Robótica – T.P. Nº1

Implementación de una matriz cinemática en DSP

Docente: Giannetta, Hernan

Alumnos: Almela, Alejandro
Bustos, Mariano
Gil, Gustavo

Índice de contenidos

Introducción sobre cinemática ---------- *Pag. 3*

Cinemática Directa ---------- *Pag. 5*

Resolución del problema cinemático directo mediante matrices de transformación homogénea ---------- *Pag. 6*

Desarrollo de la Práctica ---------- *Pag. 10*

Gráficos del sólido de Movimiento ---------- *Pag. 11*

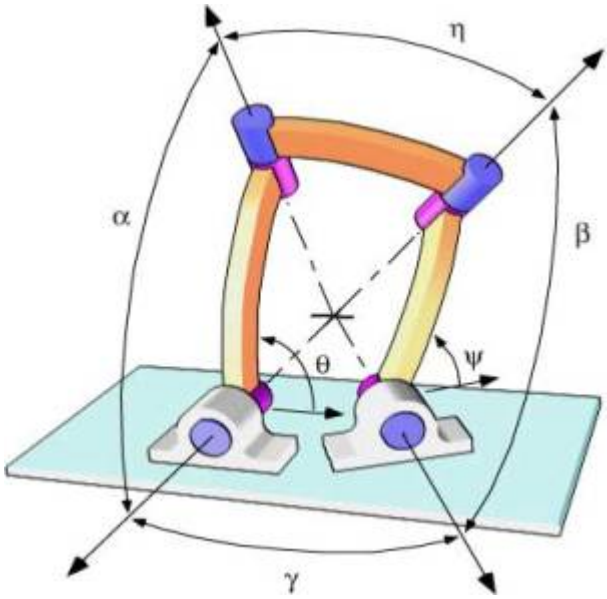
Código Matlab ---------- *Pag. 14*

Código DSP ---------- *Pag. 15*

Conclusiones Finales ---------- *Pag. 19*

Introducción sobre cinemática

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares. Existen dos problemas fundamentales para resolver la cinemática del robot, el primero de ellos se conoce como el **problema cinemático directo**, y consiste en determinar cual es la posición y orientación del extremo final del robot, 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, el segundo denominado **problema cinematico inverso** resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas.



Denavit y Hartenberg propusieron un método sistemático para descubrir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para descubrir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinematico directo a encontrar una matriz de transformación homogénea 4 X 4 que relacione la localización espacial del robot con respecto al sistema de coordenadas de su base.

Por otra parte, la cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo. Esta relación viene dada por el **modelo diferencial** expresado mediante la matriz Jacobiana.

Diagrama entre cinematica directa e inversa.

	Cinemática directa ->->	
Valor de las coordenadas Articulares (q0, q1, ... qn)		posición y orientación del extremo del robot (x, y, z, α, ß, γ)
	<-<- Cinemática inversa	

El movimiento relativo en las articulaciones resulta en el movimiento de los elementos que posicionan la mano en una orientación deseada. En la mayoría de las aplicaciones de robótica, se esta interesado en la descripción espacial del efector final del manipulador con respecto a un sistema de coordenadas de referencia fija.

La cinemática del brazo del robot trata con el estudio analítico de la geometría del movimiento de un robot con respecto a un sistema de coordenadas de referencia fijo como una función del tiempo sin considerar las fuerzas-momentos que originan dicho movimiento. Así pues, trata con la descripción analítica del desplazamiento espacial del robot como función del tiempo, en particular las relaciones entre variables espaciales de tipo de articulación y la posición y orientación del efector final del robot.

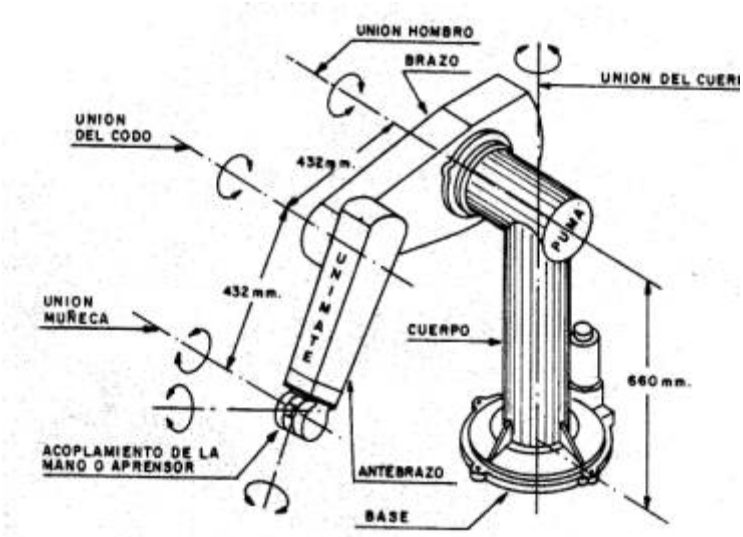


Fig. 1.7.–Esquema del manipulador correspondiente al robot PUMA 600 de UNIMA TION, con indicación del nombre de sus elmentos y el de sus articulaciones, así com la especificación de los movimientos posibles.

Aumentando la destreza de robots repetitivos.

Se usan las coordenadas redundantes para definir tareas adicionales.

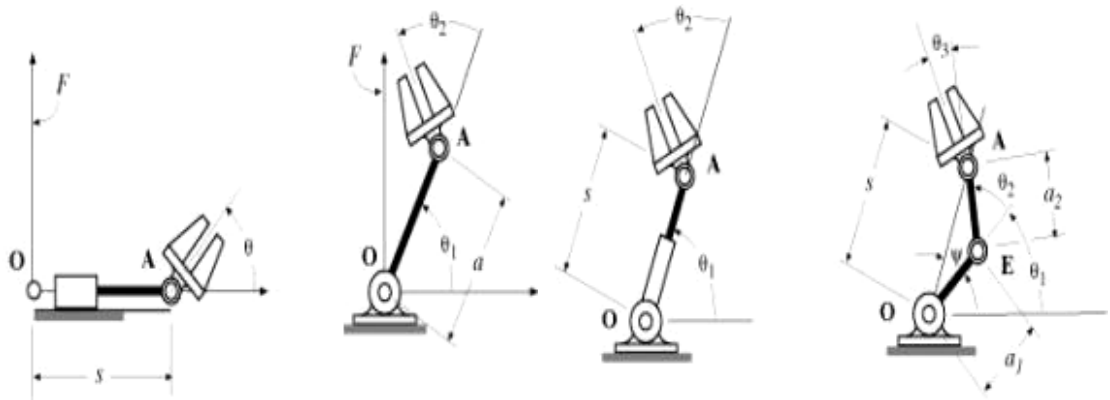
El mando de configuración está surgiendo como una manera eficaz de controlar los movimientos de un robot que tiene más grados de libertad y en el cual es necesario

definir la trayectoria del efector del extremo y / o el objeto para ser manipulado. Pueden usarse los grados extras o redundantes de libertad para dar destreza de robot y versatilidad. En mando de configuración, la configuración del robot se representa matemáticamente por un juego de variables de configuración que son un vector de coordenadas generalizado y que es más pertinente a la tarea global que es el vector de coordenadas de la juntura que aparecen en los acercamientos convencionales a controlar. El vector de la coordenada generalizado consiste en las coordenadas del efector del extremo en el espacio de la tarea, más varias funciones de cinemática que involucran grados redundantes de libertad. La tarea básica del sistema de mando es hacer las coordenadas del efector del extremo seguir la trayectoria deseada. Las funciones de la cinemática pueden seleccionarse para definir una tarea adicional por ejemplo, la anulación de obstáculos u optimización de la cinemática para reforzar la manipulabilidad. En efecto, la tarea adicional define la trayectoria en los grados redundantes de libertad. Las variables de configuración pueden usarse en un esquema de mando adaptable que no exige manipular el conocimiento del modelo matemático complicado de la dinámica del robot o los parámetros del objeto.

Cinemática Directa.

El problema cinemático directo.

Se utiliza fundamentalmente el álgebra vectorial y matricial para representar y describir la localización de un objeto en el espacio tridimensional con respecto aun sistema de referencia fijo. Dado que un robot puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. De esta forma, el problema cinematico directo se reduce a encontrar una matriz homogénea de transformación **T** que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz **T** será función de las coordenadas articulares.

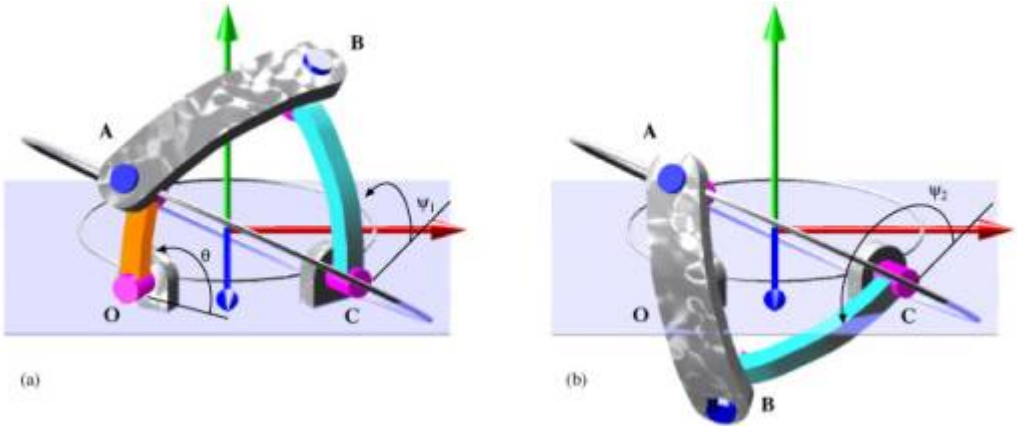


El mando adaptable de un manipulador remoto.

Un sistema de mando de robot causa a un manipulador remoto, seguir una trayectoria de referencia estrechamente en un marco de referencia Cartesiano en el espacio de trabajo, sin el recurso a un modelo matemático intensivo de dinámica del robot y sin el conocimiento del robot y parámetros de carga. El sistema, derivado de la teoría lineal multivariable, utiliza a los manipuladores delanteros relativamente simples y controladores de retroalimentacion con modelo y adaptable de referencia del mando. El sistema requiere dimensiones de posición y velocidad del extremo manipulador del efector. Éstos pueden obtenerse directamente de los sensores ópticos o por cálculo que utiliza las relaciones de la cinemática conocidas entre el manipulador modelado y el extremo de la juntura de la posición del efector. Derivando las ecuaciones de control, las ecuaciones diferenciales no lineales acopladas a la dinámica del robot, expresan primero la forma general de la cinemática, entonces la linealización por cálculo de perturbaciones sobre una especifica operación del punto en las coordenadas Cartesianas del extremo del efector. El modelo matemático resultante es un sistema multivariable lineal de orden de 2n (donde n = es el número de coordenadas espaciales independientes del manipulador) esto expresa la relación entre los incrementos del actuador de n voltajes de control (las entradas) y los incrementos de las coordenadas de n, la trayectoria de extremo del efector (los rendimientos). La trayectoria del efector incrementa la referencia, la trayectoria se incrementa: esto requiere la retroalimentación independiente y controladores de manipulación. Para este propósito, le basta aplicar posición y retroalimentación de velocidad a través de la matriz de n x n posición y velocidad, la matriz de ganancia de retroalimentación.

Resolución del problema cinemático directo mediante matrices de transformación homogénea.

La resolución del problema cinematico directo consiste en encontrar las relaciones que permiten conocer la localización espacial del extremo del robot a partir de los valores de sus coordenadas articulares.



Así, si se han escogido coordenadas cartesianas y ángulos de Euler para representar la

posición y orientación del extremo de un robot de seis grados de libertad, la solución al problema cinematico directo vendrá dada por las relaciones:

$$x = Fx\ (\ q1,q2,q3,q4,q5,q6\ )$$
$$y = Fy\ (\ q1,q2,q3,q4,q5,q6\ )$$
$$z = Fz\ (\ q1,q2,q3,q4,q5,q6\ )$$
$$\alpha = F\alpha\ (\ q1,q2,q3,q4,q5,q6\ )$$
$$ß = Fß\ (\ q1,q2,q3,q4,q5,q6\ )$$
$$\gamma = F\gamma\ (\ q1,q2,q3,q4,q5,q6\ )$$

La obtención de estas relaciones no es en general complicada, siendo incluso en ciertos casos (robots de pocos grados de libertad) fácil de encontrar mediante simples consideraciones geométricas. Por ejemplo, para el caso de un robot con 2 grados de libertad es fácil comprobar que:

$$X = I1\ \cos q1 + I2\ \cos(\ q1 + q2\ )$$
$$y = I1\ \cos q1 + I2\ \cos(\ q1 + q2\ )$$

Para robots de mas grados de libertad puede plantearse un método sistemático basado en la utilización de las matrices de transformación homogénea.
En general, un robot de n grados de libertad esta formado por n eslabones unidos por n articulaciones, de forma que cada par articulación-eslabón constituye un grado de libertad. A cada eslabón se le puede asociar un sistema de referencia solidario a el y, utilizando las transformaciones homogéneas, es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen el robot.

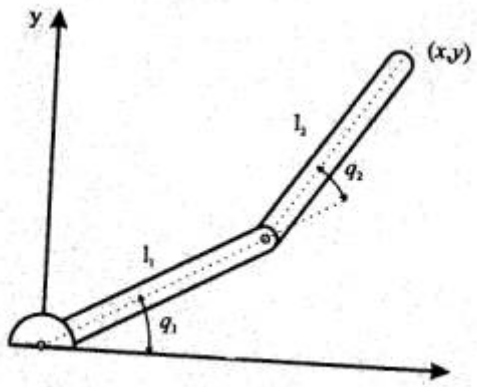


Figura 4.2. Robot planar de 2 grados de libertad.

Normalmente, la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se le suele denominar (i-1)1/Ai. Así pues, 0Ai describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, 1A2 describe la posición y orientación del segundo eslabón respecto del primero, etc. Del mismo modo, denominando 0Ak a las matrices resultantes del producto de las matrices (i-1)Ai con i desde 1 hasta k, se puede representar de forma total o parcial la cadena cinemática que forma el robot. Así, por ejemplo, la posición y orientación del sistema solidario con el segundo eslabón del robot con respecto al sistema de coordenadas de la base se puede expresar mediante la matriz 0A2:

$$0A2 = 0A1 (1A2)$$

De manera análoga, la matriz 0A3 representa la localización del sistema del tercer eslabón:

$$0A3 = 0A1 (1A2)(2A3)$$

Cuando se consideran todos los grados de libertad, a la matriz 0An se le suele denominar T. Así, dado un robot de seis grados de libertad, se tiene que la posición y orientación del eslabón final vendrá dada por la matriz T:

$$T = 0A6 = 0A1 (1A2)(2A3)(3A4)(4A5)(5A6)$$

Aunque para descubrir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada elemento, la forma habitual que se suele utilizar en robótica es la representación de Denavit-Hartenberg.

Denavit-Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas (Si) ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación D-H, escogiendo adecuadamente los sistemas de coordenadas asociados para cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.
Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento i con el sistema del elemento i-1. Las transformaciones en cuestión son las siguientes:

1. Rotación alrededor del eje Zi-1 un ángulo $\theta\iota$.
2. Traslación a lo largo de Zi-1 una distancia di; vector di (0,0,di).
3. Traslación a lo largo de Xi una distancia ai; vector ai (0,0,ai).
4. Rotación alrededor del eje Xi, un ángulo αi.

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$i\text{-}1A\ i = T(z,\theta i) T(0,0,di) T (ai,0,0) T(x,\alpha i)$$

Y realizando el producto de matrices:

$$
{}^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =
$$

$$
= \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [4.7]
$$

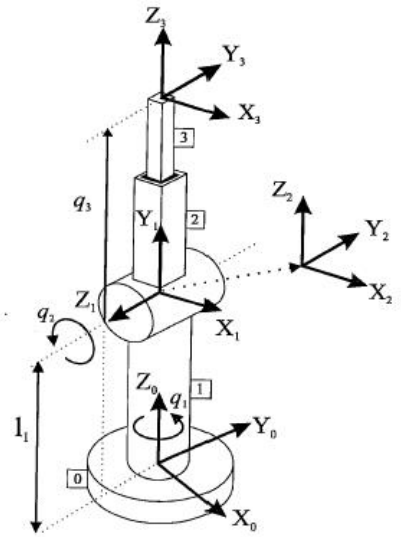
donde θi, ai, di, αi, son los parámetros D-H del eslabón i. De este modo, basta con identificar los parámetros θi, ai, di, αi , para obtener matrices A y relacionar así todos y cada uno de los eslabones del robot.

Como se ha indicado, para que la matriz i-1Ai, relacione los sistemas (Si) y (Si-1), es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Estas, junto con la definición de los 4 parámetros de Denavit-Hartenberg, conforman el siguiente algoritmo para la resolución del problema cinemático directo.

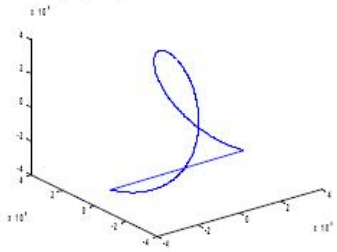
Desarrollo de la practica

- Implemente el código C en CW para el DSP56800/E de la cadena cinemática directa de la figura, usando la matriz homogénea , e usando como setpoint , una trayectoria lineal continua a cada eje. Defina los limites y área de trabajo del manipulador.

Articulación	θ	d	a	α
1	q1	L1	0	90
2	q2	0	0	-90
3	0	q3	0	0



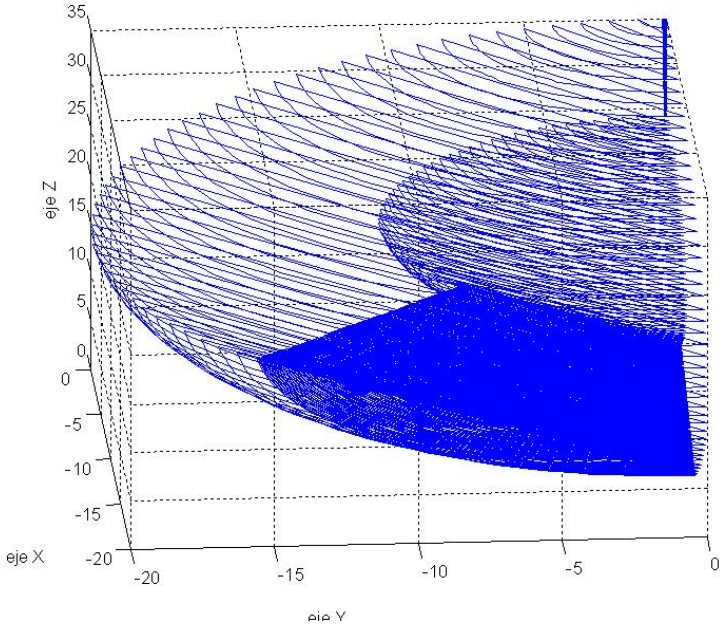
- Imprima el resultado del vector (x,y,z) usando la funcion plot3(x,y,z) de matlab

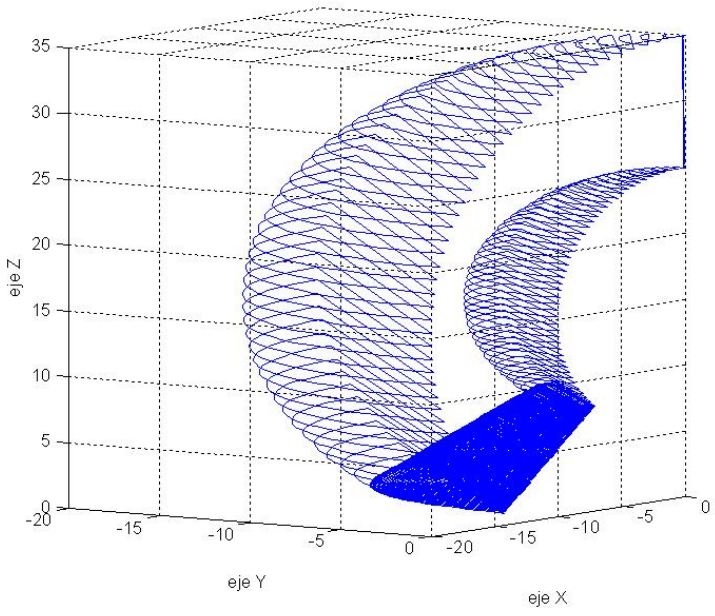


Gráficos del sólido de movimiento del manipulador (generados en Matlab):

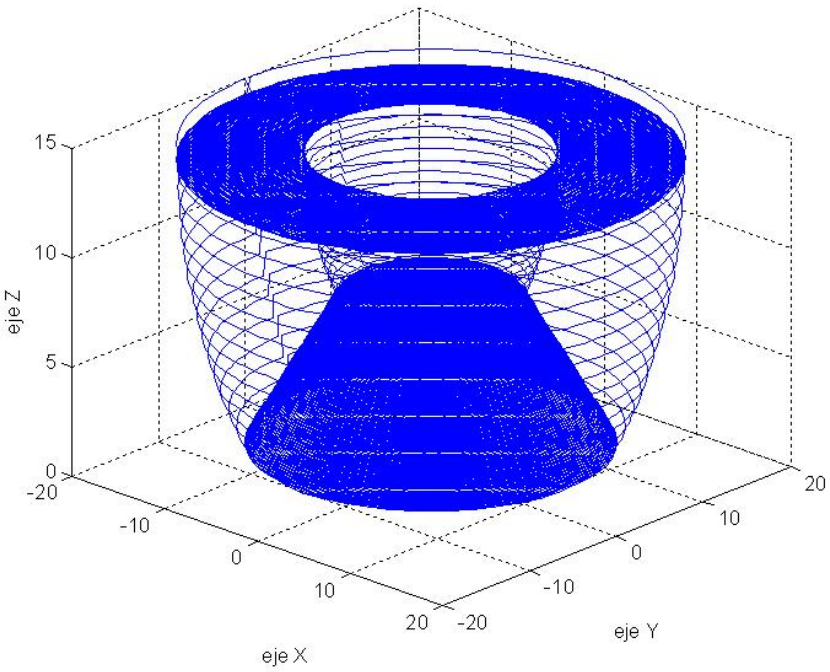
Para éstas restricciones:

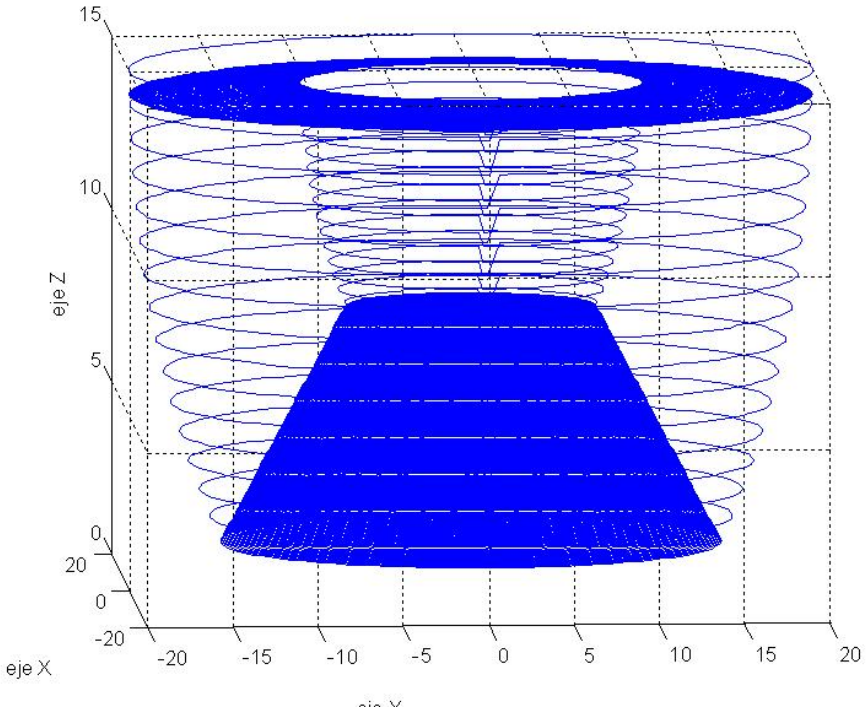
```
l1=15;          % Largo de cuepo 1

q1m=0;          % Angulo q1 minimo
q1M=pi/2;       % Angulo q1 MAXIMO
q2m=pi/4;       % Angulo q2 minimo
q2M=pi;         % Angulo q2 MAXIMO
q3m=10;         % Extension q3 minimo
q3M=20;         % Extension q3 MAXIMO
```



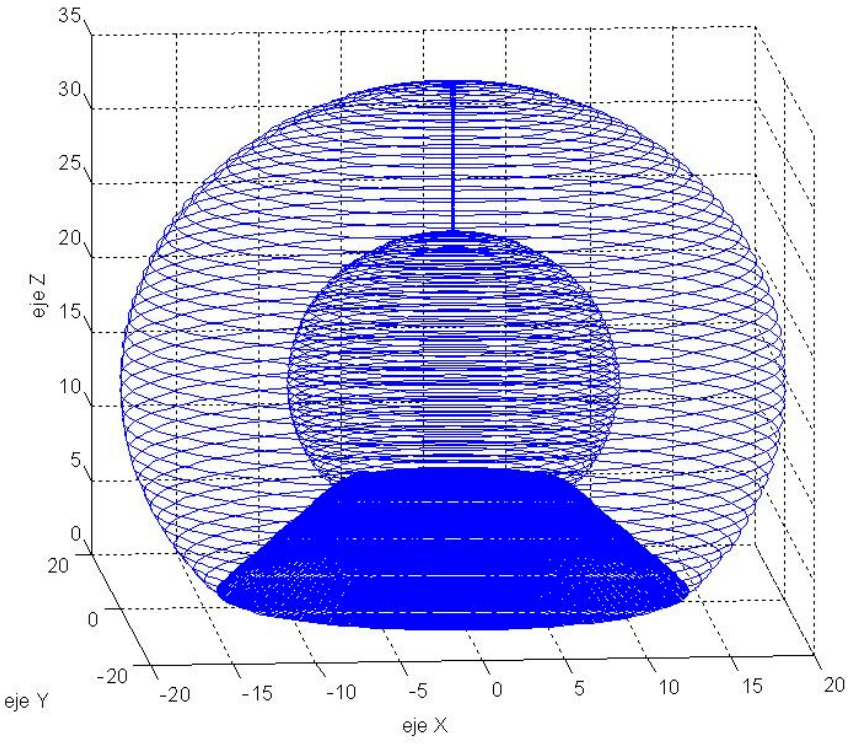


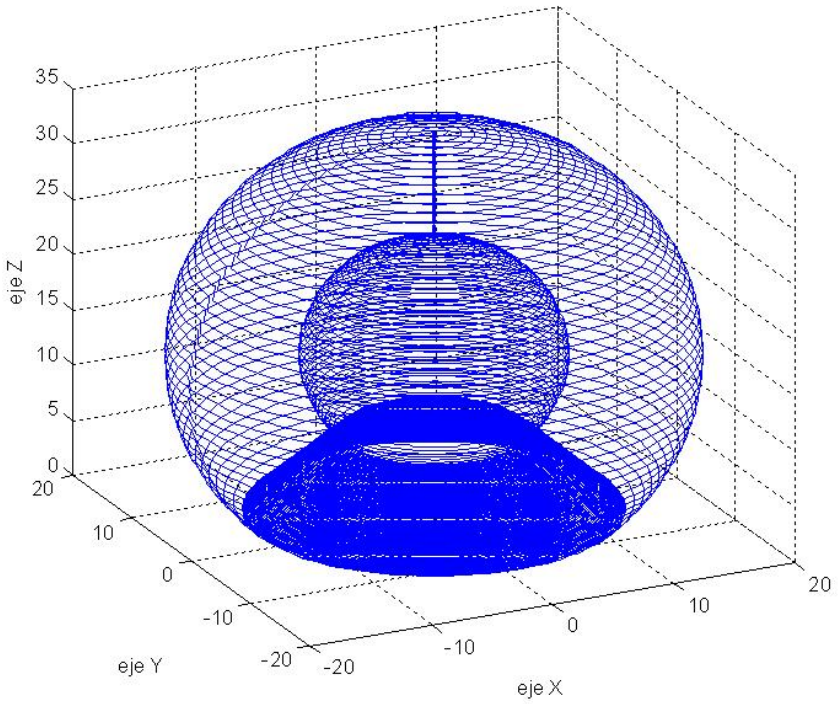
```
l1=15;          % Largo de cuepo 1

q1m=0;          % Angulo q1 minimo
q1M=2*pi;       % Angulo q1 MAXIMO
q2m=pi/4;       % Angulo q2 minimo
q2M=pi/2;       % Angulo q2 MAXIMO
q3m=10;         % Extension q3 minimo
q3M=20;         % Extension q3 MAXIMO
```





```
l1=15;          % Largo de cuepo 1

q1m=0;          % Angulo q1 minimo
q1M=2*pi;       % Angulo q1 MAXIMO
q2m=pi/4;       % Angulo q2 minimo
q2M=pi;         % Angulo q2 MAXIMO
q3m=10;         % Extension q3 minimo
q3M=20;         % Extension q3 MAXIMO
```





Los dos archivos en Matlab con los cuales se generaron los movimientos de la periferia del sólido se encuentran a continuación:

```
% TP1 - Matriz Cinematica                              SOLIDO DE MOVIMIENTO
% Manipulador de 3 grados de libertad, dos rotacionales y uno traslacional.
% ------------------------------------------------------------------------
% Rango de movimientos:
%                       q1: 0 a 2*pi
%                       q2: pi/4 a 3*pi/4
%                       q3: 10 a 20
% ------------------------------------------------------------------------
% Secuencia de movimientos para definir el sólido de cobertura:
%     1)con angulos minimos y el brazo contraido, rotamos todo q1
%     2)rotamos a tope q2
%     3)rotamos todo q1
%     4)estiramos al maximo q3
%     5)rotamos todo q1
%     6)rotamos a tope q2 (hacia la posición de origen)
%     7)rotamos todo q1
% ------------------------------------------------------------------------
clear
clc

l1=15;          % Largo de cuepo 1

q1m=0;          % Angulo q1 minimo
q1M=pi/2;       % Angulo q1 MAXIMO
q2m=pi/4;       % Angulo q2 minimo
q2M=pi;         % Angulo q2 MAXIMO
q3m=10;         % Extension q3 minimo
q3M=20;         % Extension q3 MAXIMO

res=0.05;       % Resolución de los mov

pos_gripper=[0 0 0 1]';     % Presumimos que no hay Gripper

q1=0;       % Posiciones iniciales de las articulaciones
q2=pi/4;
q3=10;

count=0;    % Contador (eje de tiempo)


for q2=q2m:res:q2M        % rotamos a tope q2
    for q1=q1m:res:q1M        % rotamos todo q1
        count=count+1;
        out=homogenia(q1,q2,q3,l1,pos_gripper);
        salida(:,count)=out;
    end
    count=count+1;
    out=homogenia(q1,q2,q3,l1,pos_gripper);
    salida(:,count)=out;
end

for q3=q3m:res:q3M        % estiramos al maximo q3
    for q1=q1m:res:q1M        % rotamos todo q1
        count=count+1;
        out=homogenia(q1,q2,q3,l1,pos_gripper);
        salida(:,count)=out;
    end
    count=count+1;
    out=homogenia(q1,q2,q3,l1,pos_gripper);
    salida(:,count)=out;
end

for q2=q2M:-res:q2m        % rotamos a tope q2 (hacia el otro lado)
    for q1=q1m:res:q1M        % rotamos todo q1
```

```
        count=count+1;
        out=homogenia(q1,q2,q3,l1,pos_gripper);
        salida(:,count)=out;
    end
    count=count+1;
    out=homogenia(q1,q2,q3,l1,pos_gripper);
    salida(:,count)=out;
end

for q3=q3M:-res:q3m        % estiramos al maximo q3 (hacia el otro lado)
    for q1=q1m:res:q1M        % rotamos todo q1
        count=count+1;
        out=homogenia(q1,q2,q3,l1,pos_gripper);
        salida(:,count)=out;
    end
    count=count+1;
    out=homogenia(q1,q2,q3,l1,pos_gripper);
    salida(:,count)=out;
end

salida_final=salida(1:3,:);   % Nos quedamos solo con los valores de x,y,z

hold on
 %comet3(salida_final(1,:),salida_final(2,:),salida_final(3,:),.01)
 plot3(salida_final(1,:),salida_final(2,:),salida_final(3,:))
xlabel('eje X'), ylabel('eje Y'), zlabel('eje Z')
hold off


function [out]=homogenia(q1,q2,q3,l1,pos_gripper)
% Matriz Homogenia del manipulador del TP1.

T=[ cos(q1)*cos(q2)     -sin(q1)        -cos(q1)*sin(q2)     -
q3*cos(q1)*sin(q2);
    sin(q1)*cos(q2)      cos(q1)        -sin(q1)*sin(q2)     -
q3*sin(q1)*sin(q2);
    sin(q2)              0               cos(q2)             -q3*cos(q2)+l1;
    0                    0               0                    1 ];

out=T*pos_gripper;
```

Los datos para realizar los gráficos, debieron ser recogidos por la implementación en DSP que se realizó sobre el Code Warrior, pero debido a nuestra falta de familiaridad con la herramienta, no obtuvimos resultados válidos, ya que nos desbordaban los registros y perdíamos información.
Se asume que con mas tiempo de dedicación, hubiera sido posible llegar a buen término.
Igualmente la lógica del programa es la correcta (ya que es la misma que se implementó en Matlab), motivo por el cuál adjuntamos el programa para el DSP.

```
/* MODULE test2 */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "TFR1.h"
#include "MFR1.h"
#include "MEM1.h"
#include "XFR1.h"
#include "AFR1.h"
```

```
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include <stdio.h>



//***************************************************************************
//              DEFINE
//***************************************************************************

#define MXRAD 361//

#define PULSE2RAD 50  //32767/MXRAD  // 32767/361 impulsos
                                                                        // Se usa 361 para poder crear pasos de 32767/361 de
manera
                                                                        // que el rango de la variable sea de 0º a 360º.

#define DESPLAZAMIENTO 10               //10 milímetros

#define CANT_DE_EXTENSIONES 80     //80 PASOS DE MOVIMIENTO LONGITUDINAL

#define L1 350                          // longitud del BODY 1 en milímetros

#define Max_Q1 300                      // max cant de pasos de ángulo Q1

#define Max_Q2 200                      // max cant de pasos de ángulo Q2

//***************************************************************************
//              STRUCTS
//***************************************************************************

typedef struct T                        // Nuestra matríz Homogenea
              {
              Frac16 Tee[4][4];
              }T;



//***************************************************************************
//              GLOBAL VARIABLE
//***************************************************************************

Frac16 i,j,k;
Frac16 pulse2rad=PULSE2RAD;
Frac16 delta_x=DESPLAZAMIENTO;

T MatrizHomogenea;


//***************************************************************************
//              MAIN
//***************************************************************************


void main(void)
{
 // Variables locales
 // Frac32 Q1_32b,Q2_32b;
 Frac16 Q1_16b,Q2_16b;

 Frac16 S1,S2,C1,C2,Q3,C1S2,S1S2;

 // Vector posición del Gripper (referido al extremo del manipulador)
 Frac16 Gripper[4][1]={{0},{0},{0},{1}};
 Frac16 Ref_Base[4][1];              // Vector a obtener (posición del Gripper referido a la base)
 //Ref_Base[0][0] *p;


  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
```

```
 /*** End of Processor Expert internal initialization.                    ***/

 /* Write your code here */

 for(;;) {


// Movimiento del TERCER grado de libertad (long), luego del SEGUNDO (Q2), y finalmente de Q1.

for(i=0;i<Max_Q1;i++)           // **
        {
                for(j=0;j<Max_Q2;j++)
                {
                        for(k=0;k<CANT_DE_EXTENSIONES;i++)
                        {



                        Q1_16b=(L_mult(pulse2rad,i));  // Multiplico un incremento para la articulación 1
                                                                                                        // por un
ángulo en radianes
                        //Q1_16b=extract_l(Q1_32b);             // Redondeamos a 16bits


                        Q2_16b=(L_mult(pulse2rad,j));  // Multiplico un incremento para la articulación 2
                                                                                                        // por un
ángulo en radianes
                        //Q2_16b=extract_l(Q2_32b);             // Redondeamos a 16bits



                        S1=TFR1_tfr16SinPIx(Q1_16b);            // Calculamos el Seno de Q1
                        C1=TFR1_tfr16CosPIx(Q1_16b);            // Calculamos el Coseno de Q1

                        S2=TFR1_tfr16SinPIx(Q2_16b);            // Calculamos el Seno de Q2
                        C2=TFR1_tfr16CosPIx(Q2_16b);            // Calculamos el Coseno de Q2

                        Q3=extract_l(L_mult(delta_x,k));                        // Desplazamiento traslacional
del cuerpo 3


                        // Cálclulos auxiliares de la matríz Homogenea que se repiten mucho.
                        C1S2=extract_l(L_mult(C1,S2));
                        S1S2=extract_l(L_mult(S1,S2));

                        // Matriz Homogenea para nuestro manipulador de 3 grados de libertad
                MatrizHomogenea.Tee[0][0]= extract_l(L_mult(C1,C2));
                        MatrizHomogenea.Tee[0][1]= negate (S1);
                        MatrizHomogenea.Tee[0][2]= negate (C1S2);
                        MatrizHomogenea.Tee[0][3]= negate (extract_l(L_mult(Q3,C1S2)));
                        MatrizHomogenea.Tee[1][0]= extract_l(L_mult(S1,C2));
                        MatrizHomogenea.Tee[1][1]= C1;
                        MatrizHomogenea.Tee[1][2]= negate (S1S2);
                        MatrizHomogenea.Tee[1][3]= negate (extract_l(L_mult(Q3,S1S2)));
                        MatrizHomogenea.Tee[2][0]= S2;
                        MatrizHomogenea.Tee[2][1]= 0;
                        MatrizHomogenea.Tee[2][2]= C2;
                        MatrizHomogenea.Tee[2][3]= extract_l(L_mult(Q3,C2))+L1;
                        MatrizHomogenea.Tee[3][0]=0;
                        MatrizHomogenea.Tee[3][1]=0;
                        MatrizHomogenea.Tee[3][2]=0;
                        MatrizHomogenea.Tee[3][3]=1;


                        // Ahora multiplicamos matricialmente a la matríz Homogenea por el Vector Posición
                        // del sistema de referencia del Gripper, y obtenemos el Vector Posición del Gripper
                        // referido a la base del manipulador.

                        /* PROTOTIPO DE LA FUNCIÓN MULTIPLICACIÓN MATRICIAL

                        ANSIC prototype: void xfr16Mult(Frac16 *pX,int xrows,int xcols,Frac16 *pY,int ycols,Frac16
*pZ)

                         pX: Pointer to Frac16 - Pointer to the first input matrix X of fractional data values.
                         xrows:int - The number of rows of the input matrix X.
```

```
                                    xcols:int - The number of columns of the input matrix X; this is also the number of rows of the
input matrix Y.
                                    pY: Pointer to Frac16 - Pointer to the second input matrix Y of fractional data values.
                                    ycols:int - The number of columns of the input matrix Y.
                                    pZ: Pointer to Frac16 - Pointer to the output matrix Z of fractional data values.
                                   */


                                   XFR1_xfr16Mult(&MatrizHomogenea.Tee[0][0],4,4,Gripper,4,(short          *)&Ref_Base[0][0]);
          // Sorry por los # mágicos


                                   //  *** Acá tenemos en el vector Ref_Base la posición a imprimir ***

                                   printf ("Índice: i j k  , Ref_Base (x,y,z) , Ángulo (Q1,Q2) , Desplazamiento (Q3)\n");
                                   printf ("%d \t %d \t %d \t %d \t %d \t %d \t %d \t %d \t %d \t
\n",i,j,k,Ref_Base[0],Ref_Base[1],Ref_Base[2],Q1_16b,Q2_16b,Q3);




                                   }
                            }
                     }

  }
}

/* END test2 */
/*
** ###################################################################
**
**     This file was created by UNIS Processor Expert 2.97 [03.74]
**     for the Freescale 56800 series of microcontrollers.
**
** ###################################################################
*/
```

Conclusiones Finales:

• Fijamos los conceptos de las matrices homogéneas.
Con la simulación de la implementación, se pudo conceptualizar los movimientos y capacidades de este menipulador.

• Es muy importante las restricciones espaciales en los movimientos, ya que definen el sólido de movimiento. Es necesario restringir los movimientos para acotarlo al uso específico del manipulador, y que choquen con otros robots o máquinas en una planta.

• Implementación en lenguaje C para el DSP DSP56800/E de Motorola.
Si bien no obtuvimos los resultados deseados, fue nuetra primer experiencia con un DSP, y aprendimos bastante de la misma. El uso de los beans del CodeWarrior, las funciones más complejas vienen implementadas en el soft de desarrollo, y pueden ser usadas en proyectos propios como el nuestro.

• Observamos que para éste tipo de manipulador, si no definimos adecuadamente el rango de movimientos, podríamos llegar al mismo punto físico en el espacio, a través de distintas combinaciones espaciales.

• Observamos que si bien se pueden desarrollar, toda clase de movimientos dentro del sólido, para ésta topología en particular de manipulador, conviene pensarlo como un sistema de coordenadas esféricas.