TRABAJO PRACTICO Nº 1

Implementación de una matriz cinemática en DSP

Alumnos:

- **Jorge, Asade.**
- **Rosende, Alejandro.**
- **Melisa, Villafañe.**

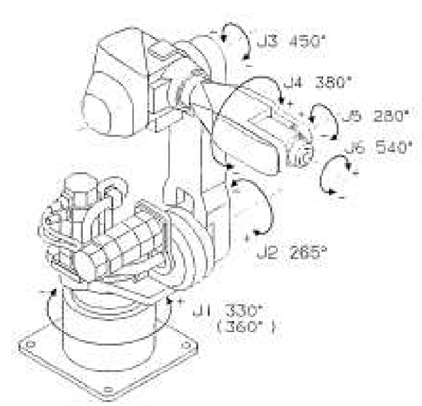
Profesor: Ing. H. Giannetta.

Implementación de una matriz cinemática en DSP

Cinemática del robot

Introducción

Un **robot** por definición es un manipulador multifuncional reprogramable capaz de mover materia con determinados grados de libertad, según trayectorias variables para realizar diversas tareas.



Se llama **grados de libertad** a cada uno de los movimientos que puede realizar una articulación respecto de la anterior.

Las **transmisiones** son las encargadas de llevar el movimiento desde los actuadores hasta las articulaciones.

Los **reductores** se utilizan para adaptar la velocidad y el par de la salida del actuador a valores adecuados para el movimiento. La relación entre el par y velocidad de salida respecto de la entrada en un actuador está dada por:

$$Ts = \eta\, Te \, \frac{\omega e}{\omega s}$$

Dónde:
- **T:** Par
- **ω:** Velocidad
- **η:** Rendimiento
- **e:** Entrada
- **s:** Salida

Los **actuadores** son los encargados de generar el movimiento de los elementos del robot según las ordenes dadas por la unidad de control. Se deben considerar las siguientes características:

- Potencia
- Controlabilidad
- Peso
- Volumen
- Precisión

- Velocidad
- Mantenimiento
- Costo

Los **actuadores** se pueden clasificar en:

- Actuadotes neumáticos
    - Cilindros o pistones neumáticos
    - Motores neumáticos

- Actuadores hidráulicos (funcionalmente en general no se distinguen de los neumáticos, sólo que en vez de utilizar aire, actuan con líquidos que en general son aceites.)

- Motores eléctricos
    - Motores de CC.
    - Motores de CA.
    - Paso a paso.

Los **sensores** se encargan de brindar la información precisa del estado (posición, velocidad, etc) a la unidad de control.

Para tener información de la posición de un rotor se utilizan los **encoders** y **resolvers.**

Básicamente un **encoder** está constituido por un disco codificado (solidario con el eje a controlar) con marcas (agujeros en su superficie) que son atravezadas por un rayo de luz, al cortarse o no el haz el encoger emite una señal cuadrada que deberá interpretar el controlador, tomando su defasaje para saber el sentido de giro y tomando la cantidad de pulsos para saber su posición.

Un **resolver** son 3 bobinas, una solidaria a la rotación con una portadora de 400 hz y otras 2 fijas al rededor, al girar la primera el acoplamiento con las otras dos varía, lo que hace que la señal resultante dependa del seno del ángulo de giro.

Los **elementos terminales** o **efectores** son los encargados de interactuar con el entorno del robot o materia, pueden ser de aprehensión, herramientas, etc.

Herramientas matemáticas

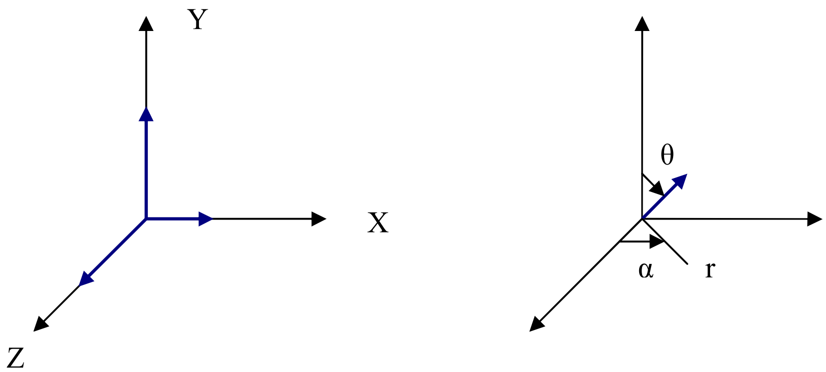
Para ubicar un punto en el espacio podemos utilizar 2 métodos:

- Coordenadas cartesianas

- Coordenadas polares

Las primeras utilizan 3 vectores que agrupándolos de a 2 forman un plano transversal al otro grupo de 2 vectores, y lograr ubicar un punto en el espacio con una componente de cada uno de ellos.

Las coordenadas polares ubican el punto con un determinado radio que se desplaza con 2 ángulos desde el origen.

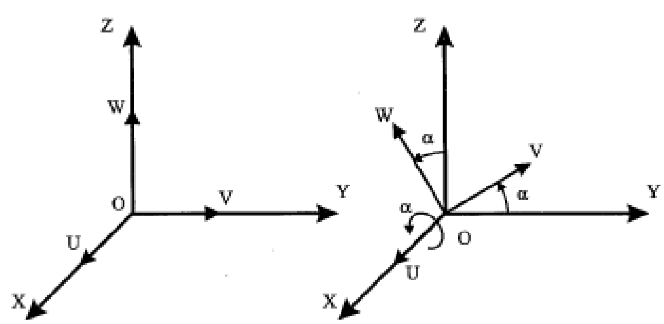


Una forma de representar y operar con los componentes de éstos sistemas de coordinas es utilizando **matrices.** Por ejemplo cuando queremos hacer girar un sistema respecto de otro utilizamos la **matriz rotación.**

Matriz rotación

La matriz rotación define la orientación de un sistema respecto de otro, veamos un ejemplo en el que se desea hacer rotar el sistema OXYZ sobre el eje X para llegar al sistema OUVW:

$$\mathbf{R}(\mathbf{x},\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\mathrm{sen}\alpha \\ 0 & \mathrm{sen}\alpha & \cos\alpha \end{bmatrix}$$



También podemos lograr un movimiento haciendo varias rotaciones, lo que se llama composición de rotaciones, cabe aclarar que no es lo mismo hacer las rotaciones descuidando el orden de las mismas, veamos un ejemplo

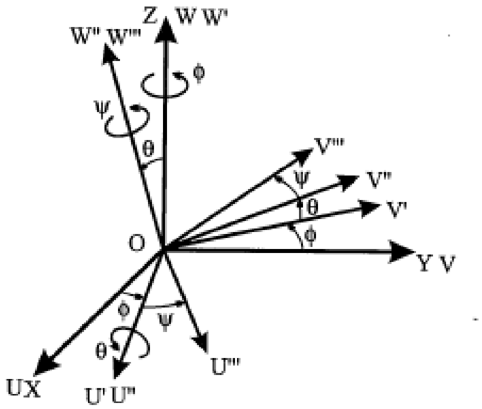
$$T = R(x,\alpha)\, R(y,\Phi)\, R(z,\theta) =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{pmatrix} \begin{pmatrix} c\Phi & 0 & s\Phi \\ 0 & 1 & 0 \\ -s\Phi & 0 & c\Phi \end{pmatrix} \begin{pmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c\Phi & 0 & s\Phi \\ s\alpha\, s\Phi & c\alpha & -s\alpha\, c\Phi \\ c\alpha\, s\Phi & s\alpha & c\alpha\, c\Phi \end{pmatrix} \begin{pmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c\Phi\, c\theta & -c\Phi\, s\theta & 1 \\ s\alpha\, s\Phi\, c\theta + c\alpha\, s\theta & -s\alpha\, s\Phi\, s\theta + c\alpha\, c\theta & -s\alpha\, c\Phi \\ -s\alpha\, s\Phi\, c\theta + s\alpha\, s\theta & s\alpha\, s\Phi\, s\theta + s\alpha\, c\theta & c\alpha\, c\Phi \end{pmatrix}$$

Existe otro método para realizar composición de rotaciones que son los **ángulos de Euler**, dónde básicamente se realizan giros sobre ejes ya girados:

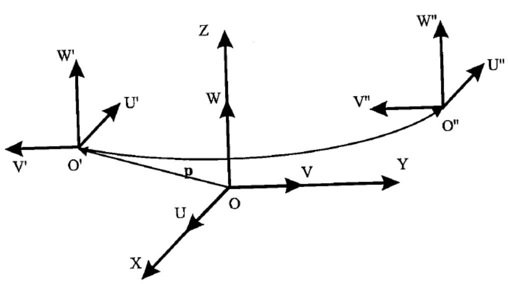


Aquí lo que se hizo fue:

- Girar OUVW un águlo α respecto de Z hasta llegar a OU'V'W'.
- Girar OU'V'W' un águlo θ respecto de U' hasta llegar a OU''V''W''.
- Y así sucesivamente.

<u>Matriz traslación</u>

De la misma manera que la matriz rotación, con la matriz traslación logramos el movimiento de un sistema de referencia a otro.



La matriz traslación se compone de la siguiente manera:

x' = x + kx

y' = y + ky

z' = z + kz

$$\begin{pmatrix} x' \\ y' \\ \\ \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & Kx \\ 0 & 1 & 0 & Ky \\ & & & \end{pmatrix} \begin{pmatrix} x \\ y \\ \\ \end{pmatrix}$$

z’	0	0	1	Kz	z
1	0	0	0	1	1

Matrices homogéneas

La representación de un punto en un espacio n-dimensional se realiza con n+1 dimensiones para las matrices homogéneas. Por ejemplo: supongamos un sistema oXYZ que contiene un vector **p**(x,y,z) = a i + b j + c k, se representará con el vector **p (**a w, b w, c w, w**)** dónde w tiene un valor arbitrario**:**

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} aw \\ bw \\ cw \\ w \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}$$

Entonces podemos definir a la **matriz de transformación homogénea T**, la cual sirve para transformar un sistema de matrices homogéneas en otro.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ \mathbf{f}_{1x3} & \mathbf{w}_{1x1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix}$$

Está compuesta por 4 matrices para determinar la rotación y traslado de un sistema (esto es considerando nula la perspectiva y el esclado cómo sucede en las aplicaciones de robótica), de ésta manera la matriz T nos queda:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ 0 & 1 \end{bmatrix}$$

Cómo conclusión la matriz de transformación homogénea se utiliza para:

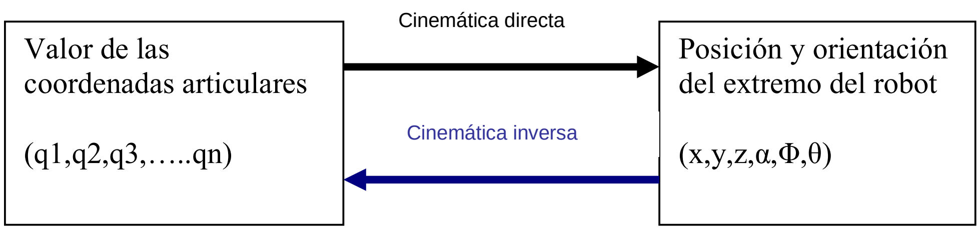
- Representar la posición y orietneación de un sistema OUVW respecto de un sistema fijo OXYZ.
- Transformar un vector de un sistema a otro.
- Rotar y trasladar un vector respecto de un sistema fijo OXYZ.

Cinemática del robot

La cinemática del robot estudia los movimientos del mismo respecto de un sistema de referencia. Se estudian dos problemas fundamentales:

- Cinemática inversa
- Cinemática directa

La **cinemática directa** determina la posición y orientación del extremo final del robot para un sistema de referencia. La **cinemática inversa** determina la configuración de un robot para una posición y orientación conocida.



El problema cinemática directo se puede resolver con los siguientes métodos:

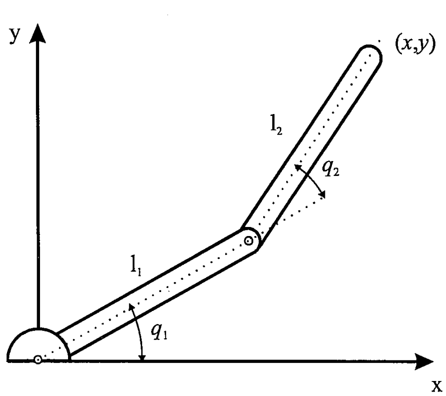
- Matriz de transformación homogénea.
- Algoritmo Denavit-Hartenberg
- Cuaternios

<u>Cinemática directa por Matriz Homogenea</u>

Utilizando las coordenadas cartesianas y los ángulos de Euler, con las siguientes relaciones representamos la orientación y posición de los elementos del robot de n grados de libertad.

x = fx (q1,q2,…….qn)
y = fy (q1,q2,…….qn)
z = fz (q1,q2,…….qn)
α = fα (q1,q2,…….qn)
θ = fθ (q1,q2,…….qn)
Φ = fΦ (q1,q2,…….qn)

Entonces, tomamos por ejemplo un robot de 2 grados de libertad como el de la siguiente figura, que tendrá como ecuaciones:



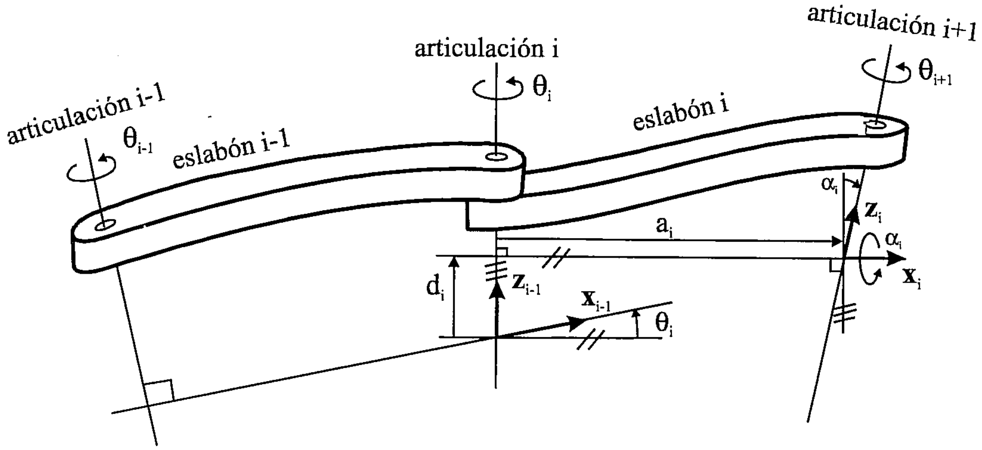
x = l1 cos q1 + l2 cos (q1+q2)

y = l1 sen q1 + l2 sen (q1+q2)

Luego aplicamos los métodos de matrices de rotación y transformación para ver la evolución del movimiento.

Algoritmo de Denavit-Hartenberg

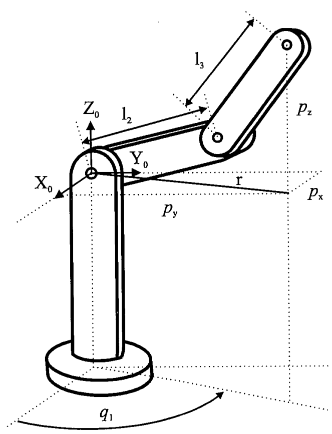
1. Numerar los eslabones de la cadena comenzando con 1 hasta el n, la base fija del robot se considera el eslabón 0.
2. Numerar las articulaciones de 1 (primer grado de libertad) hasta n.
3. Localizar el eje de cada articulación, si es rotativa el eje será su propio eje de giro, si es prismática será el eje sobre el que se produce el movimiento.
4. Para i de 0 a n-1, situar el eje Zi sobre la articulación i+1.
5. Situar la base del sistema sobre el eje Z0.
6. Para i de 1 a n-1, situar el sistema Si solidario al eslabon i, en la intersección del eje Zi con la línea normal a Zi-1. Si los ejes se cortan se situa en Si, si son paralelos en la articulación i+1.
7. Situar xi en la línea normal común a zi y zi-1.
8. Situar Yi de forma tal que forme un sistema dextrógiro con Zi y Xi.
9. Situar el sistema Sn en el extremo del robot de forma tal que Zn coincida con la dirección de Zn-1 y Xn sea normal a Zn y Zn-1.
10. θi será el ángulo con el que hay que girar en torno a Zi-1 para que xi-1 y x queden paralelos.
11. di es la distancia que habría que desplazar Si-1 a lo largo de Zi-1 para que Xi y Xi-1 queden alineados.
12. ai es la distancia a a lo largo de Xi que habría que desplazar el nuevo Si-1 para quedase cómo Si.
13. αi es el ángulo que habría que girar sobre Xi para que Si-1 coincida con Si.
14. Matrices de transformación (i-1)Ai.
15. Obtener la matriz transformación que relaciona la base con el extremo del robot. T = 0A1 1A2…(n-1)An.
16. T defina la orientación y posición del extremo referido a la base en función de las n coordenadas.



Cinemática Inversa por método geométricos

Consiste en encontrar un determinado y suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot.

Por ejemplo:



$$q_1 = \mathrm{arctg}\left(\frac{p_y}{p_x}\right)$$

$$r^2 = p_x^2 + p_y^2$$

$$r^2 + p_z^2 = l_2^2 + l_3^2 + 2l_2l_3\cos q_3$$

$$\cos q_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2l_2l_3}$$

También se puede resolver por el método de las **matrices de transformación ya visto.**

Matriz Jacobbiana

Se utiliza para determinar la relación entre la velocidad de las coordenadas articulares y de posición y orientación de un robot.

Velocidad de las articulaciones (*q1,*q2,*q3,…..*qn)	Cinemática directa → ← Cinemática inversa	Velocidad de Posición y orientación del extremo del robot (*x,*y,*z,*α,*Φ,*θ)

Se forman las siguientes ecuaciones en función a los grados de libertad:

x = fx (q1,q2,…….qn)
y = fy (q1,q2,…….qn)
z = fz (q1,q2,…….qn)
α = fα (q1,q2,…….qn)
β = f β (q1,q2,…….qn)
σ = fσ (q1,q2,…….qn)

$$\dot{x} = \sum_{1}^{n} \frac{\partial f_x}{\partial q_i} \dot{q}_i \qquad \dot{y} = \sum_{1}^{n} \frac{\partial f_y}{\partial q_i} \dot{q}_i \qquad \dot{z} = \sum_{1}^{n} \frac{\partial f_z}{\partial q_i} \dot{q}_i$$

$$\dot{\alpha} = \sum_{1}^{n} \frac{\partial f_\alpha}{\partial q_i} \dot{q}_i \qquad \dot{\beta} = \sum_{1}^{n} \frac{\partial f_\beta}{\partial q_i} \dot{q}_i \qquad \dot{\gamma} = \sum_{1}^{n} \frac{\partial f_\gamma}{\partial q_i} \dot{q}_i$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} \dot{q}_1 \\ \\ \vdots \\ \vdots \\ \\ \dot{q}_n \end{bmatrix} \qquad \text{con} \quad \mathbf{J} = \begin{bmatrix} \dfrac{\partial f_x}{\partial q_1} & \cdots & \dfrac{\partial f_x}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_\gamma}{\partial q_1} & \cdots & \dfrac{\partial f_\gamma}{\partial q_n} \end{bmatrix}$$

Entonces el valor de la matriz será diferente en cada punto del espacio.

Desarrollo e implementacion en Codewarrior DSP 56800-E

Matriz a implementar:

$$
{}^{0}\mathbf{A}_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad
{}^{1}\mathbf{A}_2 = \begin{bmatrix} C_2 & 0 & -S_2 & 0 \\ S_2 & 0 & C_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad
{}^{2}\mathbf{A}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}\mathbf{A}_2 = \begin{bmatrix} C_1C_2 & -S_1 & -C_1S_2 & 0 \\ S_1C_2 & C_1 & -S_1S_2 & 0 \\ S_2 & 0 & C_2 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad
\mathbf{T} = {}^{0}\mathbf{A}_3 = \begin{bmatrix} C_1C_2 & -S_1 & -C_1S_2 & -q_3\,C_1S_2 \\ S_1C_2 & C_1 & -S_1S_2 & -q_3\,S_1S_2 \\ S_2 & 0 & C_2 & q_3\,C_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

```
/** ###################################################################
**     Filename  : TPN1.C
**     Project   : TPN1
**     Processor : 56F8367
**     Version   : Driver 01.12
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 10/05/2009, 19.19
** ###################################################################*/
/* MODULE TPN1 */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "TFR1.h"
#include "MFR1.h"
#include "MEM1.h"

#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include "stdio.h"

//********************************************************************
//          DEFINE
//**********************************************************************
#define LON 200

//********************************************************************
//          GLOBAL VARIABLE
//********************************************************************
        Frac16 i1;
        Frac16 x,y,z,s1,s2,c1,c2,l3;
        Word16 angulo1;
        Word16 angulo2;
```

```
//*******************************************************************
//          MAIN
//*******************************************************************

void main(void)
{
/* Write your local variable definition here */
/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
/*** End of Processor Expert internal initialization.                    ***/
/* Write your code here */
for(;;)
        {
        for(i1=0;i1<MAX;i1++)
                {
                angulo1=(extract_l(L_mult((32767/360),i1)));  // mutiplico 2 de 16bits -> 32bits L_mult
                angulo2=(extract_l(L_mult((32767/360),i1)));  // extraigo a 16 con extract_l
                l3=(extract_l(L_mult((32767/360),i1)));

                c1=TFR1_tfr16CosPIx(angulo1);  // se uso el mismo angulo para los 2 giros
                c2=TFR1_tfr16CosPIx(angulo2);
                s1=TFR1_tfr16SinPIx(angulo1);
                s2=TFR1_tfr16SinPIx(angulo2);

                x=negate(extract_l(L_mult(extract_l(L_mult(l3,c1)),s2)));  // desplazamiento en X
                y=negate(extract_l(L_mult(extract_l(L_mult(l3,s1)),s2)));  // desplazamiento en Y
                z=add(extract_l(L_mult(l3,c2)),LON); // desplazamiento en Z

                printf ("%d \t %d \t %d \t %d \t \n",i1,x,y,z);
                }
        }
}
```

Resultados de la simulación

La respuesta de la simulación la ingresamos al Mathlab para graficarla y obtuvimos: