Universidad Tecnológica
Nacional
Facultad Regional Buenos Aires

***Materia**: Robótica*
***Curso**: R-6055*
***Profesor**: Ing. Hernán Gianetta*
***JTP**: Ing. Damián Granzella*
***Año**: 2010*

***Trabajo Práctico N° 1**:*

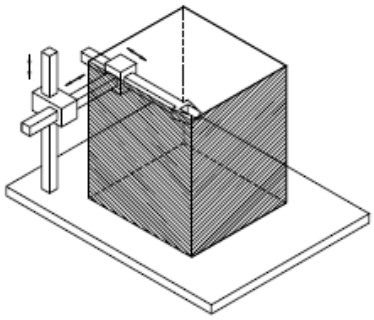
Implementación de Una Matriz Cinemática en DSP

***Alumnos**:Charec, Diego 113768-2*
Montaño Vallejos, Hector 116021-7

***Fecha de Entrega**: 21/05/2010*

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 limites y área de trabajo del manipulador.



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

Introducción Sobre Cinemática del Robot

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia, por lo que toma interés la relación entre la localización del extremo del robot y los valores de sus articulaciones, y también la descripción analítica del movimiento espacial en función del tiempo.

Para poder resolver la cinemática del robot, existen 2 problemas fundamentales:

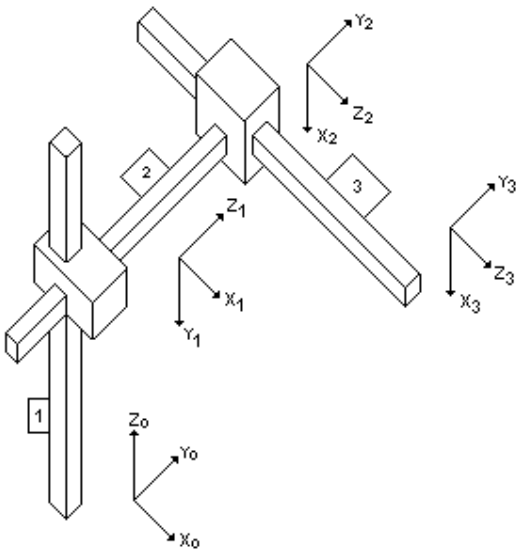
El **problema cinemático directo** que consiste en determinar la posición y orientación del extremo del robot, respecto a un sistema de referencia, conociendo los valores de movimiento de las articulaciones y los parámetros geométricos de los elementos del robot y puede ser resuelto por diferentes métodos:

- Utilizando matrices de transformación homogenea.
- Mediante el algoritmo de Denavit-Hartenberg.
- Cuaternios.

El segundo, el **problema cinemático inverso** que consiste en determinar la configuración que debe adoptar el robot para alcanzar una posición y orientación del extremo conocidas, en otras palabras, conozco mi sistema de referencia y a donde quiero llegar, mi problema es como lograrlo. Para resolverlo tenemos métodos geométricos, y nuevamente haciendo uso de la matriz de transformación homogenea.

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, que permite conocer las velocidades del extremo del robot a partir de las velocidades de cada articulación (matriz Jacobiana directa) o a la inversa (matriz jacobiana inversa).

Matriz del Brazo Robótico de 3 Grados de Libertad



Articulación	θ	d	a	α
1	0	d1	0	-90
2	90	d2	0	90
3	0	d3	0	0

- De los ejes de referencia 0 a 1, habrá una traslación en el eje Z seguida de una rotación en X de -90°.
- De los ejes 1 a 2 habrá una rotación de 90° mas una traslación en el eje Z, seguida de una rotación de 90° en el eje X.
- Finalmente, de los ejes 2 a 3 solo hay una traslación en el eje Z.

Reemplazando los valores de Denavit y Hartenberg en la matriz genérica obtengo las matrices A que relacionan los movimientos del robot.

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

De donde:

$$ A^0_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A^1_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A^2_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Y luego $T^0_3 = A^0_1 \cdot A^1_2 \cdot A^2_3$:

$$ T^0_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & d3 \\ 0 & 1 & 0 & d2 \\ -1 & 0 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

```
/**
###################################################################
**     Filename  : Traslacion_TPN1.C
**     Project   : Traslacion_TPN1
**     Processor : 56F8367
**     Version   : Driver 01.13
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 05/05/2010, 11:43
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
**
**     (c) Copyright UNIS, a.s. 1997-2008
**     UNIS, a.s.
**     Jundrovska 33
**     624 00 Brno
**     Czech Republic
**     http      : www.processorexpert.com
**     mail      : info@processorexpert.com
**
###################################################################*
/
/* MODULE Traslacion_TPN1 */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "MFR1.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 MAXPASO 5
#define PASO_NORMALIZADO 32767/(2*MAXPASO)
```

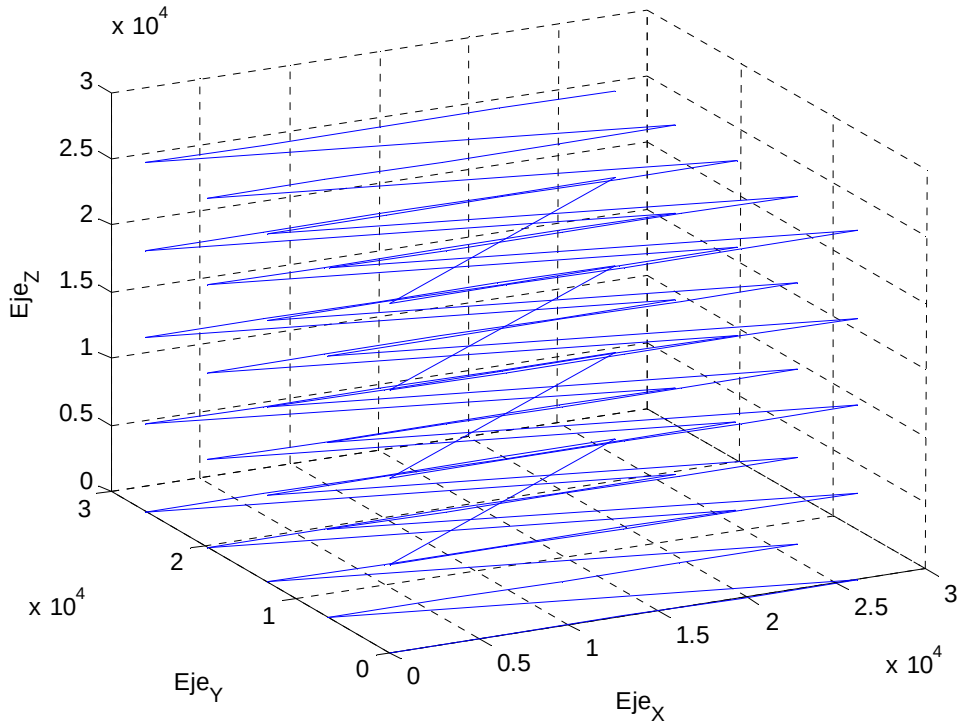
```
void main(void)
{
        Frac16 paso_normalizado=PASO_NORMALIZADO;
        Frac16 longitud1, longitud2, longitud3;
        Frac16 coordenadaX_16, coordenadaY_16, coordenadaZ_16;
        Frac32 coordenadaX_32, coordenadaY_32, coordenadaZ_32;

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

        for(longitud1=0;longitud1<MAXPASO;longitud1++)
        {
                for(longitud2=0;longitud2<MAXPASO;longitud2++)
                {
                        for(longitud3=0;longitud3<MAXPASO;longitud3++)
                        {
                                coordenadaX_32=(L_mult(paso_normalizado,longitud3));
                                coordenadaX_16=extract_l(coordenadaX_32);
                                coordenadaY_32=(L_mult(paso_normalizado,longitud2));
                                coordenadaY_16=extract_l(coordenadaY_32);
                                coordenadaZ_32=(L_mult(paso_normalizado,longitud1));
                                coordenadaZ_16=extract_l(coordenadaZ_32);

        printf("%d\t%d\t%d\n",coordenadaX_16,coordenadaY_16,coordenadaZ_16);
                        }
                }
        }
}
/* END Traslacion_TPN1 */
/*
**
###################################################################
**
**     This file was created by UNIS Processor Expert 2.99 [04.17]
**     for the Freescale 56800 series of microcontrollers.
**
**
###################################################################
*/
```

Matriz de Transformación Homogénea hecha por MATLAB

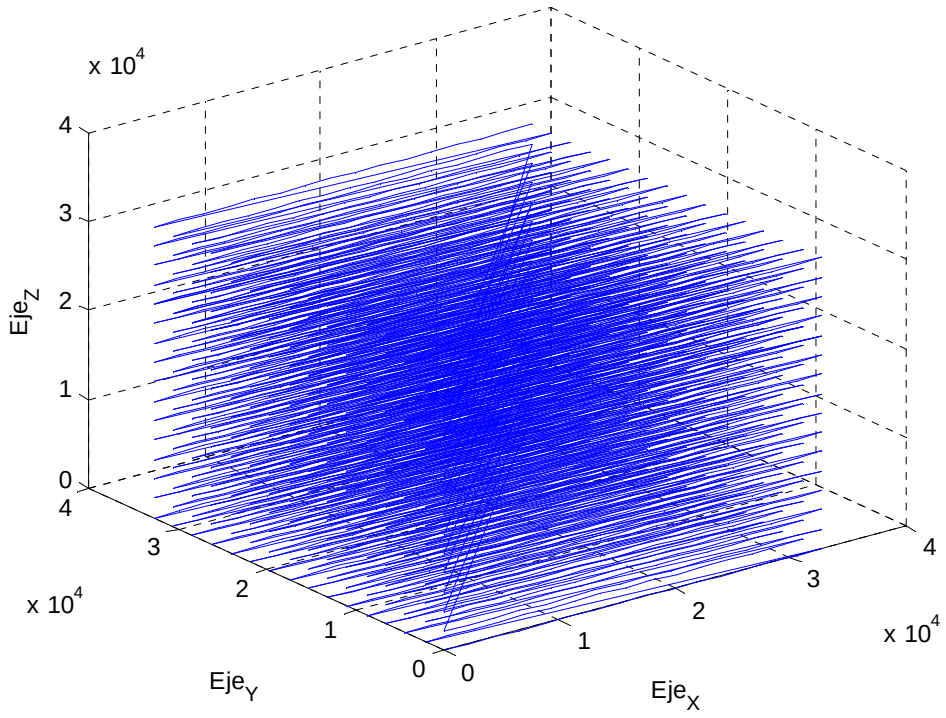
```
>> syms d1 d2 d3
>> A01=[1 0 0 0;0 0 1 0;0 -1 0 d1;0 0 0 1]

A01 =

[  1,  0,  0,  0]
[  0,  0,  1,  0]
[  0, -1,  0, d1]
[  0,  0,  0,  1]


>> A12=[0 0 1 0;1 0 0 0;0 1 0 d2;0 0 0 1]

A12 =

[  0,  0,  1,  0]
[  1,  0,  0,  0]
[  0,  1,  0, d2]
[  0,  0,  0,  1]


>> A23=[1 0 0 0;0 1 0 0;0 0 1 d3;0 0 0 1]

A23 =

[  1,  0,  0,  0]
[  0,  1,  0,  0]
[  0,  0,  1, d3]
[  0,  0,  0,  1]


>> T03=A01*A12*A23

T03 =

[  0,  0,  1, d3]
[  0,  1,  0, d2]
[ -1,  0,  0, d1]
[  0,  0,  0,  1]
```

Resultados de la simulación:

Usando MAXPASO = 5



Usando MAXPASO = 15



Conclusiones:

Para está configuración del brazo robot, al no tener articulaciones rotacionales, sino que cada una contribuye en una traslación de un punto sobre un eje, llegar a obtener el valor del punto final puede lograrse con solo observar un poco la configuración, donde se ve que el punto final al que voy a llegar va a ser mi punto de origen, desplazado en cada eje el valor que se haya desplazado cada articulación. Esto lo comprobamos después de aplicar el algoritmo de D-H y obtener la matriz de la transformación total, que luego verificamos en Matlab.

Con los gráficos de la simulación llegamos a ver el área que podría recorrer nuestro brazo robot, y comprobamos que es un cubo ya que supusimos las 3 articulaciones de la misma longitud, y por lo tanto en cada eje voy a poder desplazarme la misma distancia.

Para poder ver más en detalle hicimos 2 gráficas. En la segunda tomamos un valor mayor en MAXPASO, que según nuestro código determina el salto que realizaría nuestro brazo robot al pasar de un valor a otro en un eje. Por eso en la primer gráfica se ven tantos espacios en blanco, porque según nuestra programación esa separación sería mayor. En la 2° gráfica al haber aumentado ese valor, se reducen los espacios vacíos, y cada vez serían menos al seguir aumentando este valor, en otras palabras, disminuyendo el valor que puede desplazarse el robot en cada movimiento.

También de los gráficos vemos que podríamos mejorar nuestro código para que sea mas eficiente y mas rápido y por ej. en vez de volver al origen de un eje, para luego hacer un salto, se podría hacer directamente el salto al final, de esa manera solo tendríamos que modificar el valor de un eje y no de 2, que puede ser mas difícil a la hora de programarlo, pero obtendríamos un mejor resultado.