UNIVERSIDAD TECNOLOGICA NACIONAL

FACULTAD REGIONAL BUENOS AIRES

Trabajo Practico N°1

“Implementación de una matriz cinemática en DSP”

Materia: Robotica

Integrantes: Abaca, Daniel Alberto

Tunez, Diego

Profesor: Mas. Ing. Giannetta Hernan

JTP: Ing. Granzella Damian

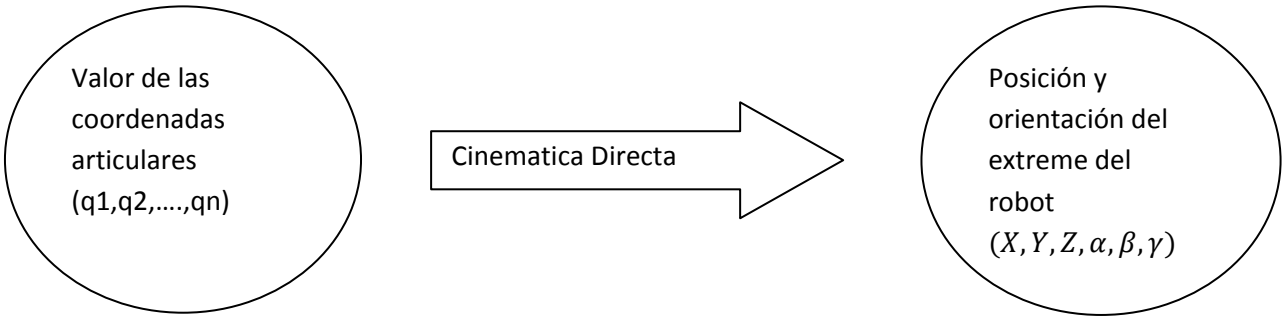
Introducción a la cinemática del robot:

A continuación se dará un resumen sobre la teoría necesaria para el desarrollo de la práctica posterior.

Definición de Cinemática: Es la parte de la Física que estudia el movimiento sin tener en cuenta las causas que lo producen, limitándose esencialmente al estudio de la trayectoria en función del tiempo.

La cinemática del robot se puede resolver mediante cinemática directa o cinemática inversa para lo que es el desarrollo de este trabajo práctico nos volcaremos al estudio de la cinemática directa del robot.

La cinemática directa consiste básicamente en determinar la posición y orientación del extremo del robot $(X, Y, Z, \alpha, \beta, \gamma)$ en función del valor de las coordenadas articulares.



El método que permite resolver la cinemática directa del robot es a través de un método matricial que propusieron Denavit y Hartenberg. Este permite establecer de manera sistemática un sistema de coordenadas {Si} ligado a cada eslabón i de una cadena articulada, pudiéndose determinar a continuación las ecuaciones cinemáticas de las ecuaciones 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 cuatro transformaciones básicas que dependen exclusivamente de las características geométricas de cada eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i con el sistema del elemento i-1.

Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema $\{S_{i-1}\}$ al $\{S_i\}$ mediante estas cuatro transformaciones está garantizado solo si los sistemas $\{S_{i-1}\}$ al $\{S_i\}$ han sido definidos de acuerdo a unas normas determinadas que se expondrán posteriormente):

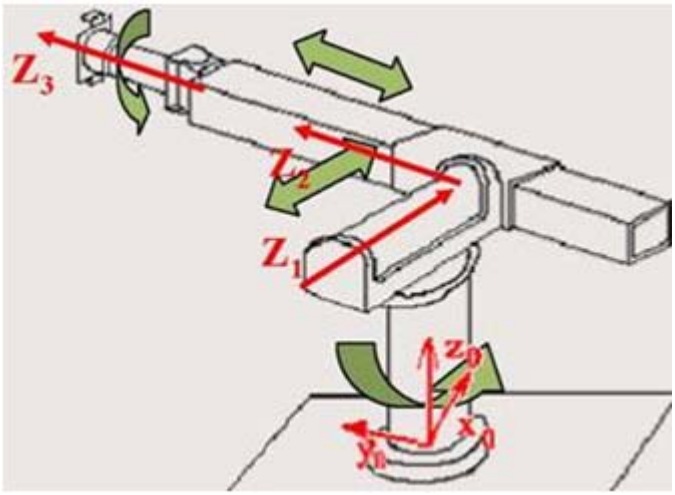
- Rotación alrededor del eje z_{i-1}un ángulo θ_i.
- Traslación a lo largo de z_{i-1}una distancia d_i; vector $d_i(0,0,d_i)$.
- Traslación a lo largo de x_i una distancia a_i; vector $a_i(0,0,a_i)$.
- Rotación alrededor del eje x_i un ángulo α_i.

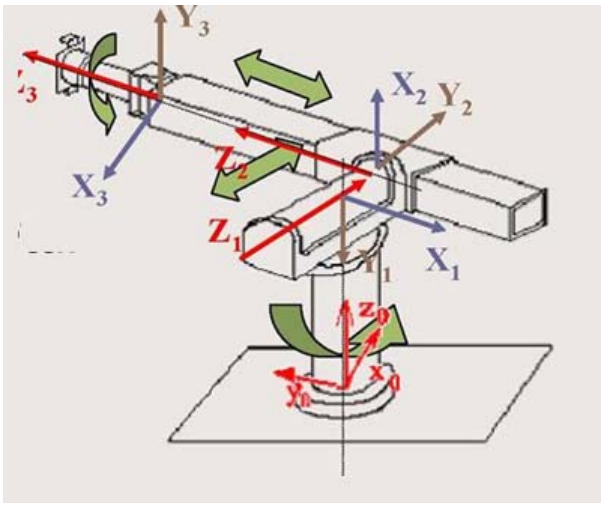
Como se dijo anteriormente por medio del método matricial Denavit y Hartenberg se obtiene el modelo cinematico del robot que es nada más y menos que ecuaciones que determinan la posición del extremo final del robot a través de los valores de las articulaciones y dimensiones del robot. Estas ecuaciones se obtienen considerando previamente las normas anteriormente mencionadas por medio del cálculo de la matriz de transformación homogénea total [T] que es el producto de matrices homogéneas que son en función de cada articulación. Para obtener dichas matrices se aplica el algoritmo de D.H.

Algoritmo de D.H:

- Elegir un sistema de coordenadas fijo (X0, Y0, Z0) asociado a la base del robot.
- Localizar el eje de cada articulación Z: Si la articulación es rotativa, el eje será el propio eje de giro. Si es prismática, el eje lleva a dirección de deslizamiento.

Ejemplo:





- Situar los ejes X a la línea normal común a Z_{i-1} y $Z_{i.}$
- Si estos son paralelos, se elige la línea normal que corta ambos ejes.
- El eje Yi debe completar el tríedro dextrógiro.

Parámetros D.H:

α_i: Angulo entre el eje Z_{i-1} y Z_i sobre el plano perpendicular X_i. El signo lo da la regla de la mano derecha.

a_i: Distancia entre los ejes Z_{i-1} y Z_i, a lo largo de X_i. El signo lo define el sentido X_i.

θ_i: Angulo que forman los ejes X_{i-1} y X_i, sobre el plano perpendicular a Z_i. El signo lo determina la regla de la mano derecha.

d_i:Distancia a lo largo del eje Z_{i-1} desde el origen del sistema S_{i-1} hasta la intersección del eje Z_i, con el eje X_i. En el caso de articulaciones prismáticas será la variable de desplazamiento.

Obtención de la Matriz homogénea Total [T]:

Una vez obtenidos los parámetros D.H para cada articulación en lo que se tendrá cuatro valores correspondiente a $(\alpha_{i,}a_{i,}\theta_i, d_i)$ para cada articulación q1, q2, q3,……, qn; estamos en condiciones de calcular la matriz homogénea para cada articulación respecto de otr articulación y finalmente la matriz de transformación homogénea total [T] como se muestra a continuación.

$${}^{i-1}A_i = Rot(Z_{i-1},\theta).Tras(0,0,d_i),Tras(a_i,0,0).Rot(X_i,\alpha_i)$$

Rot: Matriz homogénea de rotación; Tras: Matriz homogénea de traslación

$${}^{i-1}A_i = \begin{bmatrix} Cos(\theta i) & -Sen(\theta i) & 0 & 0 \\ Sen(\theta i) & Cos(\theta i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & di \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(\alpha i) & -Sen(\alpha i) & 0 \\ 0 & Sen(\alpha i) & Cos(\alpha i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{i-1}A_i = \begin{bmatrix} Cos(\theta i) & -Cos(\alpha i).Sen(\theta i) & Sen(\alpha i).Sen(\theta i) & ai.Cos(\theta i) \\ Sen(\theta i) & Cos(\alpha i)Cos(\theta i) & -Sen(\alpha i).Cos(\theta i) & aiSen(\theta i) \\ 0 & Sen(\alpha i) & Cos(\alpha i) & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

${}^{i-1}A_i$: $Matriz\ de\ transformacion\ homogenea\ desde\ el\ sistema\ i-1\ hasta\ el\ i(una\ por\ cada\ articulacion)$

Finalmente:

$$[T] = {}^{0}A_n = {}^{0}A_1.\ {}^{1}A_2 \dots\dots {}^{i-1}A_i \dots\dots {}^{n-1}A_n$$

[T]: Matriz de transformación homogénea total

Desarrollo de la práctica:

Antes de implementar el código en lenguaje C en el CodeWarrior para el DSP56800/E de la cadena cinemática directa del robot **M5 sin considerar el gripper** se procederá a calcular la matriz de transformación homogénea llamada matriz **[T]** de toda la cadena cinemática, aplicando el método matricial de **Denavit Hartenberg(D.H)**.

Primeramente se procede esquematizar el robot M5 para la aplicación del método de D.H indicando principalmente las articulaciones y los sistemas de referencias convenientes como se muestra a continuación:

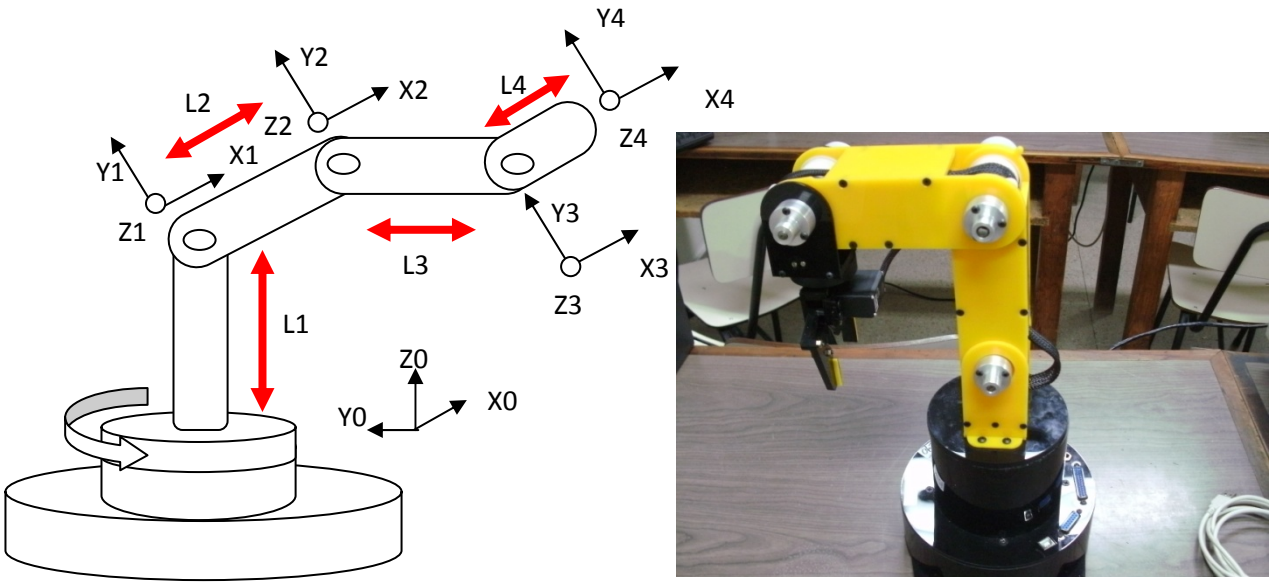


Figura_1: A la izquierda modelo del robot M5 sin considerar el gripper y a la derecha el robot M5.

En base a la figura_1 se aplica el método D.H y se determina para cada articulación los parámetros $\boldsymbol{\theta}, \mathbf{d}, \mathbf{a}, \boldsymbol{\alpha}$ que se detallan a continuación en una tabla:

	θ	d	a	α
Articulación 1	q1	L1	0	90 grados
Articulación 2	q2	0	L2	0
Articulación 3	q3	0	L3	0
Articulación 4	q4	0	L4	0

Nota: En realidad el robot M5 tiene cinco articulaciones (cinco grados de libertad) pero a fin de desarrollar este TP se nos pidió que lo consideremos solo con cuatro articulaciones (cuatro grados de libertad).

En función de la tabla anterior; se tiene una matriz homogénea particular para dos eslabones o articulaciones consecutivas **de acuerdo a una norma establecida** donde los valores de dicha matriz dependen de los parámetros $\boldsymbol{\theta}, \mathbf{d}, \mathbf{a}, \boldsymbol{\alpha}$ de la tabla y está dada por:

$$^{i-1}A_i = \begin{bmatrix} Cos(\theta i) & -Cos(\alpha i).Sen(\theta i) & Sen(\alpha i).Sen(\theta i) & ai.Cos(\theta i) \\ Sen(\theta i) & Cos(\alpha i)Cos(\theta i) & -Sen(\alpha i).Cos(\theta i) & aiSen(\theta i) \\ 0 & Sen(\alpha i) & Cos(\alpha i) & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$^{i-1}A_i$: Es una matriz de transformación homogénea que representa la orientación y posición relativa entre los sistemas asociados a dos eslabones consecutivos.

Dado que solo consideraremos el robot M5 sin gripper tendremos **solo cuatro grados de libertad** por lo tanto se tendrán solo cuatro matrices de transformación homogéneas $^{i-1}A_i$ las mismas serán las que se detallan a continuación:

Matrices de transformación homogénea referida a todos los sistemas asociados a dos eslabones consecutivos

$$^{0}A_1 = \begin{bmatrix} Cos(q1) & 0 & Sen(q1) & 0 \\ Sen(q1) & 0 & -Cos(q1) & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$^{0}A_1$:Matriz de transformación homogénea que describe la posición y orientación del sistema de referencia solidario al primer eslabón con respecto al sistema de referencia solidario a la base.

q1: articulación numero 1(es un ángulo)

L1: distancia del sistema de referencia a la articulación 1

$$^{1}A_2 = \begin{bmatrix} Cos(q2) & -Sen(q2) & 0 & L2.Cos(q2) \\ Sen(q2) & Cos(q2) & 0 & L2.Sen(q2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$^{1}A_2$:Matriz de transformación homogénea que describe la posición y orientación del sistema de referencia solidario al segundo eslabón con respecto al sistema de referencia solidario al primer eslabón.

q2: articulación numero 2(es un ángulo).

L2: distancia del sistema de referencia 1 a la articulación 2.

$$^{2}A_3 = \begin{bmatrix} Cos(q3) & -Sen(q3) & 0 & L3.Cos(q3) \\ Sen(q3) & Cos(q3) & 0 & L3.Sen(q3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

${}^{2}A_{3}$:Matriz de transformación homogénea que describe la posición y orientación del sistema de referencia solidario al tercer eslabón con respecto al sistema de referencia solidario al segundo eslabón.

q3: articulación numero 3(es un ángulo).

L3: distancia del sistema de referencia 2 a la articulación 3.

$$ {}^{3}A_{4} = \begin{bmatrix} Cos(q4) & -Sen(q4) & 0 & L4.Cos(q4) \\ Sen(q4) & Cos(q4) & 0 & L4.Sen(q4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{3}A_{4}$:Matriz de transformación homogénea que describe la posición y orientación del sistema de referencia solidario al cuarto eslabón con respecto al sistema de referencia solidario al tercer eslabón.

q4: articulación numero 4(es un ángulo).

L4: distancia del sistema de referencia 3 al extremo final del robot (X4, Y4, Z4).

<u>Obtención de la matriz de transformación homogénea **[T]** de toda la cadena cinemática</u>

Finalmente obtenidas anteriormente estas cuatro matrices de transformación homogénea se procede a determinar la matriz de transformación homogénea **[T]**, simplemente multiplicando las cuatro matrices.

$[T]=\left[{}^{0}A_{4} \right] = \left[{}^{0}A_{1} \right]\left[{}^{1}A_{2} \right]\left[{}^{2}A_{3} \right]\left[{}^{3}A_{4} \right]$

$$ \left[{}^{0}A_{1} \right].\left[{}^{1}A_{2} \right] = \begin{bmatrix} C1 & 0 & S1 & 0 \\ S1 & 0 & -C1 & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} C2 & -S2 & 0 & L2.C2 \\ S2 & C2 & 0 & L2.S2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = $$

$$ =\left[{}^{0}A_{1} \right].\left[{}^{1}A_{2} \right] = \begin{bmatrix} C1.C2 & -C1.S2 & S1 & C1.L2.C2 \\ S1.C2 & -S1.S2 & -C1 & S1.L2.C2 \\ S2 & C2 & 0 & L2.S2 + L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Nota: $\boldsymbol{C1 = Cos(q1);\ S1 = Sen(q1);\ C2 = Cos(q2); S2 = Sen(q2)}$

$$ \left[{}^{0}A_{1} \right].\left[{}^{1}A_{2} \right].\left[{}^{2}A_{3} \right] = \begin{bmatrix} C1.C2 & -C1.S2 & S1 & C1.L2.C2 \\ S1.C2 & -S1.S2 & -C1 & S1.L2.C2 \\ S2 & C2 & 0 & L2.S2 + L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C3 & -S3 & 0 & L3.C3 \\ S3 & C3 & 0 & L3.S3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Nota: $\boldsymbol{\left[{}^{0}A_{1} \right].\left[{}^{1}A_{2} \right].\left[{}^{2}A_{3} \right] = \left[{}^{0}A_{3} \right]}$

$$ \left[{}^{0}A_{3} \right] = \begin{bmatrix} C1.C2.C3 - C1.S2.S3 & -C1.C2.S3 - C1.S2.C3 & S1 & C1.C2.L3.C3 - C1.S2.L3.S3 + C1.L2.C2 \\ S1.C2.C3 - S1.S2.S3 & -S1.C2.S3 - S1.S2.C3 & -C1 & S1.C2.L3.C3 - S1.S2L3.S3 + S1.L2.C2 \\ S2.C3 + C2.S3 & -S2.S3 + C2.C3 & 0 & S2.L3.C3 + C2.L3.S3 + L2.S2 + L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$[T]=\left[{}^{0}A_{4} \right] = \left[{}^{0}A_{3} \right]\left[{}^{3}A_{4} \right]$

$$ [T] = \begin{bmatrix} C1.C2.C3 - C1.S2.S3 & -C1.C2.S3 - C1.S2.C3 & S1 & C1.C2.L3.C3 - C1.S2.L3.S3 + C1.L2.C2 \\ S1.C2.C3 - S1.S2.S3 & -S1.C2.S3 - S1.S2.C3 & -C1 & S1.C2.L3.C3 - S1.S2L3.S3 + S1.L2.C2 \\ S2.C3 + C2.S3 & -S2.S3 + C2.C3 & 0 & S2.L3.C3 + C2.L3.S3 + L2.S2 + L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C4 & -S4 & 0 & L4.C4 \\ S4 & C4 & 0 & L4.S4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = $$

$$[T] = \begin{bmatrix} T11 & T12 & T13 & T14 \\ T21 & T22 & T23 & T24 \\ T31 & T32 & T33 & T34 \\ T41 & T42 & T43 & T44 \end{bmatrix}$$

$T11 = C1.C2.C3.C4 - C1.S2.S3.C4 - C1.C2.S3.S4 - C1.S2.C3.S4$

$T12 = -C1.C2.C3.S4 + C1.S2.S3.S4 - C1.C2.S3.C4 - C1.S2.C3.C4$

$T13 = S1$

$T14 = C1.C2.C3.L4.C4 - C1.S2.S3.L4.C4 - C1.C2.S3.L4.S4 - C1.S2.C3.L4.S4 + C1.C2.L3.S3 - C1.S2.L3.S3 + C1.L2.C2$

$T21 = S1.C2.C3.C4 - S1.S2.S3.C4 - S1.C2.S3.S4 - S1.S2.C3.S4$

$T22 = -S1.C2.C3.S4 + S1.S2.S3.S4 - S1.C2.S3.C4 - S1.S2.C3.C4$

$T23 = -C1$

$T24 = S1.C2.C3.L4.C4 - S1.S2.S3.L4.C4 - S1.C2.S3.L4.S4 - S1.S2.C3.L4.S4 + S1.C2.L3.S3 - S1.S2.L3.S3 + S1.L2.C2$

$T31 = S2.C3.C4 + C2.S3.C4 - S2.S3.S4 + C2.C3.S4$

$T32 = -S2.C3.S4 - C2.S3.S4 - S2.S3.C4 + C2.C3.C4$

$T33 = 0$

$T34 = S2.C3.L4.C4 + C2.S3.L4.C4 - S2.S3.L4.S4 + C2.C3.L4.S4 + S2.L3.S3 + C2.L3.S3 + L2.S2 + L1$

$T41 = 0$

$T42 = 0$

$T43 = 0$

$T44 = 1$

Obtenida la Matriz T realizamos la siguiente operación:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [T].\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Y hacemos u=0, v=0, w=0 (en el origen) tendremos las coordenadas **(x, y, z) en función de las coordenadas articulares q1, q2, q3, q4 y las dimensiones L1, L2, L3, L4 de nuestro robot M5**.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} T11 & T12 & T13 & T14 \\ T21 & T22 & T23 & T24 \\ T31 & T32 & T33 & T34 \\ T41 & T42 & T43 & T44 \end{bmatrix}.\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} T14 \\ T24 \\ T34 \\ T44 \end{bmatrix}$$

Por lo tanto:

$\boldsymbol{X = C1.C2.C3.L4.C4 - C1.S2.S3.L4.C4 - C1.C2.S3.L4.S4 - C1.S2.C3.L4.S4 + C1.C2.L3.S3 - C1.S2.L3.S3 + C1.L2.C2}$

$\boldsymbol{Y = S1.C2.C3.L4.C4 - S1.S2.S3.L4.C4 - S1.C2.S3.L4.S4 - S1.S2.C3.L4.S4 + S1.C2.L3.S3 - S1.S2.L3.S3 + S1.L2.C2}$

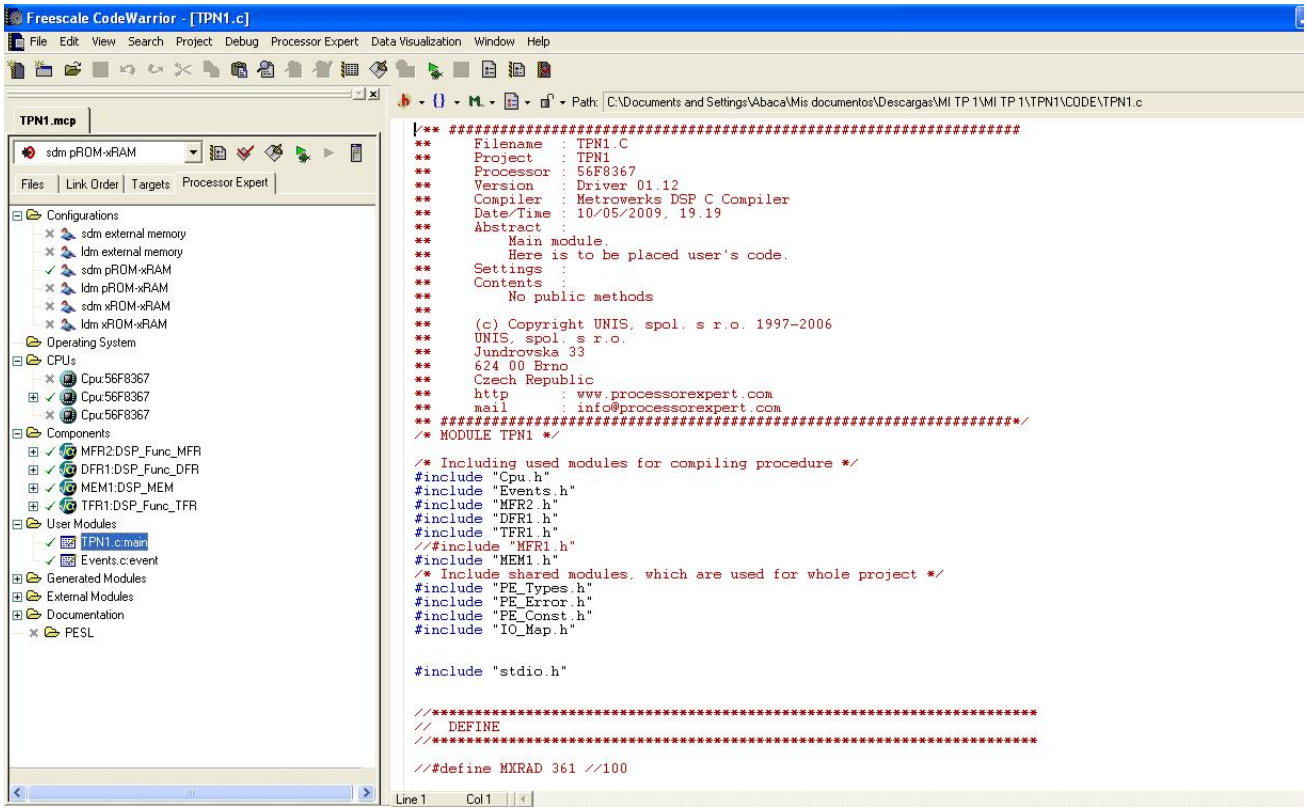
$\boldsymbol{Z = S2.C3.L4.C4 + C2.S3.L4.C4 - S2.S3.L4.S4 + C2.C3.L4.S4 + S2.L3.S3 + C2.L3.S3 + L2.S2 + L1}$

Nota: X = X4; Y = Y4; Z = Z4

Implementación del modelo cinematico directo del Robot M5 a través del CodeWarrior

Dado el modelo cinematico directo obtenido en la hoja anterior a través de operaciones complejas entre matrices este puede ser realizado por medio de un procesador digital de señales (DSP). Para ello se utilizo el procesador 56F8367 de la familia 56800E sobre un entorno de desarrollo llamado Freescale CodeWarrior en este se implementara un código en lenguaje C para que desarrolle las operaciones entre matrices para llegar al modelo cinematico directo del RobotM5 y calcule así las coordenadas (X, Y, Z) del extremo final del mismo.

A continuación se muestra una imagen de cómo se ve el entorno de desarrollo del CodeWarrior



El Codewarrior puede descargarse gratis a través de internet en la página http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-56800E-DSC&fpsp=1&tab=Design_Tools_Tab

La misma es una versión estudiantil que está limitada en código hasta 64kb.

Luego de esta pequeña introducción se mostrara el código en C para nuestro RobotM5 considerando solo la solución para cuatro grados de libertad.

```
/** ###################################################################
**     Filename  : TPN1.C
**     Project   : TPN1
**     Processor : 56F8367
**     Version   : Driver 01.12
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 10/05/2009, 19.19
**     Abstract  :
**         Main module.
**         Here is to be placed user's code.
**     Settings  :
**     Contents  :
**         No public methods
**
**     (c) Copyright UNIS, spol. s r.o. 1997-2006
**     UNIS, spol. s r.o.
**     Jundrovska 33
**     624 00 Brno
**     Czech Republic
**     http      : www.processorexpert.com
**     mail      : info@processorexpert.com
** ###################################################################*/


/* MODULE TPN1 */
/* Including used modules for compiling procedure */
#include "Cpu.h"
```

```
#include "Events.h"
#include "MFR2.h"
#include "DFR1.h"
#include "TFR1.h"
//#include "MFR1.h"
#include "MEM1.h"
/* Include 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
//*********************************************************************
//#define MXRAD 361 //100
//#define PULSE2RAD 32767/MXRAD  // 32767/100 impulsos // #define PULSE2RAD 450
//*********************************************************************
//        GLOBAL VARIABLE
//*********************************************************************
//ac16 pulse2rad=PULSE2RAD ,testResult[MXRAD],testResult2[MXRAD];

//Word16 c16[MXRAD];
//Word32 c32[MXRAD];
Frac16 Homogenea[4][4];
Frac16 C1, S1, C2, S2, C3, S3, C4, S4, L1, L2, L3, L4, q1, q2, q3, q4;
```

```
Frac16 x, y, z, x0, y0, z0;

int j,k;

void main(void)

{

 /* Write your local variable definition here */

 /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/

 PE_low_level_init();

 /*** End of Processor Expert internal initialization.                    ***/

 /* Write your code here */

                    /* Inicializo */

                    q1 = 0;

                    q2 = 0x4000;

                    q3 = 0xC000;

                    q4 = 0xC000;

                    /*Para L1,L2, L3 y L4, normalizo respecto a 13*/

                    L1 = 0x175c;

                    L2 = 0x2876;

                    L3 = 0x2876;

                    L4 = 0xf0a;

                    x0 = 0;

                    y0 = 0;

                    z0 = 0;


printf ("\n\n!!!!!!!!!! VARIANDO q1 SOLAMENTE!!!!!!!!!!!!\n\n");

for(q1=-21915; q1<22511; q1 = q1+500)

                    {
```

```
                    C1 = tfr16CosPIx (q1);

                    C2 = tfr16CosPIx (q2);

                    C3 = tfr16CosPIx (q3);

                    C4 = tfr16CosPIx (q4);

                    S1 = tfr16SinPIx (q1);

                    S2 = tfr16SinPIx (q2);

                    S3 = tfr16SinPIx (q3);

                    S4 = tfr16SinPIx (q4);

//X=

 x        = add(

/*A*/  sub (sub ( mult(mult(mult(C1,C2),C3), mult(L4,C4)),

/*B*/           mult(mult(mult(C1,S2),S3), mult(L4,C4))) ,

/*C*/       add ( mult(mult(mult(C1,C2),S3), mult(L4,S4)),

/*D*/          mult(mult(mult(C1,S2),C3), mult(L4,S4)) ) ) ,

/*E*/  add (sub (mult(mult(C1,C2), mult(L3,S3)) ,

/*F*/         mult(mult(C1,S2), mult(L3,S3)) ),

/*G*/      mult(mult(C1,L2),C2)         )  );

 //Y=

 y = add(

/*A*/  sub (sub ( mult(mult(mult(S1,C2),C3), mult(L4,C4)),

/*B*/           mult(mult(mult(S1,S2),S3), mult(L4,C4))) ,

/*C*/       add ( mult(mult(mult(S1,C2),S3), mult(L4,S4)),

/*D*/          mult(mult(mult(S1,S2),C3), mult(L4,S4)) ) ) ,

/*E*/  add (sub (mult(mult(S1,C2), mult(L3,S3)) ,

/*F*/         mult(mult(S1,S2), mult(L3,S3)) ),

/*G*/      mult(mult(S1,L2),C2)         )  );
```

```
//Z=
 z = add(
/*A*/  add (add ( mult(mult(S2,C3), mult(L4,C4)),
/*B*/          mult(mult(C2,S3), mult(L4,C4))) ,
/*C*/      add ( mult(mult(C3,C2), mult(L4,S4)),
/*D*/           mult(mult(L3,S2),S3)  ) ),
/*E*/  add (add (mult(C2, mult(L3,S3)) ,
/*F*/          mult(L2,S2)) ,
/*G*/      sub (         L1  ,
/*H*/          mult(mult(S2,S3),mult (S4,L4)))));
                        printf ("\n%d\t%d\t%d", x, y, z);
                    }
printf ("\n\n!!!!!!!!!! VARIANDO q2 SOLAMENTE!!!!!!!!!!!!\n\n");
 for(q2=16384; q2<21594; q2 = q2+500)
                    {
                        C1 = tfr16CosPIx (q1);
                        C2 = tfr16CosPIx (q2);
                        C3 = tfr16CosPIx (q3);
                        C4 = tfr16CosPIx (q4);
                        S1 = tfr16SinPIx (q1);
                        S2 = tfr16SinPIx (q2);
                        S3 = tfr16SinPIx (q3);
                        S4 = tfr16SinPIx (q4);
//X=
 x       = add(
/*A*/  sub (sub ( mult(mult(mult(C1,C2),C3), mult(L4,C4)),
```

```
/*B*/          mult(mult(mult(C1,S2),S3), mult(L4,C4))) ,
/*C*/      add ( mult(mult(mult(C1,C2),S3), mult(L4,S4)),
/*D*/          mult(mult(mult(C1,S2),C3), mult(L4,S4)) ) ) ,
/*E*/  add (sub (mult(mult(C1,C2), mult(L3,S3)) ,
/*F*/          mult(mult(C1,S2), mult(L3,S3)) ),
/*G*/      mult(mult(C1,L2),C2)         )  );
//Y=
  y = add(
/*A*/  sub (sub ( mult(mult(mult(S1,C2),C3), mult(L4,C4)),
/*B*/          mult(mult(mult(S1,S2),S3), mult(L4,C4))) ,
/*C*/      add ( mult(mult(mult(S1,C2),S3), mult(L4,S4)),
/*D*/          mult(mult(mult(S1,S2),C3), mult(L4,S4)) ) ) ,
/*E*/  add (sub (mult(mult(S1,C2), mult(L3,S3)) ,
/*F*/          mult(mult(S1,S2), mult(L3,S3)) ),
/*G*/      mult(mult(S1,L2),C2)         )  );
//Z=
  z = add(
/*A*/  add (add ( mult(mult(S2,C3), mult(L4,C4)),
/*B*/          mult(mult(C2,S3), mult(L4,C4))) ,
/*C*/      add ( mult(mult(C3,C2), mult(L4,S4)),
/*D*/           mult(mult(L3,S2),S3)  ) ),
/*E*/  add (add (mult(C2, mult(L3,S3)) ,
/*F*/          mult(L2,S2)) ,
/*G*/      sub (          L1  ,
/*H*/          mult(mult(S2,S3),mult (S4,L4)))));
                  printf ("\n%d\t%d\t%d", x, y, z);
```

```
                    }
printf ("\n\n!!!!!!!!!! VARIANDO q3 SOLAMENTE!!!!!!!!!!!!\n\n");
 for(q3=-16384; q3<4915; q3 = q3+500)
                    {
                              C1 = tfr16CosPIx (q1);
                              C2 = tfr16CosPIx (q2);
                              C3 = tfr16CosPIx (q3);
                              C4 = tfr16CosPIx (q4);
                              S1 = tfr16SinPIx (q1);
                              S2 = tfr16SinPIx (q2);
                              S3 = tfr16SinPIx (q3);
                              S4 = tfr16SinPIx (q4);
//X=
  x         = add(
/*A*/   sub (sub ( mult(mult(mult(C1,C2),C3), mult(L4,C4)),
/*B*/            mult(mult(mult(C1,S2),S3), mult(L4,C4))) ,
/*C*/       add ( mult(mult(mult(C1,C2),S3), mult(L4,S4)),
/*D*/           mult(mult(mult(C1,S2),C3), mult(L4,S4)) ) ) ,
/*E*/  add (sub (mult(mult(C1,C2), mult(L3,S3)) ,
/*F*/           mult(mult(C1,S2), mult(L3,S3)) ),
/*G*/      mult(mult(C1,L2),C2)         )  );



//Y=
  y = add(
/*A*/   sub (sub ( mult(mult(mult(S1,C2),C3), mult(L4,C4)),
```

```
/*B*/           mult(mult(mult(S1,S2),S3), mult(L4,C4))) ,
/*C*/       add ( mult(mult(mult(S1,C2),S3), mult(L4,S4)),
/*D*/           mult(mult(mult(S1,S2),C3), mult(L4,S4)) ) ) ,
/*E*/  add (sub (mult(mult(S1,C2), mult(L3,S3)) ,
/*F*/         mult(mult(S1,S2), mult(L3,S3)) ),
/*G*/      mult(mult(S1,L2),C2)         )  );
//Z=
  z = add(
/*A*/   add (add ( mult(mult(S2,C3), mult(L4,C4)),
/*B*/           mult(mult(C2,S3), mult(L4,C4))) ,
/*C*/       add ( mult(mult(C3,C2), mult(L4,S4)),
/*D*/            mult(mult(L3,S2),S3)  ) ),
/*E*/   add (add (mult(C2, mult(L3,S3)) ,
/*F*/           mult(L2,S2)) ,
/*G*/       sub (          L1  ,
/*H*/           mult(mult(S2,S3),mult (S4,L4)))));
                    printf ("\n%d\t%d\t%d", x, y, z);
                    }
printf ("\n\n!!!!!!!!!! VARIANDO q4 SOLAMENTE!!!!!!!!!!!!\n\n");
 for(q4=-16384; q4<18877; q4 = q4+500)
                    {
                              C1 = tfr16CosPIx (q1);
```

```
                    C2 = tfr16CosPIx (q2);

                    C3 = tfr16CosPIx (q3);

                    C4 = tfr16CosPIx (q4);

                    S1 = tfr16SinPIx (q1);

                    S2 = tfr16SinPIx (q2);

                    S3 = tfr16SinPIx (q3);

                    S4 = tfr16SinPIx (q4);

//X=

 x        = add(

/*A*/  sub (sub ( mult(mult(mult(C1,C2),C3), mult(L4,C4)),

/*B*/          mult(mult(mult(C1,S2),S3), mult(L4,C4))) ,

/*C*/      add ( mult(mult(mult(C1,C2),S3), mult(L4,S4)),

/*D*/          mult(mult(mult(C1,S2),C3), mult(L4,S4)) ) ) ,

/*E*/  add (sub (mult(mult(C1,C2), mult(L3,S3)) ,

/*F*/         mult(mult(C1,S2), mult(L3,S3)) ),

/*G*/      mult(mult(C1,L2),C2)        )  );

 //Y=

 y = add(

/*A*/  sub (sub ( mult(mult(mult(S1,C2),C3), mult(L4,C4)),

/*B*/          mult(mult(mult(S1,S2),S3), mult(L4,C4))) ,

/*C*/      add ( mult(mult(mult(S1,C2),S3), mult(L4,S4)),

/*D*/          mult(mult(mult(S1,S2),C3), mult(L4,S4)) ) ) ,

/*E*/  add (sub (mult(mult(S1,C2), mult(L3,S3)) ,

/*F*/         mult(mult(S1,S2), mult(L3,S3)) ),

/*G*/      mult(mult(S1,L2),C2)        )  );

//Z=
```

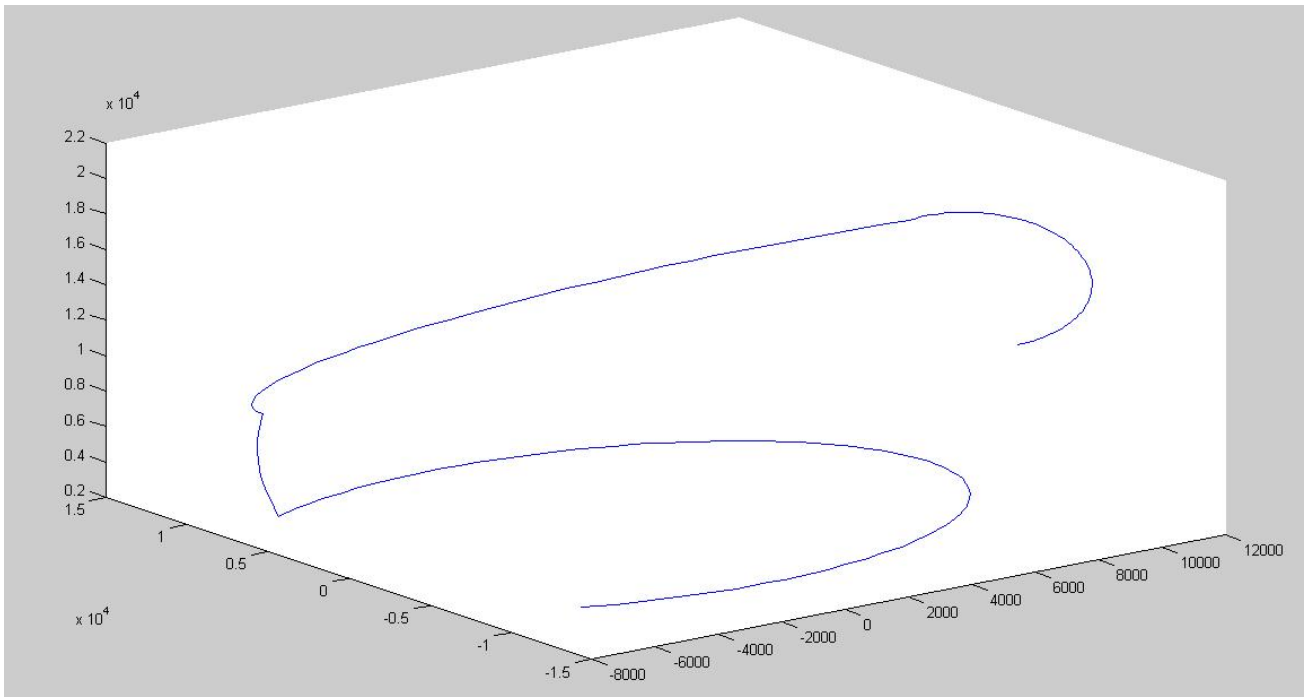
```
 z = add(
/*A*/  add (add ( mult(mult(S2,C3), mult(L4,C4)),
/*B*/          mult(mult(C2,S3), mult(L4,C4))) ,
/*C*/      add ( mult(mult(C3,C2), mult(L4,S4)),
/*D*/           mult(mult(L3,S2),S3)  ) ),
/*E*/  add (add (mult(C2, mult(L3,S3)) ,
/*F*/          mult(L2,S2)) ,
/*G*/      sub (         L1  ,
/*H*/          mult(mult(S2,S3),mult (S4,L4)))));
                  printf ("\n%d\t%d\t%d", x, y, z);
                  }
}
/* END TPN1 */
```

En el código se puede observar una instrucción **printf** la cual muestra valores diferentes de la posición final del extremo del robotM5 (X, Y, Z) para distintos valores de las coordenadas articulares (q1, q2, q3, q4). Esto se realiza gracias a que el entorno de desarrollo CodeWarrior es un simulador y a través de la acción de **DEBUG** nos va mostrando los resultados de las coordenadas finales (X, Y, Z) producto de las operaciones matemáticas que realiza nuestro **DSP**.

Graficas de trayectorias del robot M5 en Matlab obtenidas por los resultados del DSP

A través de los resultados de las coordenadas (X, Y, Z) en función de distintos valores de las articulaciones (q1, q2, q3, q4) por medio del DSP. Estos pueden ser ingresados en una grafica tridimensional para visualizar la trayectoria que realiza el extremo final de nuestro robot(X, Y, Z).

Utilizando el programa Matlab a través de la función **plot3(x, y, z)** se puede determinar la trayectoria para distintos valores de (X, Y, Z). A continuación se muestra una trayectoria como ejemplo.



Simulación del robot M5 en Matlab a través del Robotics Toolbox

Para llevar a cabo la siguiente simulación es necesario disponer de la herramienta llamada **Robotics Toolbox for Matlab (release 8)** que se debe agregar a nuestro programa Matlab; este paquete puede descargarse de la página **www.petercorke.com**.

A continuación se detalla el script para lograr la simulación de nuestro robot M5 en base a nuestra tabla de los parámetros D.H que se vuelve a mostrar a continuación:

	θ	d	a	α
Articulación 1	q1	L1	0	90 grados
Articulación 2	q2	0	L2	0
Articulación 3	q3	0	L3	0
Articulación 4	q4	0	L4	0

En base a esta tabla se realiza el siguiente script:

```
clear%borra todo el workspace

clc%limpia pantalla

l1=link([90 0 0 7.3])%articulacion 1

l2=link([0 11.95 0 0])%articulacion 2

l3=link([0 11.95 0 0])%articulacion 3

l4=link([0 4.7 0 0])%articulacion 4

robo1=robot({l1 l2 l3 l4})

syms q1 q2 q3 q4

%tr=fkine(robo1,[q1 q2 q3 q4])

%tr=simple(tr)

plot(robo1,[0 0 0 0])

drivebot(robo1)
```

Nota: L1 = 7.3cm; L2 = 11.95cm; L3 = 11.95cm; L4 = 4.7cm. Son las dimensiones de nuestro robot M5.

Finalmente se obtiene la siguiente simulación:

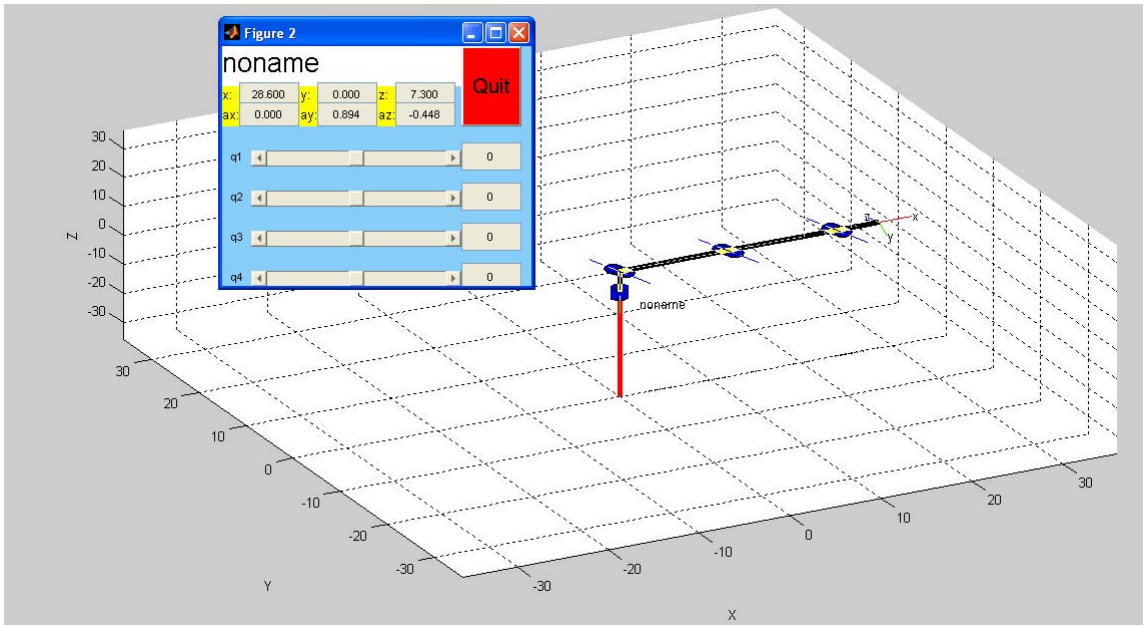


Figura_2: Robot M5 en posición inicial

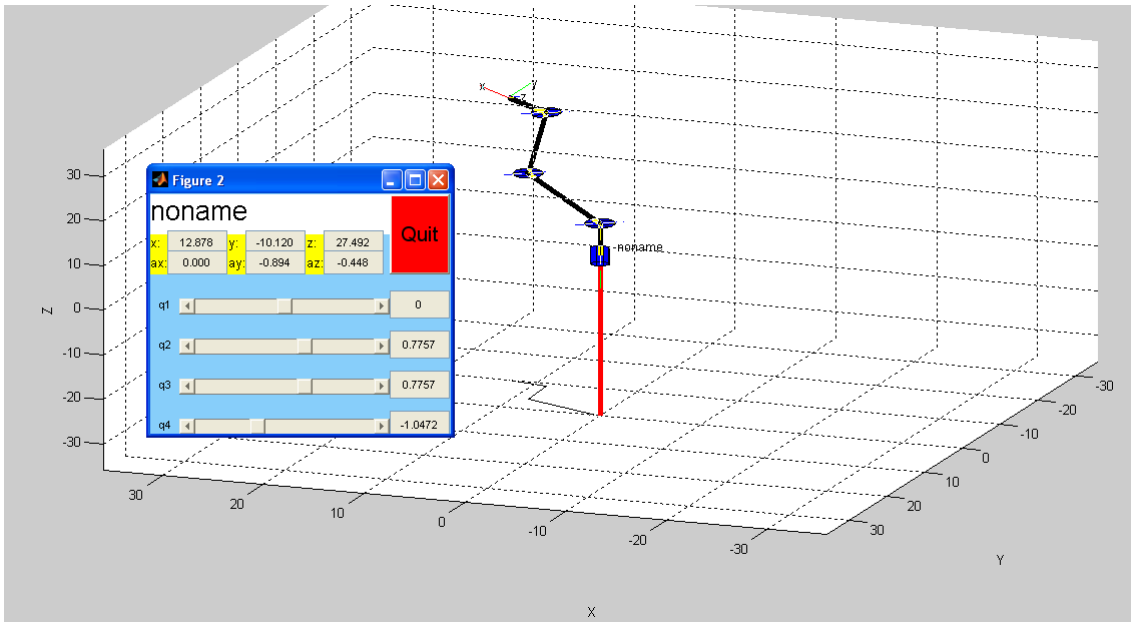


Figura_3: Robot M5 con variación en las articulaciones q2, q3, q4

Conclusiones:

La realización de este trabajo práctico nos permitió verificar la simulación de la cinemática directa de un robot.

Teniendo en cuenta que partimos de un robot en el que solo consideramos cuatro grados de libertad calculamos el modelo cinematico directo a través de método de D.H y por medio de una herramienta muy útil para Matlab(Tollbox de petercorke) de simulación pudimos verificar dicho modelo que se correspondía a nuestro robot.

Para el desarrollo de esta práctica utilizamos también herramientas tales como CodeWarrior para generar el código fuente y nuevamente Matlab para realizar los gráficos. Para dichos softwares utilizamos herramientas que fueron de gran utilidad, y en algunos casos eran desarrolladas por primera vez, según nuestra experiencia.

Cuando compilamos el código por primera vez y realizamos la gráfica correspondiente, verificamos que habíamos calculado en forma errónea los valores límite, ya que el dibujo mostraba trayectorias no deseadas y que no correspondían con nuestras directivas. Es decir, físicamente estábamos destruyendo el robot, ya que lo haríamos ir a puntos del espacio a los cuales le es imposible acceder.

Una vez solucionado esto, delimitamos en forma correcta los movimientos de cada eje y obtuvimos un resultado satisfactorio.

Bibliografía:

- Fundamentos de robótica, Barrientos, Peñin, Balaguer y Aracil, Mc Graw Hill 2001.
- Apuntes de robótica cortesía de Samuel Oporto Diaz.