



*Universidad
Tecnológica Nacional*

Facultad Regional Buenos Aires

Cátedra: Robótica - Plan 95A

Trabajo práctico N°1

Implementación de una matriz cinemática en DSP

Profesor: Ing. Hernan Giannetta.

JTP: Ing. Damián Granzella.

Integrantes: Juan Pablo Perelló. Leg (132651-0)

Nicolás Pimentel. Leg (119137-8)

Fecha de entrega: 14 / 05 / 2010.

Introducción Teórica:

Cinemática del robot

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 a resolver en 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 **cinemático 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 describir 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 describir la relación espacial entre los elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea 4×4 que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base.

Por otra parte, la cinemática del robot trata 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.



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 a un sistema de referencia fijo. Dado que un robot se 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 cinemático directo se reduce a encontrar una matriz homogénea de transformación T que relacione la posición y orientación del extremo del robot respecto al sistema de referencia fijo situado en la base del mismo. Esta matriz T será función de las coordenadas articulares.

Resolución del problema cinemático directo mediante matrices de transformación homogénea

La resolución del problema cinemático 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 6 grados de libertad, la solución del problema cinemático directo vendrá dada por las relaciones:

$x=f_x(q_1,q_2,q_3,q_4,q_5,q_6)$	$\alpha=f_\alpha(q_1,q_2,q_3,q_4,q_5,q_6)$
$y=f_y(q_1,q_2,q_3,q_4,q_5,q_6)$	$\beta=f_\beta(q_1,q_2,q_3,q_4,q_5,q_6)$
$z=f_z(q_1,q_2,q_3,q_4,q_5,q_6)$	$\gamma=f_\gamma(q_1,q_2,q_3,q_4,q_5,q_6)$

De acuerdo al robot que nos toca caracterizar en este trabajo práctico vamos a tener 3 grados de libertad, con lo que nos quedará solo x , y , z .

Para la realización del trabajo práctico nos vamos a plantear por el método sistemático basado en la utilización de las matrices de transformación homogénea.

Un robot de n grados de libertad está 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 él y , utilizando las transformaciones homogéneas, es posible representar las rotaciones y translaciones relativas entre los distintos eslabones que componen el robot. Normalmente la matriz de transformación homogénea que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del

robot se suele denominar matriz ${}^{i-1}A_i$. Así pues, 0A_1 describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, 1A_2 describe la posición y orientación del segundo eslabón respecto del primero, etc. Del mismo modo denominando 0A_k a las matrices resultantes del producto de las matrices ${}^{i-1}A_i$ 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 respecto al sistema de coordenadas de la base se puede expresar mediante la matriz 0A_2 :

$${}^0A_2 = {}^0A_1 \cdot {}^1A_2.$$

Cuando se consideran todos los grados de libertad, a la matriz 0A_n se la 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 :

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

Aunque para describir 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 (D-H).

Estos propusieron en 1995 un método matricial que permite establecer de manera sistemática un sistema de coordenadas $\{S_i\}$ ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de la cadena completa.

Según la representación de (D-H), escogiendo adecuadamente los sistemas de coordenadas asociados a 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 translaciones que permiten relacionar el sistema de referencia del objeto i con el sistema del elemento $i-1$. Las transformaciones en cuestión son las siguientes:

- 1) Rotación alrededor del eje z_{i-1} un ángulo θ_i .
- 2) Translación a lo largo de z_{i-1} una distancia d_i ; vector $d_i(0,0,d_i)$.
- 3) Translación a lo largo de x_i una distancia a_i ; vector $a_i(0,0,a_i)$.
- 4) Rotación alrededor del eje x_i un ángulo α_i .

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:

$${}^{i-1}A_i = T(z, \theta_i) T(0, 0, d_i) T(a_i, 0, 0) T(x, \alpha_i).$$

$${}^{i-1}A_i = \begin{bmatrix} \cos \theta_i & -\cos \alpha_i \cdot \sin \theta_i & \sin \alpha_i \cdot \sin \theta_i & a_i \cdot \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cdot \cos \theta_i & -\sin \alpha_i \cdot \cos \theta_i & a_i \cdot \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [\text{expresión 1}]$$

Donde θ_i , a_i , d_i , α_i son parámetros D-H del eslabón i . De este modo, basta con identificar los parámetros θ_i , a_i , d_i , α_i para obtener las matrices A y relacionar así todos y cada uno de los eslabones del robot.

En nuestro caso, vamos a utilizar el Algoritmo de Denavit y Hartenberg para la obtención del modelo cinemático directo para la resolución del trabajo práctico.

Reglas que establece el método de D-H para la obtención del modelo cinemático directo:

D-H 1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se enumerará 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 z_i sobre el eje de la articulación $i+1$.

D-H 5. Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0 . Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0 .

D-H 6. Para i de 1 a $n-1$, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i . Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación $i+1$.

D-H 7. Situar x_i en la línea normal común a z_{i-1} y z_i .

D-H 8. Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i .

D-H 9. Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n .

D-H 10. Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.

D-H 11. Obtener d_i como la distancia, medida a lo largo de z_{i-1} , que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.

D-H 12. Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidirá con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

D-H 13. Obtener α_i como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con x_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

D-H 14. Obtener las matrices de transformación ${}^{i-1}A_i$.

D-H 15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0A_1 \cdot {}^1A_2 \cdot \dots \cdot {}^{n-1}A_n$.

D-H 16. La matriz T define la orientación (submatriz de rotación) y posición (submatriz de translación) del extremo referido a la base en función de las n coordenadas articulares.

Los cuatro parámetros de D-H (θ_i , d_i , a_i , α_i) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que lo unen con el anterior y siguiente.


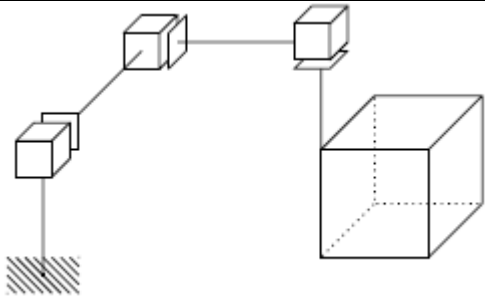
Una vez obtenidos los parámetros de D-H, el cálculo de las relaciones entre los eslabones consecutivos del robot es inmediato, ya que vienen dadas por las matrices A , que se calculan según la expresión general [expresión 1]. Las relaciones entre eslabones no consecutivos vienen dadas por las matrices T que se obtienen como producto de un conjunto de matrices A .

Obtenida la matriz T , ésta expresará la orientación y posición del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

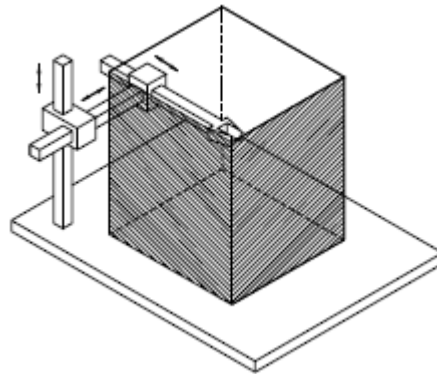
Desarrollo de la práctica:

✓ Implemente el código C en CW para el DSP56800/E de la cadena cinemática directa para el robot sin gripper, usando como setpoint, una trayectoria lineal continua a cada eje. Defina los límites y área de trabajo del manipulador.

✓ Imprima el resultado del vector (x,y,z) usando la función `plot3(x,y,z)` de matlab.

	
Robot cartesiano del trabajo práctico	Movimientos que describe el robot cartesiano

Teniendo en cuenta el tipo de robot (cartesiano) podemos apreciar de antemano que el área de trabajo del robot estará limitada a un volumen como se muestra en la siguiente figura.



En base a la teoría antes descripta podemos llegar a los parámetros D-H para el robot cartesiano.

Articulación	θ	d	a	α
1	0	q_1	0	-90
2	-90	q_2	0	-90
3	0	q_3	0	0

A continuación se exponen las diferentes matrices en forma canónica que vamos a utilizar:

Translación en 3 dimensiones – Forma canónica -

Original object position

Translated in X

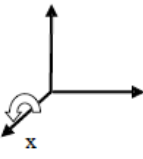
$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned}$$

Translated in Z

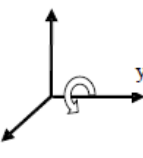
Translated in Y

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

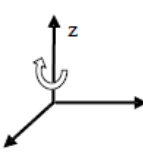
Rotación en 3 dimensiones – Forma canónica –



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{Rotación en x}$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{Rotación en y}$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{Rotación en z}$$

Utilizando las transformaciones anteriores vamos a sacar las matrices correspondientes al robot del.

0A_1 = Translación en z una distancia l1 x rotación en x de -90° .

$${}^0A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1A_2 = Rotación en z de -90° x translación en z una distancia l2 x rotación en x de -90° .

$${}^1A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & q2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2A_3 = Translación en z una distancia l3

$${}^2A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$T = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & q2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & q3 \\ 0 & -1 & 0 & q2 \\ 1 & 0 & 0 & q1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Por lo tanto las ecuaciones resultantes serán:

X = q3.

Y = q2.

Z = q1.

Lo cual concuerda perfectamente con lo que estábamos esperando.

Código Fuente (implementación en DSP568xx/E)

```
/* MODULE Pratico01 */
/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "MEM1.h"
#include "MFR1.h"
#include "TFR1.h"
#include "DFR1.h"
/* Including 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 PLANO_INCLINADO
// #define CUATRO_PLANOS
// #define PLANOS_PARALELOS

#define RESOLUCION 1
#define PUNTOS 10
#define CANTIDAD PUNTOS/RESOLUCION

#ifdef PLANO_INCLINADO
    #define LONG CANTIDAD*10
#elif defined (CUATRO_PLANOS)
    #define LONG CANTIDAD*CANTIDAD*6
#else
    #define LONG CANTIDAD*CANTIDAD*CANTIDAD
#endif

#define FIN -1 // indica el final de los vectores de resultado

int resultX[LONG],resultY[LONG],resultZ[LONG];
unsigned int index,indexPrint;
int x,y,z;

void main(void)
{
    PE_low_level_init();
    /** End of Processor Expert internal initialization.    ***/

    indexPrint=0;
```

```
index=0;

printf ("Trabajo Practico 1\n");

#ifdef PLANO_INCLINADO

for(z=0;z<CANTIDAD;)
{
    for(y=0;y<CANTIDAD;)
    {
        resultX[index]=y;
        resultY[index]=y;
        resultZ[index]=z;
        index++;
        y=y+RESOLUCION;
    }

    z=z+RESOLUCION;
}

#elif defined (CUATRO_PLANOS)

    for(x=0;x<CANTIDAD; )
    {
        for(y=0;y<CANTIDAD;)
        {
            resultX[index]=x;
            resultY[index]=y;
            resultZ[index]=0;
            index++;
            y=y+RESOLUCION;
        }
        x=x+RESOLUCION;
    }

    for(y=0;y<CANTIDAD; ) { //barro el plano YZ
        for(z=0;z<CANTIDAD;)
        {
            resultX[index]=0;
            resultY[index]=y;
            resultZ[index]=z;
            index++;
            z=z+RESOLUCION;
        }
        y=y+RESOLUCION;
    }
```

```
for(x=0;x<CANTIDAD; ) { //barro el plano XZ
    for(z=0;z<CANTIDAD;)
    {
        resultX[index]=x;
        resultY[index]=0;
        resultZ[index]=z;
        index++;
        z=z+RESOLUCION;
    }
    x=x+RESOLUCION;
}

for(x=0;x<CANTIDAD; ) {
    for(y=0;y<CANTIDAD;) //barro el plano XY
    {
        resultX[index]=x;
        resultY[index]=y;
        resultZ[index]=9;
        index++;
        y=y+RESOLUCION;
    }
    x=x+RESOLUCION;
}

for(y=0;y<CANTIDAD; ) { //barro el plano YZ
    for(z=0;z<CANTIDAD;)
    {
        resultX[index]=9;
        resultY[index]=y;
        resultZ[index]=z;
        index++;
        z=z+RESOLUCION;
    }
    y=y+RESOLUCION;
}

for(x=0;x<CANTIDAD; ) { //barro el plano XZ
    for(z=0;z<CANTIDAD;)
    {
        resultX[index]=x;
        resultY[index]=9;
        resultZ[index]=z;
        index++;
        z=z+RESOLUCION;
    }
    x=x+RESOLUCION;
}
```

```
#elif defined (PLANOS_PARALELOS)

for(z=0;z<CANTIDAD;)
{
    for(x=0;x<CANTIDAD; )
    {
        for(y=0;y<CANTIDAD;)
        {
            resultX[index]=x;
            resultY[index]=y;
            resultZ[index]=z;
            index++;
            y=y+RESOLUCION;
        }
        x=x+RESOLUCION;
    }
    z=z+RESOLUCION;
}

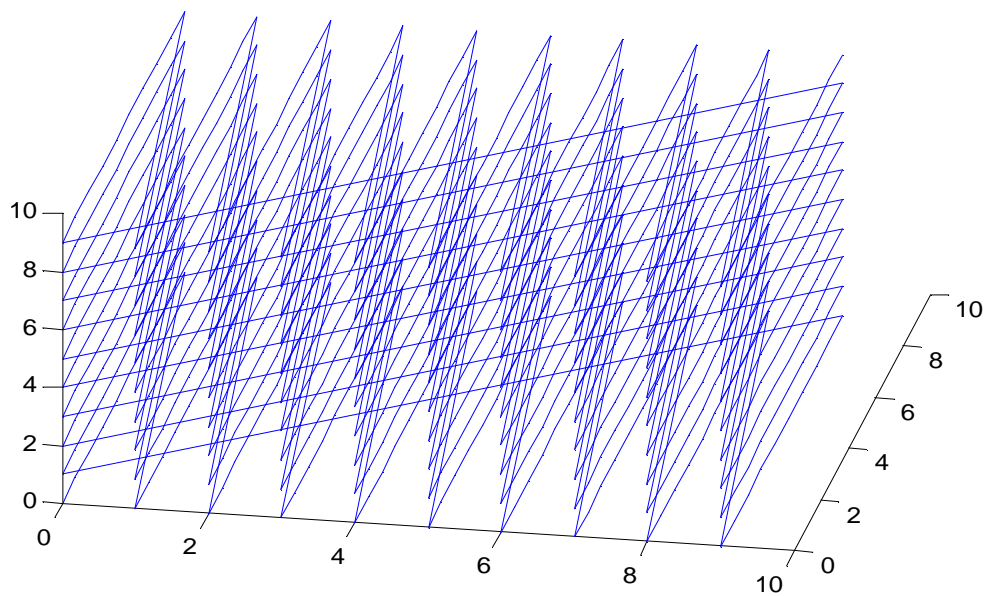
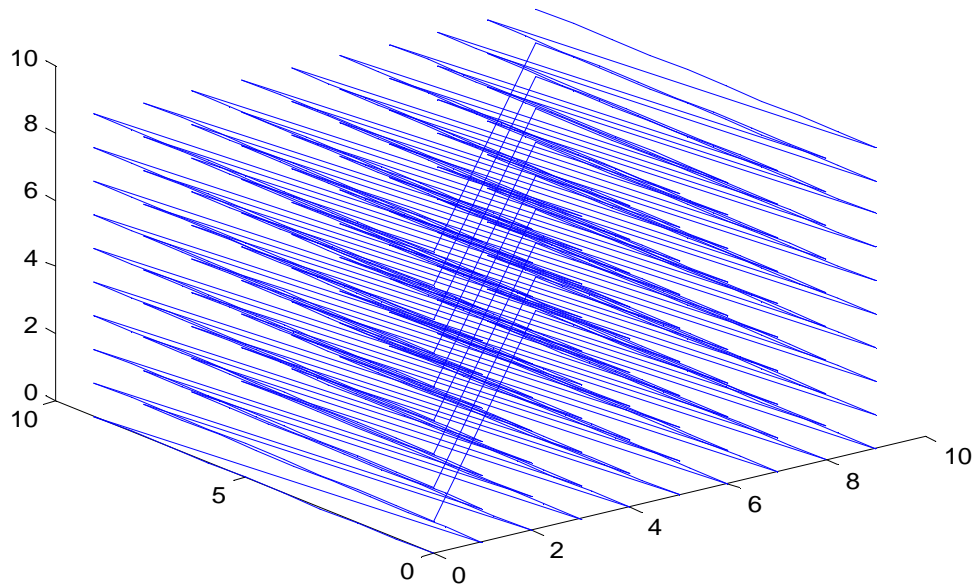
#endif

//imprimo los resultados

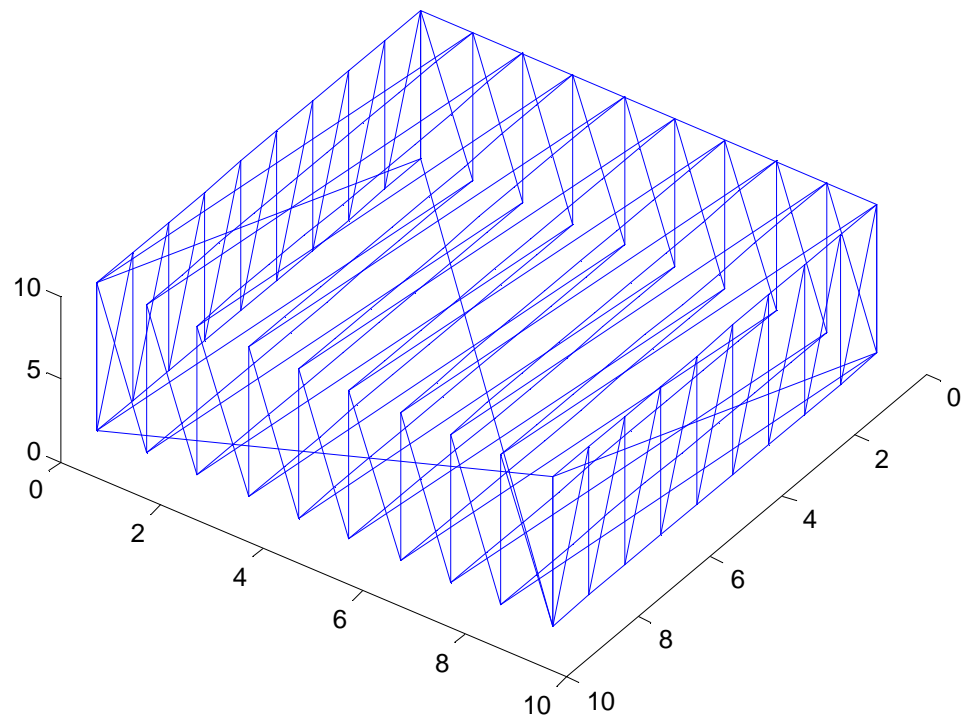
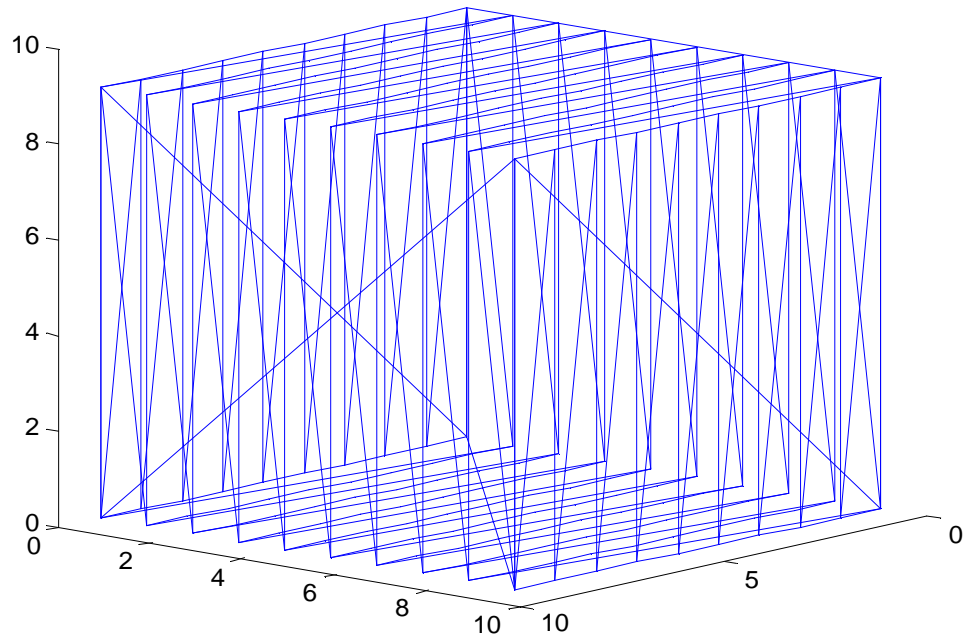
printf ("\nX\n");
while(indexPrint<=LONG)
{
    printf ("%d \t;",resultX[indexPrint]);
    indexPrint++;
}
printf ("\nY\n");
indexPrint=0;
while(indexPrint<=LONG)
{
    printf ("%d \t;",resultY[indexPrint]);
    indexPrint++;
}
printf ("\nZ\n");
indexPrint=0;
while(indexPrint<=LONG)
{
    printf ("%d \t;",resultZ[indexPrint]);
    indexPrint++;
}
}
```

Resultados:

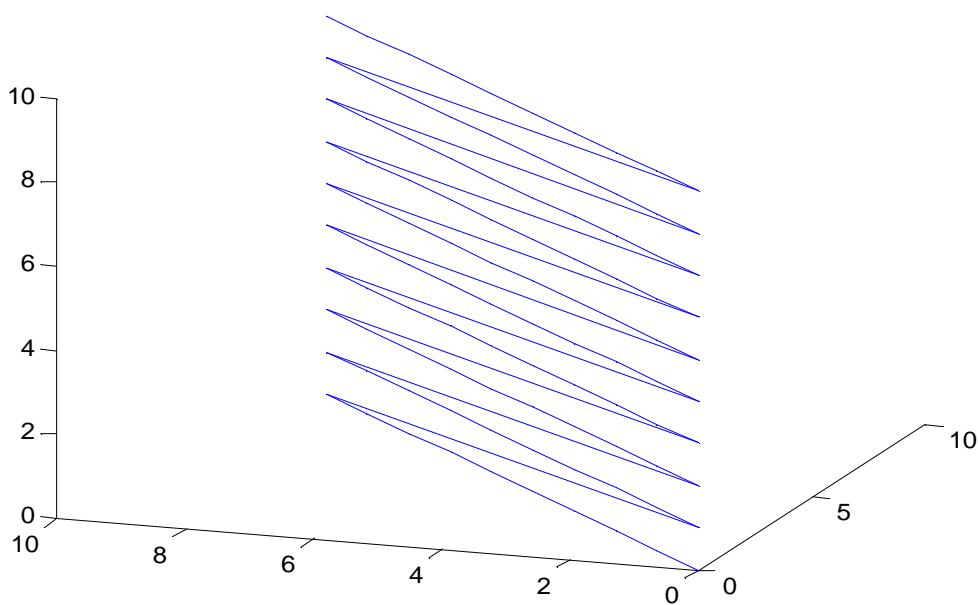
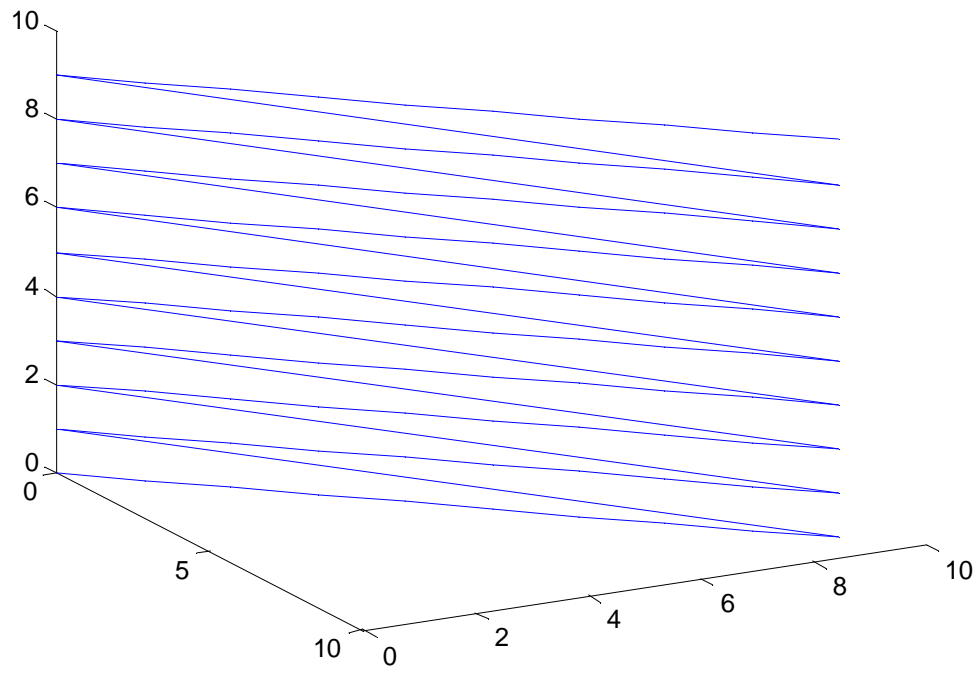
El siguiente grafico se genero barriendo el plano XY (limite 10, valor normalizado) para distintos valores en el eje Z (limite 10, valor normalizado) y con una resolución igual a 1 (mínima variación posible).



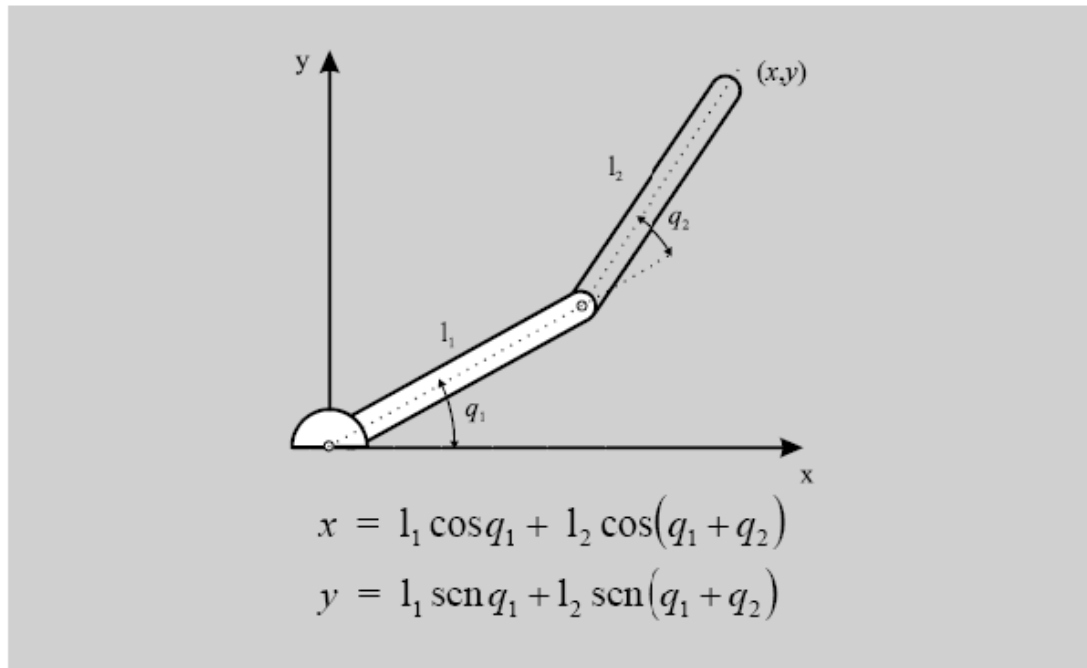
Implementado otra manera de barrer los planos; antes graficamos planos paralelos, en el siguiente grafico solo graficamos los planos limites del cubo (volumen de trabajo)



A manera de ejemplo implementamos el código necesario para barrer un plano inclinado, demostrando que es posible realizar diversas combinaciones de movimiento con distintos resultados.



En el caso del ejemplo de 2 grados de libertad obtuvimos lo siguiente:



Código Fuente:

```
/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "DFR1.h"
#include "TFR1.h"
#include "MFR1.h"
#include "MEM1.h"
/* Including 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 OPCION1

#define MXRAD 100
#define PULSE2RAD 32767/MXRAD
#define CANT 200
//Frac16 c,i;
Frac16 pulse2rad=PULSE2RAD;
Word16 c16[MXRAD];
Word32 c32[MXRAD];

unsigned char i,j,a,b,index;

void main(void)
```

```
{
    /* Write your local variable definition here */
    Frac16 testResultX[CANT],testResultY[CANT];
    Word16 local_c16;
    Word16 local2_c16;
    Word16 local3_c16;
    Word32 local_c32;
    Word32 local2_c32;
    Frac16 intermedia1[CANT],intermedia2[CANT];
    /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
    PE_low_level_init();
    /*** End of Processor Expert internal initialization.          ***/

#ifdef OPCION1 // ambas articulaciones se mueven juntas

    printf ("X\n");
    for(i=0,j=40;i<MXRAD;i++,j++)
    {
        if(j==60)
            j=60;

        local_c32=(L_mult(pulse2rad,i));
        local_c16=extract_l(local_c32);
        local2_c32=(L_mult(pulse2rad,j));
        local3_c16=extract_l(local2_c32);
        local2_c16=add(local_c16,local3_c16);

        intermedia1[i]=TFR1_tfr16CosPlx(local_c16); // cos q1
        intermedia2[i]=TFR1_tfr16CosPlx(local2_c16); // cos (q1+q2)
        testResultX[i]=add(intermedia1[i],intermedia2[i]);
        printf ("%d \t;",testResultX[i]);
    }

    printf ("\nY\n");

    for(a=0,b=40;a<MXRAD;a++,b++)
    {
        if(b==60)
            b=60;
        local_c32=(L_mult(pulse2rad,a));
        local_c16=extract_l(local_c32);
        local2_c32=(L_mult(pulse2rad,b));
        local3_c16=extract_l(local2_c32);
        local2_c16=add(local_c16,local3_c16);

        intermedia1[a]=TFR1_tfr16SinPlx(local_c16); //sin q1
        intermedia2[a]=TFR1_tfr16SinPlx(local2_c16); //sin (q1+q2)
        testResultY[a]=add(intermedia1[a],intermedia2[a]);
        printf ("%d \t;",testResultY[a]);
    }
}
```

```
printf ("\n\n");
```

```
#else // las articulaciones se mueven por separado, primero una y luego la segunda
```

```
printf ("X\n");
```

```
index=0;
```

```
j=10;
```

```
for(i=10;i<MXRAD;i++)
```

```
{
```

```
    local_c32=(L_mult(pulse2rad,i));
```

```
    local_c16=extract_l(local_c32);
```

```
    local2_c32=(L_mult(pulse2rad,j));
```

```
    local3_c16=extract_l(local2_c32);
```

```
    local2_c16=add(local_c16,local3_c16);
```

```
    intermedia1[index]=TFR1_tfr16CosPlx(local_c16);    // cos q1
```

```
    intermedia2[index]=TFR1_tfr16CosPlx(local2_c16); // cos (q1+q2)
```

```
    testResultX[index]=add(intermedia1[index],intermedia2[index]);
```

```
    printf ("%d \t;",testResultX[index]);
```

```
    index++;
```

```
}
```

```
i=10;
```

```
for(j=10;j<80;j++)
```

```
{
```

```
    local_c32=(L_mult(pulse2rad,i));
```

```
    local_c16=extract_l(local_c32);
```

```
    local2_c32=(L_mult(pulse2rad,j));
```

```
    local3_c16=extract_l(local2_c32);
```

```
    local2_c16=add(local_c16,local3_c16);
```

```
    intermedia1[index]=TFR1_tfr16CosPlx(local_c16);    // cos q1
```

```
    intermedia2[index]=TFR1_tfr16CosPlx(local2_c16); // cos (q1+q2)
```

```
    testResultX[index]=add(intermedia1[index],intermedia2[index]);
```

```
    printf ("%d \t;",testResultX[index]);
```

```
    index++;
```

```
}
```

```
printf ("\nY\n");

index=0;
b=10;
for(a=10;a<MXRAD;a++)
{

    local_c32=(L_mult(pulse2rad,a));
    local_c16=extract_l(local_c32);
    local2_c32=(L_mult(pulse2rad,b));
    local3_c16=extract_l(local2_c32);
    local2_c16=add(local_c16,local3_c16);

    intermedia1[index]=TFR1_tfr16SinPlx(local_c16); //sin q1
    intermedia2[index]=TFR1_tfr16SinPlx(local2_c16); //sin (q1+q2)
    testResultY[index]=add(intermedia1[index],intermedia2[index]);
    printf ("%d \t ;",testResultY[index]);
    index++;

}

a=10;
for(b=10;b<80;b++)
{

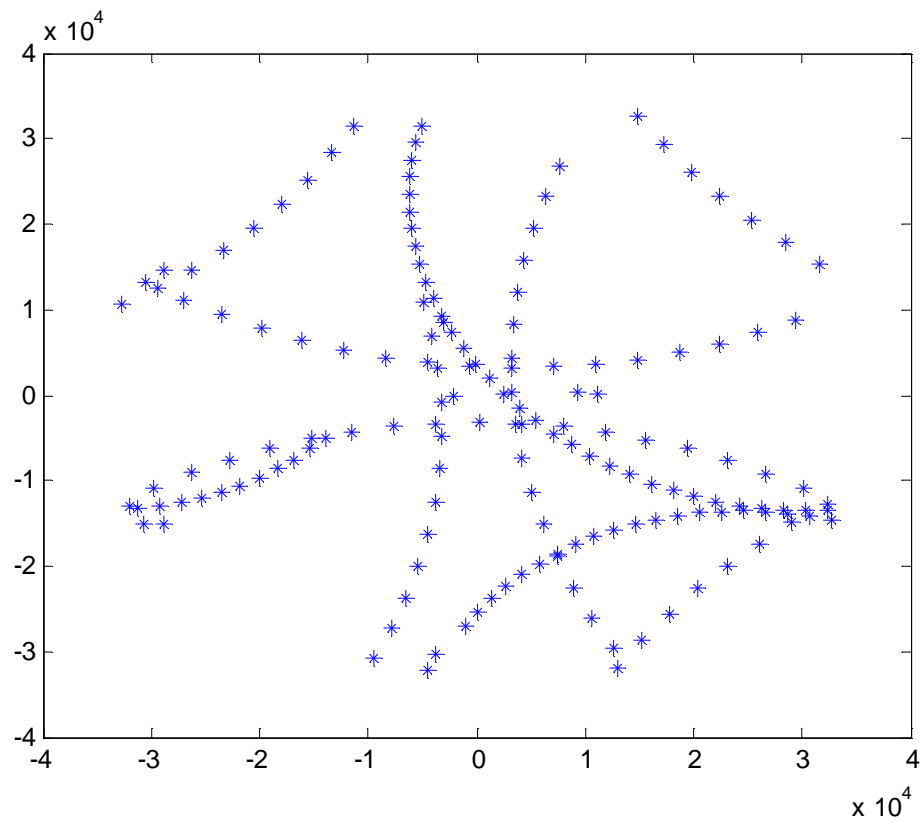
    local_c32=(L_mult(pulse2rad,a));
    local_c16=extract_l(local_c32);
    local2_c32=(L_mult(pulse2rad,b));
    local3_c16=extract_l(local2_c32);
    local2_c16=add(local_c16,local3_c16);

    intermedia1[index]=TFR1_tfr16SinPlx(local_c16); //sin q1
    intermedia2[index]=TFR1_tfr16SinPlx(local2_c16); //sin (q1+q2)
    testResultY[index]=add(intermedia1[index],intermedia2[index]);
    printf ("%d \t ;",testResultY[index]);

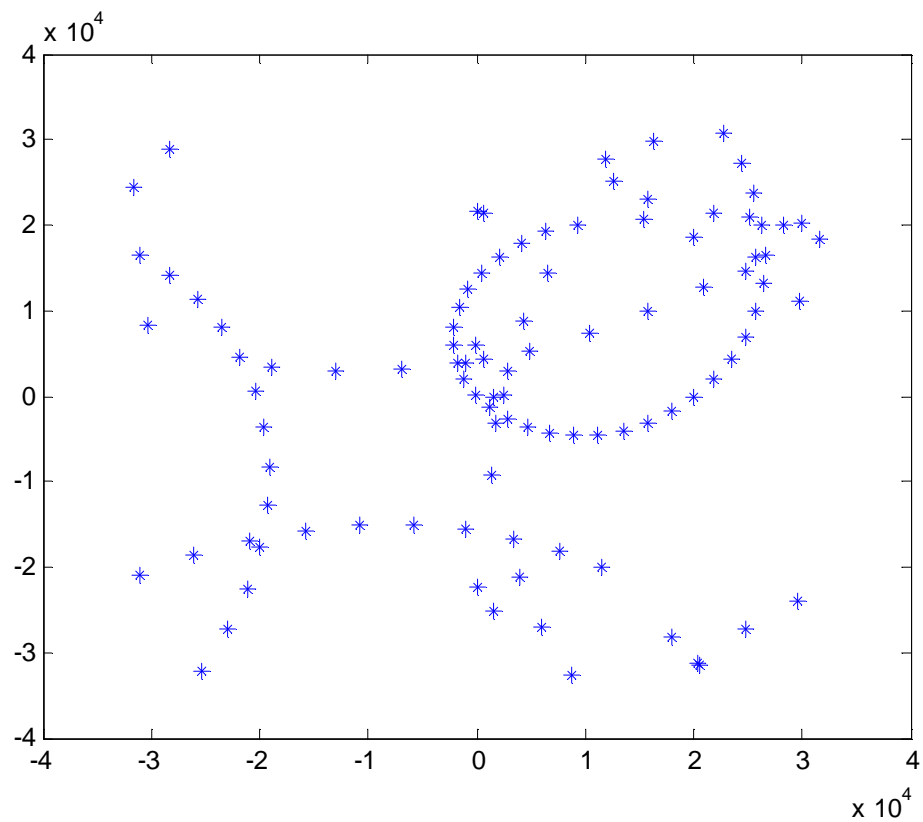
}
#endif
}
```

Resultados:

El sig. Gráfico muestra el caso en el que las articulaciones se mueven por separado.



Caso en el que se mueven simultáneamente.



Conclusiones:

Después de realizar ambas simulaciones podemos concluir que tanto en el caso de tres grados de libertad como en el de dos grados de libertad, se deben definir con cuidado los límites de área o volumen de trabajo del robot y la resolución (esta caracteriza la mínima variación entre puntos consecutivos a los que el robot puede acceder dentro del área o volumen de trabajo). Los límites, se deciden según las características físicas de la configuración del robot elegido, de manera de que no se produzca la destrucción de las piezas mecánicas del robot.

Otro aspecto destacable lo notamos, y se vea claro en los gráficos, que los resultados cambian notablemente al modificar las secuencias de los movimientos de las distintas articulaciones.

Por último concluimos que para poder definir los límites, resolución y secuencia de movimiento es importante conocer bien las características del robot que vamos a utilizar, las funciones que debe realizar y que especificaciones debe cumplir nuestro robot.