Tesis Robotica

Grupo N°:		**Año y División:**	**2010 - R 6055**
Integrantes:		**1 – Bobbio Matias** **2 - GONZALEZ, Maximiliano Ariel**	
		Fecha:26/08/2010	
Título de la Tesis:		**RT4-PUMA** **Cinematica** **Dinamica** **Compilador**	

.....
Firma y aclaración Representante

INDICE

INTRODUCCION GENERAL

El estudio cinemática del brazo robótica se ocupa de la geometría del movimiento del brazo robot con respecto a un sistema de coordenadas de referencia fijo como una función del tiempo sin tener en cuenta a las fuerzas / momentos que causan el movimiento. Por lo tanto trata la configuración espacial del robot como una función del tiempo, en particular las relaciones entre el espacio de la articulación variable y la posición y orientación de un brazo robot. El problema de la cinemática consiste habitualmente en dos subproblemas, la cinemática inversa y directa:

El problema de la cinemática directo consiste en encontrar la posición y orientación del actuador final de un manipulador con respecto a un sistema de coordenadas de referencia, dado el vector de ángulos de las articulaciones del brazo del robot. $\vartheta = (\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6)^t$

El problema de la cinemática inversa (o arm solution) es calcular el vector de ángulos de las articulaciones ϑ dada la posición y orientación del actuador final con respecto al sistema de coordenadas de referencia. Dado que las variables independientes en un brazo del robot son los ángulos de las articulaciones, y una tarea suele expresarse en términos de la base o el sistema de coordenadas del mundo, la solución cinemática inversa se utiliza con mayor frecuencia en las aplicaciones informáticas. El resultado de la cinemática directa en una matriz de 4 x 4 de transformación homogénea que se relaciona con la configuración espacial entre enlaces vecinos. Estas matrices de transformación homogéneas son útiles para derivar las ecuaciones dinámicas de movimiento del brazo robot.

La dinámica de los brazos robóticas trabaja con la formulación matemática de ecuaciones del movimiento del brazo robot. Las ecuaciones dinámicas del movimiento del manipulador son un set de ecuaciones describiendo el comportamiento dinámico del manipulador. Tales ecuaciones del movimiento son útiles para simulaciones por computadoras, el diseño adecuado de ecuaciones para el control del brazo, y la evaluación del diseño cinemático y estructural del mismo.

Representación de Denavit-Hartenberg. Para describir las relaciones de traslación y rotación entre enlaces (links) adyacentes, Denavit y Hartenberg propusieron un método matricial para establecer de forma sistemática un sistema de coordenadas (body-attached frame) para cada enlace de una cadena articulada. La representación de D-H resulta en una matriz transformación homogénea de 4x4 representando el sistema de coordenadas de cada enlace con respecto al sistema de coordenadas del enlace previo. Así a trabes de transformaciones secuénciales el actuador final expresado en "coordenadas de la mano" puede ser transformado y expresado en "coordenadas de la base", que compone el sistema de referencia inercial de este sistema dinámico.

Las seis matrices A_{i-1}^{i} de transformación para un robot PUMA están basadas en el sistema de coordenadas establecido en la figura 5. Estas matrices A_{i-1}^{i} están listadas en la figura 6.

$$A_{i-1}^{i} = \begin{bmatrix} C\vartheta_i & -C\alpha_i\, S\vartheta_i & S\alpha_i\, S\vartheta_i & a_i\, C\vartheta_i \\ S\vartheta_i & C\alpha_i\, C\vartheta_i & -S\alpha_i\, C\vartheta_i & a_i\, S\vartheta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_0^1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^3 = \begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3^4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4^5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5^6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 6. PUMA coordinate transformation matrices.

CINEMATICA DIRECTA

Ecuaciones cinemáticas para manipuladores

La matriz homogénea T_0^i, la cual especifica la posición y orientación del punto final del enlace i con respecto al sistema de coordenadas base, es el producto en cadena de sucesivas matrices transformación A_{i-1}^i, expresada como:

$$\mathbf{T}_o^i = \mathbf{A}_o^1\mathbf{A}_1^2 \ldots \mathbf{A}_{i-1}^i = \prod_{j=1}^{i} \mathbf{A}_{j-1}^j \; ; \text{for } i = 1,2 \ldots n$$
$$= \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_o^i & \mathbf{p}_o^i \\ 0 & 1 \end{bmatrix}$$

Específicamente para i=6,obtenemos la matriz T, $T = A_0^6$ la cual especifica la posición y orientación del punto final del manipulador con respecto al sistema de coordenadas base. Esta matriz T es usada tan frecuentemente en la cinemática de brazos robóticas, que es llamada “matriz del brazo”. Considere la matriz T como:

$$\mathbf{T} = \begin{bmatrix} \mathbf{x}_6 & \mathbf{y}_6 & \mathbf{z}_6 & \mathbf{p}_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde (Figura 8):

n = el vector normal de la mano. Suponiendo paralela la mordaza de la mano, que es ortogonal a los dedos del brazo robot.
s = el vector de deslizamiento de la mano. Se apunta en la dirección del movimiento del dedo, como se abre la pinza y se cierra.
a = el vector de enfoque de la mano. Se apunta en la dirección normal a la palma de la mano (normal a la herramienta placa de montaje del brazo).
p = el vector de posición de la mano. Este apunta desde el origen del sistema de coordenadas base, el cual usualmente esta ubicado en el punto central de los dedos cerrados totalmente.

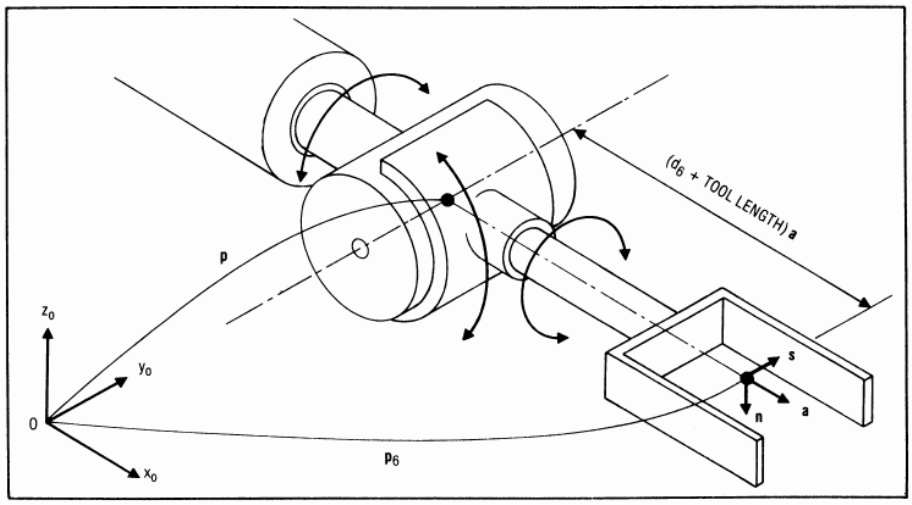


Figure 8. Hand coordinate system and [n, s, a].

La solución de la cinemática directa, es simplemente cuestión de calcular $T = A_0^6$ mediante la multiplicación en cadena de las seis A_{i-1}^i matrices o mediante la evaluación de cada elemento en la matriz T. La matriz T para el brazo PUMA se muestra en la figura 5.

$$\mathbf{T} = \mathbf{A}_o^1 \cdot \mathbf{A}_1^2 \cdot \mathbf{A}_2^3 \cdot \mathbf{A}_3^4 \cdot \mathbf{A}_4^5 \cdot \mathbf{A}_5^6 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

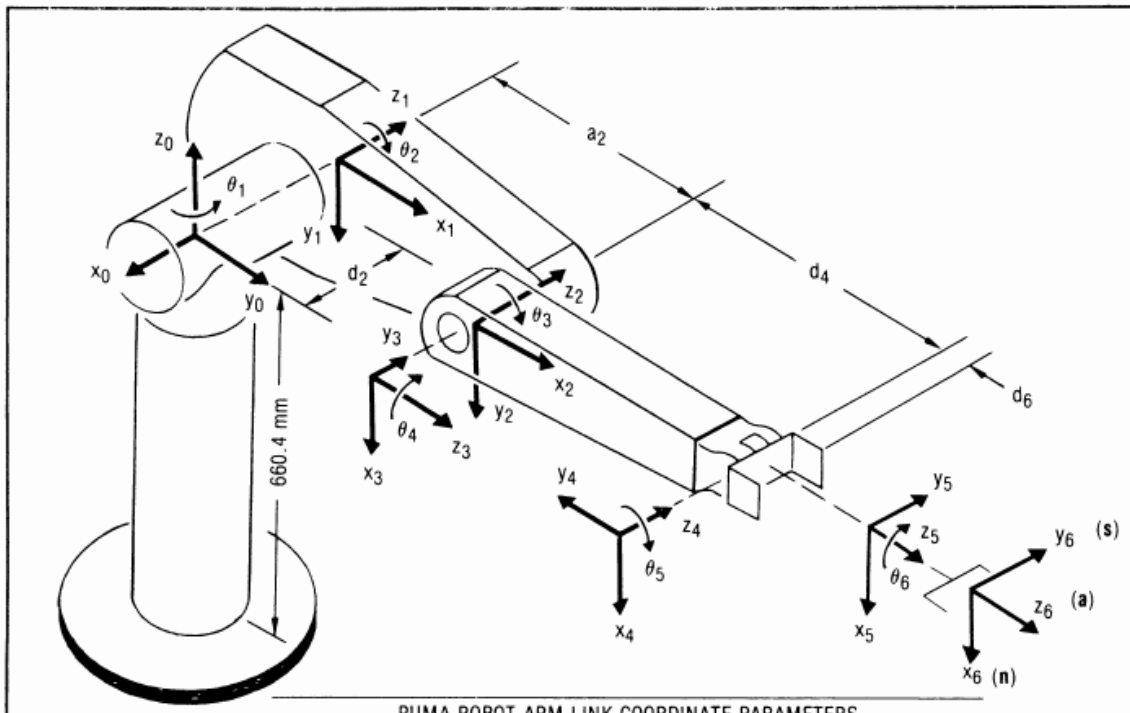
Donde:

$$\begin{aligned} n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1\,[S_4C_5C_6 + C_4S_6] \\ n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1[S_4C_5C_6 + C_4S_6] \\ n_z &= -S_{23}[C_4C_5C_6 - S_4S_6] - C_{23}S_5C_6 \end{aligned} \tag{27}$$

$$\begin{aligned} s_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\ &\quad - S_1[-S_4C_5S_6 + C_4C_6] \\ s_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\ &\quad + C_1[-S_4C_5S_6 + C_4C_6] \\ s_z &= S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \end{aligned} \tag{28}$$

$$a_x = C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5$$
$$a_y = S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5$$
$$a_z = -S_{23}C_4S_5 + C_{23}C_5$$

$$p_x = C_1(d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_2C_2) - S_1(d_6S_4S_5 + d_2)$$
$$p_y = S_1(d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_2C_2) + C_1(d_6S_4S_5 + d_2)$$
$$p_z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_2S_2$$



PUMA ROBOT ARM LINK COORDINATE PARAMETERS

JOINT i	ϑ_i	α_i	a_i	d_i	RANGE
1	90	−90	0	0	−160 to +160
2	0	0	432 mm	149.5 mm	−225 to +45
3	90	90	0	0	−45 to +225
4	0	−90	0	432.0	−110 to +170
5	0	90	0	0	−100 to +100
6	0	0	0	56.5	−266 to +266

Figure 5. Establishing link coordinate systems for a PUMA robot.

Detalle del desarrollo de la matriz transformación homogénea:

Metodo de Denavit-Hartenberg

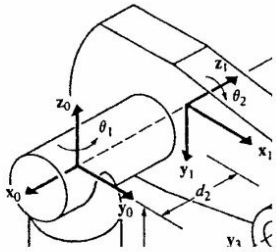
D-H 1) Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.
D-H 2) Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.
D-H 3) Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
D-H 4) Para i de 0 a n-1 situar el eje zi sobre el eje de la articulación i+1.
D-H 5) Situar el origen del sistema de la base {S0} en cualquier punto del eje z0 .Los ejes x0 e y0 se situarán de modo que formen un sistema dextrógiro con z0.
D-H 6) Para i de 1 a n-1 situar el sistema {Si}(solidario al eslabón i) en la intersección del eje zi con la línea normal común a zi-1 y zi .Si ambos ejes se cortasen se situaría {Si} en el punto de corte. Si fuesen paralelos {Si} se situaría en la articulación i + 1.
D-H 7) Situar xi en la línea normal común a zi-1 y zi.
D-H 8) Situar yi de modo que forme un sistema dextrógiro con xi y zi.
D-H 9) Situar el sistema {Sn} en el extremo del robot de modo que zn, coincida con la dirección de zn-1 y xn sea normal a zn-1 y zn.
D-H 10) Obtener i como el ángulo que hay que girar en torno a zi-1 para que xi-1 y xi queden paralelos.
D-H 11) Obtener di, como la distancia, medida a lo largo de zi-1, que habría que desplazar {Si-1} para que xi y xi-1 quedasen alineados.
D-H 12) Obtener ai como la distancia medida a lo largo de xi (que ahora coincidiría con xi-1) que habría que desplazar el nuevo {Si-1} para que su origen coincidiese con {Si}.
D-H 13) Obtener i como el ángulo que habría que girar entorno a xi, (que ahora coincidiría con xi-1),para que el nuevo {Si-1} coincidiese totalmente con {Si}.
D-H 14) Obtener las matrices de transformación i-1 Ai definidas anteriormente.
D-H 15) Obtener la matriz de transformación T que relaciona el sistema de la base con el del extremo del robot.
D-H 16) La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.
Los cuatro parámetros de D-H (i, di, ai, i) dependen únicamente de las características geométricas de
cada eslabón.

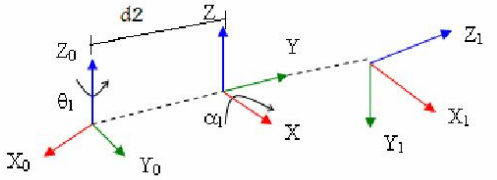
• Matriz de Transformación elemental de Denavit-Hartenberg

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

Procedimiento para el PUMA

Variables de articulación de la articulación 0 a la 1





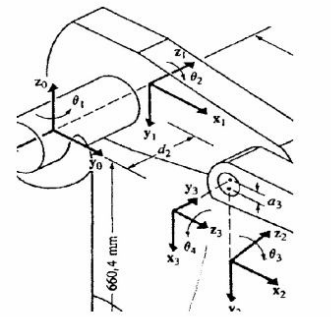
Rotación en Z	θ_i	90º
Traslación en Z	d_i	d2
Traslación en X	a_i	0
Rotación en X	α_i	-90º

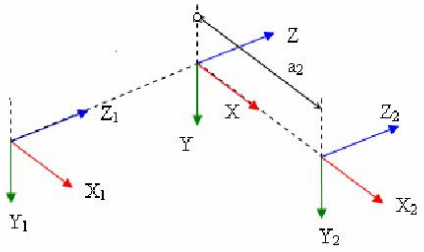
d_2 un desplazamiento que nos sitúa en el eje del enlace J_2

Matriz de transformación de la articulación 0 a la articulación1:

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{0}A_1 = \begin{pmatrix} \cos\theta 1 & 0 & -\sin\theta 1 & 0 \\ \sin\theta 1 & 0 & \cos\theta 1 & 0 \\ 0 & -1 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Variables de articulación de la articulación 1 a la 2





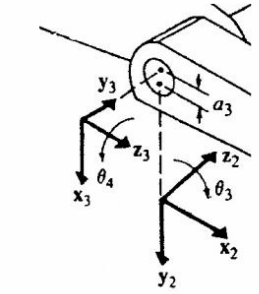
Rotación en Z	θ_i	0º
Traslación en Z	d_i	0
Traslación en X	a_i	a_2
Rotación en X	α_i	0º

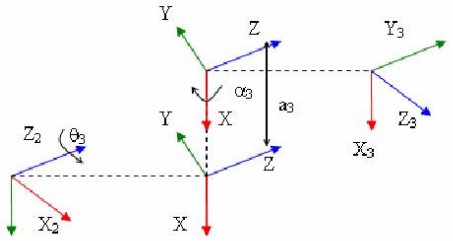
a_2 nos lleva hasta el eje de la articulación J_3

Matriz de transformación de la articulación 1 a la articulación2

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{1}A_2 = \begin{pmatrix} \cos\theta 2 & -\sin\theta 2 & 0 & a2\cos\theta 2 \\ \sin\theta 2 & \cos\theta 2 & 0 & a2\sin\theta 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Variables de articulación de la articulación 2 a la 3





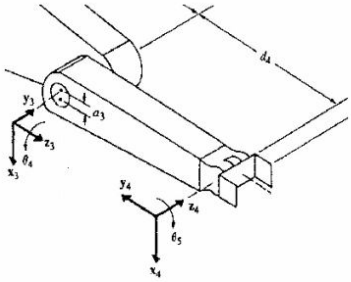
Rotación en Z	θ_i	90º
Traslación en Z	d_i	0
Traslación en X	a_i	a_3
Rotación en X	α_i	90º

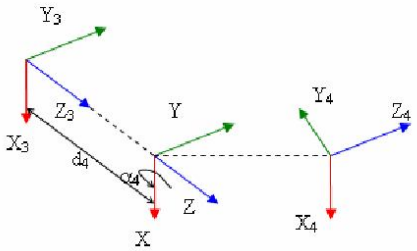
a_3 nos sitúa el eje Z en el giro de la muñeca

Matriz de transformación de la articulación 2 a la articulación3

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{2}A_3 = \begin{pmatrix} \cos\theta 3 & 0 & \sin\theta 3 & a3\cos\theta 3 \\ \sin\theta 3 & 0 & -\cos\theta 3 & a3\cos\theta 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Variables de articulación de la articulación 3 a la 4





Rotación en Z	θ_i	0º
Traslación en Z	d_i	d_4
Traslación en X	a_i	0
Rotación en X	α_i	-90º

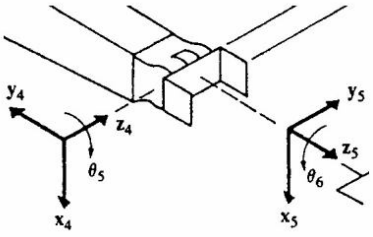
d_4 es un desplazamiento que nos sitúa en el enlace L_4

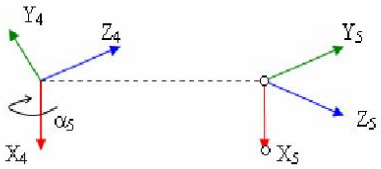
Matriz de transformación de la articulación 3 a la articulación 4

Rotación en Z	θ_i	0º
Traslación en Z	d_i	d_4
Traslación en X	a_i	0
Rotación en X	α_i	-90º

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{3}A_4 = \begin{pmatrix} \cos\theta 4 & 0 & -\sin\theta 4 & 0 \\ \sin\theta 4 & 0 & \cos\theta 4 & 0 \\ 0 & -1 & 0 & d4 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Variables de articulación de la articulación 4 a la 5





Rotación en Z	θ_i	0º
Traslación en Z	d_i	0
Traslación en X	a_i	0
Rotación en X	α_i	90º

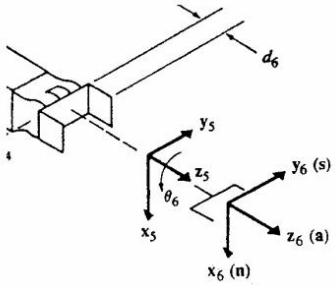
$X_4Y_4Z_4$ y $X_5Y_5Z_5$ tienen el mismo origen

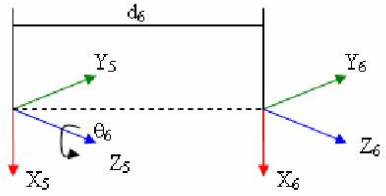
Matriz de transformación de la articulación 4 a la articulación 5

Rotación en Z	θ_i	0º
Traslación en Z	d_i	0
Traslación en X	a_i	0
Rotación en X	α_i	90°

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{4}A_5 = \begin{pmatrix} \cos\theta 5 & 0 & \sin\theta 5 & 0 \\ \sin\theta 5 & 0 & -\cos\theta 5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Variables de articulación de la articulación 5 al efector final





Rotación en Z	θ_i	0º
Traslación en Z	d_i	d_6
Traslación en X	a_i	0
Rotación en X	α_i	0°

El giro θ_6 y el desplazamiento d_6 nos llevan al centro de la pinza (SdR final)

Matriz de transformación de la articulación 5 al efector final

Rotación en Z	θ_i	0º
Traslación en Z	d_i	d_6
Traslación en X	a_i	0
Rotación en X	α_i	0°

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow {}^{5}A_6 = \begin{pmatrix} \cos\theta 6 & -\sin\theta 6 & 0 & 0 \\ \sin\theta 6 & \cos\theta 6 & 0 & 0 \\ 0 & 0 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Obtención de la matriz Homogénea con MATLAB

```
%> Script de Matlab
%> Calculo la matriz Homogénea de la arquitectura PUMA

clear all;
clc;

syms d1;syms d2;syms d3;syms d4;syms d5;syms d6;
syms C1;syms C2;syms C3;syms C4;syms C5;syms C6;
syms S1;syms S2;syms S3;syms S4;syms S5;syms S6;
syms a2;syms a3;


A01=[C1 0 -S1 0;S1 0 C1 0;0 -1 0 d2 ; 0 0 0 1];
A12=[C2 -S2 0 a2*C2;S2 C2 0 a2*S2 ; 0 0 1 0 ; 0 0 0 1];
A23=[C3 0 S3 a3*C3;S3 0 -C3 a3*C3 ; 0 1 0 0 ; 0 0 0 1];
A34=[C4 0 -S4 0;S4 0 C4 0;0 -1 0 d4 ; 0 0 0 1];
A45=[C5 0 S5 0;S5 0 -C5 0;0 1 0 0 ; 0 0 0 1];
A56=[C6 -S6 0 0;S6 C6 0 0;0 0 1 d6 ; 0 0 0 1];


Th=A01*A12*A23*A34*A45*A56
```

Dando el siguiente resultado:

```
Th =
[                                  (((C2*C1*C3-S2*C1*S3)*C4-S1*S4)*C5+(-C2*C1*S3-S2*C1*C3)*S5)*C6+(-(C2*C1*C3-S2*C1*S3)*S4-S1*C4)*S6,
                                   -(((C2*C1*C3-S2*C1*S3)*C4-S1*S4)*C5+(-C2*C1*S3-S2*C1*C3)*S5)*S6+(-(C2*C1*C3-S2*C1*S3)*S4-S1*C4)*C6,
                                                                      ((C2*C1*C3-S2*C1*S3)*C4-S1*S4)*S5-(-C2*C1*S3-S2*C1*C3)*C5,
          (((C2*C1*C3-S2*C1*S3)*C4-S1*S4)*S5-(-C2*C1*S3-S2*C1*C3)*C5)*d6+(C2*C1*S3+S2*C1*C3)*d4+C2*C1*a3*C3-S2*C1*a3*C3+C1*a2*C2]
[                                  (((C2*S1*C3-S2*S1*S3)*C4+C1*S4)*C5+(-C2*S1*S3-S2*S1*C3)*S5)*C6+(-(C2*S1*C3-S2*S1*S3)*S4+C1*C4)*S6,
                                   -(((C2*S1*C3-S2*S1*S3)*C4+C1*S4)*C5+(-C2*S1*S3-S2*S1*C3)*S5)*S6+(-(C2*S1*C3-S2*S1*S3)*S4+C1*C4)*C6,
                                                                      ((C2*S1*C3-S2*S1*S3)*C4+C1*S4)*S5-(-C2*S1*S3-S2*S1*C3)*C5,
          (((C2*S1*C3-S2*S1*S3)*C4+C1*S4)*S5-(-C2*S1*S3-S2*S1*C3)*C5)*d6+(C2*S1*S3+S2*S1*C3)*d4+C2*S1*a3*C3-S2*S1*a3*C3+S1*a2*C2]
[                                                                     ((-S2*C3-C2*S3)*C4*C5+(S2*S3-C2*C3)*S5)*C6-(-S2*C3-C2*S3)*S4*S6,
                                                                      -((-S2*C3-C2*S3)*C4*C5+(S2*S3-C2*C3)*S5)*S6-(-S2*C3-C2*S3)*S4*C6,
                                                                                         (-S2*C3-C2*S3)*C4*S5-(S2*S3-C2*C3)*C5,
                                                    ((-S2*C3-C2*S3)*C4*S5-(S2*S3-C2*C3)*C5)*d6+(-S2*S3+C2*C3)*d4-S2*a3*C3-C2*a3*C3-a2*S2+d2]
                                                                                  [  0,                 0,                 0,                 1]
```

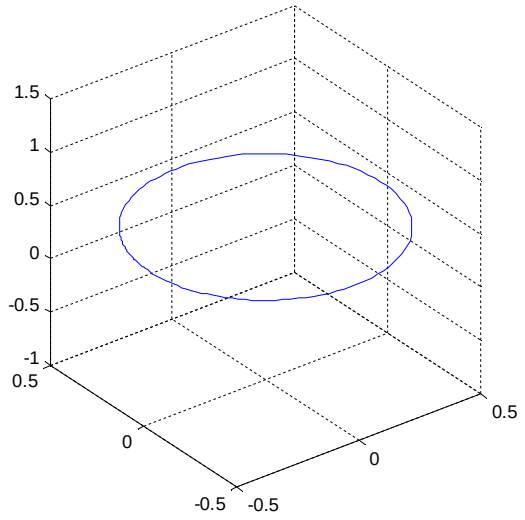
Matriz que puede ser simplificada (ej.: S(1+-2)=S1*C2+-S2*C1)y nos permite llegar a las ec. Antes expuestas.

Ejemplo con robotic toolbox de Matlab

Código fuente:

```
%> \file AreaTrabajo1.m
%> \brief genera un brazo Puma560 animado
%> \author Maximiliano A. Gonzalez, Matias Bobbio
%> \author UTN FRBA - Robotica
%> PRACTICA CON ROBOTIC TOOLBOX
%> Fecha creacion 2010.08.10
%> Fecha ultima modificacion: 2010.08.10
%> Descripcion: ploteo area de trabajo

close all
clear all

% Consider the Puma 560
puma560;
qz  %the joint coordinates of zero% which are defined by qz

% The forward kinematics may be computed using fkine() with an appropropriate
% kinematic description, in this case, the matrix p560 which defines
% kinematics for the 6-axis Puma 560.
fkine(p560, qz)

% returns the homogeneous transform corresponding to the last link of the
% manipulator

% fkine() can also be used with a time sequence of joint coordinates, or
% trajectory, which is generated by jtraj()
%
    t = [0:.056:10];    % generate a time vector
    qi=[-pi 0 0 0 0 0]
    qf=[pi 0 0 0 0 0]

    q = jtraj(qi, qf, t); % compute the joint coordinate trajectory

% then the homogeneous transform for each set of joint coordinates is given by
    T = fkine(p560, q);

% Ahora hago el plot 3D del area de trabajo
     figure;
     plot3(squeeze(T(1,4,:)), squeeze(T(2,4,:)),squeeze(T(3,4,:)));
     grid on
     axis square
```

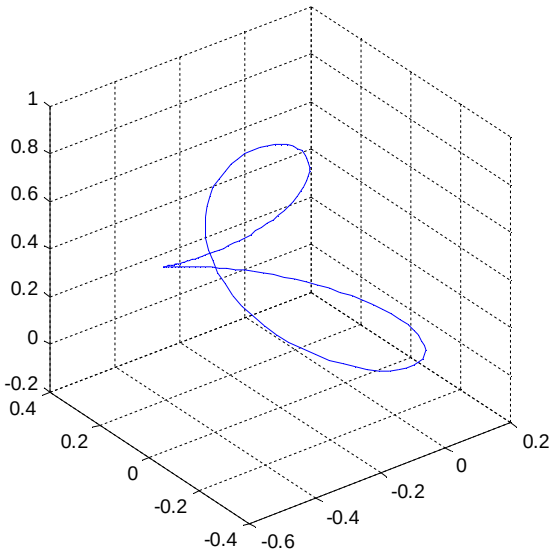
Trayectoria barriendo $\vartheta 1$ 360°:



Ahora ploteamos la trayectoria para

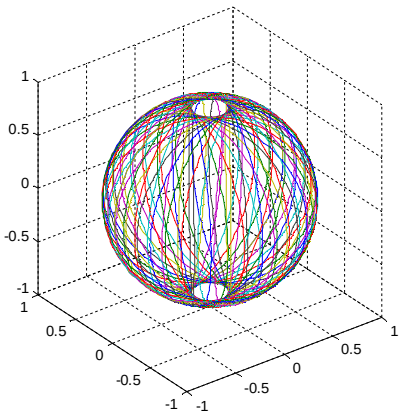
```
qi=[-pi pi/2 -pi 0 0 0]
qf=[pi pi/2 pi 0 0 0 ]
```

siendo estos los valores de $\vartheta = (\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6)^t$

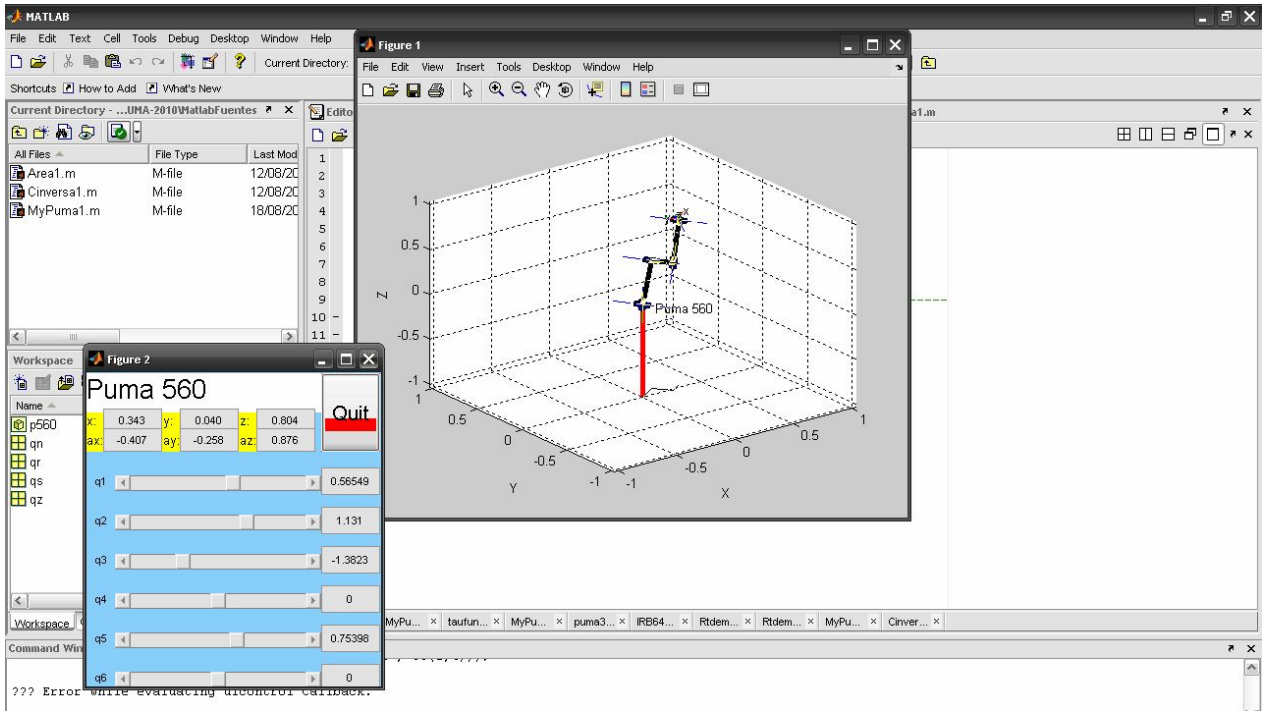


Nota: no se tubo en cuanta limitaciones mecanicas(esto seria barrer menos de 360°)

```
%> \file AreaTrabajoMaximaCompleta1.m
%> \brief genera un brazo Puma560 animado
%> \author Maximiliano A. Gonzalez, Matias Bobbio
%> \author UTN FRBA - Robotica
%> PRACTICA CON ROBOTIC TOOLBOX
%> Fecha creacion 2010.08.25
%> Fecha ultima modificacion: 2010.08.25
%> Descripcion: ploteo area de trabajo

close all
clear all

% Consider the Puma 560
puma560;

% fkine() can also be used with a time sequence of joint coordinates, or
% trajectory, which is generated by jtraj()
%
t = [0:.056:10];    % generate a time vector

figure;
k=1;
for m = -1:0.05:k
        qi=[m*pi -pi -pi/2 0 0 0];  %juego con las primeras tres variables
        %articulares x que son las que posicionan la muñeca
        qf=[m*pi pi -pi/2 0 0 0 ];
        q = jtraj(qi, qf, t); % compute the joint coordinate trajectory

% then the homogeneous transform for each set of joint coordinates is given by
        T = fkine(p560, q);

        % where T is a 3-dimensional matrix, the first two dimensions are a 4x4
        % homogeneous transformation and the third dimension is time.
        % Ahora hago el plot 3D del area de trabajo

         plot3(squeeze(T(1,4,:)), squeeze(T(2,4,:)),squeeze(T(3,4,:)));
%plot3(x,y,z)
         hold all
         grid on
         axis square
end
```



```
%> \file MyPuma1.m
%> \brief genera un brazo Puma560 animado
%> \author Maximiliano A. Gonzalez, Matias Bobbio
%> \author UTN FRBA - Robotica
%> PRACTICA CON ROBOTIC TOOLBOX
%> Fecha creacion 2010.08.10
%> Fecha ultima modificacion: 2010.08.10
%> Nota: Ejemplo de función.
%--------------------------------------------------------------------------
close all
clear all
puma560 % define the robot
plot(p560,qz) % draw it
drivebot(p560) % now drive it
```



Este ejemplo permite controlar un modelo del brazo puma, variando el valor de los q.

CINEMATICA INVERSA

Solución de la cinemática inversa para un brazo robótica PUMA

Dada la posición y orientación del actuador final T_0^6 de un robot necesitamos encontrar el correspondiente vector de ángulos de uniones $\vartheta = (\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6)^t$ del robot para que el actuador final pueda ser posicionado donde se desee.
La solución es calculada en dos etapas. Primero un vector posición apuntando desde el hombre hacia la muñeca es derivado. Este es luego usado para derivar la solución de las tres primeras uniones. Las ultimas tres uniones son solucionadas usando los valores calculados de las tres primeras y las submatrices "orientación" de T_0^i ,i= 4,5,6.

Lo calcularemos con el método de desacoplo cinemática, ya que este separa en 2 problemas: posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los 3 últimos ejes (muñeca del robot) calculándose los valores de las tres primeras variables articulares (q1,q2,q3) que consiguen posicionar este punto. A continuación, a partir de los datos de orientación y de los ya calculados (q1, q2,q3) obtiene los valores del resto de las variables articulares.

Cinemática Inversa 1° parte

Solución del brazo para $(\vartheta_1, \vartheta_2, \vartheta_3)$ de un brazo PUMA.

El vector posición p, que apunta desde el origen del sistema de coordenadas del hombro (Xo, Yo, Zo) hacia el punto final del enlace 3 (Figure 8), es calculado como:

$$\mathbf{p} = \mathbf{p}_6 - d_6\mathbf{a} = (p_x, p_y, p_z)^t$$

Recordando que,

$$\mathbf{T}_o^i = \prod_{j=1}^{i} A_{j-1}^j$$

Y encontrando el vector de posición de T_0^4 ,tenemos:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} C_1(a_2C_2 + d_4S_{23}) & -d_2S_1 \\ S_1(a_2C_2 + d_4S_{23}) & +d_2C_1 \\ d_4C_{23} - a_2S_2 \end{bmatrix} \qquad (32)$$

De esta ultima ecuación, usando las ecuaciones px y py para resolver ϑ_1 , tenemos

$$\vartheta_1 = \tan^{-1}\left[\frac{\pm p_y\sqrt{p_x^2+p_y^2-d_2^2}-d_2p_x}{\pm p_x\sqrt{p_x^2+p_y^2-d_2^2}+d_2p_y}\right]; \qquad (33)$$

$$-\pi \le \vartheta_1 \le \pi$$

where $\tan^{-1}$ is defined as

$$\vartheta = \tan^{-1}\left(\frac{y}{x}\right)$$

$$= \begin{cases} 0° \le \vartheta \le 90°; & \text{for } +x \text{ and } +y \\ 90° \le \vartheta \le 180°; & \text{for } -x \text{ and } +y \\ -180° \le \vartheta \le -90°; & \text{for } -x \text{ and } -y \\ -90° \le \vartheta \le 0°; & \text{for } +x \text{ and } -y \end{cases}$$

Por el signo ± de la función raíz cuadrada, que tiene dos soluciones correspondientes a dos configuraciones de brazo (el signo positivo de la raíz cuadrada para el brazo el hombro izquierdo y el signo negativo de la raíz cuadrada para el brazo el hombro derecho)
De la ecuación 32, cuadrando los componentes y la sumandolos, tenemos:

$$\vartheta_3 = \tan^{-1}\left[\frac{p_x^2+p_y^2+p_z^2-d_4^2-a_2^2+d_2^2}{\pm\sqrt{4d_4^2a_2^2-(p_x^2+p_y^2+p_z^2-d_4^2-a_2^2-d_2^2)^2}}\right];$$

$$-\pi \le \vartheta_3 \le \pi \qquad (34)$$

From equation 32, using the pz-component equation
El signo positivo de la raíz cuadrada de la ecuación anterior da la configuración del codo por debajo de la mano, y el negativo de la raíz cuadrada es para el codo por encima.
De la ecuación 32, usando la ecuación pz y $r\sin(\alpha+\beta) = r\sin\alpha\cos\beta + r\cos\alpha\sin\beta$,
obtenemos ϑ_2 como sigue:

$$\vartheta_2 = \tan^{-1}\left[\frac{-\left\{p_z(a_2 + d_4S_3) + (d_4C_3)(\pm\sqrt{p_x^2 + p_y^2 - d_2^2})\right\}}{p_z(d_4C_3) - (a_2 + d_4S_3)(\pm\sqrt{p_x^2 + p_y^2 - d_2^2})}\right];$$

$$-\pi \le \vartheta_2 \le \pi \qquad (35)$$

El signo negativo de la raíz cuadrada de la ecuación anterior da la configuración del brazo izquierdo, y la raíz cuadrada positiva da la configuración del brazo derecho. Conociendo los primeros tres ángulos articulares $(\vartheta_1, \vartheta_2, \vartheta_3)$ ahora podemos evaluar la matriz T_0^3.

Cinematica Inversa 2° parte

Solución del brazo $(\vartheta_4, \vartheta_5, \vartheta_6)$ para un robot PUMA
Para encontrar la solución de las últimos tres ángulos articulares de un brazo robótica PUMA, seteamos estas uniones para satisfacer el siguiente criterio (Figura 8):
(1) Seteamos la unión 4 de forma tal que una rotación sobre la unión 5 alinea los ejes del movimiento de la unión 6 con el vector dado de acercamiento. (a de T)
(2) Seteamos la unión 5 para alinear los ejes de movimiento de la unión 6 con el vector de acercamiento.
(3) Seteamos la unión 6 para alinear el vector dado de orientación (o vector de deslizamiento Y6) y el vector normal
Matemáticamente, el criterio anterior se traduce a:

$$\mathbf{z}_4 = \frac{\pm(\mathbf{z}_3 \times \mathbf{a})}{\| \mathbf{z}_3 \times \mathbf{a} \|}\ ;\ given\ \mathbf{a} = (a_x, a_y, a_z)^t$$
$$\mathbf{a} = \mathbf{z}_5\ ;\ given\ \mathbf{a} = (a_x, a_y, a_z)^t$$
$$\mathbf{s} = \mathbf{y}_6\ ;\ given\ \mathbf{s} = (s_x, s_y, s_z)^t \text{ and } \mathbf{n} = (n_x, n_y, n_z)^t$$

De aqui podemos llegar a: $S_4 = -(\mathbf{x}_3 \cdot \mathbf{z}_4)\ ;\ C_4 = (\mathbf{y}_3 \cdot \mathbf{z}_4)$

Donde x3 e y3 son los vectores columna x e y de T_0^3 respectivamente. Entonces,

$$\vartheta_4 = \tan^{-1}\left[\frac{C_1a_y - S_1a_x}{C_1C_{23}a_x + S_1C_{23}a_y - S_{23}a_z}\right]; \qquad (36)$$

Para encontrar $\vartheta 5$, usamos el criterio de alinear los ejes de la rotación de la unión 6 con el vector de acercamiento (o a = z5). Usando S5 =x4 *a y C5 = - (Y4* a), donde X4 e Y4 son los vectores columna x e y de T_0^4 respectivamente y a es el vector de acercamiento,

$$\vartheta_5 = \tan^{-1}\left[\frac{(C_1C_{23}C_4 - S_1S_4)a_x + (S_1C_{23}C_4 + C_1S_4)a_y - C_4S_{23}a_z}{C_1S_{23}a_x + S_1S_{23}a_y + C_{23}a_z}\right]; \quad (37)$$

$$-\pi \le \vartheta_5 \le \pi$$

Si $\vartheta 5 \approx 0$, ocurre un caso degenerado. En el cual, en este posición / orientación particular, un brazo robótica de 5 ejes satisfacería.
Hasta ahora, tenemos alineados los ejes de la unión 6 con el vector de acercamiento. A continuación necesitamos alinear la garra para facilitar el levantar un objeto. El criterio para hacer esto es setear s=y6. Usando S6=y5*n y C6=y5*s, donde y5 es el vector columna de T_0^5 y n y s son los vectores normal y deslizante de T_0^6, respectivamente

$$\vartheta_6 = \tan^{-1}\left[\frac{(-S_1C_4 - C_1C_{23}S_4)n_x + (C_1C_4 - S_1C_{23}S_4)n_y + (S_4S_{23})n_z}{(-S_1C_4 - C_1C_{23}S_4)s_x + (C_1C_4 - S_1C_{23}S_4)s_y + (S_4S_{23})s_z}\right]; \quad (38)$$

$$-\pi \le \vartheta_6 \le \pi$$

Si el caso degenerado ocurre, entonces ($\vartheta 4 + \vartheta 6$)= al Angulo total requerido para alinear al vector deslizante s.
Resumiendo, tenemos cuatro soluciones para el problema de la cinemática inversa, dos para la configuración de brazo derecho y dos para la de brazo izquierdo. Para cada configuración, las ecuaciones 33 a 38 dan un set de soluciones ($\vartheta 1, \vartheta 2, \vartheta 3, \vartheta 4, \vartheta 5, \vartheta 6$) y ($\vartheta 1, \vartheta 2, \vartheta 3, \vartheta 4 + \Pi, -\vartheta 5, \vartheta 6 + \Pi$) dan el otro set de soluciones.
Las soluciones de los ángulos de las uniones de un brazo PUMA son mostradas en la Figura 9.

$$\vartheta_1 = \tan^{-1}\left[\frac{\pm p_y\sqrt{p_x^2+p_y^2-d_2^2} - d_2 p_x}{\pm p_x\sqrt{p_x^2+p_y^2-d_2^2} + d_2 p_y}\right]$$

$$\vartheta_2 = \tan^{-1}\left[\frac{-\left(p_z(a_2+d_4S_3)+(d_4C_3)(\pm\sqrt{p_x^2+p_y^2-d_2^2})\right)}{p_z(d_4C_3)-(a_2+d_4S_3)(\pm\sqrt{p_x^2+p_y^2-d_2^2})}\right]$$

$$\vartheta_3 = \tan^{-1}\left[\frac{p_x^2+p_y^2+p_z^2-d_4^2-a_2^2-d_2^2}{\pm\sqrt{4d_4^2a_2^2-(p_x^2+p_y^2+p_z^2-d_4^2-a_2^2-d_2^2)^2}}\right]$$

$$\vartheta_4 = \tan^{-1}\left[\frac{C_1a_y-S_1a_x}{C_1C_{23}a_x+S_1C_{23}a_y-S_{23}a_z}\right]$$

$$\vartheta_5 = \tan^{-1}\left[\frac{(C_1C_{23}C_4-S_1S_4)a_x+(S_1C_{23}C_4+C_1S_4)a_y-C_4S_{23}a_z}{C_1S_{23}a_x+S_1S_{23}a_y+C_{23}a_z}\right]$$

$$\vartheta_6 = \tan^{-1}\left[\frac{(-S_1C_4-C_1C_{23}S_4)n_x+(C_1C_4-S_1C_{23}S_4)n_y+(S_4S_{23})n_z}{(-S_1C_4-C_1C_{23}S_4)s_x+(C_1C_4-S_1C_{23}S_4)s_y+(S_4S_{23})s_z}\right]$$

$$-180° \le \vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6 \le 180°$$

- Degenerate case ($\vartheta_5 \approx 0$)

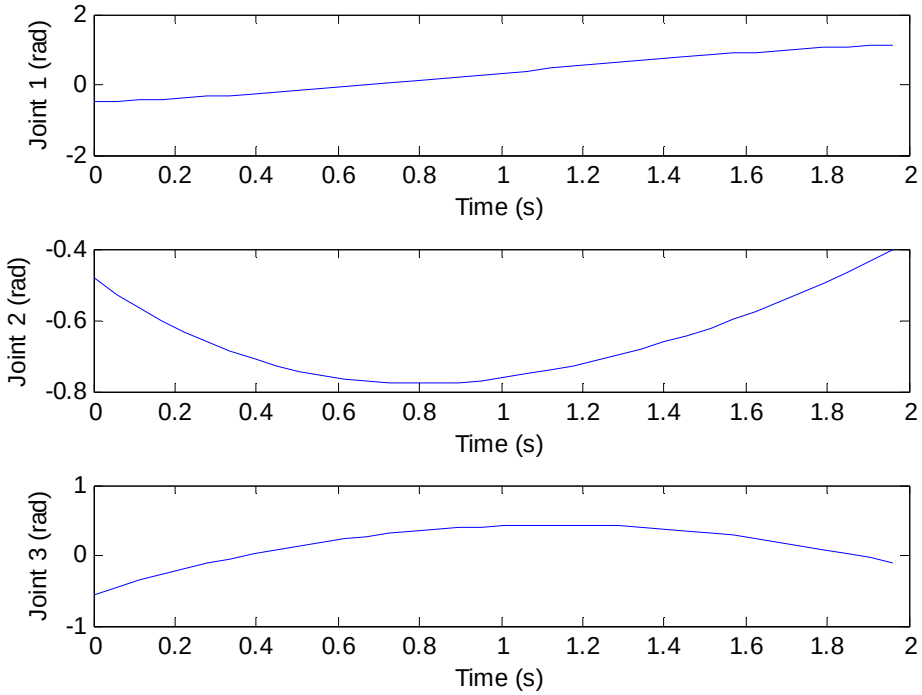
  ϑ_4 = current value of ϑ_4 or 0 and
  $(\vartheta_4 + \vartheta_6)$ = total angle required to align the orientation.
- For a given arm configuration,
  $(\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6)$ is a set of solutions and
  $(\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4 + \pi, -\vartheta_5, \vartheta_6 + \pi)$ is another set of solutions.

Figure 9. Solutions for the joint angles of a PUMA robot arm.

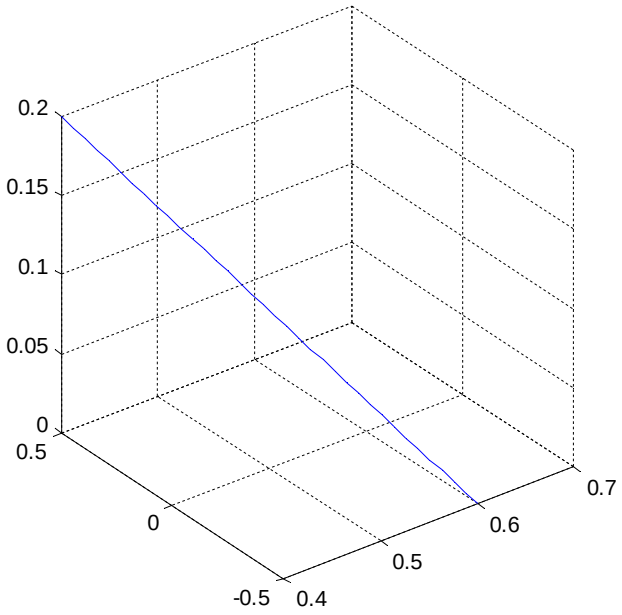
Ejemplo con robotic toolbox de Matlab

```
%> \file CInversa1.m
%> \brief genera un brazo Puma560 animado
%> \author Maximiliano A. Gonzalez, Matias Bobbio
%> \author UTN FRBA - Robotica
%> PRACTICA CON ROBOTIC TOOLBOX
%> Fecha creacion 2010.08.10
%> Fecha ultima modificacion: 2010.08.10
%> Descripcion: aplicacion de conceptos de cinematica inversa

close all
clear all

% Consider the Puma 560
puma560;

% Inverse kinematics may also be computed for a trajectory.
% If we take a Cartesian straight line path
    t = [0:.056:2];         % create a time vector
    T1 = transl(0.6, -0.5, 0.0) % define the start point

    T2 = transl(0.4, 0.5, 0.2)  % and destination

    T = ctraj(T1, T2, length(t));   % compute a Cartesian path

% now solve the inverse kinematics.  When solving for a trajectory, the
% starting joint coordinates for each point is taken as the result of the
% previous inverse solution.

    q = ikine(p560, T);

% Let's examine the joint space trajectory that results in straightline
% Cartesian motion
    figure;
    subplot(3,1,1)
    plot(t,q(:,1))
    xlabel('Time (s)');
    ylabel('Joint 1 (rad)')
    subplot(3,1,2)
    plot(t,q(:,2))
    xlabel('Time (s)');
    ylabel('Joint 2 (rad)')
    subplot(3,1,3)
    plot(t,q(:,3))
    xlabel('Time (s)');
    ylabel('Joint 3 (rad)')

    % Ahora hago el plot 3D del area de trabajo
     figure;
     plot3(squeeze(T(1,4,:)), squeeze(T(2,4,:)),squeeze(T(3,4,:)));
     grid on
     axis square
```

Aquí vemos como se modifican las varialbles articulares a medida que se avansa en la trayectoria:



Aquí vemos la trayectoria realizada desde el punto inicial T1 hasta el final T2:



DINAMICA

Introducción

Las ecuaciones cinemáticas describen el movimiento del robot sin considerar las fuerzas y torques que afectan el movimiento, las ecuaciones dinámicas por otro lado describen la relación entre las fuerzas y el movimiento. Las ecuaciones del movimiento son importantes al considerar el diseño del robot, para la simulación y animación del movimiento del robot, y en el diseño de los algoritmos de control. Esta relación se obtiene mediante el denominado modelo dinámico, que relaciona matemáticamente:

- La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.
- Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot).
- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

El problema de la obtención del modelo dinámico de un robot es, por lo tanto, uno de los aspectos más complejos de la robótica, lo que ha llevado a ser obviado en numerosas ocasiones.

- Simulación del movimiento del robot.
- Diseño y evaluación de la estructura mecánica del robot.
- Dimensionado de los actuadores.
- Diseño y evaluación del control dinámico del robot.

Este último fin es evidentemente de gran importancia, pues de la calidad del control dinámico del robot depende la precisión y velocidad de sus movimientos. La gran complejidad ya comentada existente en la obtención del modelo dinámico del robot, ha motivado que se realicen ciertas simplificaciones, de manera que así pueda ser utilizado en el diseño del controlador.

Es importante hacer notar que el modelo dinámico completo de un robot debe incluir no sólo la dinámica de sus elementos (barras o eslabones) sino también la propia de sus sistemas de transmisión, de los actuadores y sus equipos electrónicos de mando.

Estos elementos incorporan al modelo dinámico nuevas inercias, rozamientos, saturaciones de los circuitos electrónicos, etc. aumentando aún más su complejidad.

Por último, es preciso señalar que si bien en la mayor parte de las aplicaciones reales de la robótica, las cargas e inercias manejadas no son suficientes como para originar deformaciones en los eslabones del robot, en determinadas ocasiones no ocurre así, siendo preciso considerar al robot como un conjunto de eslabones no rígidos. Aplicaciones de este tipo pueden encontrarse en la robótica espacial o en robots de grandes dimensiones, entre otras.

Existen 2 formas básicas de encarar el problema del modelo dinámico:

1 - Modelo Dinámico Directo: Expresa la evolución temporal de las coordenadas articulares del robot en función de las fuerzas y pares que intervienen.

2 - Modelo Dinámico Inverso: Expresa las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas.

Características del modelo dinámico

El modelo dinámico de un manipulador es complejo, a continuación se listan algunos aspectos que hacen que dicho modelo matemático no pueda ser exacto, sin importar cuan precisos sean los cálculos o mediciones:

- Modelo no lineal
- Alto grado de acoplamiento entre las articulaciones
- Sistema variante en el tiempo

Así mismo, se producen incertidumbres que pueden clasificarse en estructuradas y no estructuradas:

- Incertidumbres estructuradas: Imprecisión en las propiedades del enlace.
- Incertidumbres no estructuradas: Dinámica no modelada (por Ej. la fricción que puede ser estática, de Coulomb, viscosa,etc).

En síntesis, el objetivo del modelo dinámico es obtener una solución simbólica (modelo matemático) que represente los movimientos de los elementos del manipulador en función de los torques aplicados teniendo en cuenta sus parámetros geométricos (distancias, ángulos relativos,etc) e inerciales (masas, momentos de inercia,etc) que se pueden utilizar para simular, diseñar e implementar leyes de control.

Existen varios métodos básicos para obtener el modelo dinámico de un manipulador:

- NewtonBEuler

- LagrangeBEuler
- Variables de estado
- Kane

Método de Newton Euler

Es un método iterativo y recursivo, se propagan aceleraciones y torques por los distintos elementos del manipulador. A continuación algunas características:

- Esta basado en la mecánica Newtoniana.
- Difícil obtención.
- Se obtienen ecuaciones recursivas. Cinemática hacia delante, dinámica hacia atrás.
- Menor tiempo de cálculo, útil en tiempo real.

Método de Lagrange Euler

Es un método cerrado, se basa en el balance de energías del sistema. Así, resulta de la diferencia de las energías
cinéticas y potenciales de todos los elementos que componen el manipulador.
Este método produce las mismas ecuaciones de movimiento para el mismo manipulador.
Algunas características de este método:

- Obtención simple y sistemática
- Mucho tiempo de cálculo, no recomendable para tiempo real.
- Se obtienen ecuaciones diferenciales

Desarrollo del Análisis Dinámico Utilizando el enfoque energético de Lagrange-Euler

Calculo de la energía cinética

Considérese un vector posición expresado en las coordenadas homogéneas, $p = (x, y, z)^t$, el cual apunta desde es sistema de coordenadas base hacia una masa diferencial dm localizada en el i enésimo enlace. La energía cinética dKi de esta masa diferencial es $\frac{1}{2} Tr(\mathbf{v}_o^i (\mathbf{v}_o^i)^t) dm$, la cual iguala

$$dK_i = \frac{1}{2} \sum_{j=1}^{i} \sum_{k=1}^{i} Tr \left\{ \frac{\partial \mathbf{T}_o^i}{\partial \vartheta_j} \mathbf{r}_i (\mathbf{r}_i)^t dm \left(\frac{\partial \mathbf{T}_o^i}{\partial \vartheta_k} \right)^t \dot{\vartheta}_j \dot{\vartheta}_k \right\} \quad (39)$$

donde Tr es el operador. Si cada enlace es integrado sobre su masa total y las energías cinéticas de todos los enlaces son sumadas, entonces la energía cinética KE del brazo robótico con respecto al sistema de referencia inercial es

$$KE = \sum_{i=1}^{n} \int dK_i$$

$$= \sum_{i=1}^{n} \left\{ \frac{1}{2} \operatorname{Tr} \left\{ \sum_{j=1}^{i} \sum_{k=1}^{i} \frac{\partial \mathbf{T}_o^i}{\partial \vartheta_j} \mathbf{J}_i \left(\frac{\partial \mathbf{T}_o^i}{\partial \vartheta_k} \right)^t \dot{\vartheta}_j \dot{\vartheta}_k \right\} \right\} \quad (40)$$

where $\mathbf{J}_i$ can be expressed as:

$$m_i \cdot \begin{bmatrix} \dfrac{-k_{i11}^2 + k_{i22}^2 + k_{i33}^2}{2} & k_{i12}^2 & k_{i13}^2 & \bar{x}_i \\ k_{i12}^2 & \dfrac{k_{i11}^2 - k_{i22}^2 + k_{i33}^2}{2} & k_{i23}^2 & \bar{y}_i \\ k_{i13}^2 & k_{i23}^2 & \dfrac{k_{i11}^2 + k_{i22}^2 - k_{i33}^2}{2} & \bar{z}_i \\ \bar{x}_i & \bar{y}_i & \bar{z}_i & 1 \end{bmatrix} \quad (41)$$

Donde Kijk es el radio de giro del enlace i sobre los ejes j-k

Calculo de la energía potencial

La energía potencial total del brazo robótico es la suma de la energía potencial PE de cada enlace expresada en el sistema de coordenadas base.

$$PE = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} -\overline{m}_i \mathbf{g} \mathbf{T}_o^i \bar{\mathbf{r}}_i \qquad (42)$$

donde $\bar{\mathbf{r}}_i$ es la posición del centro de masa del enlace i con respecto al i enésimo sistema de coordenadas y g, el vector fila de la gravedad, es (gx,gy.gz,0); |g|=9,8062 m/s².

Ecuaciones del movimiento de Lagrange-Euler

A partir de las ecuaciones 40, 41 y 42, podemos formar la función Lagrangiana L=KE-PE, y y aplicando la formula Lagrange-Euler a esta función, obtenemos el torque necesario generalizado τ_i, para la articulación i para conducir el i enésimo enlace del brazo.

$$\tau_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}_i}\right) - \frac{\partial L}{\partial \vartheta_i} = \sum_{k=i}^{n} \sum_{j=1}^{k} Tr\left\{ \frac{\partial \mathbf{T}_o^k}{\partial \vartheta_j} \mathbf{J}_k \left(\frac{\partial \mathbf{T}_o^k}{\partial \vartheta_i}\right)^t \right\} \ddot{\vartheta}_j$$

$$+ \sum_{r=i}^{n} \sum_{j=1}^{r} \sum_{k=1}^{r} Tr\left\{ \frac{\partial^2 \mathbf{T}_o^r}{\partial \vartheta_j \partial \vartheta_k} \mathbf{J}_r \left(\frac{\partial \mathbf{T}_o^r}{\partial \vartheta_i}\right)^t \right\} \dot{\vartheta}_j \dot{\vartheta}_k \qquad (43)$$

$$- \sum_{j=i}^{n} m_j \mathbf{g} \left\{ \frac{\partial \mathbf{T}_o^j}{\partial \vartheta_i} \right\} \bar{\mathbf{r}}_j \quad ; \text{for } i = 1,2,\ldots,n$$

or expressed in matrix form

$$\sum_{k=1}^{n} D_{ik} \ddot{\vartheta}_k + \sum_{k=1}^{n} \sum_{m=1}^{n} H_{ikm} \dot{\vartheta}_k \dot{\vartheta}_m + \mathbf{G}_i = \tau_i ; \qquad (44)$$

$$\text{for } i = 1,2,\ldots,n$$

or

$$\mathbf{D}(\vartheta) + \mathbf{H}(\vartheta, \dot{\vartheta}) + \mathbf{G}(\vartheta) = \tau \qquad (45)$$

where

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Tr\left\{ \frac{\partial \mathbf{T}_o^j}{\partial \vartheta_k} \mathbf{J}_j \left(\frac{\partial \mathbf{T}_o^j}{\partial \vartheta_i}\right)^t \right\} \qquad (46)$$

$$; \text{for } i = 1,2,\ldots,n$$

$$H_{ikm} = \sum_{j=\max(i,k,m)}^{n} Tr\left\{ \frac{\partial^2 \mathbf{T}_o^j}{\partial \vartheta_k \partial \vartheta_m} \mathbf{J}_j \left(\frac{\partial \mathbf{T}_o^j}{\partial \vartheta_i}\right)^t \right\} \qquad (47)$$

$$; \text{for } i,k,m = 1,2,\ldots,n$$

$$G_i = \sum_{j=i}^{n} \left(-m_j \mathbf{g} \left\{ \frac{\partial \mathbf{T}_o^j}{\partial \vartheta_i} \right\} \bar{\mathbf{r}}_j \right) ; \text{ for } i = 1,2,\ldots,n \qquad (48)$$

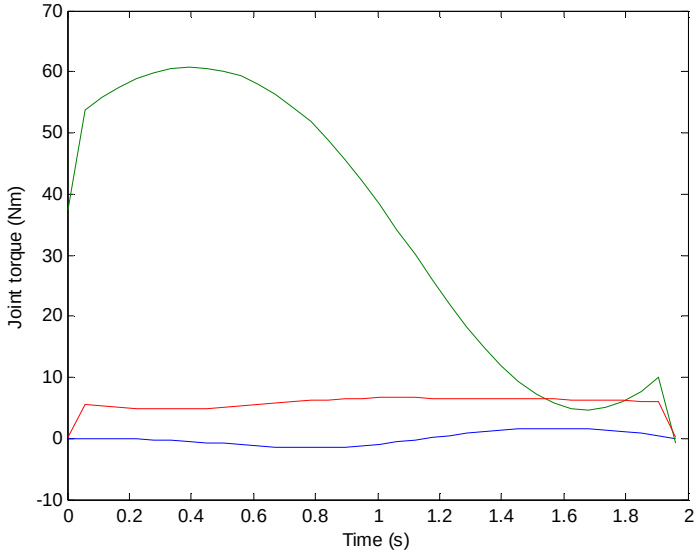
Los coeficientes Gi, Dik y Hikm en las ec. 46,47 y 48 son funciones tanto de las variables de las articulaciones y los parámetros inerciales del manipulador y pueden ser llamados los *coeficientes dinámicos* del manipulador. La interpretación física de estos coeficientes dinámicos puede verse fácilmente desde las ecuaciones de movimiento de L-E, dadas por la ec. 44.

1- El coeficiente Gi representa los términos de carga de la gravedad para los enlaces y es definida por la ec. 48.
2- El coeficiente de Dik se relaciona con la aceleración de la las variables de la articulación y se define por la ecuación 46. En particular, para k = i, Dii se relaciona con la aceleración de la articulación i donde el par motor τ i actúa, mientras que para i ≠ k, Dik está relacionado con el par de reacción (o fuerza) inducida por la aceleración de la articulación k y actuando en la articulación i, o viceversa. Dado que la matriz de inercia es simétricas y $Tr(\mathbf{A}) = Tr(\mathbf{A}^t)$, podemos demostrar que $D_{ik} = D_{ki}$.
3- El coeficiente Hikm se relaciona con la velocidad de la variables de las articulaciones y se define por la ecuación 47. Los dos últimos índices k y m se refieren a las velocidades de las articulaciones k y m cuyas velocidades inducen un torque de reacción (o fuerza) en la articulación i. Asi, el primer índice i esta siempre relacionado con la articulación donde la velocidad induce torques de reacción. En particular, cuando k=m, Hikk esta relacionado con la fuerza Centrifuga, generada por la velocidad angular de la articulación K y sentida en la articulación i, mientras que cuando k≠m, Hikm esta relacionada con la fuerza de Coriolis generada por las velocidades de las articulaciones k y m y sentida en la articulación i.

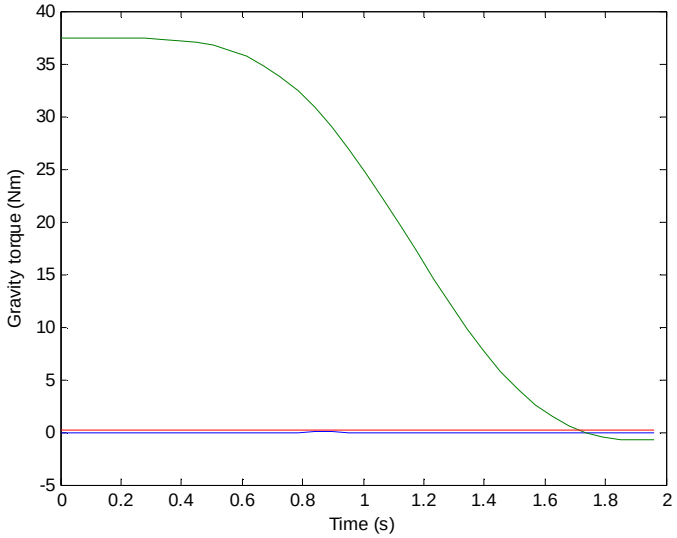
Debido a su estructura matricial, esta formulación es atractiva desde un punto de vista de control, ya que ofrece un conjunto de ecuaciones de estado como los de la ecuación 45. Esta forma nos permite diseñar una ley de control que fácilmente compense todos los efectos no lineales. Computacionalmente, sin embargo, estas ecuaciones de movimiento son extremadamente ineficientes, en comparación con otras formulaciones.

Análisis con Roobotic Toolbox de Matlab

```
%Dinámica inversa calcula los torques requeridos para alcanzar una posición, a
%una velocidad y aceleración especificadas.
%La formulación recursiva de Newton-Euler es una algoritmo matricial eficiente
%para realizar este calculo, y es implementado en la función rne().
%La dinámica inversa requiere los parámetros de inercia y masa de cada enlace
%tanto como los parámetros cinemáticas
%Esto se logra adicionando columnas para estos datos a la matriz cinemática.
%Por ej., para un Puma560, en la posición ángulo cero, con  todas las
%velocidades de uniones de 5rad/s y aceleraciones de 1 rad/s/s, los torques
%requeridos son:
   tau = rne(p560, qz, 5*ones(1,6), ones(1,6))
%tau =
%   -7.940476566323304e+001    3.716938195250001e+001
%1.354546214243801e+001    1.072787264523680e+000
%9.398861746570001e-001    5.119329466680000e-001

%Como con otras fusiones la dinámica inversa puede ser computada para cada
%punto de una trayectoria. Crear un conjunto de coordenadas de trayectoria y
%calcular la velocidad y aceleración
   t = [0:.056:2];              % create time vector
   [q,qd,qdd] = jtraj(qz, qr, t); % compute joint coordinate trajectory
   tau = rne(p560, q, qd, qdd); % compute inverse dynamics

%Ahora los torques pueden ser ploteados como función del tiempo:
   plot(t, tau(:,1:3))
   xlabel('Time (s)');
   ylabel('Joint torque (Nm)')
```

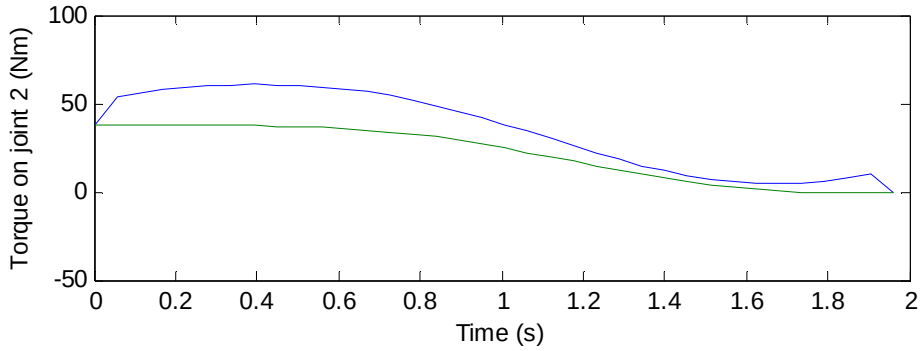


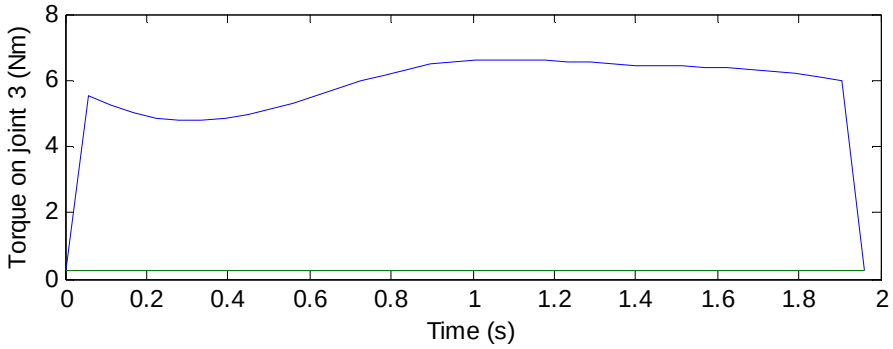

```
% Gran parte del torque de las articulaciones 2 y 3 del Puma 560 (montado
%convencionalmente) se debe a la gravedad. Esa componente puede ser
%calculada usando gravload()
   taug = gravload(p560, q);
   plot(t, taug(:,1:3))
   xlabel('Time (s)');
   ylabel('Gravity torque (Nm)')
```




```
% Ahora la ploteamos como una fracción del torque total requerido sobre la
%trayectoria:
```

```
subplot(2,1,1)
plot(t,[tau(:,2) taug(:,2)])
xlabel('Time (s)');
ylabel('Torque on joint 2 (Nm)')
subplot(2,1,2)
plot(t,[tau(:,3) taug(:,3)])
xlabel('Time (s)');
ylabel('Torque on joint 3 (Nm)')
```



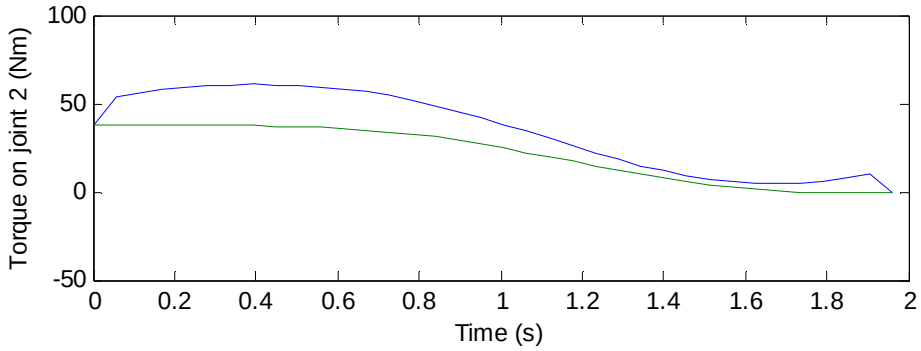


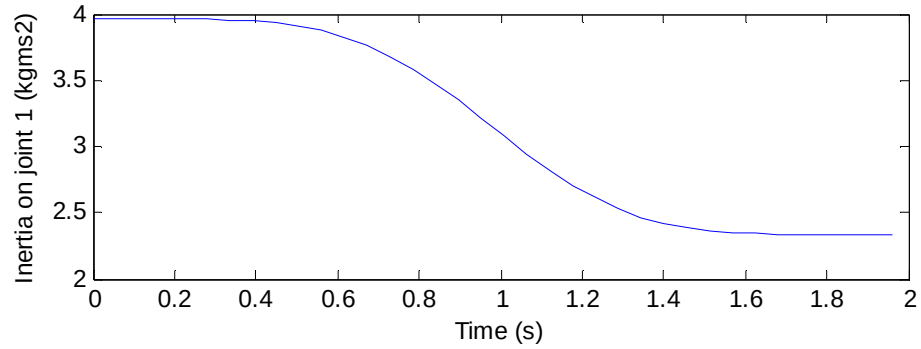

```
% La inercia vista por el motor de la cintura (articulación 1) cambia notablemente
%con la configuración del robot. La función inertia() calcula la matriz de inercia
para %una configuración dada.

%  Si calculamos la variación de la inercia en la articulación 1, que es M(1,1),
%mientras el manipulador se mueve a lo largo de una trayectoria
   M = inertia(p560, q);
   M11 = squeeze(M(1,1,:));
   plot(t, M11);
   xlabel('Time (s)');
   ylabel('Inertia on joint 1 (kgms2)')
```

% Claramente la inercia vista por la articulación 1 varia considerablemente en %este camino.
% Este es uno de muchos desafíos para controlar en el diseño de robots, %buscando estabilidad y alta performance frente a la variación de planta. De hecho %para este ejemplo la inercia varia por un factor de:

```
max(M11)/min(M11)
```

%ans =
% 1.694707391268889e+000



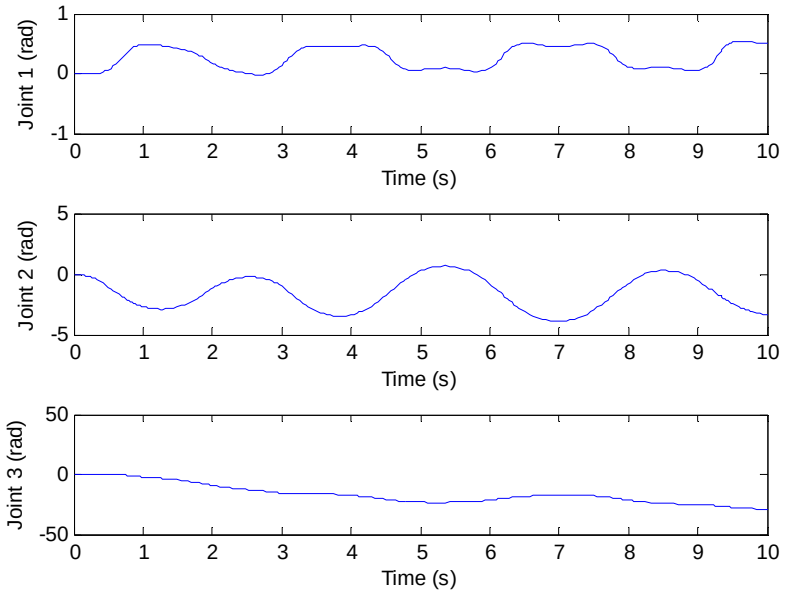


% Dinámica Directa es el calculo de las aceleraciones de las articulaciones dada %la posición, velocidad y torques actuantes. Es útil en la simulación de sistemas %de control de robots.
%
% Considerando un puma 560 en descanso en la posición ángulo cero, con %torques cero. La aceleración de las articulaciones estaria dada por:

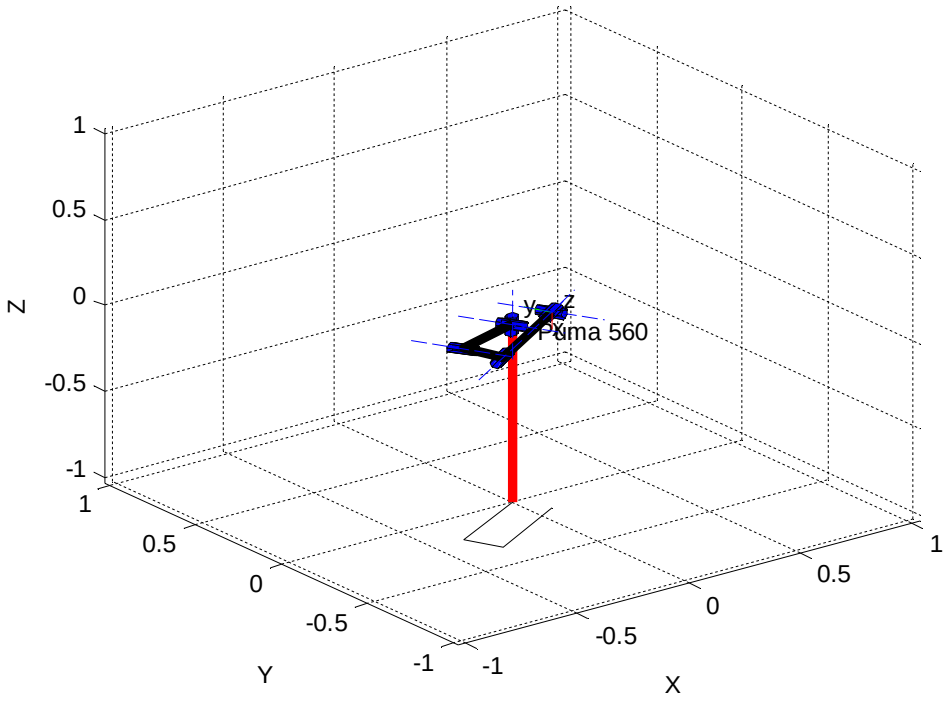
```
accel(p560, qz, zeros(1,6), zeros(1,6))
```

%ans =
% -2.462261497743049e-001
% -8.682930235967222e+000
% 3.146208071580471e+000
% 2.098464685294656e-003
% 6.031239470186990e-002
% 5.030850453447406e-005

```
%
% Para ser util para simulación, esta función debe ser integrada. fdyn() usa la
%función de MATLAB ode45() para esto. También permite a una función escrita
%por el usuario calcular los torques de las articulaciones como función del estado
%del manipulador.
% Para simular el movimiento de un Puma560 desde el descanso el la posición de
ángulo cero con torques  cero:
    tic
    [t q qd] = fdyn(nofriction(p560), 0, 10);
    toc
%Elapsed time is 43.000000 seconds.
%
% Y el movimiento resultante puede ser ploteado como función del tiempo
     subplot(3,1,1)
    plot(t,q(:,1))
    xlabel('Time (s)');
    ylabel('Joint 1 (rad)')
    subplot(3,1,2)
    plot(t,q(:,2))
    xlabel('Time (s)');
    ylabel('Joint 2 (rad)')
    subplot(3,1,3)
    plot(t,q(:,3))
    xlabel('Time (s)');
    ylabel('Joint 3 (rad)')
```



```
% Claramente el robot esta colapsando bajo la gravedad, pero es interesante
%notar que la velocidad rotacional del brazo superior e inferior  ejercen torques
% centripetal y coriolis en la articulación de cintura (arti. 1), causando que rote.

% This can be shown in animation also
   clf
   plot(p560, q)
```



esta ultima imagen es en realidad una animación en Matlab.

COMPILADOR

Introducción al antlrWorks

ANTLR es una herramienta creada principalmente por Terence Parr, que opera sobre lenguajes, proporcionando un marco para construir reconocedores (parsers), intérpretes, compiladores y traductores de lenguajes a partir de las descripciones gramaticales de los mismos (conteniendo acciones semánticas a realizarse en varios lenguajes de programación).
Partiendo de la descripción formal de la gramática de un lenguaje, ANTLR genera un programa que determina si una sentencia o palabra pertenece a dicho lenguaje (reconocedor), utilizando algoritmos LL(*) de parsing. Si a dicha gramática, se le añaden acciones escritas en un lenguaje de programación, el reconocedor se transforma en un traductor o interprete.
Hasta el dia de la fecha la ultima versión es la **3.0, sumado a un ambiente de trabajo** para la creación y edición de gramáticas para ANTLR llamado "ANTLRWorks".
ANTLRWorks es un entorno de desarrollo con interfaz gráfica que permite el desarrollo de gramáticas para la versión 3.0 o superior de ANTLR. Consiste en una aplicación independiente Java, que se puede ejecutar directamente desde un jar.
Actualmente ANTLR genera código Java, C, C++, C#, Python, Perl, Delphi, Ada95, JavaScript y Objective-C.
La pagina oficial para la descarga es: http://www.antlr.org/works/index.html
Allí encontrara una versión AntlrWorks que incluye todo lo del ANTLR mas la interfase grafica.

Pasos previos

Luego de la descarga, necesitará el paquete Java. Tener cuidado con esto, que no es lo mismo que lo que requieren los navegadores. Yo recomiendo descargar lo siguiente:

http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html

Que es un JDK que soporta base de datos Oracle. De esta manera se evitan problemas.

Inicio a la programación

Para arrancar nuestro compilador, dado que usamos java y todo se traducirá a este lenguaje podemos optar por meter todo el código del compilador dentro del fichero .g o escribirlo en ficheros aparte. Si lo hacemos en ficheros aparte deberemos compilar desde la línea de comandos todos los .java que hayamos añadido. Para hacerlo mas fácil, escribiremos todo el código en un solo fichero.

Vamos a dividir el programa en bloques principales:

Grammar *micompilador.g****;***

Con esto definimos el encabezado del fichero. ATENCION, el nombre del archivo debe tener este mismo nombre para que compile.

Tokens { … }

Acá se agregan todos los elementos que queremos que reconozca nuestro compilador (la gramática), tales como IGUAL = '='; SUMA = '+'; etc.

@header { … }

Permite insertar el código que queramos colocar al principio del fichero java correspondiente al analizador sintáctico.

@members { … }

Acá va el código de nuestra aplicación (en nuestro caso Java). Podemos poner un Main para procesar el archivo de entrada luego de que sea aceptado por nuestro compilador.
Permite insertar atributos y métodos definidos por el usuario en esa clase.

Reglas del parser

En este bloque va la sintaxis de nuestro compilador. Para que el compilador sepa que pertenecen a reglas del parser, debemos incluir encabezados en minúscula (como una etiqueta en assembler).

Ej: ***expr : ID IGUAL op { … };***
op: … ;

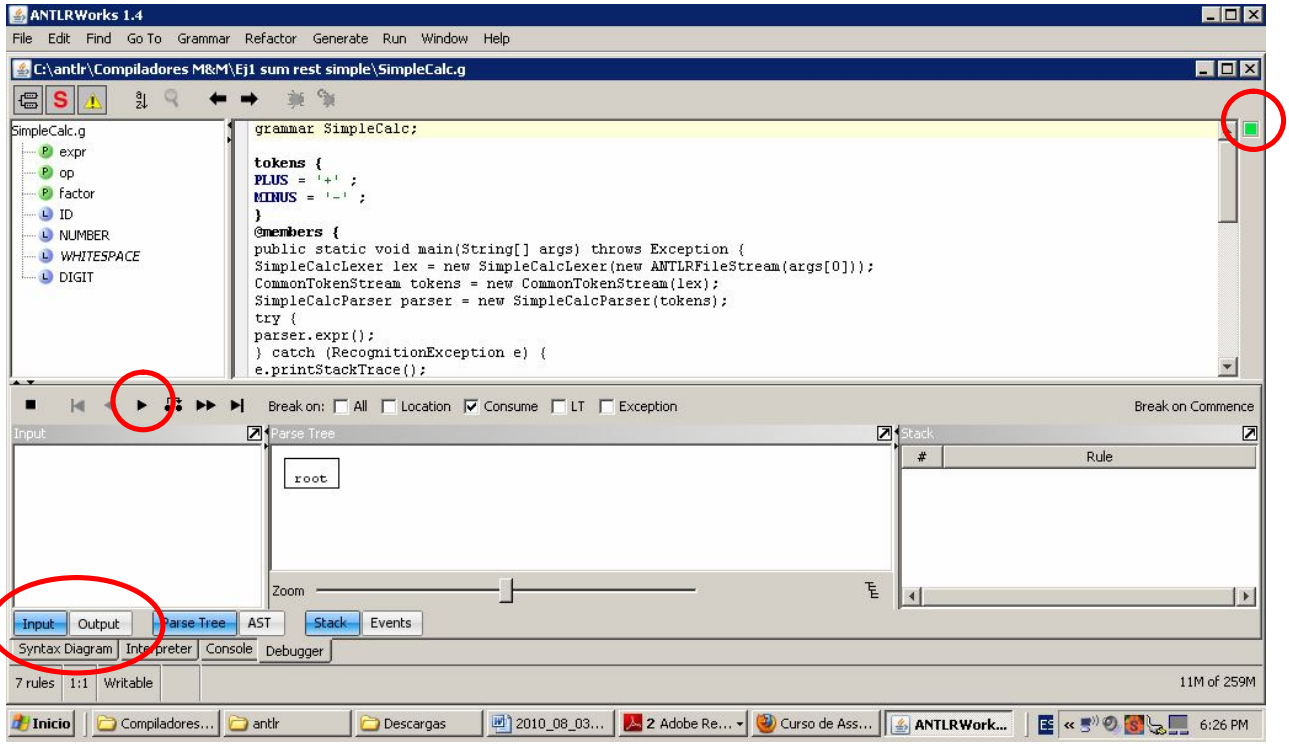
Reglas del lexer

En este bloque se pone el léxico, es decir, la declaración de los tipos. Para que el compilador sepa que pertenece a reglas del lexer, debemos incluir encabezados en minúscula (como una etiqueta en assembler).

Ej: ***ID : ('a'..'z'|'A'..'Z')+ ;***

Una vez escrito nuestro código, para generar el código del parser y el lexer, en el ANTLRWorks, vamos a Generate → Generate Code.
NOTA: a la derecha hay un cuadrado que debe estar verde, lo que indica que no hay errores. Si esta rojo, revisar las zonas donde el mismo programa nos marca con un subrayado rojo.
Esto nos creará dos ficheros java que los pondrá en una carpeta llamada Outputs en el directorio donde se encuentra nuestro fichero .g
Finalmente, para compilar el programa y probarlo ir a Run → Debug…

Esto hará que aparezca una pantalla de input donde tenemos dos opciones: elegimos Text para escribir nuestra entrada alli o bien elegimos File para indicarle donde esta nuestro archivo de texto con la entrada.



Para ir viendo paso a paso pulsar el simbolo ► que esta a mitad de pantalla.

Debajo de todo tambien en Syntax diagram podemos ir viendo como es el diagramaa en bloques de nuestro programa.
Finalmente en Input y Output tenemos 2 consolas para ir observando las entradas y las salidas de nuestro compilador.

Funciones claves:

Función para imprimir en pantalla Output

System.out.println($ID.text +"="+ $op.value);

Función para guardar datos en una tabla de memoria

variables.put(KEY, new Integer(INT));

Previamente hay que importar la clase en el encabezado **header** con: ***import java.util.HashMap;*** y luego en **members** definir alguna tabla de variables con: ***HashMap variables = new HashMap();***

Función para leer datos de la tabla de memoria

Integer v = (Integer)variables.get(KEY);

EJEMPLOS ***"Breve explicación"***

Ver ***anexo*** para ver el código.

Ejemplo 1

Archivo de entrada: myinput0.txt

a=6+4

Con el código de este ejemplo, lo que vamos hacer es que nuestro compilador reconozca la suma o resta de dos números y como salida si es exitoso, el resultado de la operación.

Output:
a=10

Archivo de entrada: myinput1.txt

b=4

```
a=3+b
```

Lo que hacemos ahora es tratar de guardar una variable y luego usarla como operador en una ecuación.
Como resultado podemos ver que no es posible ya que nuestro compilador solo esperaba una expresión, y además, no la estamos almacenando en ningún lado.

Ejemplo 2

Archivo de entrada: myinput1.txt

```
a=2
b=a+2
c=a+b+6
```

Ahora el código posee la tabla para almacenar variables y a su vez modificamos las reglas de la gramática para que puedan entrar varias variables. Ver **Syntax diagram** para entender mejor los cambios.

Output:
```
a=2
b=4
c=10
```

Archivo de entrada: myinput2.txt

```
b=4
a=6*b
```

Aquí lo que ocurre, instantáneamente que el compilador recibe el carácter *, nos indica que no pertenece al lenguaje y nos emite un mensaje de error.
Para solucionarlo, debemos agregar éste como un token.

Ejemplo 3

Archivo de entrada: myinput2.txt

En este ejemplo se agregó dicho token y en la parte de sintaxis le indicamos lo que debe hacer cuando reciba dicho token.

Output:
```
b=4
a=24
```

Archivo de entrada: myinput3.txt

Finalmente como prueba final vamos a pasar el archivo de entrada 3. En este caso, cuando llegue la palabra mover, la va a tomar como una variable y al llegar el paréntesis, el compilador nota que no es lo que espera y nos avisa con un error. Por ello en nuestro ejemplo final, vamos a definir la palabra mover como token y a su vez le indicamos el formato completo para que pueda reconocer esa sintaxis y así comenzar a definir nuestro compilador.

Output:
line 8:5 no viable alternative at character '('

***Ejemplo 4:* "Compilador puma"**

Archivo de entrada: myinput3.txt

Ahora ya definimos el token y definimos lo que queremos que haga cuando este llegue obteniendo como resultado:

```
Output:
a=1
b=2
c=3
d=4
e=5
f=6
v=10
EJEMPO DE TRAUCIR MI ENTRADA A CODIGO EN C
#include <math.h> ...
#define  PULSOS_POR_GRADO 150 ...
float CinematicaInversa( **char ); ...
Void Main (String[] args) {
int aux, ...
Trayectoria[n][3]=ArmarTrayectoria(1,2,3,4,5,6,10);
Q[n][6]=CinematicaInversa(Trayectoria[n][3]);
PwmMotor1(Q[n][1],10);
PwmMotor2(Q[n][2],10);
PwmMotor3(Q[n][3],10);
PwmMotor4(Q[n][4],10);
PwmMotor5(Q[n][5],10);
PwmMotor6(Q[n][6],10);
 ... Y asi hasta completar el código ... }
```

Cabe aclarar que además de esto, tendría que salir la definición de las funciones, variables, y el código de las funciones para que un compilador como el Gcc pueda compilar sin errores. Esto nada mas es una demostración muy simple de cómo se haría.

CONCLUCION

El analizar detalladamente esta manipulador tanto desde la cinemática como desde la dinámica, nos permitió comprender a fondo su funcionamiento.
Además pusimos en practica los conceptos teóricos que incorporamos a lo largo de la cursada. Los cuales reforzamos al realizar esta tesis.
Pudimos apreciar el potencial del Robotic toolbox de Matlab, el cual utilizamos a lo largo del desarrollo de la tesis para plotaer áreas de trabajo, calcular trayectoria, caculos dinámicos directos e inversos, etc. Lo que nos permitió afirmar que los desarrollos teóricos eran correctos.
El mayor desafío fue realizar el compilador, ya que optamos por utilizar una herramienta nueva, el ANTLR, siendo esta de uso actual, a pesar de no estar familiarizados con esta ni con su lenguaje de programación (Java).
A pesar de esto gracias a varios ejemplos encontrados en la red y algunos tutoriales pudimos ir avanzando y entendiendo algunos pasos básicos para así alcanzar los objetivos.
Si bien aún quedan varios puntos pendientes, este trabajo tiene una introducción de los programas que requiere su sistema para poder ejecutar el Antlr sin problemas y algunos ejemplos claves bien resumidos para entender como continuarlo sin perder tiempo. Básicamente lo que resta es lenguaje en C del cual estamos todos familiarizados.

Resumiendo, el realizar esta tesis nos puso a prueba ya que tuvimos que aprender a utilizar nuevas herramientas en corto tiempo, lo que implico gran esfuerzo, pero de igual magnitud es la gratificación de haber concluido el trabajo.
Sin embargo, algunos de los resultados obtenidos seguramente serán mejoras por cursadas posteriores, por lo cual hemos tratado de documentar lo mejor posible todos los temas desarrollados.

BIBLIOGRAFIA

Lee_puma-CinematicaDirecta.pdf (se respeto la numeración de formulas y graficas)

TPs 2009

Robotic Toolbox

The Definitive ANTLR Reference

http://www.antlr.org/

http://es.wikipedia.org/wiki/Antlr

http://www.albertnogues.com

http://www.java-tips.org/java-se-tips/java.util/how-to-use-of-hashmap.html

http://javadude.com/articles/antlr3xtut/

http://www.antlr.org/works/help/tutorial/calculator.html