Trabajo Práctico Nº 1

Implementación de una matriz cinemática en DSP

Cátedra: Robótica - Plan 95A

Grupo Nº

Integrantes:

ALONSO, NICOLAS ALEJANDRO 120641-2
PEREZ, HERNAN FACUNDO 120971-1

Fecha de Entrega:

10-05-2010

Fecha Aprobación:

Docente: Ing. Hernán Giannetta

Introducción a la cinemática del Robot 3
Cinemática directa del robot sin gripper 6
Matriz de roto traslación universal 7
Desarrollo e implementación en Codewarrior DSP 56800-E 8
Rutina 9
Resultado de la simulación 10
Simulación Nº1: 10
Simulación Nº2: 10
Simulación Nº3: 11
Simulación Nº4: 12
Obtención del espacio de trabajo 12
Conclusiones 13
Cinemática directa del robot con gripper 15
Matriz de roto traslación universal 16
Simulación 18
Conclusiones 21

Introducción a la 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 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 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 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 cinemático 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.

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:

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. 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, ${}^{0}A_1$ describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base, ${}^{1}a_2$ describe la posición y orientación del segundo eslabón respecto del primero, etc. Del mismo modo, denominando ${}^{0}A_k$ 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.

a_i = Distancia desde la intersección de x_i y z_{i-1} hasta O_i, a lo largo de x_i.
d_i = Distancia desde O_{i-1} a la intersección de x_i y z_{i-1}, a lo largo de z_{i-1}.
θ_i = Angulo entre z_{i-1} y z_i medido alrededor de x_i.
α_i = Angulo entre x_{i-1} y x_i alrededor de z_{i-1}.

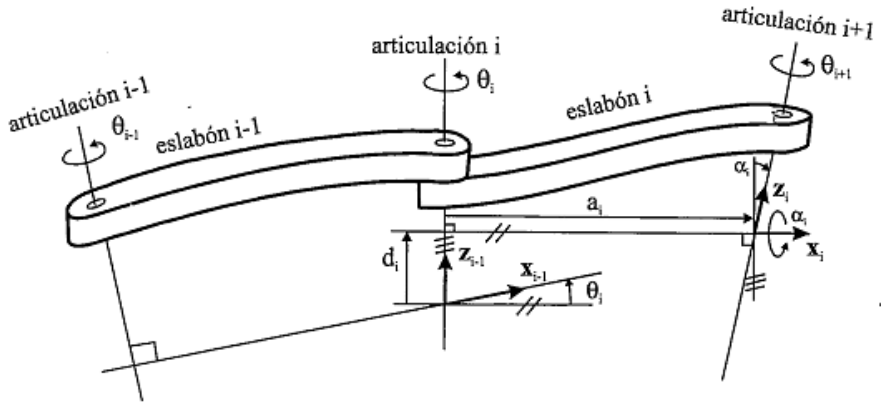


Figura 4.3. Parámetros D-H para un eslabón giratorio.

- *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 Z_i, sobre el eje de la articulación i+1.

- *DH5:* 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 situaran dé modo que formen un sistema dextrógiro con Z_0.

- *DH6:* 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 (Si) en el punto de corte. Si fuesen paralelos (Si) se situaría en la articulación i+1.

- *DH7:* Situar X_i en la línea normal común a Z_{i-1} y Z_i.

- *DH8:* Situar Y_i de modo que forme un sistema dextrógiro con X_i y Z_i.

- *DH9:* Situar el sistema (Sn) 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.

- *DH10:* Obtener $Ø_i$ como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y X_i queden paralelos.

- *DH11:* Obtener di 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.

- *DH12:* Obtener a_i como la distancia medida a lo largo de X_i (que ahora coincidiría con X_{i-1}) que habría que desplazar el nuevo (S_{i-1}) para que su origen coincidiese con (S_i).

- *DH13:* Obtener a_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).

- *DH14:* Obtener las matrices de transformación ${}^{i-1}a_i$.

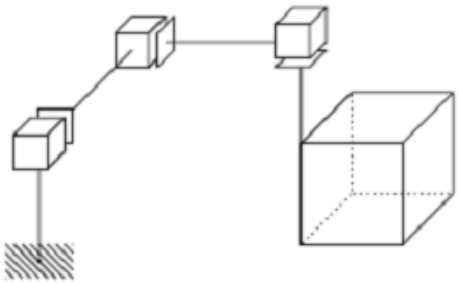
- *DH15:* Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot T = ${}^0A_1$, ${}^1A_2$... ${}^{n-1}A_n$.

- *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.

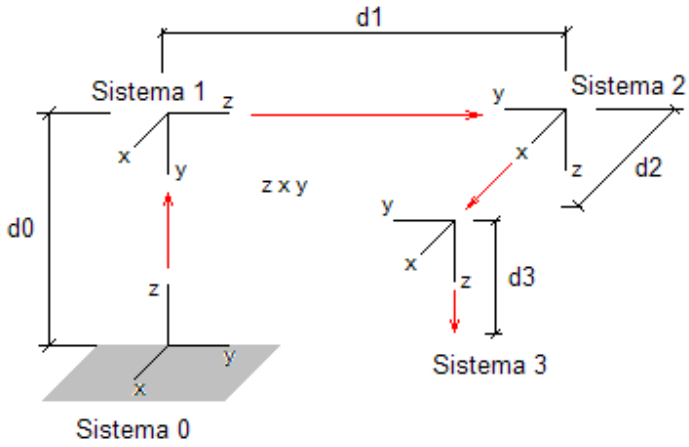
$$T = {}^0A_1 * {}^1A_2 * {}^2A_3 * {}^3A_4 * ..... * {}^{n-1}A_n$$

Cinemática directa del robot sin gripper

El modelo del robot a esquematizar y controlar es el siguiente:



Aplicando las reglas D-H, se logra esquematizar el sistema de manera tal que:



Sistema	θ_i	d_i	a_i	α_i
1	0	d_0	0	-90
2	0	d_1	0	-90
3	0	0	d_2	0
4	0	d_3	0	0

Considerando la posición del Sistema 0, a la posición x-y extremo del robot

Matriz de roto traslación universal

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

$${}^0A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2A_3 = \begin{bmatrix} 1 & 0 & 0 & d_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^3A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = {}^0A_1 * {}^1A_2 * {}^2A_3 * {}^3A_4 = \begin{bmatrix} 1 & 0 & 0 & d_2 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & -1 & d_0 - d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para saber la locación de la herramienta desde el punto de vista de las coordenadas (0,0,0) de nuestro sistema de referencia (Sistema 0):

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_4^0 * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_2 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & -1 & d_0 - d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d_2 \\ d_1 \\ d_0 - d_3 \\ 1 \end{bmatrix}$$

Desarrollo e implementación en Codewarrior DSP 56800-E

Para la realización del programa, se optó por utilizar el tipo de variable que provee el fabricante, y por el cual optimiza el manejo de variables con punto flotante, tales como las funciones trigonometriítas, cuadrados y raíces. La misma se basa en un concepto similar al del punto fijo. El mismo se conoce con el nombre de Notación Fraccional.

Lo que se buscará, es trabajar siempre con este tipo de variables, y luego para realizar la escala, recién ahí cambiar a otro tipo de variable que no tenga soporte por hardware, de manera tal de mejorar el rendimiento del programa en cuestión.

Para realizar esto, en primer momento normalizaremos todas nuestras variables, de manera tal que entren dentro del rango de [-1,1). Este es el momento de decidir si utilizamos simple precisión (16bits) o doble (32bits). Claro esta que solamente se debe de recurrir a la precisión doble en casos que así lo requiera, ya que el DSP seleccionado solo posee un acumulador de 16bits, y por consiguiente, las operaciones de palabras mayores incurrirán en rutinas de software que den soporte al mismo (mas lentas). Este tipo de variables optimizará el programa, sobre todo al trabajar con funciones trigonométricas, las cuales se utilizan mucho en robótica para confeccionar las matrices de roto translación.

En nuestro caso, al no tener articulaciones con movimientos angulares, y solamente se tienen movimientos lineales, se podría utilizar otro tipo de variable y luego escalarlas, pero con el objetivo de tomar practica con las herramientas del IDE, se optará realizarlo con este tipo de variables. Por lo cual, se normalizarán las distancias d1, d2 y d3 con respecto de si mismas, y en el caso de tener que hacer alguna operación de algebraica, recién ahí se volverá a escala.

El próximo punto en cuestión, será la elección del encoder, el cual nos traducirá los movimientos (en este caso lineales) a señales eléctricas que nosotros podremos observar para controlar el avance que nosotros deseamos realizar. Este nos dará, la sensibilidad máxima que va tener nuestro robot (mínimo paso el cual nosotros podremos controlar).

Debido al lento comportamiento del simulador del IDE, se mostrará el comportamiento para 3 tipos de encoders distintos. Uno hipotético de 10 pulsos en la longitud d_x, y otro de 100. En el caso del encoder sobre el eje Z, no se hicieron modificaciones, ya que el mismo, a no tener movimientos rotacionales en su extremo, no se observarán en la simulación (aun así, se realizo el programa que lo controla adecuadamente).

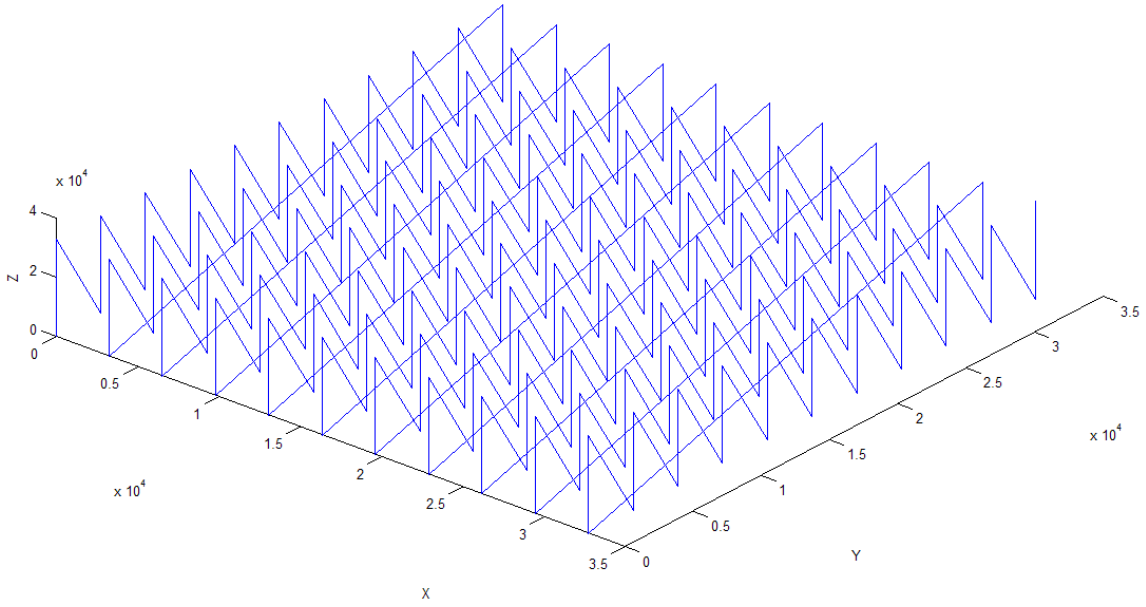
Rutina

```
#define   ENCODER1 100              // Pulsos del encoder en
                                    //la longitud d1
#define   ENCODER2 100              // Pulsos del encoder en
                                    //la longitud d2
#define   ENCODER3 100              // Pulsos del encoder en
                                    //la longitud d3

#define   PULSEd1   32767/ENCODER1 // “Distancia” que
                                    //conlleva cada pulso
#define   PULSEd2   32767/ENCODER2
#define   PULSEd3   32767/ENCODER3


Word16 index1,index2,index3;


void main(void)
{

Frac16 d1,d2,d3,z;
PE_low_level_init();
d1=0;
  for(index1=0;index1<ENCODER1;index1++)
     {
     d2=0;
     for(index2=0;index2<ENCODER2;index2++)
          {
          d3=0;
          for(index3=0;index3<ENCODER3;index3++)
               {
               z=sub(32767,d3);
               printf("%d\t%d\t%d\n",d1,d2,z);
               d3=add(d3,PULSEd3);
               }
               d2=add(d2,PULSEd2);
          }
     d1=add(d1,PULSEd1);
      }
}
```

Resultado de la simulación.

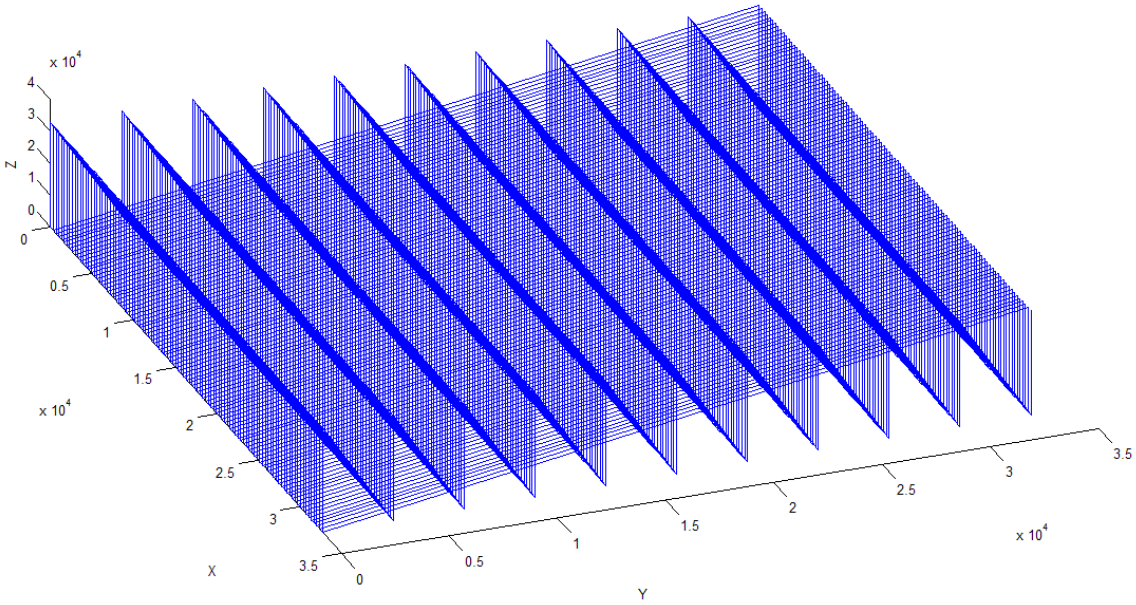
Simulación Nº1:

ENCODER1 (X) = 10 pulsos
ENCODER2 (Y) = 10 pulsos
ENCODER3 (Z) = 100 pulsos



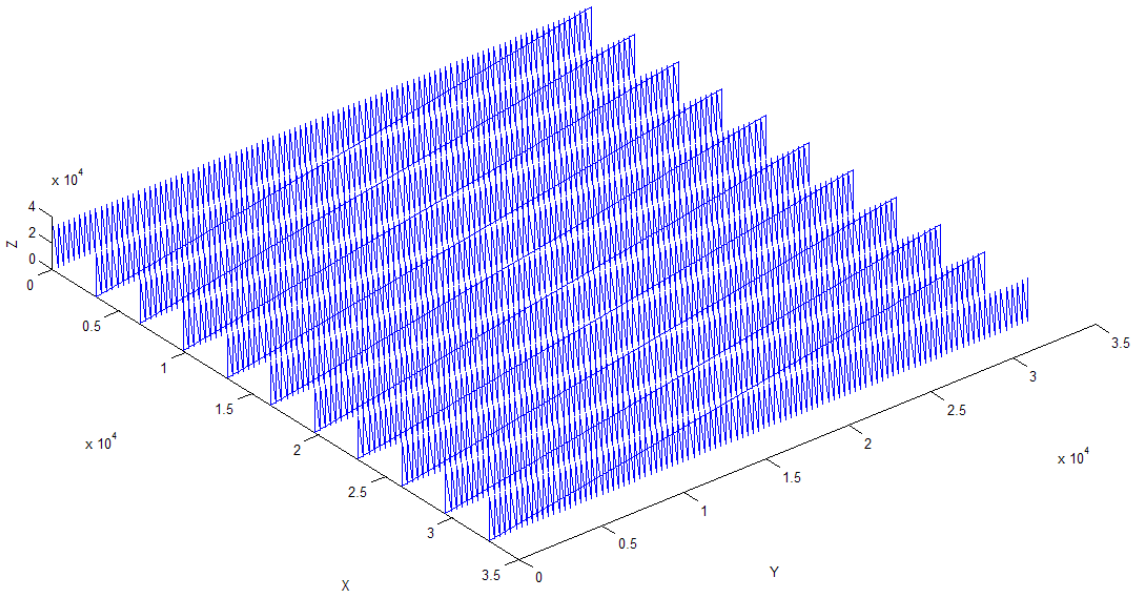
Simulación Nº2:

ENCODER1 (X) = 100 pulsos
ENCODER2 (Y) = 10 pulsos
ENCODER3 (Z) = 100 pulsos



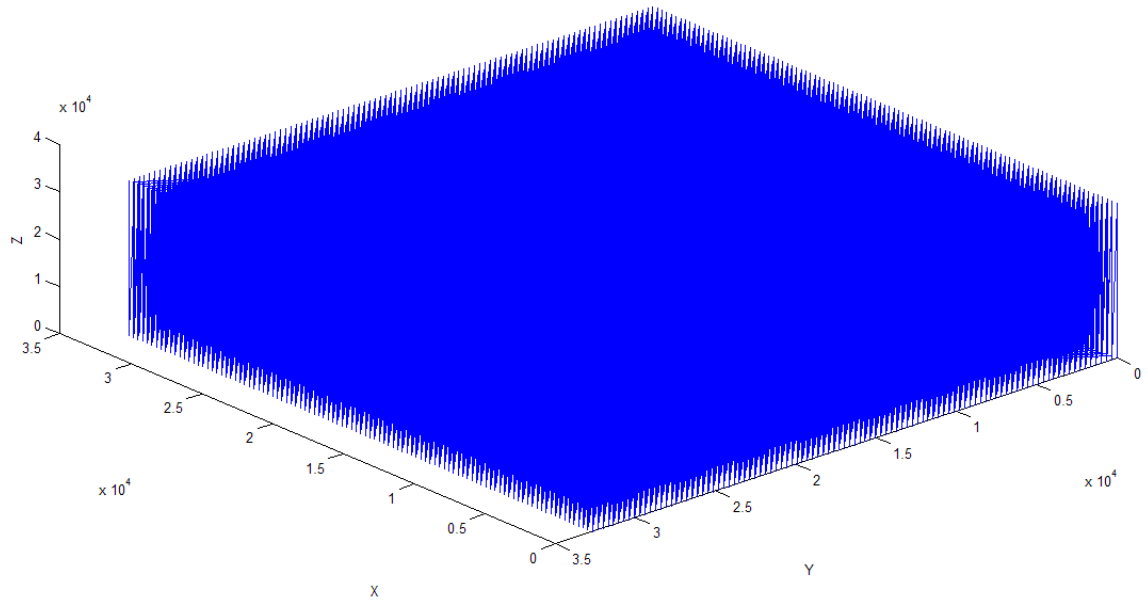
Simulación Nº3:

ENCODER1 (X) = 10 pulsos
ENCODER2 (Y) = 100 pulsos
ENCODER3 (Z) = 100 pulsos



Simulación Nº4:

ENCODER1 (X) = 100 pulsos
ENCODER2 (Y) = 100 pulsos
ENCODER3 (Z) = 100 pulsos



Obtención del espacio de trabajo

Una cosa importante en nuestro estudio, es saber cual es el espacio de trabajo de nuestro robot, ya que como medida de seguridad se suele aislar al mismo, de manera que no sea posible entrar en contacto directo.

En este caso, el área de trabajo la podemos observar en el caso de Simulación Nº4, en donde se le dio al robot la máxima resolución (para este caso). Claramente se observa que el espacio de trabajo es un cubo de dimensiones $d_1 x d_2 x d_3$.

Analíticamente, se debería generar un programa que, analizando el vector de salida, haga:

- Se genera un plano z=k, con k desde 0 hasta d_3.
- Se genera un plano de y=v, con v desde 0 hasta d_2.
- Se buscan los elementos que pertenezcan a la recta resultante de la intersección de los planos anteriores y al espacio de trabajo, y nos quedamos con el valor máximo y mínimo de x.
- Dichas coordenadas (x,y,z), una vez recorridos todos los k y v posibles, dan la superficie de trabajo del robot.

Conclusiones

Este tipo de robot con 3 grados de libertad, es ampliamente utilizado en la industria en aplicaciones de corte de planchuelas (hidráulico, láser, etc.) o fresadoras. Dicho tipo de movimientos lineales le da las limitaciones obvias del caso, pero facilita drásticamente la mecánica, y por consiguiente, su confiabilidad y costo.

Una cosa fundamental de dicho robot, es la sensibilidad que tenga el mismo para posicionarse en un punto de su espacio de trabajo. Para lograr esto, se debe trabajar sobre 3 pilares:

- Motores
- Reductores - Transmisión
- Sensores de posición

La combinación motor-reductores/transmisión, nos dará el torque y sensibilidad máxima capaz de ejecutar la maquina (idea). Un caso a nombrar, son aquellos sistemas con motores paso a paso, que permiten movimientos rápidos y precisos, pero con pasos fijados en grados de giro. En este caso, con bloques reductores, se puede ganar en exactitud y torque.

Al mismo tiempo, los sensores de posición nos dictan la posibilidad de poder controlar al sistema completo cerrando el lazo. De nada nos sirve tener un sistema de posicionamiento excelente, si la detección del mismo es inadecuada. De más esta decir, que si el sensor se conecta sobre el eje del motor, antes de las reducciones, también se ganará en precisión para el posterior control.

En nuestro caso, hemos supuesto que la sensibilidad del conjunto motor-reductor es mucho más grande que la del sensor de posición, y por consiguiente, este último es el que limita la sensibilidad del sistema.

Mención especial deberíamos darle a la utilización de números fraccionales para mejorar la performance de procesamiento del DSP. Su utilización (simplificación) acarrea también un error, y se deberá seleccionar el tipo de variable para hacer despreciable el mismo, con respecto a los errores que introduce el sistema externo.

Dicha resolución estará dada por la cantidad de bits de precisión con la que se declara la variable. Por consiguiente, podemos decir que si se trabaja con un sensor de posición que tenga una sensibilidad tal, que pueda dividir la distancia a sensar hasta en 2^{16}, (considerando un esquema es la que se utilicen tanto los números positivos como los negativos) se puede utilizar Frac16. Más allá de eso, hay que pasar al Frac32. Cabe destacar que sea un encoder incremental, como un

servomotor a tensión continua, este último siempre terminará pasando por un conversor AD, por lo que la discretización esta garantizada.

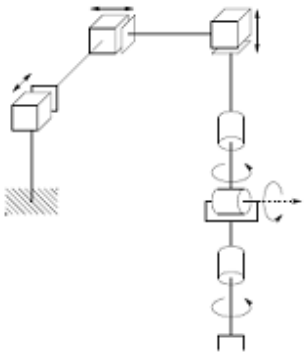
En nuestro caso, hemos optado por simular un sistema con un encoder hipotético de 10 y 100 pasos en la longitud total de trabajo, que por la estructura que hemos utilizado, únicamente utilizamos números positivos. Esto se hizo así ya que la velocidad de entrega de los datos a la hora de simular es realmente lenta, por lo que dificultaba mucho la recolección de datos para su procesamiento en el Matlab.

Como era de esperarse, se observo claramente como el aumento en la resolución del encoder mejora la sensibilidad del instrumento, y cuando dicha resolución crece, se evidencia cada vez más el espacio de trabajo del robot. Es importante que para el estudio del espacio de trabajo se tomen pasos pequeños, ya que tomando pasos discretos más grandes, la herramienta puede pasar por sectores que la simulación no muestre.

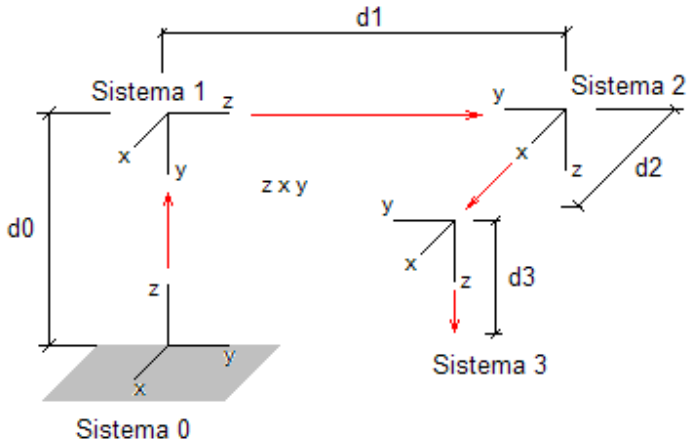
Al mismo tiempo, la elección del sistema es un compromiso costo-utilidad que hay que balancear. De nada sirve incrementar los costos en un encoder de mucha resolución, si el mercado final del robot no lo necesita, y la competencia del mercado hace que sus costos sean demasiado elevados.

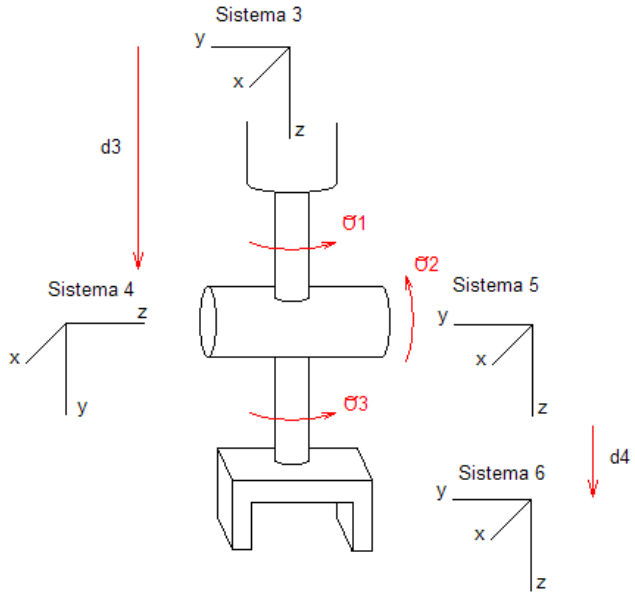
Cinemática directa del robot con gripper

El modelo del robot a esquematizar y controlar es el siguiente:



Aplicando las reglas D-H, se logra esquematizar el sistema de manera tal que:





Sistema	θ_i	d_i	a_i	α_i
1	0	d_0	0	-90
2	0	d_1	0	-90
3	0	0	d_2	0
4	θ_1	d_3	0	90
5	θ_2	0	0	-90
6	θ_3	d_4	0	0

Considerando la posición del Sistema 0, a la posición x-y extremo del robot

Matriz de roto traslación universal

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

$$^0A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^1A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{2}A_{3} = \begin{bmatrix} 1 & 0 & 0 & d_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{3}A_{4} = \begin{bmatrix} \cos(\sigma_1) & 0 & \sin(\sigma_1) & 0 \\ \sin(\sigma_1) & 0 & -\cos(\sigma_1) & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}A_{5} = \begin{bmatrix} \cos(\sigma_2) & 0 & -\sin(\sigma_2) & 0 \\ \sin(\sigma_2) & 0 & \cos(\sigma_2) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{5}A_{6} = \begin{bmatrix} \cos(\sigma_3) & -\sin(\sigma_3) & 0 & 0 \\ \sin(\sigma_3) & \cos(\sigma_3) & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ T = {}^{0}A_{1} * {}^{1}A_{2} * {}^{2}A_{3} * {}^{3}A_{4} * {}^{4}A_{5} * {}^{5}A_{6} = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ a_{11} = \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_3) - \sin(\theta_1) * \sin(\theta_3) $$
$$ a_{13} = -\cos(\theta_1) * \cos(\theta_2) * \sin(\theta_3) - \sin(\theta_1) * \cos(\theta_3) - \cos(\theta_1) * \sin(\theta_2) $$
$$ a_{14} = -\cos(\theta_1) * \sin(\theta_2) * \mathrm{d}_4 + \mathrm{d}_2 $$

$$ a_{21} = -\sin(\theta_1) * \cos(\theta_2) * \cos(\theta_3) - \cos(\theta_1) * \sin(\theta_3) $$
$$ a_{23} = \sin(\theta_1) * \cos(\theta_2) * \sin(\theta_3) - \cos(\theta_1) * \cos(\theta_3) + \sin(\theta_1) * \sin(\theta_2) $$
$$ a_{24} = \sin(\theta_1) * \sin(\theta_2) * \mathrm{d}_4 + \mathrm{d}_1 $$

$$ a_{31} = -\sin(\theta_2) * \cos(\theta_3) $$
$$ a_{33} = \sin(\theta_2) * \sin(\theta_3) - \cos(\theta_2) $$
$$ a_{34} = \mathrm{d}_0 - \mathrm{d}_3 - \cos(\theta_2) * \mathrm{d}_4 $$

Para saber la locación de la herramienta desde el punto de vista de las coordenadas (0,0,0) de nuestro sistema de referencia (Sistema 0):

$$ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_4^0 * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = T = \begin{bmatrix} a_{11} & 0 & a_{12} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d_2 - \cos(\theta_1) * \sin(\theta_2) * d_4 \\ d_1 + \sin(\theta_1) * \sin(\theta_2) * d_4 \\ d_0 - d_3 - \cos(\theta_2) * d_4 \\ 1 \end{bmatrix}$$

Como primera verificación, podemos llevar al caso particular del robot sin gripper. En dicho caso, todos los ángulos se hacen 0:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d_2 \\ d_1 \\ d_0 - (d_3 + d_4) \\ 1 \end{bmatrix}$$

y observamos que la única diferencia que hay esta en la coordenada Z, en donde quedaría $d_0 - (d_3 + d_4)$, ya que se agrego una nueva distancia hasta el gancho del gripper.

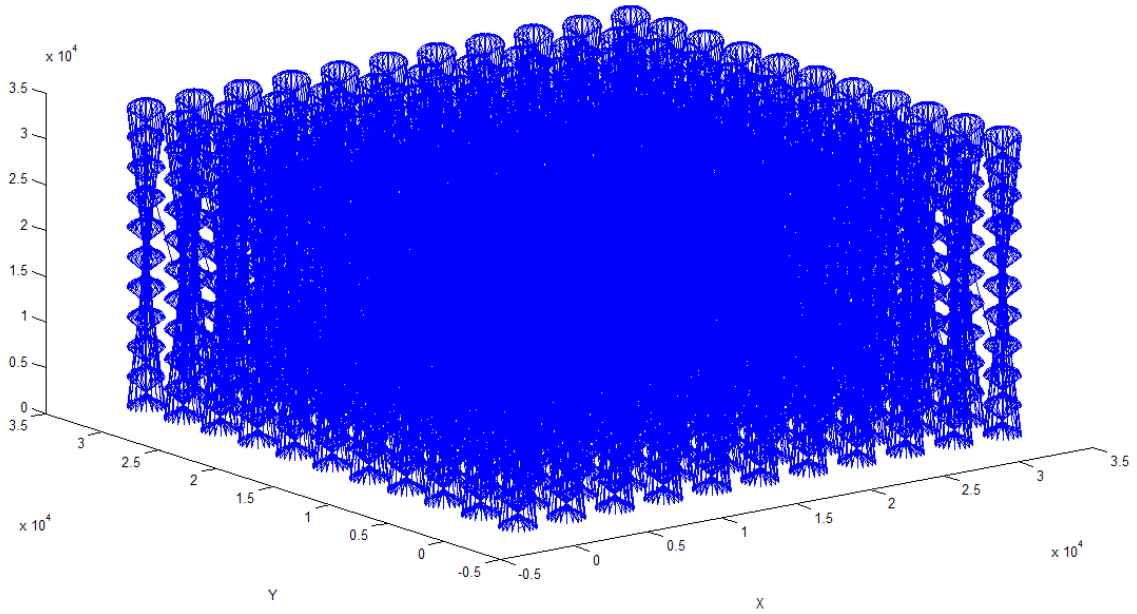
Simulación

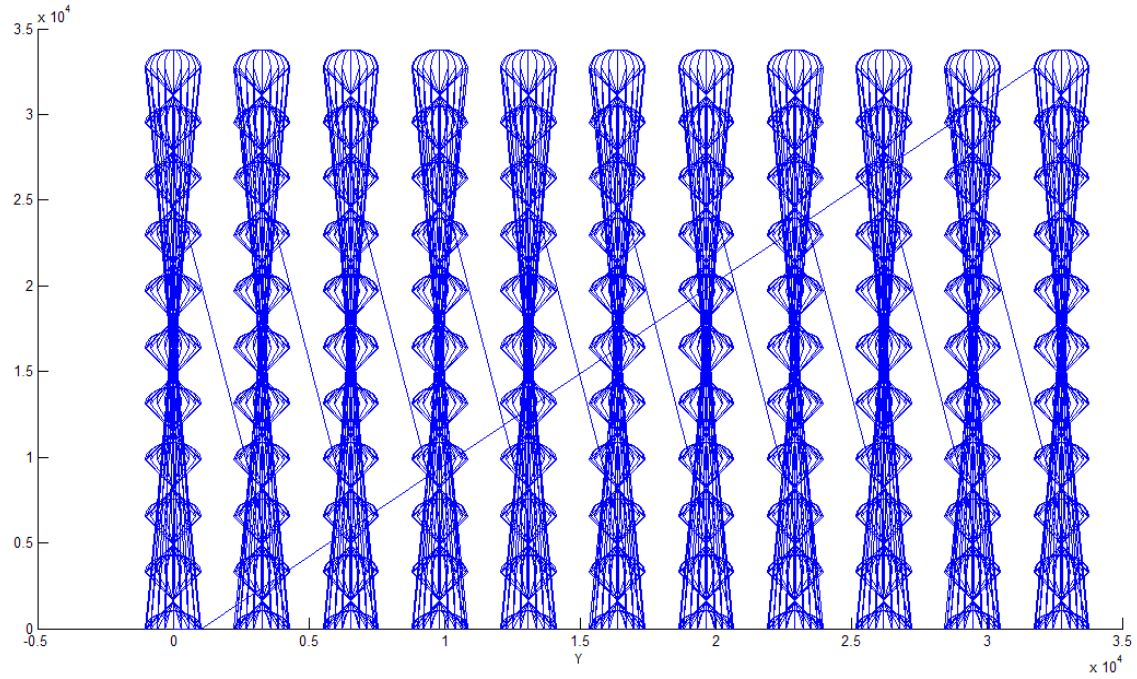
Debido a los tiempos, se opto por realizar la simulación directamente en Matlab. El script utilizado es el siguiente:

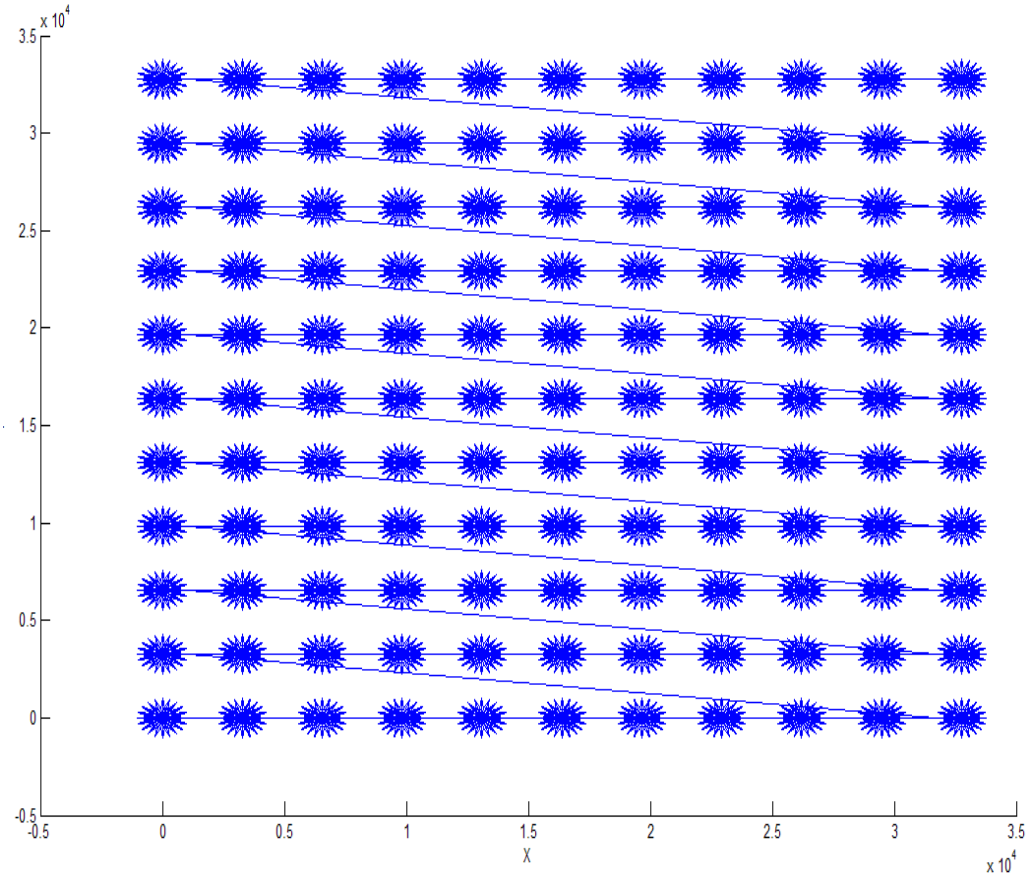
```
d1=0;d2=0;d3=0;
o2=-pi/2:pi/10:pi/2;

a=ones(1,11);

for x1 = 0:3276:32767
    for x2 = 0:3276:32767
        for o1=0:pi/10:2*pi
            for x3 = 0:3276:32767

                d1=cat(2,d1,a*x2-1024*cos(o1)*sin(o2));
                d2=cat(2,d2,a*x1+sin(o1)*sin(o2)*1024);
                d3=cat(2,d3,x3+cos(o2)*1024);

            end
        end
    end
end
plot3(d1,d2,d3)
```

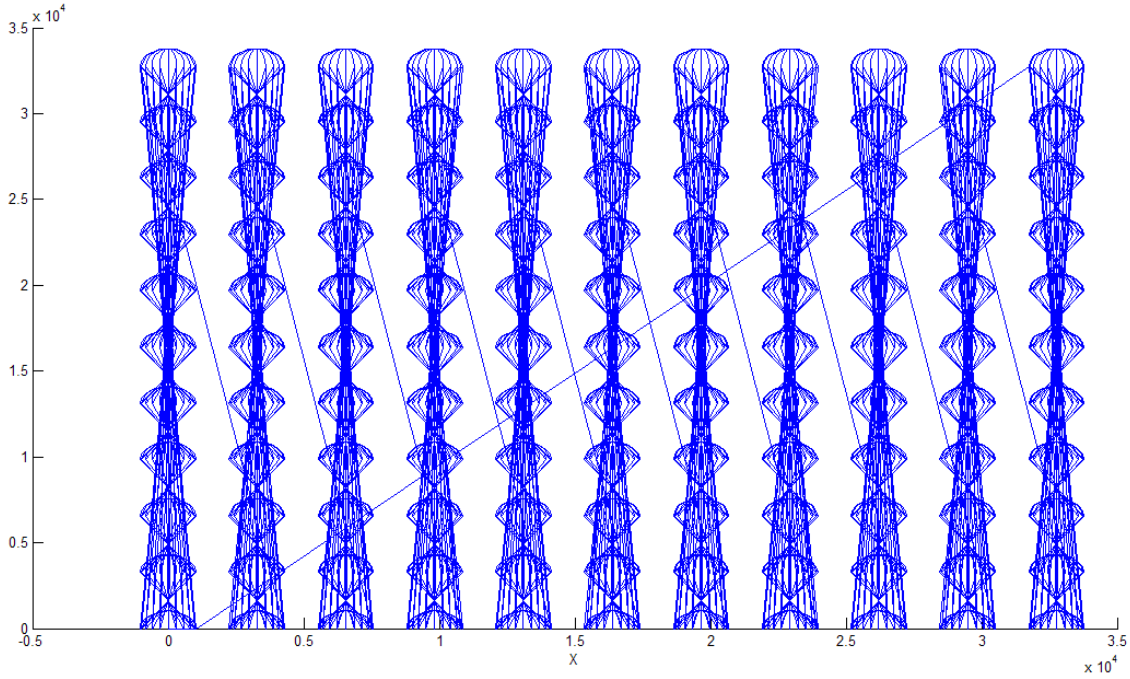
Como se puede observar, no se hicieron limitaciones para que no se superen los limites negativos, de manera de obtener los limites a implementar via soft luego de la simulación.

Al mismo tiempo, se ha limitado el movimiento de θ_2 entre -180º y +180º, mientras sobre θ_1 rota los 360º.






x 10^4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
X
x 10^4


x 10^4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
X
x 10^4

Conclusiones

Observamos como el campo de trabajo con el gripper se expande, añadiendo 2 grados de libertad al movimiento (θ_1 y θ_2), mas otro en el posicionamiento del gripper (θ_3). A costa de esto, se obtiene un mayor costo computacional, debido a que se deberán calcular límites y también dejar de trabajar con números fraccionados, al momento de sumar y restar distancias.

Se observa que el movimiento de θ_3 no tiene ninguna intervención dentro del espacio de trabajo matemáticamente. Esto en realidad no es tan así, ya que el gripper al girar, sobre su eje, moverá todo su volumen, apareciendo puntos de interés que aquí no se ven. Se podría decir que este es el caso para el cual el gripper tiene dimensiones despreciables con respecto a toda la estructura en general, y que los márgenes de seguridad que se le agregarán, salvaguardan este hecho.