UNIVERSIDAD TECNOLOGICA NACIONAL

ROBOTICA

FACULTAD REGIONAL BUENOS AIRES

“SIMULACIÓN DEL ESPACIO DE TRABAJO DE UN ROBOT DE 3 Y 6 GRADOS DE LIBERTAD”

AÑO 2010

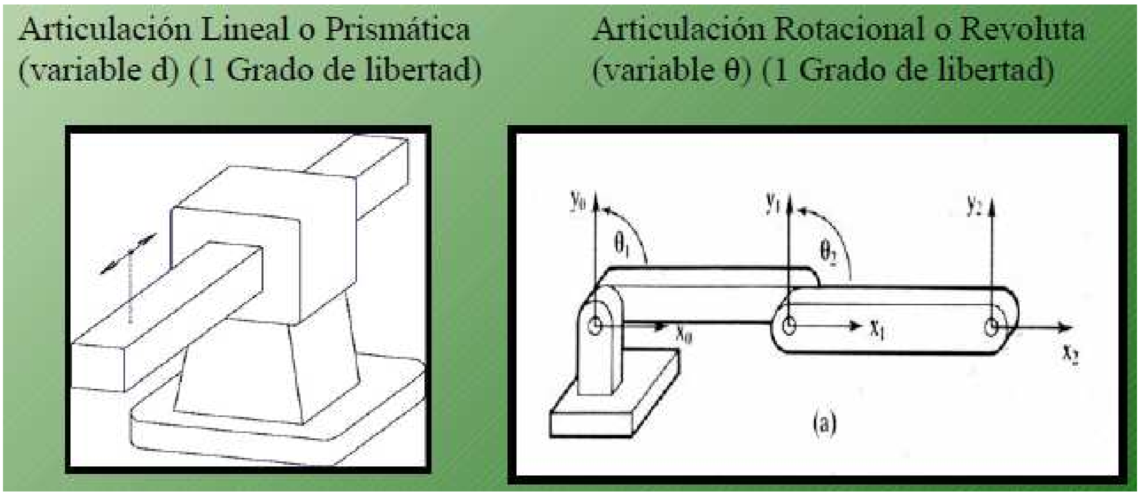
Montalti, Pablo

Alonso, Gustavo

1) Introducción a la cinemática del robot

Mecanicamente, un robot es una cadena cinematica formada de eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos.
La forma fisica de la mayoria de los robots industriales es similar a la de la anatomia del brazo humano.

Existen varios tipos de articulaciones, pero en la practica se emplean mayoritariamente articulaciones prismaticas y de rotacion.



1.1 La Matriz de Transformacion Homogenea

Es una matriz T de 4 x 4 que representa la transformacion de un vector de un sistema de coordenadas a otro. Esta matriz esta compuesta por 4 submatrices:

R_{3x3} SubMatriz de Rotacion
P_{3x1} SubMatriz de Traslacion
F_{1x3} SubMatriz de Perspectiva
E_{1x1} SubMatriz de Escalado Global

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ F_{1x3} & E_{1x1} \end{bmatrix}$$

1.3 El problema Cinematico

La **cinematica del robot** estudia el movimiento del mismo con respecto a un sistema de referencia. La cinematica se interesa por la descripcion analitica del movimiento espacial del robot como una funcion del tiempo, y en particular por las relaciones entre la posicion y la orientacion de la herramienta del robot con los valores que toman sus coordenadas de sus articulaciones.

Existen dos problemas fundamentales a resolver con respecto a la cinematica del robot:

A) **Cinematica Directa**. Consiste en determinar la posicion y orientacion del extremo final del robot con respecto al sistema de la base del robot a partir de conocer los valores de las articulaciones y los parametros geometricos.

B) **Cinematica Inversa**. Resuelve la configuracion que debe adoptar el robot para una posicion y orientacion conocidas del extremo.

1.4 El problema Cinematico Directo

El problema cinematico directo se reduce a encontrar la matriz de transformacion homogenea (T) que relacione la posicion y orientacion del extremo del robot respecto a su sistema de referencia fijo (base del robot). La matriz T esta en funcion de los parametros de las articulaciones del robot. Para un robot de n grados de libertad tenemos:

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

Donde:

$q_{1 \cdots n}$ = Son las variables de las articulaciones.
Para articulaciones revolutas las variables son ángulos.
Para articulaciones prismáticas las variables son distancias.

x, y, z = Coordenadas de la posición del extremo del robot.

α, β, γ = Ángulos de la orientación del extremo del robot.

Para robots de mas de 2 grados de libertad es dificil aplicar metodos geometricos para la solucion de su cinematica directa.

A cada eslabon se le asocia un sistema coordenado y utilizando transformaciones homogeneas es posible representar las rotaciones y traslaciones relativas entre los diferentes eslabones que componen el robot.

Siendo la matriz ${}^{i}A_{i-1,}$ la matriz de transformacion homogenea que representa la posicion y orientacion relativa entre los sistemas asociados a dos eslabones consecutivos del robot. Se puede representar de forma parcial o total la cadena cinematica que forma el robot:

$${}^{n}A_0 = {}^{1}A_0 . {}^{2}A_1 . {}^{3}A_2 \ldots\ldots . {}^{i}A_{i-1}$$

1.5 Algoritmo de Denavit-Hartenberg

En 1955 Denavit y Hartenberg propusieron un metodo matricial que permite establecer de manera sistematica un sistema de coordenadas. La representacion de Denavit-Hartenberg (D-H) establece que seleccionandose adecuadamente los sistemas de coordenadas asociados a cada eslabon, sera posible pasar de uno al siguiente mediante 4 transformaciones basicas que dependen exclusivamente de las caracteristicas geometricas del eslabon. Reduciendose al siguiente patron de transformaciones que permiten relacionar el sistema de referencia del elemento i con respecto al sistema del elemento i-1:

- Rotacion alrededor del eje Z_{i-1} un angulo θ_i
- Traslacion a lo largo Z_{i-1} de una distancia d_i
- Traslacion a lo largo de x_i una distancia a_i
- Rotacion alrededor del eje x_i un angulo α_i

Algoritmo de Denavit-Hartenberg

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

Desarrollando la expresión:

$$A^{i}_{i-1} = \begin{bmatrix} \cos\theta_i & -sen\theta_i & 0 & 0 \\ sen\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -sen\alpha_i & 0 \\ 0 & sen\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obtenemos la <u>**expresión general de DH**</u>, donde θ_i, d_i, a_i, α_i son los parámetros DH del eslabón i :

$$A^{i}_{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i\, sen\theta_i & sen\alpha_i\, sen\theta_i & a_i \cos\theta_i \\ sen\theta_i & \cos\alpha_i \cos\theta_i & -sen\alpha_i \cos\theta_i & a_i sen\theta_i \\ 0 & sen\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Como se ha indicado, para que la matriz i-1Ai , definida anteriormente, relacione los sistemas{ Si-1 } y { Si }, es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Estas, junto con la definicion de los 4 parametros de Denavit Hartenberg, conforman el siguiente algoritmo para la resolucion del problema cinematico directo:

• **DH1.**Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (ultimo eslabón móvil). Se numerara como eslabón 0 a la base fija del robot.

• **DH2.**Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad y acabando en n).

• **DH3.**Localizar el eje de cada articulación. Si esta 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.

• **DH4.**Para i de 0 a n-1, situar el eje Zi, sobre el eje de la articulación i+1.

• **DH5.**Situar el origen del sistema de la base (S0) en cualquier punto del eje Z0. Los ejes X0 e Y0 se situaran dé modo que formen un sistema dextrógiro con Z0.

• **DH6.**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.

• **DH7.**Situar Xi en la línea normal común a Zi-1 y Zi.

• **DH8.**Situar Yi de modo que forme un sistema dextrógiro con Xi y Zi.

• **DH9.**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.

• **DH10.**Obtener Øi como el ángulo que hay que girar en torno a Zi-1 para que Xi-1 y Xi queden paralelos.

• **DH11.**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.

• **DH12.**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).

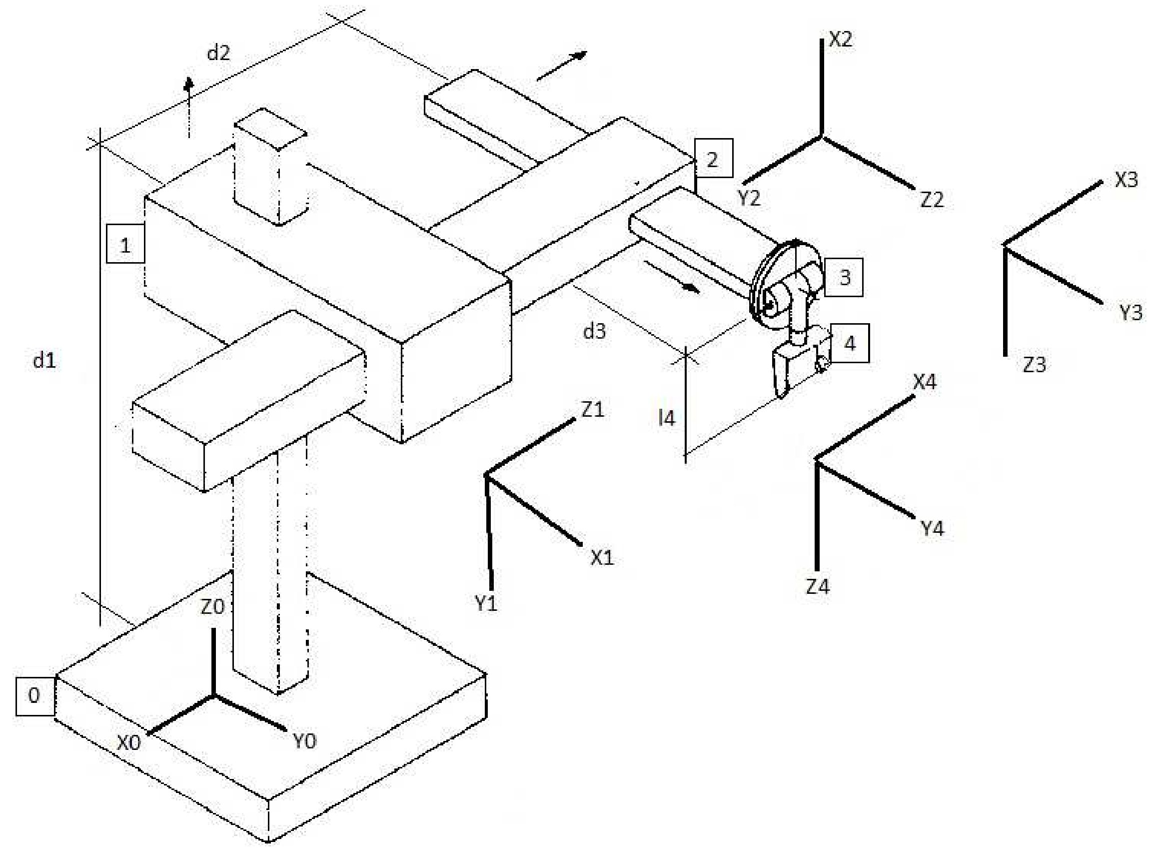
• **DH13.**Obtener ai como el ángulo que habría que girar entorno a Xi (que ahora coincidiría con Xi-1), para que el nuevo (Si-1) coincidiese totalmente con (Si).

• **DH14.**Obtener las matrices de transformación i-1Ai.

• **DH15.**Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot T = 0Ai, 1A2... n-1An.

• **DH16.**La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido ala base en función de las n coordenadas articulares.

1.6 Robot Cartesiano de 3DOF



	θi	di	ai	αi
1	90°	D1	0	-90°
2	-90°	D2	0	-90°
3	-90°	D3	0	90°
4	0	L4	0	0

$$^{1}A_0 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

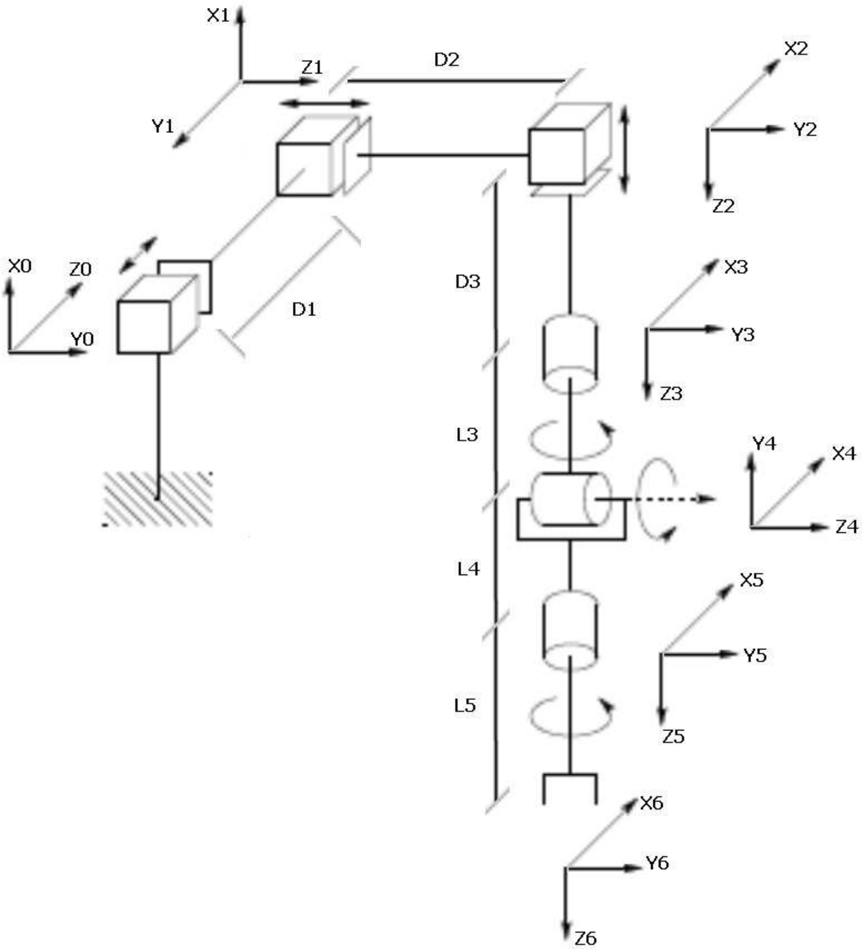
$$^{3}A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}A_{3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = {}^{4}A_{0} = {}^{4}A_{3}.{}^{3}A_{2}.{}^{2}A_{1}.{}^{1}A_{0} = \begin{bmatrix} -1 & 0 & 0 & -d2 \\ 0 & 1 & 0 & d3 \\ 0 & 0 & -1 & d1-l4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} X0 \\ Y0 \\ Z0 \\ 1 \end{bmatrix} = T.\begin{bmatrix} X4 \\ Y4 \\ Z4 \\ 1 \end{bmatrix} = T.\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -d2 \\ d3 \\ d1-l4 \\ 1 \end{bmatrix}$$

1.7 Robot Cartesiano de 6DOF



	θi	di	ai	αi
1	0	D1	0	-90°

2	-90°	D2	0	90°
3	0	D3	0	0
4	θ4	L3	0	-90°
5	θ5	L4	0	90°
6	θ6	L5	0	0°

$$^{0}A_{1}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&D1\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&-1&0&0\\0&0&0&1\end{bmatrix}$$

$$^{1}A_{2}=\begin{bmatrix}0&1&0&0\\-1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&D2\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

$$^{2}A_{3}=.\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&D3\\0&0&0&1\end{bmatrix}$$

$$^{3}A_{4}=\begin{bmatrix}C\theta_4&-S\theta_4&0&0\\S\theta_4&C\theta_4&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&L3\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&-1&0&0\\0&0&0&1\end{bmatrix}$$

$$^{4}A_{5}=\begin{bmatrix}C\theta_5&-S\theta_5&0&0\\S\theta_5&C\theta_5&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&L4\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

$$^{5}A_{6}=\begin{bmatrix}C\theta_6&-S\theta_6&0&0\\S\theta_6&C\theta_6&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&L5\\0&0&0&1\end{bmatrix}$$

$$T={}^{0}A_{1}={}^{0}A_{1}.{}^{1}A_{2}.{}^{2}A_{3}.{}^{3}A_{4}.{}^{4}A_{5}.{}^{5}A_{6}$$

$$\begin{bmatrix}X0\\Y0\\Z0\\1\end{bmatrix}=T.\begin{bmatrix}X6\\Y6\\Z6\\1\end{bmatrix}=T.\begin{bmatrix}0\\0\\0\\1\end{bmatrix}=\begin{bmatrix}-C\theta_5*L5-L3-D3\\S\theta_4*S\theta_5*L5+C\theta_4*L4+D2\\C\theta_4*S\theta_5*L5-S\theta_4*L4+D1\\1\end{bmatrix}$$

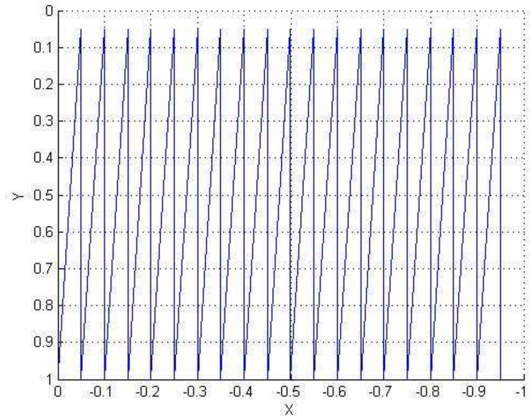
2) Desarrollo e implementacion en Codewarrior DSP 56800-E del robot cartesiano de 3DOF

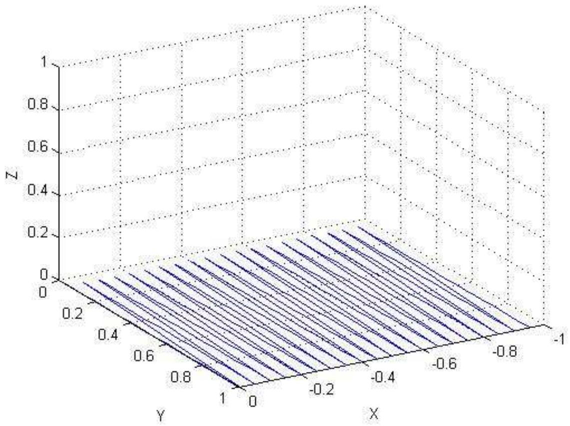
2.1 Superficie del área de trabajo del robot:

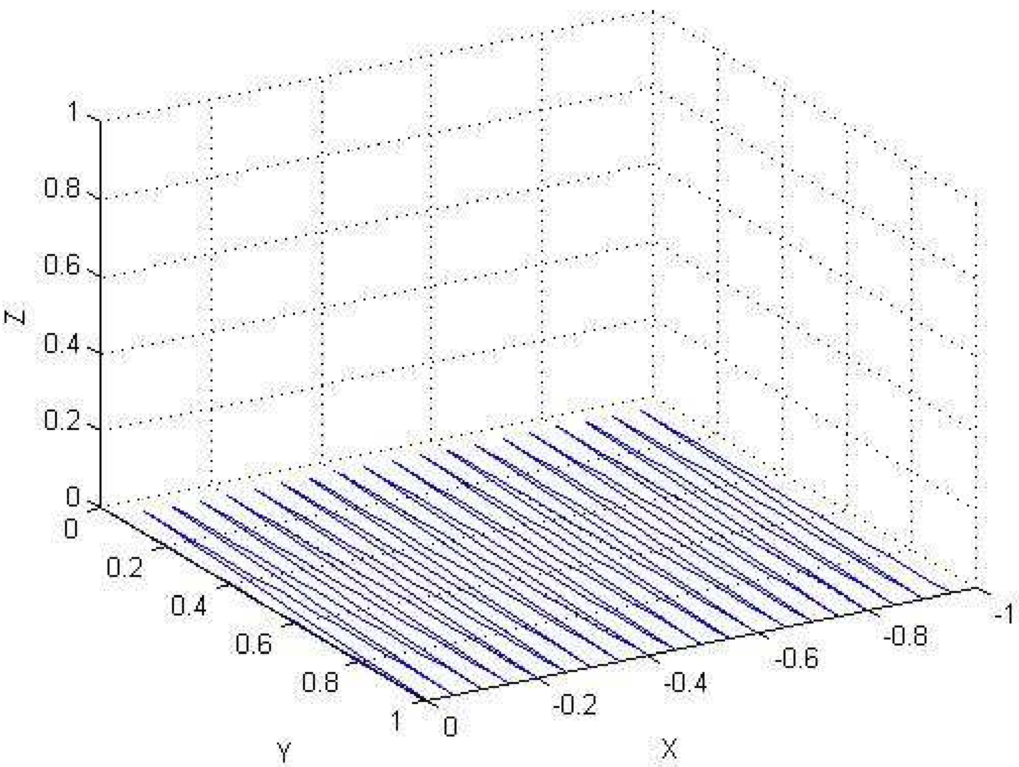
2.1.a Base – Plano XY, Z=0

```
#define     XLONG     20
#define     YLONG     20
#define     ZLONG     20

#define     PULSEX  32767/XLONG
#define     PULSEY  32767/YLONG
#define     PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
 /* Write your local variable definition here */

 Word16 d1=0,d2=0,d3=0;
 Frac16 xyz[4];

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

 /* Write your code here */
 for(;;) {

 /* Base del cubo */

  xyz[0]=0;
  xyz[1]=0;
  xyz[2]=0;
  xyz[3]=1;

  printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

        for(d2=0;d2<XLONG;d2++)
        {
          for(d3=0;d3<YLONG;d3++)
          {

                xyz[1]+=PULSEY;

                printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
          }
          xyz[1]=0;
          xyz[0]-=PULSEX;
        }
  }
```

}

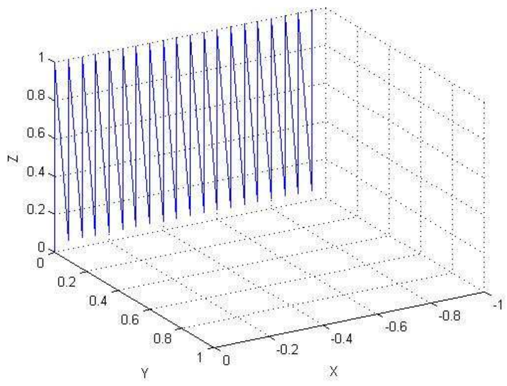

Y
X

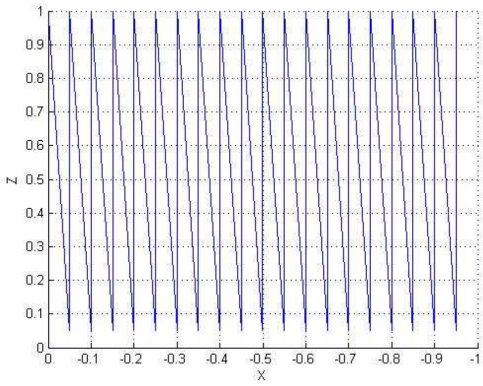

Z
Y
X

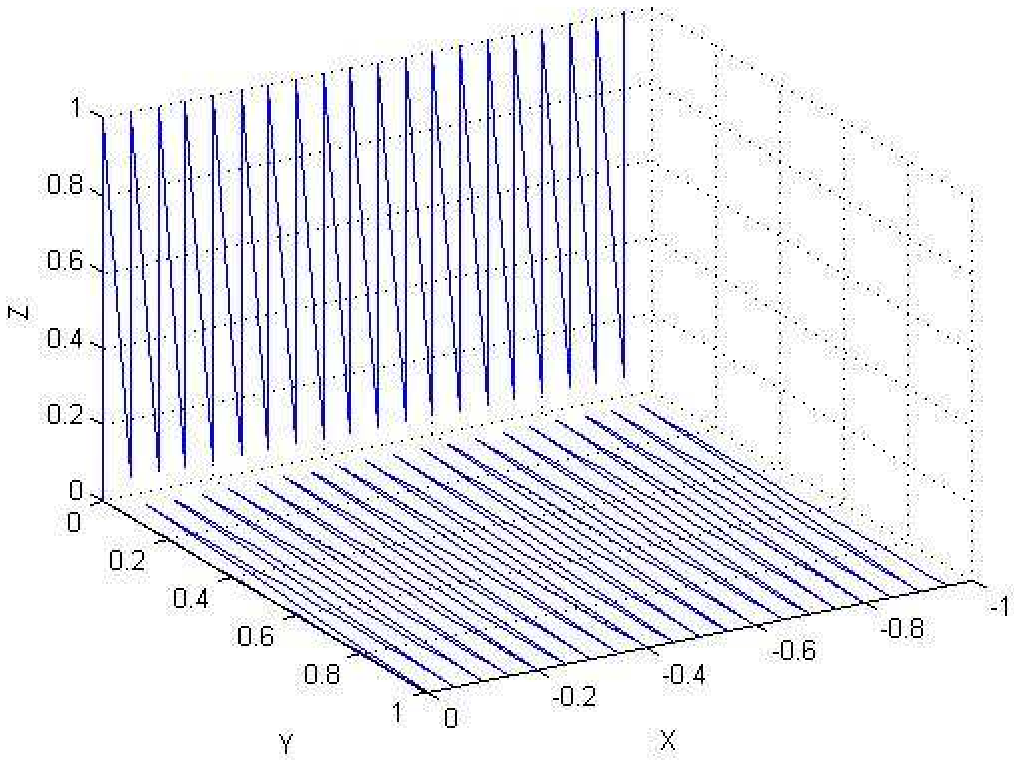

Z
Y
X

2.1.b Dorso – Plano XZ, Y=0

```
#define      XLONG      20
#define      YLONG      20
#define      ZLONG      20

#define      PULSEX  32767/XLONG
#define      PULSEY  32767/YLONG
#define      PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
  for(;;) {

  /* Dorso */

    xyz[0]=0;
    xyz[1]=0;
    xyz[2]=0;
    xyz[3]=1;

    printf("Dorso\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

        for(d2=0;d2<XLONG;d2++)
        {
          for(d1=0;d1<ZLONG;d1++)
          {

                xyz[2]+=PULSEZ;

                printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
          }
          xyz[2]=0;
          xyz[0]-=PULSEX;
        }
    }
}
```



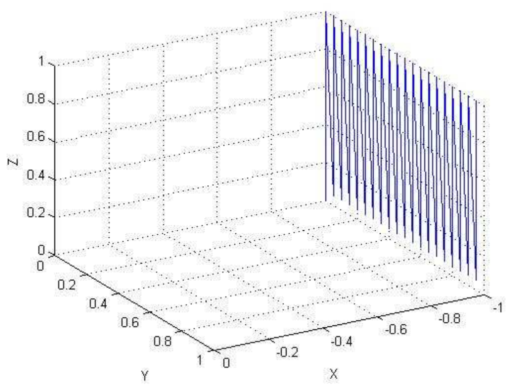


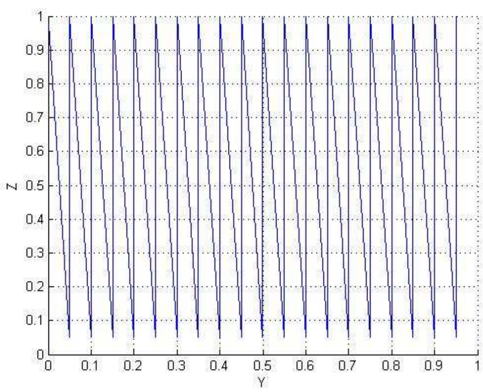


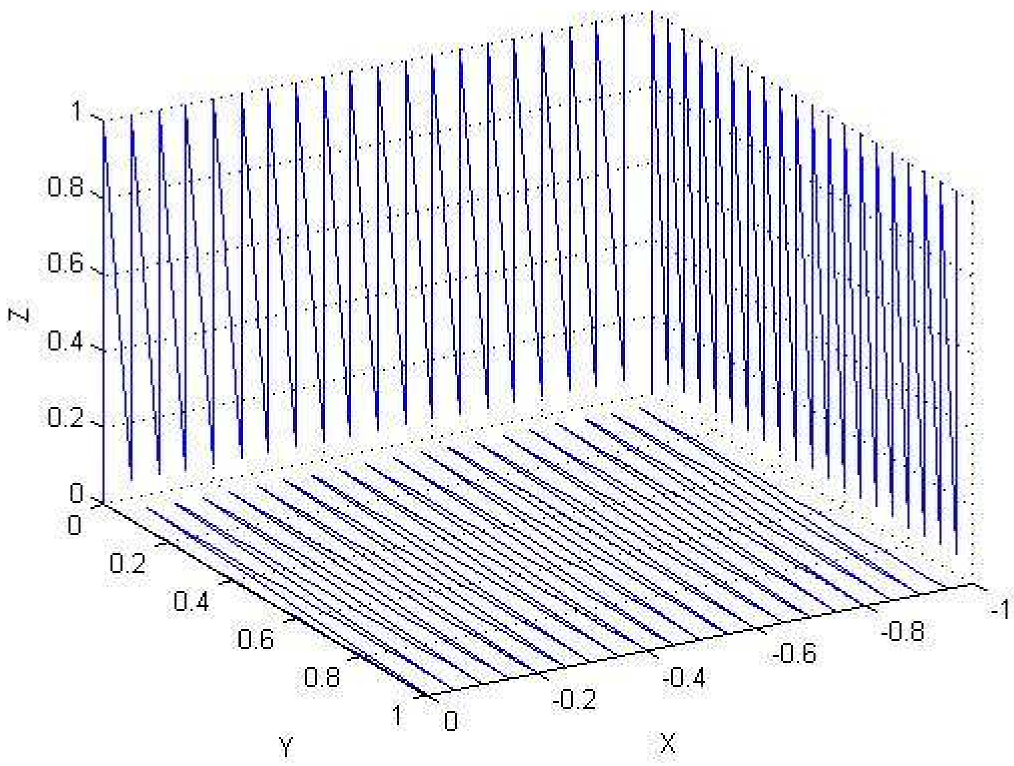
2.1.c Plano derecho – Plano YZ, X=MAX

```
#define     XLONG     20
#define     YLONG     20
#define     ZLONG     20

#define     PULSEX  32767/XLONG
#define     PULSEY  32767/YLONG
#define     PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */
```

```
Word16 d1=0,d2=0,d3=0;
Frac16 xyz[4];

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

/* Write your code here */
for(;;) {

/* Lateral derecho */

  xyz[0]=-32767;
  xyz[1]=0;
  xyz[2]=0;
  xyz[3]=1;

  printf("Lateral Derecho\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

        for(d3=0;d3<YLONG;d3++)
        {
          for(d1=0;d1<ZLONG;d1++)
          {

                xyz[2]+=PULSEZ;

                printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
          }
          xyz[2]=0;
          xyz[1]+=PULSEY;
        }

  }
}
```



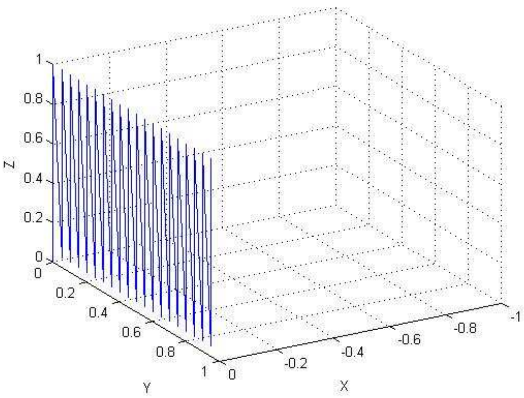


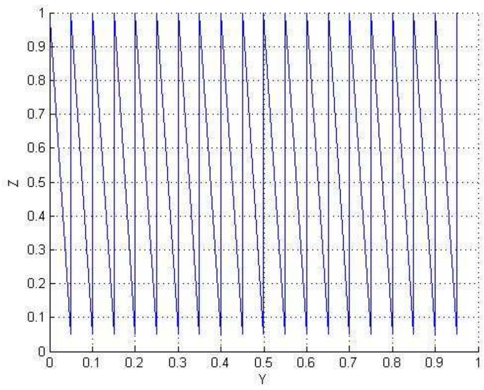


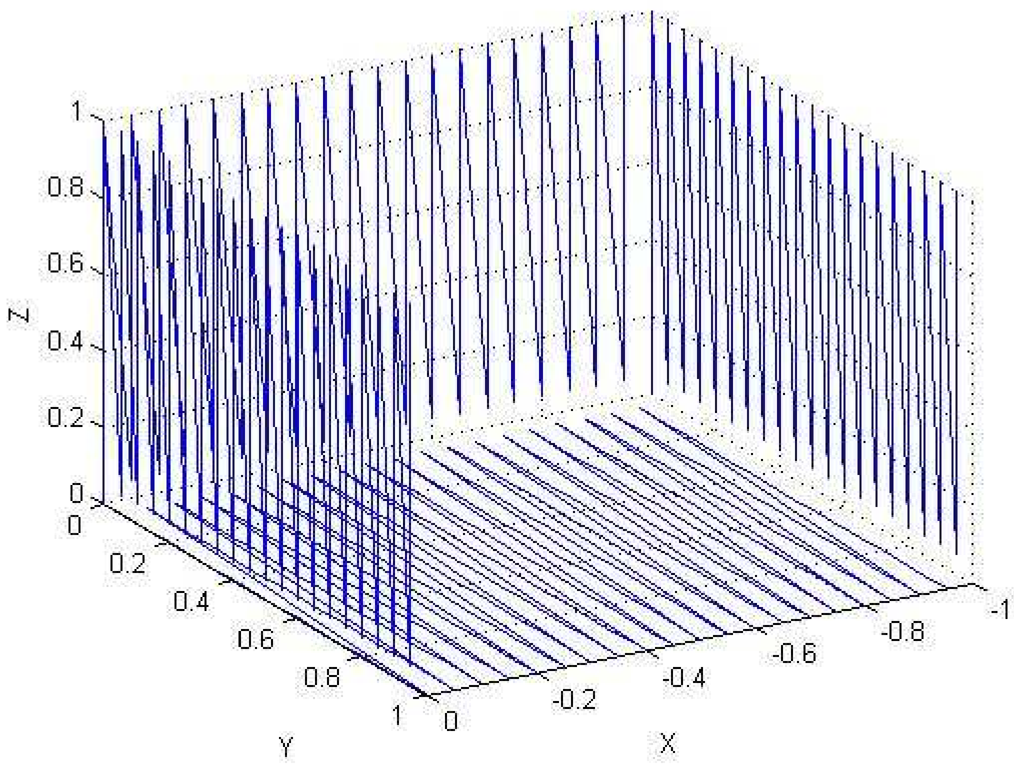
2.1.d Plano Izquierdo – Plano YZ, X=0

```
#define      XLONG     20
#define      YLONG     20
#define      ZLONG     20

#define      PULSEX  32767/XLONG
#define      PULSEY  32767/YLONG
#define      PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
  for(;;) {
```

```
/*Lateral izquierdo */

  xyz[0]=0;
  xyz[1]=0;
  xyz[2]=0;
  xyz[3]=1;

  printf("Lateral Izquierdo\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

        for(d3=0;d3<YLONG;d3++)
        {
          for(d1=0;d1<ZLONG;d1++)
          {

                  xyz[2]+=PULSEZ;

                  printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
          }
          xyz[2]=0;
          xyz[1]+=PULSEY;
        }

    }
}
```



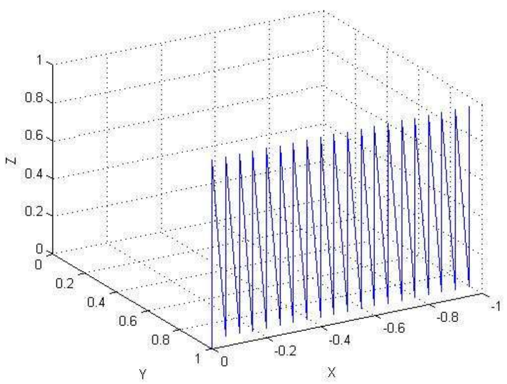


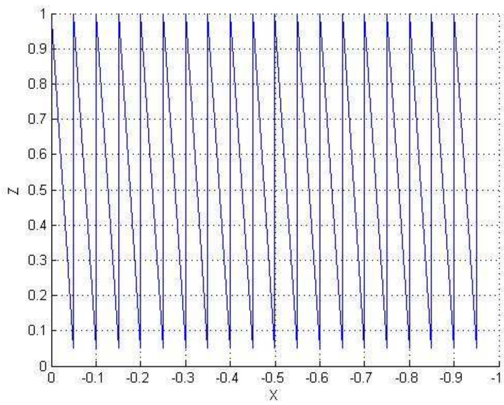


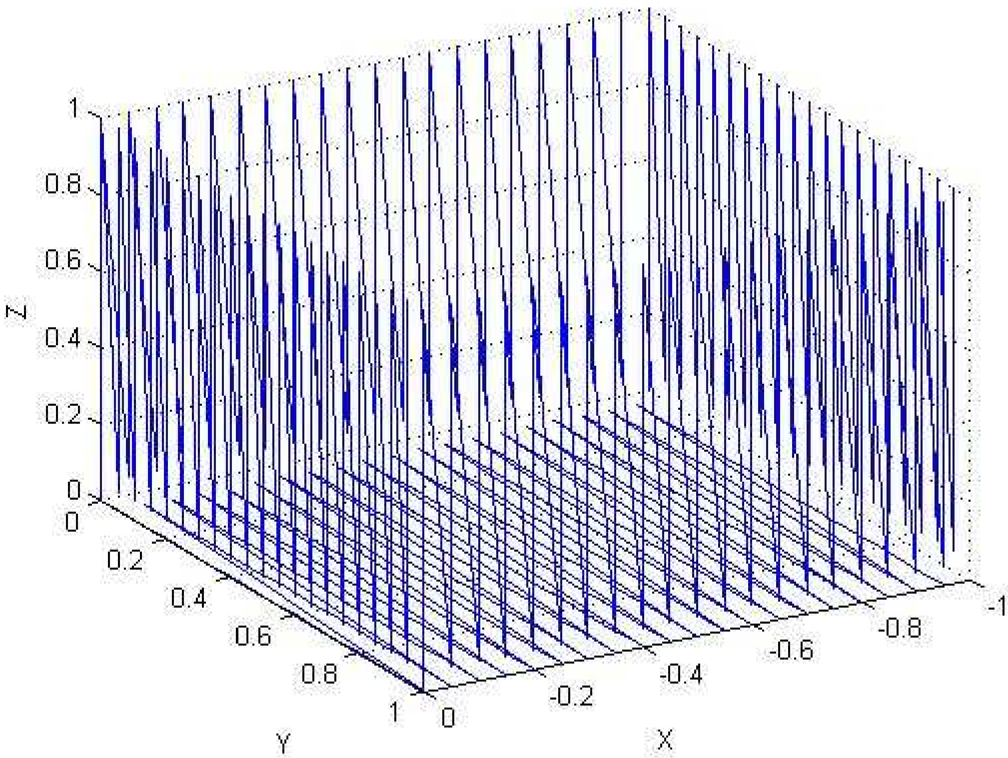
2.1.e Frente – Plano XZ, Y=MAX

```
#define      XLONG      20
#define      YLONG      20
#define      ZLONG      20

#define      PULSEX  32767/XLONG
#define      PULSEY  32767/YLONG
#define      PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
  for(;;) {
  /* Frente */
```

```
    xyz[0]=0;
    xyz[1]=32767;
    xyz[2]=0;
    xyz[3]=1;

    printf("Frente\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

          for(d2=0;d2<XLONG;d2++)
          {
            for(d1=0;d1<ZLONG;d1++)
            {

                    xyz[2]+=PULSEZ;

                    printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
            }
            xyz[2]=0;
            xyz[0]-=PULSEX;
          }
    }
}
```



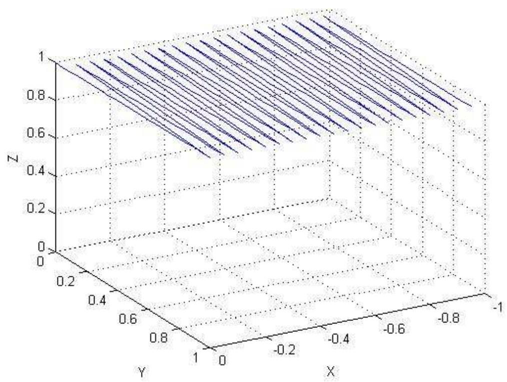


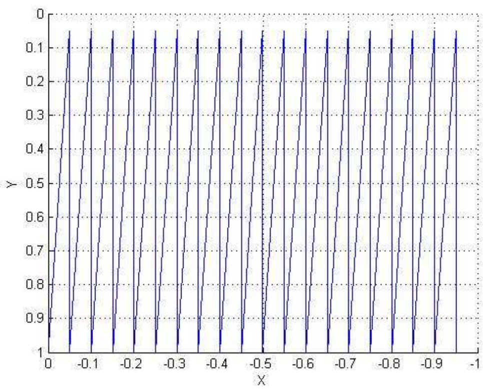


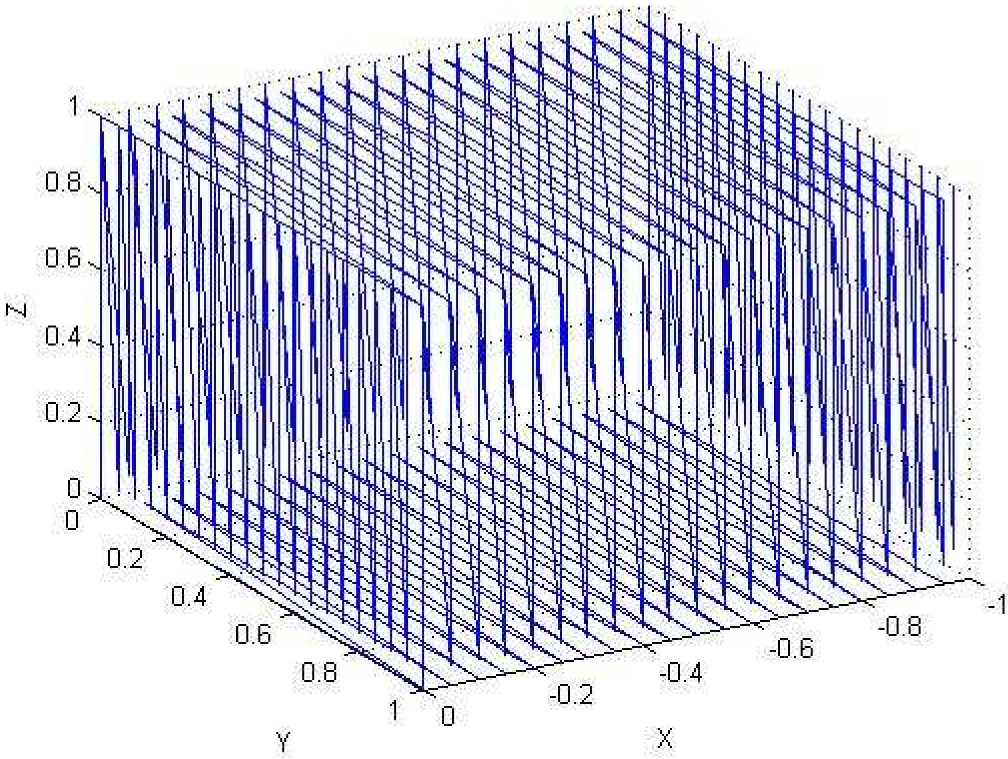
2.1.f Techo – Plano XY, Z=MAX

```
#define      XLONG     20
#define      YLONG     20
#define      ZLONG     20

#define      PULSEX  32767/XLONG
#define      PULSEY  32767/YLONG
#define      PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
  for(;;) {
```

```
/* Techo */

   xyz[0]=0;
   xyz[1]=0;
   xyz[2]=32767;
   xyz[3]=1;

   printf("Techo\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

          for(d2=0;d2<XLONG;d2++)
          {
            for(d3=0;d3<YLONG;d3++)
            {

                     xyz[1]+=PULSEY;

                     printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
            }
            xyz[1]=0;
            xyz[0]-=PULSEX;
          }

   }
}
```





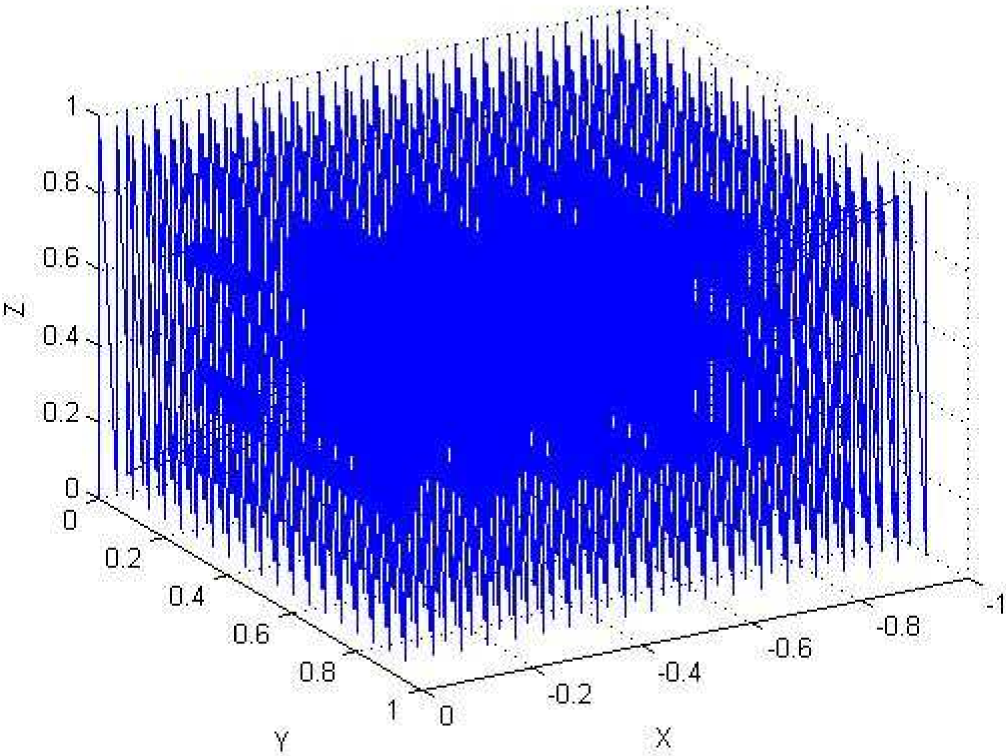


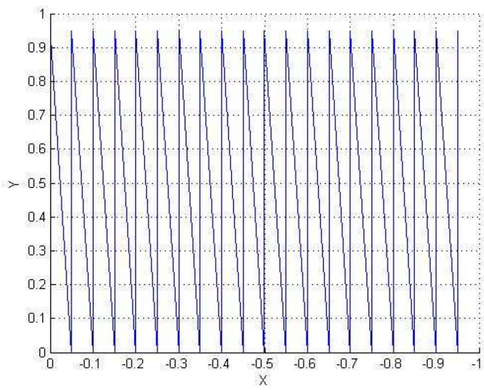
2.2 Volumen del área de trabajo del robot:

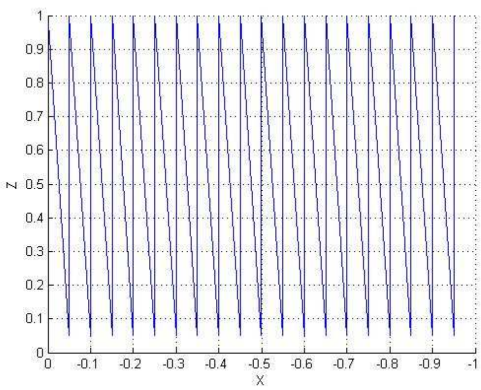
2.2.1 Resolución de 20 pasos

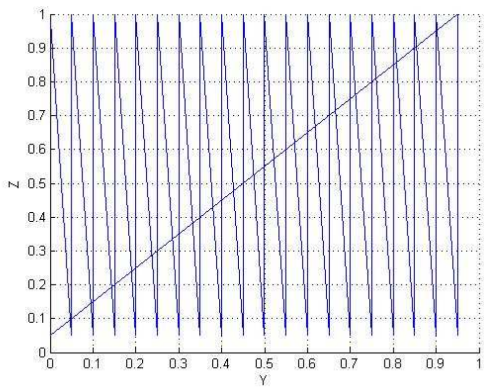
```
#define     XLONG     20
#define     YLONG     20
#define     ZLONG     20

#define     PULSEX  32767/XLONG
#define     PULSEY  32767/YLONG
#define     PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
```

```
  for(;;) {

/* Volumen */

   xyz[0]=0;
   xyz[1]=0;
   xyz[2]=0;
   xyz[3]=1;

   printf("Volumen\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

         for(d2=0;d2<XLONG;d2++)
         {
           for(d3=0;d3<YLONG;d3++)
           {
                   for(d1=0;d1<ZLONG;d1++)
                   {
                    xyz[2]+=PULSEZ;

                    printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
                   }

             xyz[2]=0;
             xyz[1]+=PULSEY;
           }
           xyz[1]=0;
           xyz[0]-=PULSEX;
         }

   }
}
```







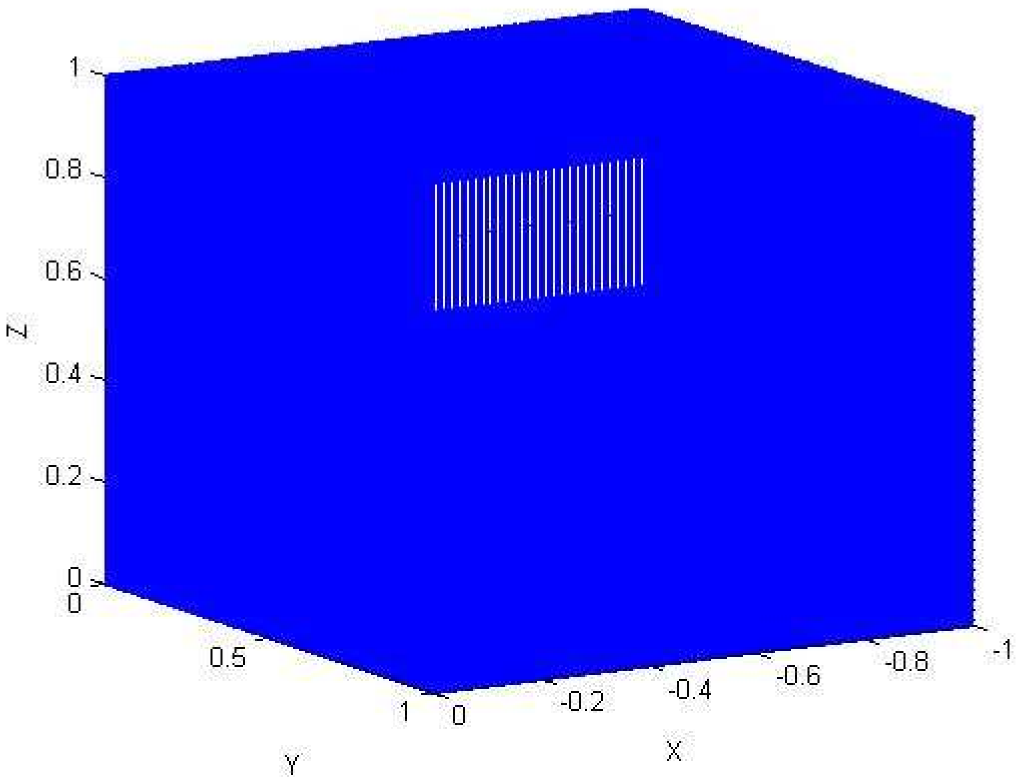


2.2.2 Resolución de 100 pasos

```
#define XLONG 100
```

```
#define     YLONG      100
#define     ZLONG      100

#define     PULSEX  32767/XLONG
#define     PULSEY  32767/YLONG
#define     PULSEZ  32767/ZLONG

Frac16 C[4];

void main(void)
{
  /* Write your local variable definition here */

  Word16 d1=0,d2=0,d3=0;
  Frac16 xyz[4];

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

  /* Write your code here */
  for(;;) {

/* Volumen */

    xyz[0]=0;
    xyz[1]=0;
    xyz[2]=0;
    xyz[3]=1;

    printf("Volumen\n%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);

        for(d2=0;d2<XLONG;d2++)
        {
          for(d3=0;d3<YLONG;d3++)
          {
                for(d1=0;d1<ZLONG;d1++)
                {
                 xyz[2]+=PULSEZ;

                 printf("%d\t%d\t%d\n",xyz[0],xyz[1],xyz[2]);
                }

            xyz[2]=0;
            xyz[1]+=PULSEY;
          }
          xyz[1]=0;
          xyz[0]-=PULSEX;
        }
```

```
    }
}
```



2.3 Conclusión:

Se ha completado la simulación de un Robot cartesiano de 3 grados de libertad haciendo énfasis en dos aspectos principales:

- La superficie de trabajo, que equivale a determinar los límites máximos de operación del robot.
- El volúmen del área de trabajo, principalmente para reflexionar sobre la capacidad de posicionamiento del robot dentro de dicha área.

Con respecto al primero, se ha supuesto que los mecanismos de movimiento tienen un limitada cantidad de pasos en los tres ejes cartesianos. Más precisamente se ha supuesto que se tienen 20 pasos en cada eje.

Para controlar dichos pasos se implementó un DSP y variables *Fraccionales* de 16 bits, con el objetivo de normalizar las coordenadas de posición al valor de longitud máxima en cada eje. De esta manera se puede obtener un paso mínimo cuya dimensión puede ser de 2^{-15} de la longitud total positiva o negativa, y se daría en el caso que se emplearan 32767 pasos distribuidos en la longitud máxima positiva o negativa.

En nuestra simulación, empleando 20 pasos distribuidos en la longitud total, estamos permitiendo una resolución de 0.049987792968750 de la máxima longitud en sentido positivo o negativo ((Frac16) 32767/20 = 1638).

Los algotirmos propuestos y las gráficas permiten visualizar éstos pasos y demuestran la manera de determinar la superficie de trabajo en cuestión.

Para la segunda simulación, volúmen del área de trabajo, se presentan dos variantes con distinta cantidad de pasos. La primera empleando 20 pasos distribuidos en la longitud total, permitiendo una resolución de 0.049987792968750 de la máxima longitud en sentido positivo o negativo ((Frac16) 32767/20 = 1638). La segunda empleando 20 pasos distribuidos en la longitud total, permitiendo una resolución de 0.009979248046875 de la máxima longitud en sentido positivo o negativo ((Frac16) 32767/100 = 327).

Se pretende mostrar aquí que la capacidad de posicionamiento del robot, es decir su presición, depende exclusivamente de la cantidad de pasos que sea capaz de dar en la longitud total.

3) Desarrollo e implementacion en Codewarrior DSP 56800-E del robot cartesiano de 6DOF

```
#define MXRAD 100
#define PMAX 50
#define FRACMAX 32796

#define L3 FRACMAX/5 // 0.2*FRACMAX
#define L4 FRACMAX/5 // 0.2*FRACMAX
#define L5 FRACMAX/5 // 0.2*FRACMAX

#define QMAX (FRACMAX/180*60)/PMAX
#define PULSE2RAD FRACMAX/PMAX

void main(void)
{
  /* Write your local variable definition here */

 Frac16 /*c, i,*/ d1, d2, d3, q1, q2; //, q3;
 Frac16 pulse2rad = PULSE2RAD, qmax2rad = QMAX;
 float COS_Q4, COS_Q5, SIN_Q4, SIN_Q5,auxf1,auxf2,auxf3,auxf4,auxf5;
 Frac32 x = 0, y = 0, z = 0, x_aux, y_aux, z_aux;

  /*** 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( d1=0; d1<PMAX; d1++ )
  {
  for ( d2=0; d2<PMAX; d2++ )
   {
   for ( d3=0; d3<PMAX; d3++)
    {
    for ( q1=-PMAX; q1<PMAX; q1++)      // 360º totales
     {
     for ( q2=-PMAX; q2<PMAX; q2++)      // 120º totales
      {
 COS_Q4=(float)TFR1_tfr16CosPIx(extract_l((L_mult(pulse2rad,q1))))/FRACMAX;
 COS_Q5=(float)TFR1_tfr16CosPIx(extract_l((L_mult(pulse2rad,q2))))/FRACMAX;
 SIN_Q4=(float)TFR1_tfr16SinPIx(extract_l((L_mult(pulse2rad,q1))))/FRACMAX;
 SIN_Q5=(float)TFR1_tfr16SinPIx(extract_l((L_mult(pulse2rad,q2))))/FRACMAX;

 auxf1 = L5*COS_Q5;
 x_aux = x;
 x = - d3 - L3 - (Frac16)auxf1;

 if ( L_abs(x) > (FRACMAX) )
         x = x_aux;

 auxf2 = L4*COS_Q4;
 auxf3 = L5*SIN_Q5*SIN_Q4;
 y_aux = y;
 y = d2 + (Frac16)auxf2 + (Frac16)auxf3;

 if ( L_abs(y) > (FRACMAX) )
         y = y_aux;

 auxf4 = L5*SIN_Q4*COS_Q4;
 auxf5 = L4*SIN_Q4;
 z_aux = z;
 z = d1 - (Frac16)auxf5 + (Frac16)auxf4;

 if ( L_abs(z) > (FRACMAX) )
         z = z_aux;

printf ( "x=%d \t y=%d \t z=%d \n",x,y,z);


      }
     }
    }
   }
  }
 }
}
```

Debido a la complejidad y tiempo de procesamiento que presentaría simular el espacio de trabajo con el Matlab y, como las unicas diferencias con el robot anterior son los 3 ejes giratorios de la punta de la pinza, dejaremos los desplazamientos fijos y graficaremos el espacio de trabajo de la herramienta en la punta.

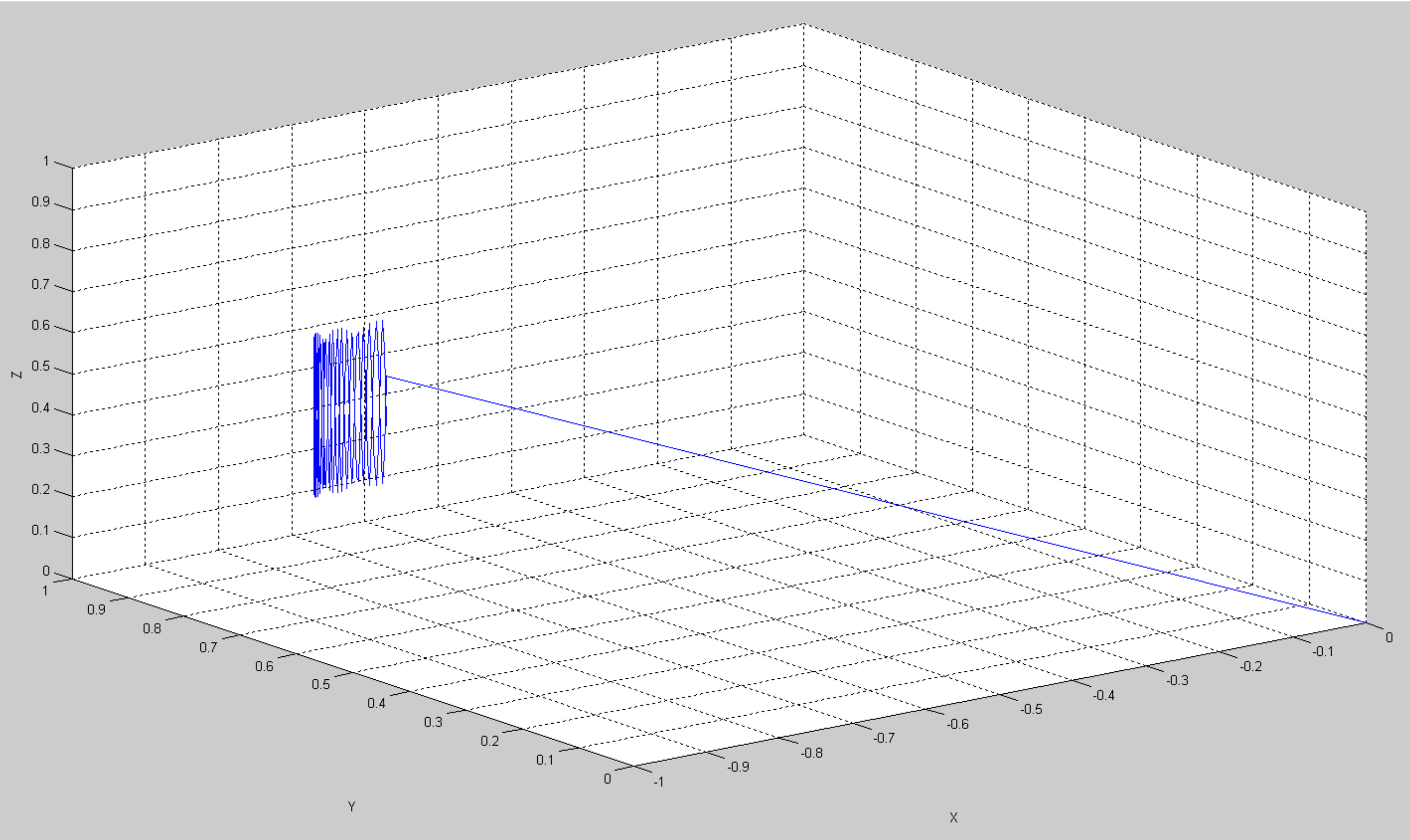
3.1 Superficie del área de trabajo del robot:

Los planos que a continuación se presentan fueron hechos en Matlab y con los siguientes valores fijos:

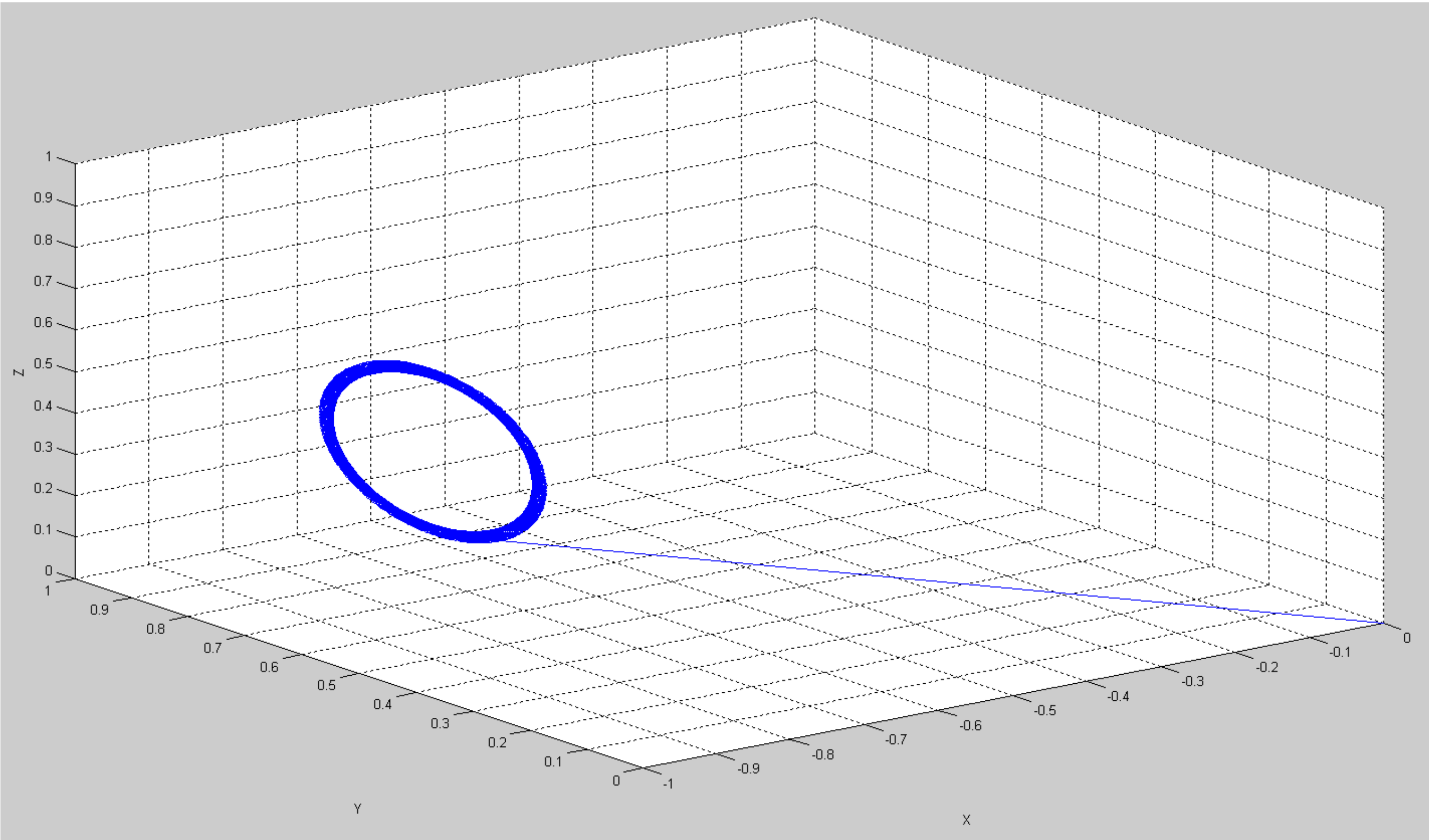
D1=0.5 D2=0.5 D3=0.5 L3=0.1 L4=0.1 L5=0.1

El ángulo de recorrido de θ_5 será de -60º a 60º con 10 pasos de resolución.

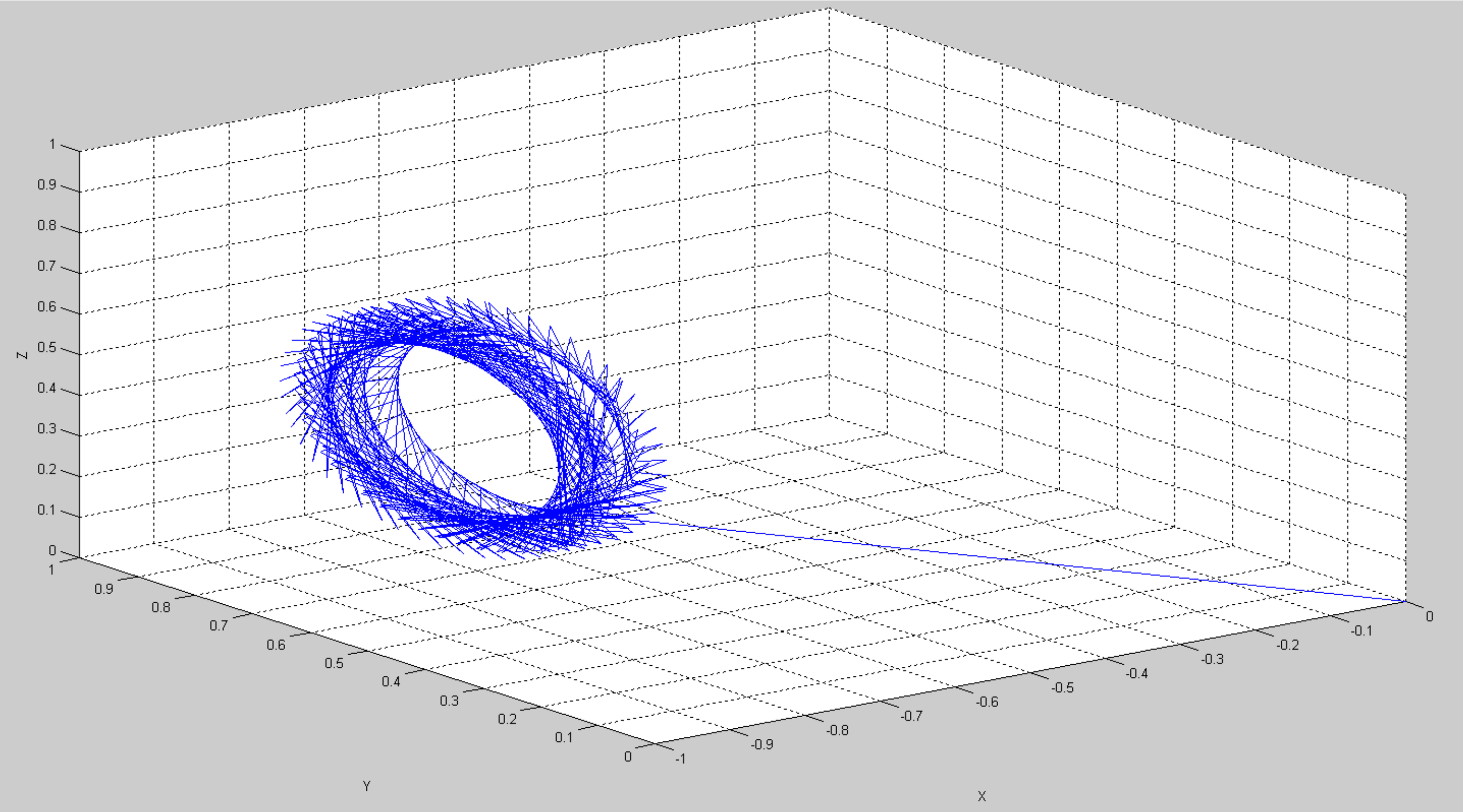
3.1.a Plano XYZ, θ_4=0



3.1.b Plano XYZ, θ_5=0



3.1.c Plano XYZ completo



Conclusiones

Los criterios establecidos para esta simulación son exactamente los mismos que para la anterior (DSP, variables, etc). Pero además, al introducirle 3 grados de libertad adicionales, el cálculo de la matriz transformación tiene un nivel más alto de complejidad.

Como al interesarnos únicamente el punto (u,v,w)=(0,0,0) perteneciente al extremo del robot, observamos que el eje giratorio de la punta (última articulación) no produce efectos en el extremo.

En cuanto a la resolución, la de los 2 primeros gráficos es de 180 pasos, por tratarse de simulaciones con una sola variable independiente. La última simulación se opto por una resolución de 50 pasos del total para facilitar la vista del espacio de trabajo.

De este modo podemos concluir que para un robot con 6 grados de libertad, la resolución que usemos es un factor muy importante. En el ejemplo anterior las coordenadas finales del extremo del robot son bastante simples de calcular mientras que en la última simulación el procesamiento será más exhaustivo.