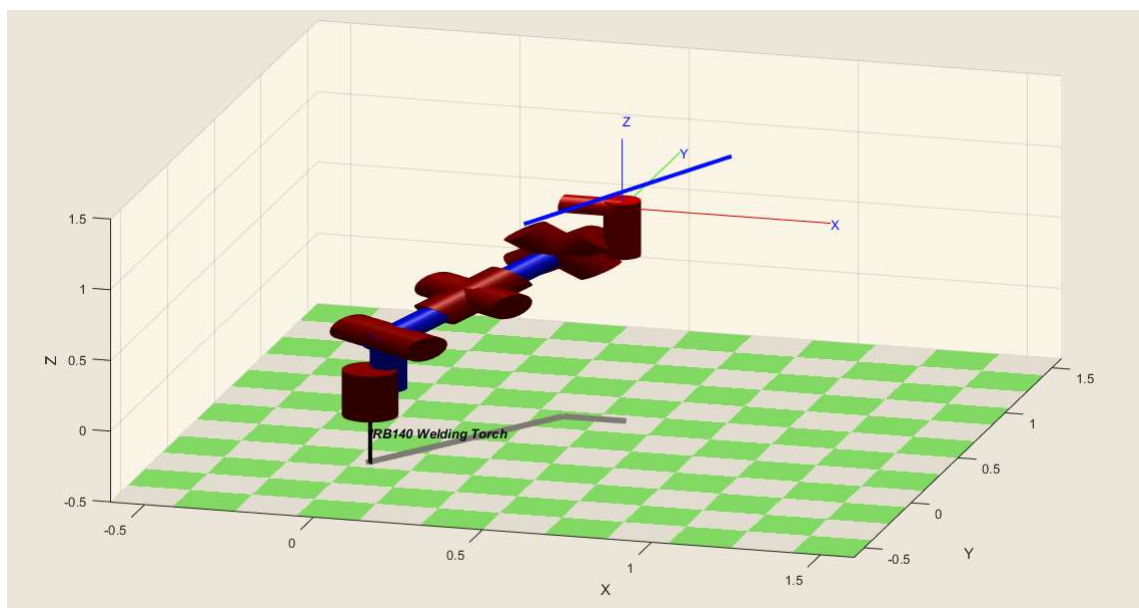
Para evaluar este punto se hace uso de la función 'jacob0' y se calcula su determinante.

```
% *****  
    DetJacobiano = det( robot.jacob0(q_i) );  
% *****
```

En vista que se encuentran 8 soluciones, o trayectorias distintas, tenemos que tomar medidas para elegir la que resulte más adecuada. Una de estas, es analizar en cada una de las soluciones el valor del determinante del Jacobiano, y si en alguna se tiene un valor menor a un límite, se lo envía al final de la matriz de soluciones, indicando que no es una buena solución. Finalmente, terminan quedando 8 soluciones ordenadas en orden creciente de la más adecuada a la menos.

Las siguientes graficas muestran un ejemplo de las singularidades presentes cuando se intenta alcanzar un punto que esta fuera de su espacio de trabajo. La línea azul representa la trayectoria deseada, mientras que la roja es la que sigue.

En la primera, se observa el robot extendido y no puede llegar al punto inicial. Luego, conforme se van cambiando de puntos comienza a seguir la trayectoria deseada desde el primer punto que le es posible alcanzar.





### 4.3. Límites articulares

Como medida de seguridad se limitan los movimientos de cada articulación a fin de evitar que los eslabones entren en contacto entre sí y pudieran dañarse, al igual que la a la herramienta. El fabricante impone los límites articulares, y supone que el robot se encuentra en una posición inicial tal que pueda moverse sólo entre estos límites. Los rangos dados son los siguientes:

Angular(°)	1	2	3	4	5	6
Inferior	180	110	50	200	120	400
Superior	-180	-90	-230	-200	-120	-400

Sin embargo, al crear el robot usando las librerías ya mencionadas anteriormente, no se tiene esa posición inicial que pueda cumplir con los rangos establecidos, por lo que al realizar la simulación los elementos se atraviesan. A causa de esto, tenemos que considerar un valor offset al iniciar con el proyecto a fin de lograr que cada articulación pueda moverse en el rango impuesto. Aunque, también se podría haber hecho que cada articulación tomara como valor cero un extremo, y el rango de movimiento fuera la suma, en valor absoluto, del valor mínimo y máximo permitido.

Para poner los límites articulares usamos la siguiente línea de código.

```
% LÍMITES de las articulaciones -----
qlimites = [deg2rad(-180) deg2rad(-100) deg2rad(-220) deg2rad(-200) deg2rad(-120) deg2rad(-400);
            deg2rad(180) deg2rad(100) deg2rad(60) deg2rad(200) deg2rad(120) deg2rad(400) ];
R.qlim = qlimites';
```

### Trayectoria seleccionada

Cuando el robot tiene que realizar un movimiento para alcanzar un punto ubicado dentro de su espacio de trabajo, puede darse el caso que puede llegar de más de una forma posible. Para seleccionar la más adecuada no solo se tienen en cuenta las limitaciones anteriores y el determinante del Jacobiano, sino que también se considera la que produce menos cambios abruptos en las articulaciones para producir el movimiento. Las que cumplen con estos requisitos son las que tienen mayor probabilidad de ser seleccionados como la mejor solución.

Para poder realizar la tarea de soldadura de la forma más adecuada, el robot debe cumplir con algunas características como continuidad, velocidad de soldadura, distancia al punto de soldado y ángulo de soldado.

En el programa “**Trayectoria(R, P1, P2, Rot, opción)**”, ubicado en el **Anexo E**, se selecciona la mejor de las trayectorias calculadas. El mismo recibe como argumentos el Robot, que en nuestro caso es el IRB140; la posición inicial; la final; y una matriz para poder calcular la rotación deseada en la herramienta durante su funcionamiento.

En cuanto a la selección de los puntos iniciales y finales, se exponen dos formas de obtenerlos. En la primera, el usuario elige los valores P1, P2, y los ángulos Roll, Pitch y Yaw a fin de armar la matriz de rotación. Mientras que, en la segunda, obtenemos esos valores de la matriz de transformación obtenida al aplicar la cinemática directa a los valores angulares deseados.

Dentro de la función ‘Trayectoria’ se calculan 8 soluciones, y haciendo algunas consideraciones, como el de la menor distancia articular y el determinante de

Jacobiano distinto de cero y mayor a un cierto valor, se puede seleccionar la mejor entre las soluciones calculadas.

Estas, en cierta medida están limitadas al valor deseado en las orientaciones que se le dé a la herramienta. Por lo que se podría dar el caso que sea posible alcanzar una posición dentro del espacio de trabajo para una orientación, pero no para otra.

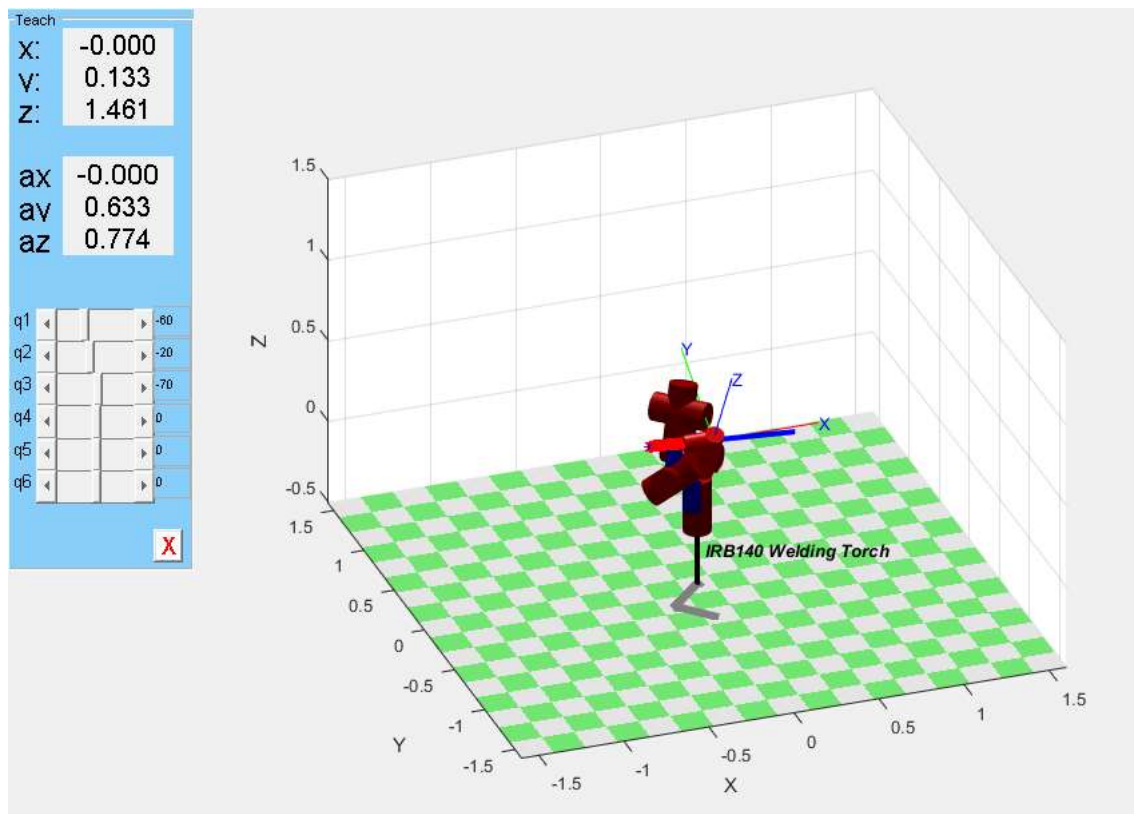
Una vez obtenida la trayectoria optima se procede a hacer la simulación. Para ello se usa la función '**AnimarRobot()**' ubicado en el **Anexo D.2**.

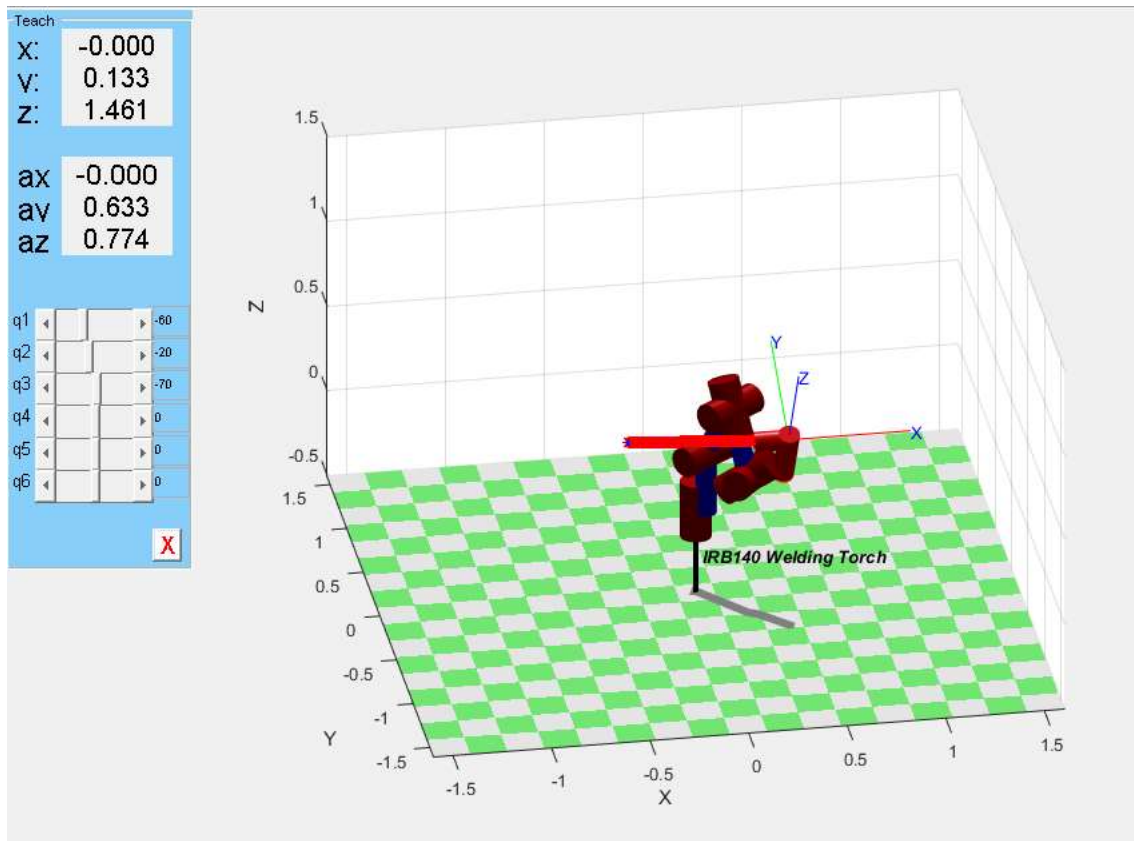
Los resultados obtenidos de aplicar estas últimas funciones mencionadas se muestran a continuación.

En primer lugar, se observan los parámetros usados para generar la trayectoria.

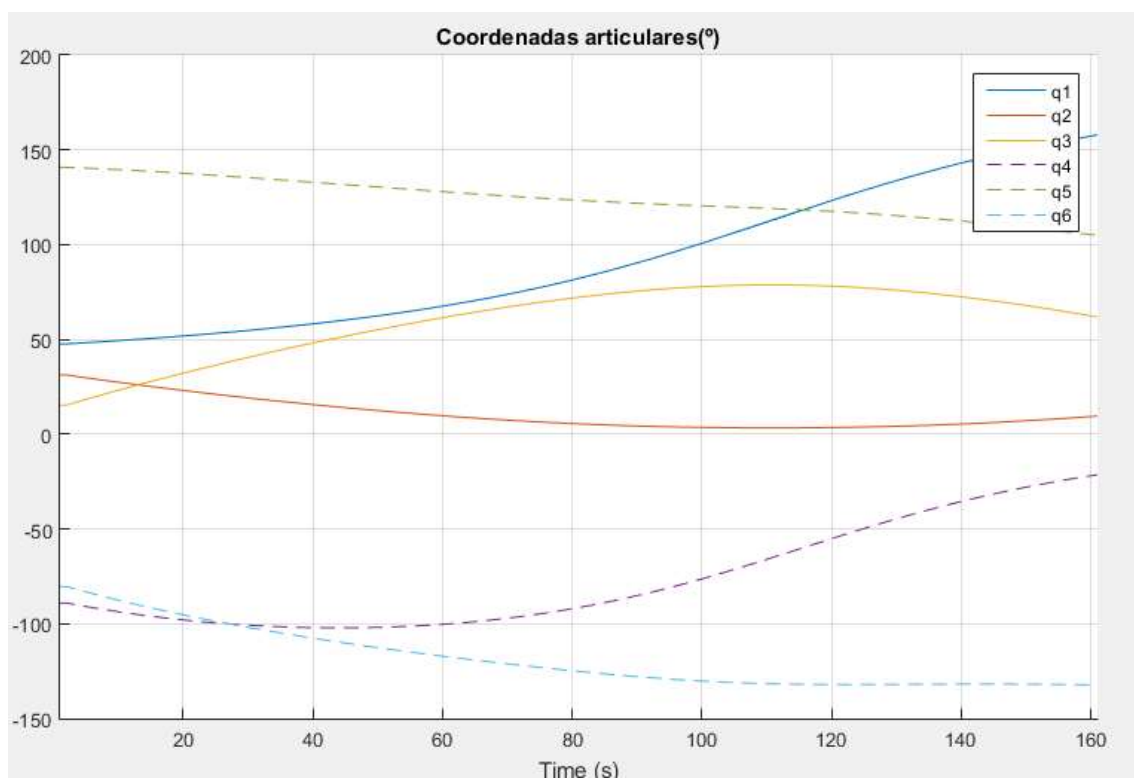
```
% ===== OPCION 1 =====  
Roll=pi/6;  
Pitch=0;  
Yaw=0;  
Rot=[Roll Pitch Yaw];  
P1=[-.4; -.35; .6]; %% POSICION INICIAL  
P2=[.4; -.45; .6]; %% POSICION FINAL  
[qs,Ps,singularidades]=Trayectoria(R,P1,P2,Rot,1);  
% =====
```

Ahora, se puede observar un punto intermedio durante el seguimiento de la mejor trayectoria, y luego la trayectoria completa. En azul se representa la consigna, y en rojo se observan las posiciones seguidas por el robot.





Finalmente, se muestran las posiciones articulares seguidas expresados en grados, donde se puede apreciar que ninguna tiene sobre picos y están dentro de los límites previamente definidos.



## 5. Conclusiones

Con el trabajo realizado concluimos que:

1. Las librerías Robotics Toolbox y Arte, ambos del software Matlab, resultaron ser herramientas muy valiosas para el estudio de robots industriales, permitiendo conocer de forma sencilla, efectiva y gráfica, los principios en los que se basa su control y simulación.
2. Tanto software, como el robot, son ampliamente conocidos, utilizados y accesibles en el mundo industrial y académico, han sido muy valiosos para introducirnos en el estudio y manejo de la robótica, permitiéndonos afianzar los conceptos visto en la materia.
3. Los fabricantes de ABB suministran la información suficiente para el estudio de sus robots. Esta incluye algunas cosas como los modelos CAD, parámetros propios de los eslabones y herramientas, como así también las velocidades límites de giro de las articulaciones y sus capacidades. Además, brindan la posibilidad de contactarse con su personal.
4. Al combinar las librerías Robotics y Arte se pudo hacer uso de los aportes de cada uno para poder llevar a cabo un buen desarrollo del proyecto. Por un lado, Arte brindaba mucha información en cuanto a funciones para posibilitar la simulación, control y entorno gráfico, entre otras cosas, pero al trabajar todo con estructuras y optimizar poco el entorno gráfico producía un poco de retardo en el momento de realizar las simulaciones. Por otro lado, Robotics Toolbox no ofrece información tan amplia de muchas marcas de robot, ni su control y cálculos, aunque se pueden obtener buenos resultados de los mismos, pero si optimiza al momento de realizar el entorno gráfico. Por lo que el proyecto pudo contar con funciones muy completas optimizadas al no trabajar tanto con estructuras, y al usar el entorno gráfico de Robotics.
5. Los resultados obtenidos al realizar el control cinemático son muy buenos, en donde el seguimiento de la trayectoria deseada. Cabe aclarar que el mismo no se tuvieron en cuenta los esfuerzos/torques aplicados en las articulaciones, al igual que los términos de inercia, rozamiento, términos gravitatorios, entre otros. Los mismos se verán en la materia Robótica 2.

### Líneas futuras de mejoras

Desarrollar mediante algún método de control, por ejemplo, mediante redes neuronales, en el cual el robot pueda aprender cuando se produzca la situación de tener que continuar avanzando en un sentido, pero ya está en su límite articular, y se produce un salto articular desde un extremo al otro dentro de los límites. De esta podría aprender de estas situaciones y no repetir las, e intentar buscar algún método para evitarlas.

## Bibliografía

- Fundamentos de Robótica. 2da edición. Antonio Barrientos.
- Apuntes de catedra 2016.
- Apuntes de catedra 2017.
- Robots y Sistemas Sensoriales. Fernando Torres.
- Robotics Toolbox. Corke.
- Robotics Vision and Control. Corke.
- Datasheet del soldador serie PKI.
- Datasheet robot IRB 140.
- Página de ABB:  
<http://new.abb.com/products/robotics/es/robots-industriales/irb-140>

## Anexos

### A. Código para cinemática directa

#### A.1. Cinemática Directa

```
function [Ttotal] = CinematicaDirecta(robot,q)
    % NOTA: T.tool es un objeto, por eso lo convierto para usarlo
    THerramienta = double(robot.tool);
    A_0 = double(robot.base);      % Por defecto es la identidad de 4x4
    T = A_0;
    for i=1:length(q),
        Tant = T*D_H(robot,q,i,-1);
        T = Tant;
    end
    Ttotal = T*THerramienta;

end
```

### B. Matriz de Denavit Hartenberg

```
function A = D_H(th, d, a, alpha)
    if(alpha == -1) % HAY QUE RESTARLE EL OFFSET
        irb = th;
        q = d + irb.offset;
        i = a;

        th = q(i);
        d = irb.d(i);
        a = irb.a(i);
        alpha = irb.alpha(i);

        A=[cos(th),-cos(alpha)*sin(th),sin(alpha)*sin(th),a*cos(th);
           sin(th),cos(alpha)*cos(th),-sin(alpha)*cos(th), a*sin(th);
           0,      sin(alpha),      cos(alpha),      d;
           0,      0,      0,      1];
    else if(alpha == 1) % NO HAY QUE RESTARLE EL OFFSET
        irb = th;
        q = d;
        i = a;

        th = q(i);
        d = irb.d(i);
        a = irb.a(i);
        alpha = irb.alpha(i);

        A=[cos(th),-cos(alpha)*sin(th),sin(alpha)*sin(th),a*cos(th);
           sin(th),cos(alpha)*cos(th),-sin(alpha)*cos(th), a*sin(th);
           0,      sin(alpha),      cos(alpha),      d;
           0,      0,      0,      1];
    else % INGRESO LOS 4 PARAMETROS Y CALCULA LA MATRIZ
        A=[cos(th),-cos(alpha)*sin(th),sin(alpha)*sin(th),a*cos(th);
           sin(th),cos(alpha)*cos(th),-sin(alpha)*cos(th), a*sin(th);
           0,      sin(alpha),      cos(alpha),      d;
           0,      0,      0,      1];
    end
end
```

## C. Código para cinemática inversa

### C.1. Cinemática Inversa

```
function q = CinematicaInversa(robot, T_07)
    q = zeros(6,8);
    THerramienta = double(robot.tool);
    T = T_07*inv(THerramienta);

    Z = T(1:3,3);
    L6 = robot.d(6);

    Px = T(1,4);
    Py = T(2,4);
    Pz = T(3,4);

    % Pm: posicion de la muñeca
    Pm = [Px Py Pz]' - L6*Z;
    % si q(1) es solucion, entonces q(1) + pi tambien es solucion
    q1 = atan2(Pm(2), Pm(1));
    q2_1 = Theta2(robot, [q1 0 0 0 0 0], Pm);
    q2_2 = Theta2(robot, [q1+pi 0 0 0 0 0], Pm);
    q3_1 = Theta3(robot, [q1 0 0 0 0 0], Pm);
    q3_2 = Theta3(robot, [q1+pi 0 0 0 0 0], Pm);

    q = [ q1,      q1,      q1,      q1,      q1+pi, q1+pi, q1+pi, q1+pi;
          q2_1(1),q2_1(1),q2_1(2),q2_1(2),q2_2(1),q2_2(1),q2_2(2),q2_2(2);
          q3_1(1),q3_1(1),q3_1(2),q3_1(2),q3_2(1),q3_2(1),q3_2(2),q3_2(2);
          0,      0,      0,      0,      0,      0,      0,      0;
          0,      0,      0,      0,      0,      0,      0,      0;
          0,      0,      0,      0,      0,      0,      0,      0];

    % se dejan las partes reales+
    q = real(q);

    % q3 tiene un rango no simetrico, por eso se evita la
    % normalizacion de -pi a pi como en todos los otros qi

    q(1,:) = Norma(q(1,:));
    q(2,:) = Norma(q(2,:));

    % 3 ultimas articulaciones para las combinaciones de las primeras 3
    for i=1:2:size(q,2)
        % Metodo algebraico/ Metodo geometrico
        qtemp = Pieper(robot, q(:,i), T, 1); % codo arriba
        qtemp(4:6) = Norma(qtemp(4:6));
        q(:,i) = qtemp;

        qtemp = Pieper(robot, q(:,i), T, -1); % codo abajo
        qtemp(4:6) = Norma(qtemp(4:6));
        q(:,i+1) = qtemp;
    end

    % q = q - robot.offset' * ones(1,8);
    q = Norma(q);
end
```

## C.2. Theta2

```
function q2 = Theta2(robot, q, Pm)

L2 = robot.a(2);
L3 = robot.d(4);

T01 = D_H(robot,q,1,1);

% Se expresa Pm en referencia al sistema 1
p1 = inv(T01)*[Pm ; 1];
r = sqrt(p1(1)^2 + p1(2)^2);

beta = atan2(-p1(2), p1(1));
gamma = (acos((L2^2 + r^2 - L3^2)/(2*r*L2)));

if ~isreal(gamma)
    disp('WARNING: punto para q2 no alcanzable, soluciones imaginarias');
end

q2(1) = - beta - gamma;% + pi/2; % codo arriba
q2(2) = - beta + gamma;% + pi/2; % codo abajo
end
```

## C.3. Theta3

```
function q3 = Theta3(robot, q, Pm)

L2 = robot.a(2);
L3 = robot.d(4);

T01 = D_H(robot, q, 1,1);

p1 = inv(T01)*[Pm; 1];
r = sqrt(p1(1)^2 + p1(2)^2);
eta = (acos((L2^2 + L3^2 - r^2)/(2*L2*L3)));

if ~isreal(eta)
    disp('WARNING:punto para q3 no alcanzable, soluciones imaginarias');
end

q3(1) = pi/2 - eta;
q3(2) = eta - 3*pi/2;
end
```



#### C.4. Pieper

```
function q = Pieper(robot, q, T, wrist, method)

A01 = D_H(robot, q, 1,1);
A12 = D_H(robot, q, 2,1);
A23 = D_H(robot, q, 3,1);
Q = inv(A23)*inv(A12)*inv(A01)*T;
% Si q5 = 0 -> hay infinitas soluciones. Por eso es que usa un thresh
thresh = 1e-12;

if abs(Q(3,3)-1)>thresh
    if wrist==1
        q(4)=atan2(Q(2,3),Q(1,3));
        q(6)=atan2(Q(3,2),-Q(3,1));
    else
        q(4)=atan2(Q(2,3),Q(1,3))+pi;
        q(6)=atan2(Q(3,2),-Q(3,1))+pi;
    end

    if abs(cos(q(6)+q(4)))>thresh
        cq5=(-(-Q(1,1)-Q(2,2))/cos(q(4)+q(6))-1;
    end

    if abs(sin(q(6)+q(4)))>thresh
        cq5=-(Q(1,2)-Q(2,1))/sin(q(4)+q(6))-1;
    end

    if abs(sin(q(6)))>thresh
        sq5=-Q(3,2)/sin(q(6));
    end

    if abs(cos(q(6)))>thresh
        sq5=Q(3,1)/cos(q(6));
    end
    q(5)=atan2(sq5,cq5);
else
    if wrist==1
        q(4)=0;
        q(5)=0;
        q(6)=atan2(Q(1,2)-Q(2,1),-Q(1,1)-Q(2,2));
    else
        q(4)=-pi;
        q(5)=0;
        q(6)=atan2(Q(1,2)-Q(2,1),-Q(1,1)-Q(2,2))+pi;
    end

end

end

end
```

#### C.5. Norma

```
% Normaliza de -pi a pi
function q = Norma(q)
    for i=1:size(q,2)
        q(:,i)=atan2(sin(q(:,i)),cos(q(:,i)));
    end
end
```

## D. Código para planificación de trayectorias

### D.1. Trayectoria

```
% DEVUELVE LAS OCHO TRAYECTORIAS SOLUCION, DONDE LA 1ra ES LA MEJOR,
% LUEGO VIENE LA SEGUNDA MEJOR, Y ASI... ---> qs( qi, Npuntos, 8 sol)
% Ps: vector de puntos que recorre el robot ---> Ps(3, Npuntos)
% singularidades: solo indica si alguna solucion tiene un punto singular
% actualmente no lo estamos usando
function [qs,Ps,singularidades] = Trayectoria(robot,P1,P2,Rot, opcion)
singularidades=zeros(8,1);
% ----- Calculamos el vector director de la recta y normalizamos
Pd=P2-P1;
dist=0.005;

% ---Discretizamos los vectores y los guardamos en una matriz -----
Psx=linspace(P1(1),P2(1),norm(Pd)/dist);
Psy=linspace(P1(2),P2(2),norm(Pd)/dist);
Psz=linspace(P1(3),P2(3),norm(Pd)/dist);
Ps=[Psx;Psy;Psz];
% printstring=sprintf('Distancia real del trazado: %f\n',(norm(Ps(:,2)-
Ps(:,1)))));
% disp(printstring);

% ----- Tomamos los valores de Roll, Pitch y Yaw deseados -----
if opcion == 1
    Roll=Rot(1);
    Pitch=Rot(2);
    Yaw=Rot(3);
    % ----- armamos las matrices de Roll Pitch Yaw -----
    Rm=[1 0 0;
        0 cos(Roll) -sin(Roll);
        0 sin(Roll) cos(Roll)];

    Pm=[cos(Pitch) 0 sin(Pitch);
        0 1 0;
        -sin(Pitch) 0 cos(Pitch)];

    Ym=[cos(Yaw) -sin(Yaw) 0;
        sin(Yaw) cos(Yaw) 0;
        0 0 1];
    % ----- armamos la matriz de rotacion -----
    R = Ym*Pm*Rm;
elseif opcion == 2
    R = Rot;
end

q = zeros(6,size(Ps,2),8); % SOLUCIONES (6 art, N puntos, 8 soluc )
T=[R,Ps(:,1);0,0,0,1];
qprovisorio=CinematicaInversa(robot,T); % TOMO UN PUNTO INICIAL

for i=1:1:size(q,3)
    q(:,1,i)= qprovisorio(:,i);
end

for i=2:1:size(q,2) % N PUNTOS
    T=[R,Ps(:,i-1);0,0,0,1];
    qprovisorio=CinematicaInversa(robot,T); % Calculo La Ci para cada punto

    for j=1:1:size(q,3) % 8 SOLUCIONES
        distmin=1000;
```

```

        for k=1:1:8
            DistanciaArticular=norm(qprovisorio(:,k)-q(:,i-1,j));
            if distmin>DistanciaArticular
                next=k;
                distmin=DistanciaArticular;
                if 0.001<abs( det(robot.jacob0(transpose(qprovisorio(:,k))))
                    );
                    next=k;
                    distmin=DistanciaArticular;
                end
            end
        end
        q(:,i,j)=qprovisorio(:,next); % ARMO SOLUCIONES CON LA MENOR DISTANCIA
        ARTICULAR
    end

end

% ----- CALCULO LAS DISTACIAS TOTALES RECORRIDAS -----
DistanciaTotal=zeros(size(q,3),1);
for i=1:1:size(q,3)
    for j=2:1:size(q,2)
        DistanciaArticular=norm(q(:,j,i)-q(:,j-1,i));
        DistanciaTotal(i)=DistanciaTotal(i)+DistanciaArticular;
    end
end

qs=zeros(6,size(Ps,2),8);
for i=1:1:size(q,3)
    [~,m]=min(DistanciaTotal);
    qs(:,i)=q(:,m);
    DistanciaTotal(m)=10000000;
end

%Reordeno enviando las soluciones con jacobiano con valor abs <0.005 al
%final de un vector auxiliar qss, y manteniendo las soluciones con
%determinante jacobiano >0.005 al inicio, valor 0.005 de toolbox
inicio=1;
final=8;
qss=qs*0;
for i=1:1:size(q,3)
    minJacobiano=1000;
    for j=2:1:size(q,2)
        Jacobiano=robot.jacob0(transpose(qs(:,j,i)));
        DetJacobiano=det(Jacobiano);
        if minJacobiano>abs(DetJacobiano)
            minJacobiano=abs(DetJacobiano);
        end
    end
    if minJacobiano<0.005
        qss(:,final)=qs(:,i);
        singularidades(final)=1;
        final=final-1;
    else
        qss(:,inicio)=qs(:,i);
        inicio=inicio+1;
    end
end
end
%Asigno el valor de qss a qs
qs=qss;
end

```

## D.2. AnimarRobot

```
function [] = AnimarRobot(robot,qs,Ps,ws)

    qs=transpose(qs);
    %imprimimos
    robot.plot(qs(1,:), 'workspace',ws, 'scale',0.9, 'delay',0.005);
    for i=2:1:size(qs,1)
        robot.animate(qs(i,:));
        T = double (robot.fkine(qs(i,:)))
        %CinematicaDirecta(robot,qs(i,:));
        hold on
        plot3(T(1,4),T(2,4),T(3,4), '*r');
        plot3(Ps(1,:),Ps(2,:),Ps(3,:), 'b');
        hold off
    end
end
```

## D.3. Space

```
function Space(robot,option)
    x=[]; y=[]; z=[];
    hold on;
    if option==1
        jump=pi/180*20;
        for q1=robot.qlim(1,1):jump:robot.qlim(1,2)
            for q2=robot.qlim(2,1):jump:robot.qlim(2,2)
                for q3=robot.qlim(3,1):jump:robot.qlim(3,2)
                    for q5=robot.qlim(5,1):jump:robot.qlim(5,2)
                        q=[q1 q2 q3 0 q5 0];
                        qprint=CinematicaDirecta(robot,q);
                        x=[x,qprint(1,4)]; y=[y,qprint(2,4)]; z=[z,qprint(3,4)];
                    end
                end
            end
        end
    end
    plot3(x,y,z, 'g. ');
    view(0,0);
end
```

## E. PROYECTO\_ROBOTICA1

```
% -----  
%                               VERSION 10.1 DE Robotics Toolbox( Peter Corke)  
% -----  
clc, clear, close all,  
% ---- DIMENSIONES -----  
d1 = 0.352;  
a1 = 0.070;  
a2 = 0.360;  
d4 = 0.380;  
d6 = 0.065; %  
  
% ----- PARAMETROS DH -----  
%   theta  d  a  alpha  R/T  (0:R / 1:T)  
d_h = [0.0    d1    a1   -pi/2  0;  
        0.0    0.0   a2    0.0  0;    % theta -pi/2  
        0.0    0.0   0.0  -pi/2  0;  
        0.0    d4    0.0   pi/2  0;  
        0.0    0.0   0.0  -pi/2  0;  
        0.0    d6    0.0    0.0  0];  
  
% ----- OBJETO -----  
R = SerialLink(d_h, 'name','IRB140 Welding Torch');  
  
% ----- HERRAMIENTA -----  
L = 0.295; % largo  
H = 0.112; % alto  
R.tool = transl([-H, 0, L])*troty(-pi/4);%* trotx(pi/3); % ES UN OBJETO  
  
% OFFSET en articulaciones -----  
R.offset = [5*pi/6 1.6*pi 1.9*pi pi 0.1 -pi];  
  
% LIMITES de las articulaciones -----  
Qlimites = [deg2rad(-180) deg2rad(-100) deg2rad(-220) deg2rad(-200)  
deg2rad(-120) deg2rad(-400);  
            deg2rad(180) deg2rad(100) deg2rad(60) deg2rad(200)  
deg2rad(120) deg2rad(400) ];  
R.qlim = Qlimites';  
  
% POSICION A MOSTRAR -----  
qpos=[-60*2*pi/360 -20*2*pi/360 -70*2*pi/360 0 0 0];  
  
% ESPACIO DE TRABAJO (-x, +x, -y, +y, -z, +z) -----  
ws = [-1.5, 1.5,-1.5, 1.5, -0.5 1.5];  
  
% ----- GRAFICAR -----  
%R.plot3d(qpos,'path',pwd,'workspace',ws)  
R.plot(qpos,'workspace',ws,'scale',0.9)  
  
% PANEL para mover articulaciones:  
R.teach('approach')  
  
% =====  
% CINEMATICA (SOLO PARA MOSTRAR QUE LOS VALORES CINEMATICOS COINCIDEN)  
% =====  
% Ttotal = CinematicaDirecta(R,qpos); % SUMAR OFFSET  
% disp('Ttotal: '), disp(Ttotal);  
% qinvSoluciones = CinematicaInversa(R,Ttotal); % RESTAR OFFSET  
% disp('qinvSoluciones: '), disp(qinvSoluciones), disp('');  
% disp('Posiciones deseadas: '),disp(qpos),  
% disp('Offset: '),disp(R.offset),  
% disp('Qlim: '),disp(R.qlim)
```

```
% ===== GRAFICAR EL ESPACIO DE TRABAJO =====
% Space(R,1);

% ===== SELECCION DE LOS PUNTOS PARA REALIZAR LA TRAYECTORIA
% TENEMOS DOS OPCIONES:
% 1RA: SELECCIONAMOS P1, P2 Y ROLL, PITCH, YAW
% 2DA: PARA UN Q 'SEGURO' LE SACAMOS LA CD Y DE AHI OBTENEMOS EL PUNTO 1 Y
% LA ORIENTACION DESEADA, LUEGO SE REPITE PARA EL PUNTO 2

% ===== OPCION 1 =====
Roll=pi/6;
Pitch=0;
Yaw=0;
Rot=[Roll Pitch Yaw];
P1=[-.4; -.35; .6]; %% POSICION INICIAL
P2=[.4; -.45; .6]; %% POSICION FINAL
[qs,Ps,singularidades]=Trayectoria(R,P1,P2,Rot,1);
% =====

% ===== OPCION 2 =====
% qposP1 = [deg2rad(-60)    deg2rad(55.1)    deg2rad(-140)    deg2rad(-159)
deg2rad(46.5)    deg2rad(0) ];
% qposP2 = [deg2rad(-107)    deg2rad(55.1)    deg2rad(-123)    deg2rad(-118)
deg2rad(24)    deg2rad(0) ];
% TpuntoP1 = CinematicaDirecta(R,qposP1);
% TpuntoP2 = CinematicaDirecta(R,qposP2);
% =====

%
[qs,Ps,singularidades]=Trayectoria(R,TpuntoP1(1:3,4),TpuntoP2(1:3,4),TpuntoP1(
1:3,1:3), 2);
% qs( qi, Npuntos, 8 soluciones)  ->> 8 trayectorias soluciones
% Ps: vector de puntos que recorre el robot --> Ps(3, Npuntos)

% NOTA: HASTA ESTE MOMENTO YA TENEMOS LAS 8 SOLUCIONES, DONDE LA 1era ES LA
% MEJOR DE TODAS, DONDE SE HA CONSIDERADO LAS DISTANCIAS Y EL JACOBIANO.
% COMO UNA ULTIMA ADICION, DE LAS 8 SOLUCIONES SE ARMA UNA EN LA QUE TOMAN
% LAS MENORES DISTANCIAS. ESTO ES LO QUE SE GRAFICA A CONTINUACION.
% LUEGO, SE PUEDE GRAFICAR LA MEJOR OBTENIDA CON "Trayectoria", O TODAS
Q = zeros(size(Ps,2), 6);
% *-*-*-*-* GRAFICO LA SOLUCION ARMADA A PARTIR DE LAS 8 *-*-*-*-*
% *-*-*-*-*
% Con las 8 soluciones se
% indice = zeros(size(Ps,2),1);
% indice(1) = 1; % ELIJO EL PRIMER PUNTO COMO PUNTO INICIAL
% Q(1,:) = qs(:,1,indice(1));
%
% for i=2:size(Ps,2)
%     distmenor = 10000;
%     distInterm = distmenor;
%     indice(i) = -1;
%
%     for j = 1:8
%         distInterm = norm( Q(i-1,:) - qs(:,i,j)');
%         if distInterm < distmenor
%             distmenor = distInterm;
%             indice(i) = j;
%         end
%     end
%     indice
%     Q(i,:) = qs(:,i,indice(i));%EL SIGUIENTE PUNTO ES EL QUE TIENE
%         % MENOR DISTANCIA A LAS 8 SIGUIENTES POSIBLES SOLUCIONES
% end
```



[illegible]