UTN – FRBA

Robótica

TPN1
Implementación de una matriz cinemática en DSP

Profesor: Hernán Giannetta

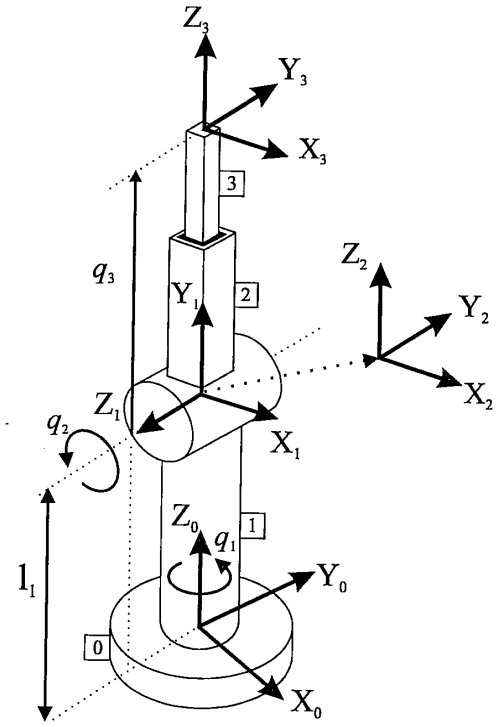
Alumnos:

Martín Cabrera	111492-0
Christian Gutierrez	96985-5

1C/2009

Enunciado:

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



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

Parámetros D-H para el robot polar de la figura

2. Imprima el resultado del vector (x,y,z) usando la función *plot3(x,y,z)* de Matlab®.

Resolución:

Introducción sobre cinemática del robot

Para hallar el modelo cinemático del robot propuesto, utilizamos el Algoritmo Denavit-Hartenberg que establece, por medio de reglas, un procedimiento normalizado cuyo resultado es la matriz homogénea.

Para el segmento 1:

$$^{0}A_{1} = [Rot\ q_1\ en\ z] \times [Rot\ 90^{o}\ en\ x + Traslación\ l_1\ en\ z]$$

$$^{0}A_{1} = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde: $C_1 = \cos(q_1)$ y $S_1 = \sin(q_1)$

Para el segmento 2:

$$^{1}A_{2} = [Rot\ q_2\ en\ z] \times [Rot\ -90^{o}\ en\ x]$$

$$^{1}A_{2} = \begin{bmatrix} C_2 & -S_2 & 0 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_2 & 0 & -S_2 & 0 \\ S_2 & 0 & C_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde: $C_2 = \cos(q_2)$ y $S_2 = \sin(q_2)$

Para el segmento 3:

$$^{2}A_{3} = [Traslación\ q_3\ en\ z]$$

$$^{2}A_{3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finalmente hallamos la matriz homogénea *T:*

$$ {}^0A_2 = {}^0A_1 \times {}^1A_2 = \begin{bmatrix} C_1 \cdot C_2 & -S_1 & -C_1 \cdot S_2 & 0 \\ S_1 \cdot C_2 & C_1 & -S_1 \cdot S_2 & 0 \\ S_1 & 0 & C_2 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ T = {}^0A_3 = {}^0A_2 \times {}^2A_3 = \begin{bmatrix} C_1 \cdot C_2 & -S_1 & -C_1 \cdot S_2 & -q_3 \cdot C_1 \cdot S_2 \\ S_1 \cdot C_2 & C_1 & -S_1 \cdot S_2 & -q_3 \cdot S_1 \cdot S_2 \\ S_1 & 0 & C_2 & q_3 \cdot C_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Para la realización de este Trabajo Práctico, sólo nos interesan las coordenadas de la posición de la punta del robot. Utilizaremos un sistema de coordenadas con origen en la base del robot (0,0,0) de manera que la submatriz *Traslación* nos de directamente las coordenadas del extremo.

$$ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} Rotación(3x3) \end{bmatrix} & \begin{bmatrix} Traslación(3x1) \end{bmatrix} \\ \begin{bmatrix} Perspectiva(1x3) \end{bmatrix} & Escalado \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

$$ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} Rxx & Rxy & Rxz & Tx \\ Ryx & Ryy & Ryz & Ty \\ Rzx & Rzy & Rzz & Tz \\ Px & Py & Pz & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} Tx \\ Ty \\ Tz \\ 1 \end{bmatrix} \Rightarrow \begin{cases} x = Tx \\ y = Ty \\ z = Tz \end{cases} $$

Para nuestro caso queda:

$$ \begin{cases} x = -q_3 \cdot C_1 \cdot S_2 \\ y = -q_3 \cdot S_1 \cdot S_2 \\ z = q_3 \cdot C_2 + l_1 \end{cases} $$

Que son las ecuaciones que utilizaremos en el DSP.

Desarrollo e implementación en CodeWarrior DSP 56800-E

La idea es utilizar el DSP 56800-E para simular el movimiento de las articulaciones del robot y obtener las coordenadas de su extremo. El programa imprimirá en la consola los valores de salida separados por tabulaciones. Luego estos valores serán utilizados para graficar la trayectoria en 3D utilizando Matlab.

Utilizamos la representación Frac16, que emplea 16 bits para representar todo el rango normalizado de una función. Es por ello que debemos normalizar los valores de entrada y desnormalizar los resultados para representarlos.

Normalización de ángulos:

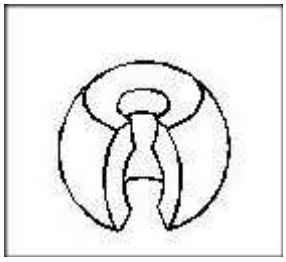
Grados	Radianes	Frac16
-180°	- π	-32768
0	0	0
180°	π	32767

Normalización de distancias:

Distancia	Frac16
-100 cm	-32768
0	0
100 cm	32767

Utilizamos los siguientes límites de movimiento para las articulaciones, que definen el espacio de trabajo del robot:

Articulación	Mínimo	Máximo
1 (rotación)	-135°	135°
2 (rotación)	30°	150°
3 (traslación)	25 cm	45 cm



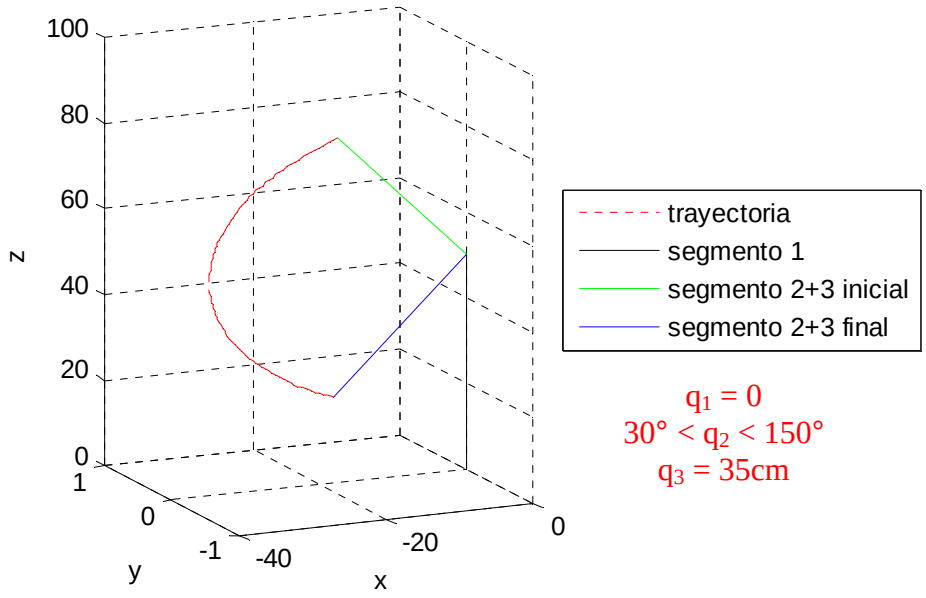
Espacio de trabajo

A continuación transcribimos el código fuente del programa principal:

```
#define L1     (short)(((long)50*32768)/100)        //defino largo fijo del "antebrazo"
                                                    // brazo vertical en 50cm

#define Q1INI  (short)(((long)(-135)*32768)/180)           //giro de la base entre -135
#define Q1FIN  (short)(((long)(135)*32768)/180)            // y 135 grados

#define Q2INI  (short)(((long)(30)*32768)/180)             //giro el codo entre 30 y 150
#define Q2FIN  (short)(((long)(150)*32768)/180)            //grados

#define Q3INI  (short)(((long)45*32768)/100)               //Extensión del brazo
#define Q3FIN  (short)(((long)25*32768)/100)               //entre 25 y 45cm

#define PUNTOS 101            //Resolución del movimiento

#define INC1 ((long)Q1FIN-Q1INI)/PUNTOS
#define INC2 ((long)Q2FIN-Q2INI)/PUNTOS
#define INC3 ((long)Q3FIN-Q3INI)/PUNTOS

Frac16 l1=L1,c1,c2,s1,s2,i,q1,q2,q3,x,y,z;
Frac16 inc1=INC1,q1fin=Q1FIN,q1ini=Q1INI;
Frac16 inc2=INC2,q2fin=Q2FIN,q2ini=Q2INI;
Frac16 inc3=INC3,q3fin=Q3FIN,q3ini=Q3INI;

Frac16 Origen[4][1],Pos[4][1];

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

  for(;;)
  {

    q1=q1ini;
    q2=q2ini;
    q3=q3ini;

    for(i=0;i<PUNTOS;i++)
    {
        c1=TFR1_tfr16CosPIx(q1);      //Cálculo de los cosenos y senos
        s1=TFR1_tfr16SinPIx(q1);
        c2=TFR1_tfr16CosPIx(q2);
        s2=TFR1_tfr16SinPIx(q2);

        x=negate(mult(q3,mult(c1,s2)));              //x=-q3 C1 S2

        y=negate(mult(q3,mult(s1,s2)));              //y=-q3 C1 S2

        z=add(L1,mult(q3,c2));                       //z=L1 + q3 C2

        printf ("%d \t %d \t %d \t %d \t %d \t %d \t %d \t \n",i,q1,q2,q3,x,y,z);

        q1=add(q1, inc1);             //incremento los valores de las articulaciones
        q2=add(q2, inc2);
        q3=add(q3, inc3);
    }

  }
}
```

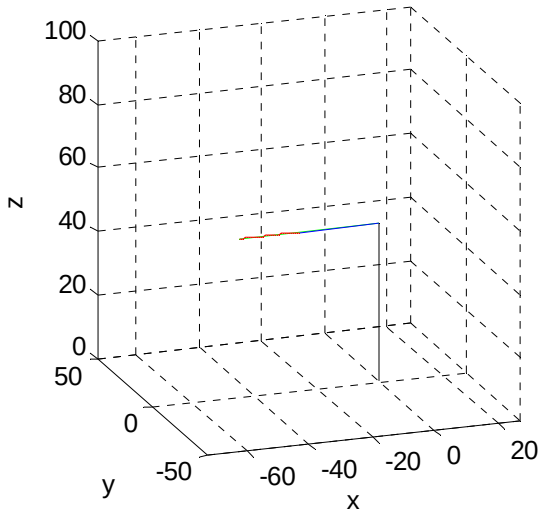
Resultados de la simulación

En primer lugar simulamos el movimiento de cada uno de los ejes por separado, manteniendo los restantes en su posición central:

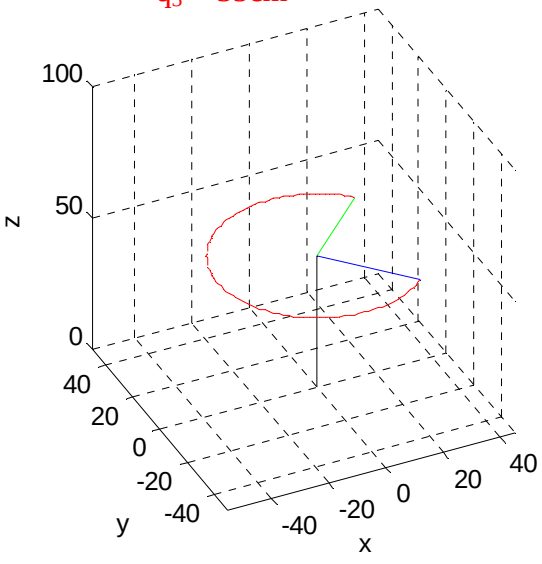


$q_1 = 0$
$30° < q_2 < 150°$
$q_3 = 35cm$

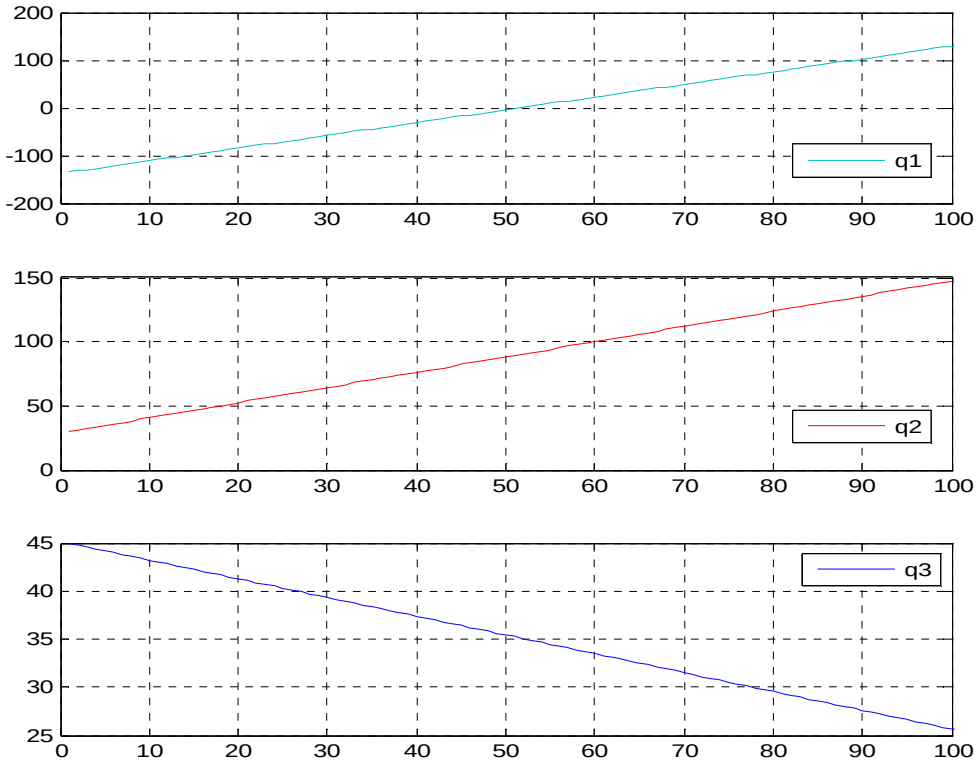
$q_1 = 0$
$q_2 = 90°$
$45cm > q_3 > 25cm$

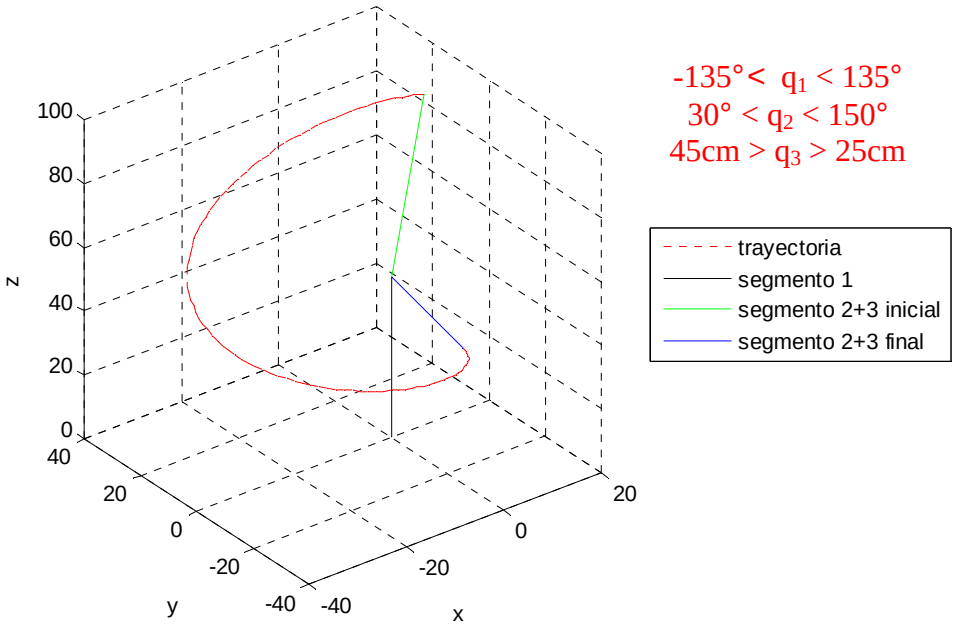


$-135° < q_1 < 135°$
$q_2 = 90°$
$q_3 = 35cm$



Finalmente simulamos el movimiento simultáneo de los 3 ejes, con variaciones lineales:





$-135° < q_1 < 135°$

$30° < q_2 < 150°$

$45cm > q_3 > 25cm$

Conclusiones finales

Pudimos comprobar mediante simulación el funcionamiento cinemático de un robot de 3 grados de libertad, pudiendo variar los límites de su campo de trabajo y viendo los resultados en 3 dimensiones. También comprobamos el modelo de matrices para representar la cadena cinemática del robot.