UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Simulación de una trayectoria tridimensional mediante la plataforma ADEFID".

TESIS

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero en Mecatrónica

PRESENTA:

Nathan Gasca Martínez

ASESOR:

Dr. Max A. González Palacios

SALAMANCA, GTO. Noviembre, 2018

Agradecimientos

Muchas han sido las dificultades presentadas en este transcurso de tiempo, pero gracias Dios y a muchas personas, he podido lograr este objetivo que a veces parecía imposible.

Primeramente quiero agradecer a mi mamá Ovelina Rosa Martínez García, por su ejemplo y que a pesar de sus problemas de salud siempre ha estado animándome. También agradezco a mi papá Enedino Gasca Cabrera por siempre creer en mí.

Un agradecimiento especial al Dr. Max González Palacios por su comprensión y asesoría, quien con amabilidad siempre estuvo dispuesto a resolver cada una de las dudas presentadas a lo largo de este proyecto.

Al M. en I. Ricardo Martínez Martínez, por su amistad y asesoría a lo largo de mi carrera. También a todos mis amigos y compañeros que por ser muchos no puedo mencionar por nombre, pero han sido parte de este proceso.

A mis sinodales los Doctores Gustavo Cerda Villafaña y J. Jesús Cervantes Sánchez, por su disposición y ayuda en la revisión de esta tesis.

Índice general

Índice de figuras

Índice de cuadros

Índice de códigos

Capítulo 1

Introducción

La constante automatización en la industria, en tareas tales como corte, soldadura, montaje y aplicación de materiales, han hecho de vital importancia la búsqueda de soluciones en la róbotica. La selección de una arquitectura, tomando en cuenta una serie de factores dependiendo de la tarea a realizar, requiere el analisís cinemático de manipuladores, para poder realizar una comparativa entre las ventajas y desventajas de seleccionar determinada arquitectura.

1.1. Justificación

En la robótica de manipuladores es importante tomar en cuenta la posibilidad de colisión entre el robot y objetos que se encuentran en el espacio de trabajo. Para esto se asume que las configuraciones inicial y final del robot son especificadas y el problema es encontrar una trayectoria libre de colisiones que conecte dichas configuraciones. El problema anteriormente descrito se conoce como planeación de trayectorias. La planeación de trayectorias nos provee una descripción del movimiento del robot, pero no especifica ninguno de los aspectos dinámicos del movimiento [1].

Es de interés conocer los aspectos dinámicos de las trayectorias, en este proyecto no se pretende realizar planeación de trayectorias, puesto que el proceso que aquí se hará, es adquirir la trayectoria para después simularla, pero también se analizarán los aspectos dinámicos para realizar la simulación.

1.2. Objetivo

Haciendo uso de un manipulador de tres grados de libertad, se obtendrán los datos necesarios para realizar la simulación de una trayectoria tridimensional, realizando la simulación mediante la plataforma de desarrollo ADEFID. Dentro de la simulación se probarán arquitecturas distintas a la del manipulador previamente utilizado para la adquisición de la trayectoria, con lo que se podrá

hacer una comparativa del comportamiento de las distintas arquitecturas. También se realizará el cálculo de los parámetros de velocidad y aceleración de la trayectoria adquirida.

1.3. Definiciones básicas

A continuación se dan algunos conceptos básicos utilizados en el estudio de manipuladores robóticos.

Robot: Manipulador multifuncional reprogramable diseñado para mover material, partes, herramientas o dispositivos especializados a través de movimientos variables programados. Para este proyecto, la palabra clave es la reprogramabilidad. Debido a que el cerebro de la computadora es el que le da la utilidad y adaptabilidad al robot.

Los robots aquí definidos surgen de dos tecnologías previas: los teleoperadores y las máquinas herramientas de control numérico. Los teleoperadores fueron desarrollados durante la segunda guerra mundial para manejar materiales radioactivos. El control numérico por computadora fue desarrollado por la alta precisión requerida en el maquinado de ciertas piezas. Los primeros robots esencialmente combinaban los elementos mecánicos de los teleoperadores con la autonomia y programabilidad de las máquinas de CNC.

1.4. Estructura de los robots

Los manipuladores robóticos seriales se componen de eslabones conectados por articulaciones en una cadena cinemática abierta. Cada articulación representa la interconexión entre dos eslabones. Las articulaciones son típicamente rotatorias (revolutas) o lineales (prismáticos). Las variables de articulación representan el desplazamiento relativo entre eslabones adyacentes, y son representados por θ para revolutas y d para prismáticos.

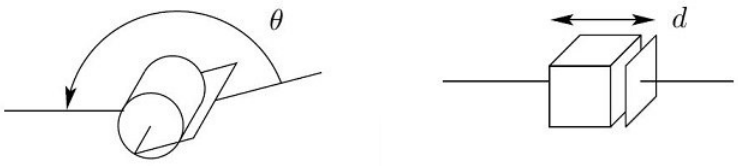

Figura 1.1: Representación simbólica de articulaciones[1].

Un concepto importante en manipuladores robóticos es el de grados de libertad. Se dice que un objeto tiene n grados de libertad (DOF) si su configuración puede ser especificada mediante n parámetros. En el caso de manipuladores robóticos el número de articulaciones determina los grados de libertad.

El espacio de trabajo de un manipulador es el total del volumen barrido por el efector final. El espacio de trabajo se divide en espacio de trabajo alcanzable

y espacio de trabajo diestro. El espacio de trabajo alcanzable es el conjunto de puntos que el manipulador puede cubrir. Mientras que el espacio de trabajo diestro se refiere a aquellos puntos que el manipulador puede cubrir con una orientación arbitraria del órgano terminal.

1.5. Clasificación de manipuladores

Los manipuladores robóticos pueden ser clasificados mediante varios criterios, tales como su fuente de energía, su geometría, su areá de aplicación o su método de control. Aunque hay muchas maneras posibles de usar articulaciones prismáticas y revolutas, sólo unos cuantos arreglos son comúnmente usados. A continuación se describen brevemente algunos de los arreglos más típicos.

1.5.1. Cartesiano(PPP)

Este manipulador se caracteriza por que sus tres primeras articulaciones son primáticas. Como es de esperarse la descripción cinemática de este manipulador es la mas simple, puesto que las variables de articulación corresponden a las coordenadas del efector final. Son útiles en tareas de transferencia de materiales y ensamble.

Figura 1.2: Manipulador cartesiano[2].

1.5.2. Cilíndrico(RPP)

Recibe su nombre debido a que las variables de articulación son las coordenadas cilíndricas del efector final con respecto a la base. La primer articulación es una revoluta que produce alrededor de la base, mientras que la segunda y tercera articulación son prismáticas.

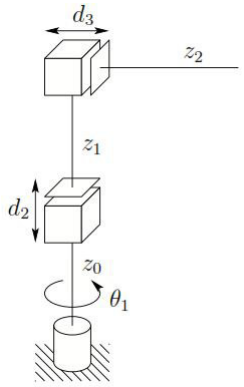

Figura 1.3: Manipulador cilíndrico [1].

1.5.3. Esférico(RRP)

De igual modo que el manipulador esférico, este manipulador recibe su nombre gracias a que las variables de articulación definen en coordenadas esféricas la posición del efector final respecto a la intersección de los ejes de las articulaciones. En este manipulador sus primeras dos articulaciones son revolutas mientras que la tercera es prismática.

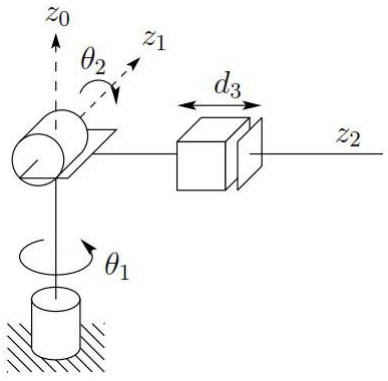

Figura 1.4: Manipulador esférico [1].

1.5.4. SCARA(RRP)

Acrónimo que corresponde a sus siglas en inglés **S**elective **C**ompliant **A**rticulated **R**obot for **A**ssembly, como su nombre lo sugiere, este manipulador fue diseñado para tareas de ensamble. A diferencia del manipulador esférico, los ejes de las articulaciones del manipulador SCARA son paralelos.

Figura 1.5: Manipulador SCARA [4].

1.5.5. Articulado(RRR)

También llamado manipulador antropomórfico. Sus tres articulaciones son revolutas. Los ejes de la segunda y tercer articulación son paralelos, y estos a su vez son perpendiculares al eje de la primera articulación. Este manipulador es ampliamente usado en la industria pues proveé una amplia libertad de movimiento en un espacio compacto.

Figura 1.6: Manipulador articulado[3]

1.6. ADEFID

El proyecto se desarolló mediante la plataforma de desarrollo ADEFID. ADvanced Engineering platForm for Industrial Development (Plataforma avanzada de ingeniería para desarrollo industrial), creada por el Dr. Max A. González Palacios [5]. Es una plataforma de desarrollo que involucra bibliotecas que permiten la creación de paquetes de software dedicados no solo a aplicaciones industriales, sino también a aquellas aplicaciones donde el diseño, modelado y/o simulación en línea son requeridas.

El principal objetivo de ADEFID es proveer una plataforma flexible mediante la que el desarrollador sea capaz de construir aplicaciones que satisfagan sus necesidades. Además, cualquier aplicación de ADEFID puede considerarse como una plataforma para generar otras aplicaciones con un nivel más alto de especialización.

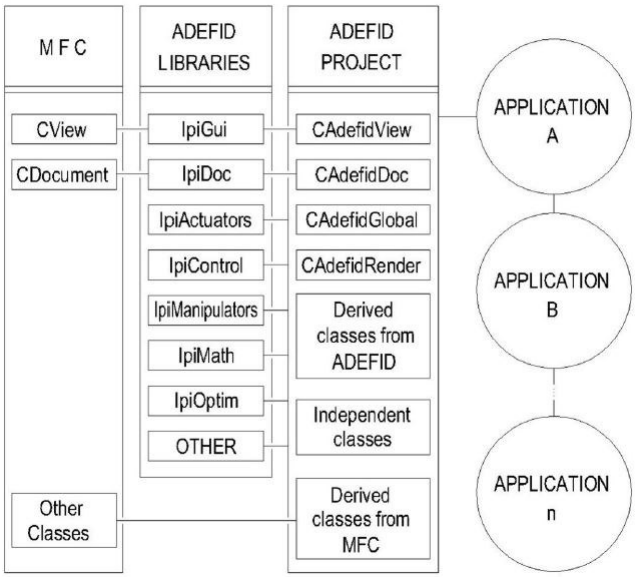

Figura 1.7: Estructura de una aplicación de ADEFID [5].

En la figura 1.7, se muestra la estructura de una aplicación derivada de un proyecto de ADEFID. El desarrollador es capaz de crear aplicaciones personalizadas empezando desde cero, o a partir de una aplicación de ADEFID ya desarrollada, todo esto para producir, como ya fue mencionado, un nivel más alto de especialización.

ADEFID cuenta con una colección de bibliotecas, las cuales han sido creadas con base en las necesidades que han surgido a través del desarrollo de paquetes de software previos, tales como, USyCaMs, SixPaq, SVIS, etc. Cada biblioteca es actualizada continuamente y nuevas clases son implementadas conforme a las

necesidades surgidas durante la implementación de nuevas aplicaciones.

Un proyecto de ADEFID se deriva de una plantilla de una aplicación MFC y esta tiene una arquitectura de Vista/Documento. El proceso principal de un proyecto de ADEFID reside en "CAdefidDoc", y el proceso es sintetizado como es indicado en el diagrama de flujo en la figura 1.8.

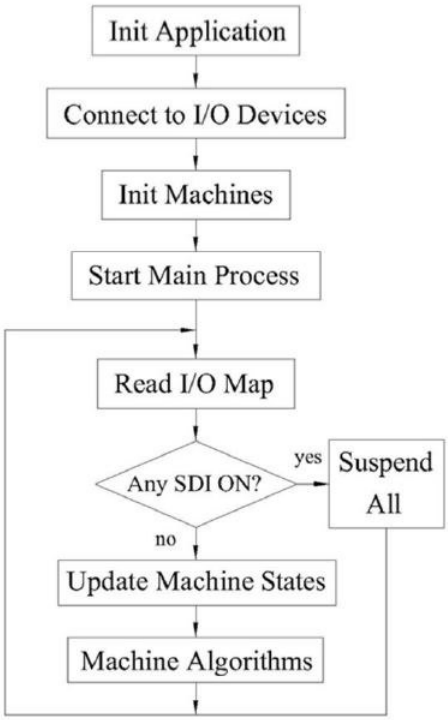

Figura 1.8: Diagrama de flujo del proceso principal de ADEFID [5].

Cuando la aplicación es iniciada, se conectan los dispositivos de entrada y salida (I/O Devices). Una vez que la conexión es establecida se procede a la inicialización de las máquinas. El inicio del proceso principal (Start Main Process) es en sí un ciclo infinito, que termina solo si el usuario cierra la aplicación, de modo que se permite la aplicación corra de manera independiente, mayormente cuando se trata de una aplicación industrial. En cada ciclo el mapa de entradas/salidas (I/O map) es leído y, si cualquiera de las entradas de seguridad (SDI) están activas, el sistema entra en un estado de suspensión y no se realiza ninguna otra acción. De modo contrario, si las SDI's permanecen desactivadas, el estado de cada máquina es evaluado y actualizado de acuerdo al estado de las I/O's. Después se tiene acceso a cada máquina a tráves de su miembro de función .ªlgorithm()".

A continuación se describen brevemente algunas aplicaciones desarrolladas mediante ADEFID, que tienen el propósito de investigación y enseñanza.

Mechanism-O: Este paquete fue creado para facilitar la enseñanza en cursos de mecanismos para los estudiantes de ingeniería, así como con el propósito de generar mecanismos inovadores.

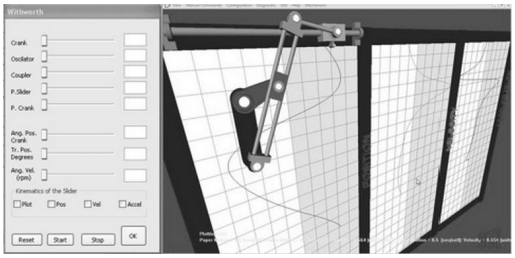

Figura 1.9: Mechanism-O: Simulación de un mecanismo Withworth [5].

Vibrato: Vibrato esta dedicado al estudio de vibraciones en sistemas mecánicos. Fue creado como una herramienta auxiliar en los cursos de teoría de vibración.

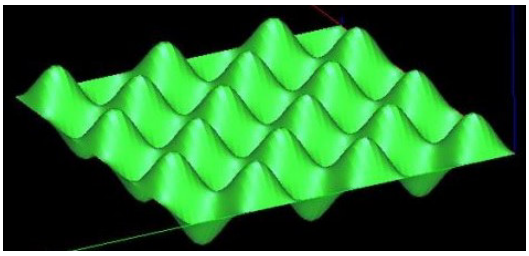

Figura 1.10: Vibrato: Simulación de una membrana rectangular [6].

OptimPlot2D: Con este software, el usuario ingresa cualquier función del tipo $y = f(x)$, escribiéndola como una cadena de caracteres en una línea de comando.

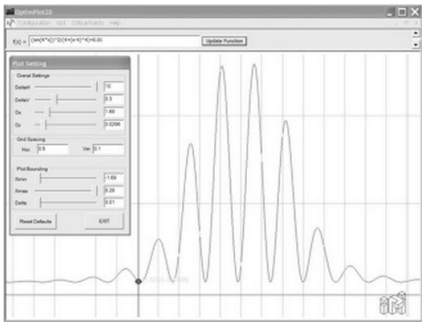

Figura 1.11: OptimPlot2D [5].

OptimPlot3D: Este paquete es similar a OptimPlot2D, pero en este caso la función es del tipo $z = f(x, y)$.

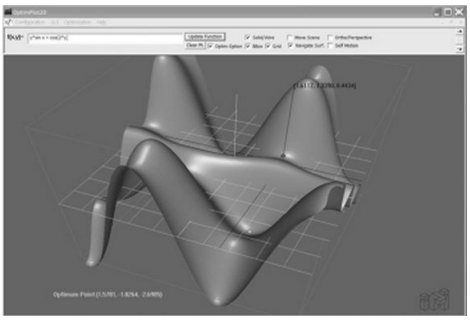

Figura 1.12: OptimPlot3D [5].

ADRS: Acrónimo para Architecture Design and Robot Simulation. Este paquete fue creado como una herramienta auxiliar en el diseño de manipuladores seriales.

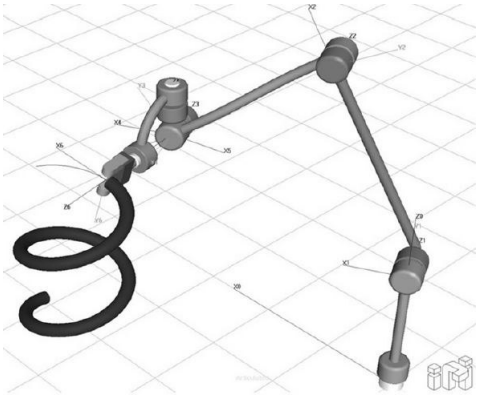

Figura 1.13: ADRS: Seguimiento de trayectoria [5].

Capítulo 2

Análisis cinemático

Es de interés representar la posición y la orientación de manipuladores robóticos. Para resolver este problema es necesario el establecimiento de varios sistemas coordenados y las transformaciones entre estos sistemas. Para ello cada una de las articulaciones de un manipulador robótico tendrá ligado un sistema coordenado. A continuación se revisan todas las herramientas necesarias para el análisis del movimiento de los manipuladores estudiados en el proyecto.

2.1. Matriz de rotación.

Considere dos sistemas coordenados mostrados en la figura 2.1, $o_0x_0y_0$ y $o_1x_1y_1$, el sistema $o_1x_1y_1$ fue obtenido mediante la rotación de un ángulo θ del sistema $o_0x_0y_0$. Se busca describir la orientación de un sistema $o_1x_1y_1$ con respecto de $o_0x_0y_0$.

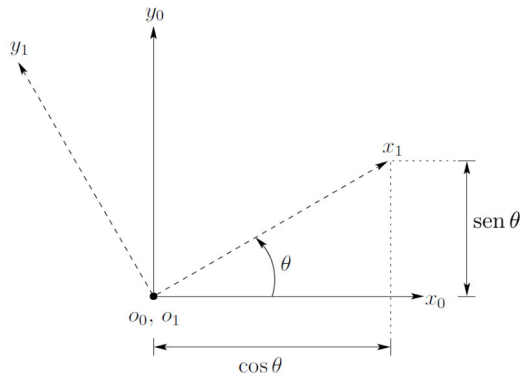

Figura 2.1: Sistemas coordenados $o_0x_0y_0$ y $o_1x_1y_1$ [1].

Para ello se puede especificar mediante vectores los ejes del sistema $o_1x_1y_1$ con respecto al sistema coordenado $o_0x_0y_0$.

$$Q_0^1 = \left[x_0^1 \mid y_0^1 \right]$$

donde x_0^1 y y_0^1 son las coordenadas en el sistema S_0 de los vectores unitarios x_1 y y_1. Estás coordenadas son las proyecciones de los ejes del sistema S_1 en el sistema S_0, de modo que:

$$x_0^1 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \qquad y_0^1 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

de donde se obtiene:

$$Q_0^1 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.1}$$

Una matriz de la forma de la ecuación (2.1) es conocida como matriz de rotación. Otra forma de obtener dicha matriz es mediante el concepto de producto punto de dos vectores unitarios con el que obtenemos la proyección de un vector unitario en el otro. De este modo:

$$x_0^1 = \begin{bmatrix} \mathbf{i}_1 \cdot \mathbf{i}_0 \\ \mathbf{i}_1 \cdot \mathbf{j}_0 \end{bmatrix} \qquad y_0^1 = \begin{bmatrix} \mathbf{j}_1 \cdot \mathbf{i}_0 \\ \mathbf{j}_1 \cdot \mathbf{j}_0 \end{bmatrix}$$

Los cuales pueden ser combinados para obtener la matriz de rotación:

$$Q_0^1 = \begin{bmatrix} \mathbf{i}_1 \cdot \mathbf{i}_0 & \mathbf{j}_1 \cdot \mathbf{i}_0 \\ \mathbf{i}_1 \cdot \mathbf{j}_0 & \mathbf{j}_1 \cdot \mathbf{j}_0 \end{bmatrix} \tag{2.2}$$

La técnica de proyección descrita con anterioridad puede ser escalada fácilmente al caso tridimensional. En tres dimensiones. cada eje del sistema $o_1x_1y_1z_1$ es proyectado en el sistema coordenado $o_0x_0y_0z_0$. La matriz de rotación resultante es:

$$Q_0^1 = \begin{bmatrix} \mathbf{i}_1 \cdot \mathbf{i}_0 & \mathbf{j}_1 \cdot \mathbf{i}_0 & \mathbf{k}_1 \cdot \mathbf{i}_0 \\ \mathbf{i}_1 \cdot \mathbf{j}_0 & \mathbf{j}_1 \cdot \mathbf{j}_0 & \mathbf{k}_1 \cdot \mathbf{j}_0 \\ \mathbf{i}_1 \cdot \mathbf{k}_0 & \mathbf{j}_1 \cdot \mathbf{k}_0 & \mathbf{k}_1 \cdot \mathbf{k}_0 \end{bmatrix} \tag{2.3}$$

Las matrices de esta forma son ortogonales, con determinante igual a 1. Los productos internos de la matriz son conmutativos, es decir $\mathbf{i}_0 \cdot \mathbf{j}_1 = \mathbf{j}_1 \cdot \mathbf{i}_0$, por lo tanto:

$$Q_0^1 = \left(Q_1^0\right)^T$$

En un sentido geométrico, la orientación de $o_0x_0y_0z_0$ con respecto al sistema $o_1x_1y_1z_1$ es la inversa de la orientación de $o_1x_1y_1z_1$ con respecto al sistema $o_0x_0y_0z_0$.

$$\left(Q_1^0\right)^T = \left(Q_1^0\right)^{-1}$$

Matrices de rotación básicas

Se habla de matrices de rotación básicas cuando la rotación entre el sistema coordenado $o_0x_0y_0z_0$ y el sistema $o_1x_1y_1z_1$ esta dada por un ángulo θ, alrededor de uno de los ejes del sistema $o_0x_0y_0z_0$. Las ecuaciones (2.4), (2.5) y (2.6) muestran las matrices básicas de rotación. Se usa la notación $Q_x(\theta)$, $Q_y(\theta)$ y $Q_z(\theta)$, pues resulta más descriptiva.

$$Q_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.4}$$

$$Q_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.5}$$

$$Q_x(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \cos\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Composición de rotaciones

Supóngase que se tienen S_0, S_1 y S_2 coincidentes. Primero se rotan S_1 junto con S_2 con relación a S_0 de acuerdo a Q_0^1. Entonces, con S_1 y S_2 coincidentes, se rota S_2 respecto a S_1 de acuerdo Q_1^2. El sistema coordenado sobre el cual se efectua la transformación se le denomina **sistema actual**.
Dado un sistema fijo S_0 y un sistema actual S_1 junto a la matriz de rotación Q_0^1 que los relaciona, si un tercer sistema S_2 se obtiene por la rotación Q_1^2 realizada respecto al sistema actual entonces:

$$Q_0^2 = Q_0^1 Q_1^2$$

Si el tercer sistema S_2 se obtiene por la rotación Q_1^2 realizada con respecto al **sistema fijo**, entonces tenemos:

$$Q_0^2 = Q_1^2 Q_0^1$$

2.2. Transformaciones rotacionales

En la figura 2.2 se muestra un objeto rígido S al cual esta unido un sistema coordenado $o_1x_1y_1z_1$. Se dan las coordenadas del punto p con respecto al sistema coordenado $o_1x_1y_1z_1$, es decir p_1. Las coordenadas $p_1 = (u, v, w)^T$ satisfacen la ecuación

$$p = u\mathbf{i}_1 + v\mathbf{j}_1 + w\mathbf{k}_1$$

Se puede proyectar el punto p en los ejes coordenados del sistema $o_0x_0y_0z_0$ y así obtener una expresión para p_0.

$$p_0 = \begin{bmatrix} p \cdot \mathbf{i}_0 \\ p \cdot \mathbf{j}_0 \\ p \cdot \mathbf{k}_0 \end{bmatrix}$$

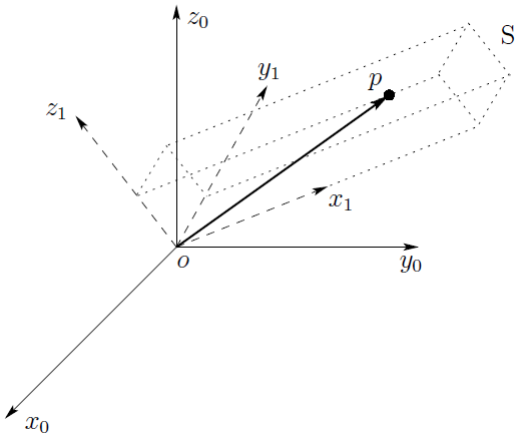

Figura 2.2: Sistema coordenado unido a un cuerpo rígido[1].

Si se combinan estas ecuaciones se obtiene:

$$\begin{aligned} p_0 &= \begin{bmatrix} (u\mathbf{i}_1 + v\mathbf{j}_1 + w\mathbf{k}_1) \cdot \mathbf{i}_0 \\ (u\mathbf{i}_1 + v\mathbf{j}_1 + w\mathbf{k}_1) \cdot \mathbf{j}_0 \\ (u\mathbf{i}_1 + v\mathbf{j}_1 + w\mathbf{k}_1) \cdot \mathbf{k}_0 \end{bmatrix} \\ &= \begin{bmatrix} u\mathbf{i}_1 \cdot \mathbf{i}_0 + v\mathbf{j}_1 \cdot \mathbf{i}_0 + w\mathbf{k}_1 \cdot \mathbf{i}_0 \\ u\mathbf{i}_1 \cdot \mathbf{j}_0 + v\mathbf{j}_1 \cdot \mathbf{j}_0 + w\mathbf{k}_1 \cdot \mathbf{j}_0 \\ u\mathbf{i}_1 \cdot \mathbf{k}_0 + v\mathbf{j}_1 \cdot \mathbf{k}_0 + w\mathbf{k}_1 \cdot \mathbf{k}_0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{i}_1 \cdot \mathbf{i}_0 & \mathbf{j}_1 \cdot \mathbf{i}_0 & \mathbf{k}_1 \cdot \mathbf{i}_0 \\ \mathbf{i}_1 \cdot \mathbf{j}_0 & \mathbf{j}_1 \cdot \mathbf{j}_0 & \mathbf{k}_1 \cdot \mathbf{j}_0 \\ \mathbf{i}_1 \cdot \mathbf{k}_0 & \mathbf{j}_1 \cdot \mathbf{k}_0 & \mathbf{k}_1 \cdot \mathbf{k}_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \end{aligned} \tag{2.7}$$

La matriz presentada en la ecuación (2.7), es la matriz de rotación Q_0^1 mostrada en la ecuación (2.3), podemos reescribir la ecuación (2.7) como:

$$\mathbf{p}_0 = Q_0^1 \mathbf{p}_1 \tag{2.8}$$

Por lo tanto, la matriz de rotación Q_0^1 puede ser usada no sólo para representar la orientación del sistema coordenado $o_1x_1y_1z_1$ con respecto al sistema $o_0x_0y_0z_0$. Si un punto es expresado relativo a $o_1x_1y_1z_1$ por las coordenadas $\mathbf{p}_1$, $Q_0^1\mathbf{p}_1$ representa al mismo punto relativo a $o_0x_0y_0z_0$.

2.3. Movimientos Rígidos.

Un movimiento rígido es una rotación pura combinada con una traslación pura. Por ejemplo, si el marco $o_1x_1y_1z_1$ es obtenido mediante el marco $o_0x_0y_0z_0$, aplicando primeramente una rotación especificada por Q_0^1 seguida de una traslación con respecto a $o_0x_0y_0z_0$ dada por $\mathbf{d}_0^1$, entonces las coordenadas de $\mathbf{p}_0$ están dadas por:

$$\mathbf{p}_0 = Q_0^1\mathbf{p}_1 + \mathbf{d}_0^1 \tag{2.9}$$

Ahora, si se tienen dos movimientos rígidos, el primero igual a la ecuación (2.9) y un segundo igual a:

$$\mathbf{p}_1 = Q_1^2\mathbf{p}_2 + \mathbf{d}_1^2 \tag{2.10}$$

Lo que definirá un tercer movimiento rígido, el cual se obtiene substituyendo la ecuación ($\mathbf{p}_1$) de la ecuación (2.10) en la ecuación (2.9).

$$\mathbf{p}_0 = Q_0^1Q_1^2\mathbf{p}_2 + Q_0^1\mathbf{d}_1^2 + \mathbf{d}_0^1 \tag{2.11}$$

Puesto que la relación entre $\mathbf{p}_0$ y $\mathbf{p}_2$ es un movimiento rígido. El movimiento puede ser descrito por:

$$\mathbf{p}_0 = Q_0^2\mathbf{p}_2 + \mathbf{d}_0^2 \tag{2.12}$$

Comparando las ecuaciones (2.11) y (2.12) obtenemos las siguientes relaciones.

$$Q_0^2 = Q_0^1Q_1^2 \tag{2.13}$$

$$d_0^2 = Q_0^1\mathbf{d}_1^2 + \mathbf{d}_0^1 \tag{2.14}$$

De la ecuación (2.13) podemos decir que se trata de una transformación con respecto al sistema actual. Y la ecuación (2.14) muestra que el vector del origen o_0 al origen o_2 es igual a la suma de el vector que va de o_0 a o_1 ($\mathbf{d}_0^1$), más el vector que va de o_1 a o_2 ($Q_0^1\mathbf{d}_1^2$), todo referido al sistema coordenado $o_0x_0y_0z_0$.

2.4. Transformaciones homogéneas

Es notorio que la ecuación (2.11) es difícil de manejar cuando la secuencia de movimientos rígidos se hace larga. Las transformaciones homogéneas combinan las operaciones de rotación y traslación en una sola matriz de multiplicación, éstas nos permiten representar de manera simultánea posición y orientación de un sistema coordenado con respecto de otro de una manera más sencilla.
Retomando las ecuaciones (2.13) y (2.14), se pueden reacomodar de manera matricial:

$$\begin{bmatrix} Q_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} Q_1^2 & \mathbf{d}_1^2 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} Q_0^1Q_1^2 & Q_0^1\mathbf{d}_1^2 + \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.15}$$

Donde $\mathbf{0}$ representa al vector $(0,0,0)$. Por lo tanto, los movimientos rígidos pueden ser representados por el conjunto de matrices de la forma:

$$T = \begin{bmatrix} Q & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.16}$$

La matrices de transformación de la forma de la ecuación (2.16) son conocidas como transformaciones homógeneas. La composición y el orden que se utiliza en las transformaciones de matrices 3×3 se aplica de igual manera en las transformaciones homogéneas de matrices de 4×4. Puesto que Q es ortogonal, y cumple con la propiedad de que $Q^{-1} = Q^T$, la inversa de la matriz de transformación es:

$$T^{-1} = \begin{bmatrix} Q^T & -Q^T\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.17}$$

Para poder expresar las transformaciones de los vectores de posición como una multiplicación matricial.

$$\mathbf{P}_0 = \begin{bmatrix} \mathbf{p}_0 \\ 1 \end{bmatrix} \tag{2.18}$$

$$\mathbf{P}_1 = \begin{bmatrix} \mathbf{p}_1 \\ 1 \end{bmatrix} \tag{2.19}$$

$$\tag{2.20}$$

A $\mathbf{P}_0$ y $\mathbf{P}_1$ se les conoce como representaciones homogéneas de $\mathbf{p}_0$ y $\mathbf{p}_1$. Ahora podemos escribir una ecuación equivalente a la ecuación (2.9) haciendo uso de la matriz de transformación homogénea.

$$\mathbf{p}_0 = T_0^1 \mathbf{P}_1 \tag{2.21}$$

Transformación homogénea general

La forma general de la matriz de transformación homogénea es:

$$T = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

Donde $\mathbf{n} = (n_x, n_y, n_z)^T$ es un vector que representa la dirección de x_1 en el sistema $o_0x_0y_0z_0$, $\mathbf{s} = (s_x, s_y, s_z)^T$ representa la dirección de y_1 y $\mathbf{a} = (a_x, a_y, a_z)^T$ representa la dirección de z_1. $\mathbf{d} = (d_x, d_y, d_z)^T$ es el vector que va del origen o_0 al origen o_1 expresado en el marco $o_0x_0y_0z_0$.

2.5. Cinemática directa

El problema de la cinemática directa consiste en dadas las variables de articulación del robot, determinar la posición y orientación del efector final. Para

realizar el análisis cinemático de manipuladores robóticos es necesario definir un conjunto de convenciones, que facilitarán el análisis y comprensión de la solución.

Convenciones

- Un robot con n articulaciones tiene $n+1$ eslabones numerados de 0 a n y empezando por la base del robot.
- Las articulaciones se enumeran de 1 a n y la i-ésima es el lugar geométrico donde los eslabones $i-1$ e i se conectan.
- Cada articulación tiene un sólo grado de libertad, la i-ésima variable de la articulación se denota por q_i.
  - θ_i si la articulación i es revoluta.
  - d_i si la articulación i es prismática.
- Se asigna un sistema de coordenadas a cada eslabón.

Por último, supóngase que B_i es una matriz homogénea que transforma las coordenadas de un punto del sistema S_i al sistema S_{i-1}. La matriz B_i no es constante, pero varia conforme cambia la configuración del robot. Pero es función de una sola variable.

$$B_i = B_i\left(q_i\right) \tag{2.23}$$

La matriz homogénea que transforma las coordenadas de un sistema S_j a un sistema S_i, se llama matriz de transformación y normalmente se denota T_i^j. La posición y orientación del efector final (S_n) en el sistema S_0, esta dado por:

$$T_0^n = B_1\left(q_1\right) B_2\left(q_2\right) \ldots B_i\left(q_i\right) \ldots B_n\left(q_n\right) \tag{2.24}$$

donde cada matriz B_i esta dada por:

$$B_i = \begin{bmatrix} Q_{i-1}^i & \mathbf{d}_{i-1}^i \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.25}$$

Por lo tanto,

$$T_i^j = B_{i+1} \cdots B_j = \begin{bmatrix} Q_i^j & \mathbf{d}_i^j \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.26}$$

La matriz Q_i^j expresa la orientación del sistema S_j relativa al sistema S_i y está dado por el conjunto de rotaciones $Q_i^j = Q_i^{i+1} \ldots Q_{j-1}^j$. Por otra parte el vector $\mathbf{d}_i^j$ está dada recursivamente por la ecuación $\mathbf{d}_i^j = \mathbf{d}_i^{j-1} + Q_i^{j-1}\mathbf{d}_{j-1}^j$.

2.6. Representación Hartenberg-Denavit

Una convención comúnmente utilizada para la selección de los marcos de referencia en aplicaciones robóticas es la de Denavit-Hartenberg, o convención HD. Ésta convención provee una descripción completa de la geometría del manipulador, mediante la generación de transformaciones homogéneas. El uso de esta convención y las anteriormente mostradas permiten simplificar el análisis considerablemente. Además, da lugar a un lenguaje universal con el que los ingenieros de robots se pueden comunicar. En esta convención, cada transformación homogénea B_i está representada como el producto de cuatro transformaciones básicas:

$$B_i = T_z\left(\theta_i\right) T_z\left(b_i\right) T_x\left(a_i\right) T_x\left(\alpha_i\right) \tag{2.27}$$

$$B_i = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

donde las cuatro cantidades θ_i, b_i, a_i y α_i son los parámetros H-D del i-ésimo eslabón y la i-ésima articulación y se conocen con los siguientes nombres:

Parámetros	Nombre
θ_i	Ángulo
b_i	Desviación
a_i	Longitud
α_i	Torsión

Cuadro 2.1: Parámetros H-D

Puesto que B_i es función de una sola variable, 3 de los 4 parámetros H-D son constantes para el eslabón dado, así, θ_i es la variable de articulación para un par de revoluta y b_i para un par prismático.

2.7. Asignación de ejes coordenados

Es imposible representar cualquier transformación homogénea haciendo uso de solo cuatro parámetros. Para que las transformaciones homogéneas puedan ser expresadas en la forma de la ecuación (2.27), es necesario que exista sólo

una matriz de transformación homogénea que relacione S_i con S_{i-1}. Para que esto suceda tienen que cumplirse las siguientes condiciones:

- El eje x_1 es perpendicular al eje z_0.
- El eje x_1 intersecta al eje z_0.

Teniendo en mente que las dos condiciones de arriba tienen que ser satisfechas se define un procedimiento general a seguir para la asignación de ejes coordenados. Para empezar, se asignan los ejes $z_0, \cdots, z_{n-1}$ de manera que z_i para que represente el eje de acción para la articulación $i+1$. Si la articulación $i+1$ es revoluta, z_i es el eje de rotación de la articulación $i+1$. Si la articulación $i+1$ es prismático, z_i es el eje de traslación de la articulación $i+1$.

Ya asignados los ejes z, se establece el marco base. El origen o_0 puede situarse en cualquier punto a lo largo de z_0, normalmente se establece en la base del manipulador. Después se orientan x_0 y y_0 de la manera más conveniente, de modo que el marco sea dextrógiro.

Por último se definen los sistemas de coordenadas restantes, definimos el marco i haciendo uso del marco $i-1$. Para realizar este proceso se consideran tres casos:

1. z_{i-1} y z_i no son coplanares.
   Si z_{i-1} y z_i no son coplanares, entonces existe un segmento de línea único perpendicular a z_{i-1} y z_i que conecta a las líneas con una longitud mínima. La linea que contiene a esta normal común entre z_{i-1} y z_i, define a x_i y el punto donde esta línea intersecta con z_i es el origen o_i.

2. z_{i-1} es paralelo a z_i.
   Si los ejes z_{i-1} y z_i son paralelos, entonces hay un número infinito de normales comunes entre z_{i-1} y z_i, por lo que o_i se coloca en cualquier punto a lo largo de z_i. Al colocar o_i se buscará que las ecuaciones se simplifiquen. x_i se coloca a lo largo de la normal común entre z_{i-1} y z_i, desde o_i en dirección a z_{i-1}.

3. z_{i-1} intersecta a z_i.
   En este caso x_i se escoge normal al plano formado por z_{i-1} y z_i. La dirección positiva de x_i es arbitraria. Para este caso el origen o_i se coloca en la intersección de z_{i-1} y z_i.

Con esto se termina la asignación de ejes coordenados, de modo que la ecuación (2.27) se satisfaga. Ahora se puede dar una interpretación física a cada uno de los cuatro parámetros D-H:

θ_i Ángulo entre x_{i-1} y x_i, medido en el eje z_{i-1}.

b_i Distancia desde el origen o_{i-1} hasta la intersección de los ejes x_i y z_{i-1}, medido a lo largo del eje z_{i-1}.

a_i Distancia entre los ejes z_{i-1} y z_i, medida a lo largo del eje x_i.

α_i Ángulos entre z_{i-1} y z_i medido en el eje x_i.

2.8. Cinemática inversa

Como su nombre lo sugiere el problema de la cinemática inversa, contrario a la cinemática directa, consiste en determinar las variables de articulación a partir de la posición de orientación del efector final, se plantea como sigue: Dada una transformación homogénea

$$H = \begin{bmatrix} Q & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.29}$$

Se requiere encontrar todas las soluciones de la ecuación T_0^n

$$T_0^n(q_1, q_2, \ldots, q_n) = H \tag{2.30}$$

$$T_0^n = B_1, B_2, \ldots, B_n \tag{2.31}$$

H representa la posición y orientación deseada para el efector final, la tarea consiste en encontrar los valores de las variables de articulación $q_1, q_2, \ldots, q_n$ que cumplan con H. De la ecuación (2.30) resultan 12 ecuaciones no lineales con n incógnitas, las cuales pueden ser escritas como:

$$T_{ij}(q_1, \ldots, q_n) = h_{ij}, \quad i = 1, 2, 3, \quad j = 1, \ldots, 4 \tag{2.32}$$

donde T_{ij} y h_{ij} se refieren a las doce ecuaciones no triviales de T_0^n y H, respectivamente.

En la solución de problemas de cinemática inversa es deseable encontrar soluciones de forma cerrada.

$$q_k = f_k(h_{11}, \ldots, h_{34}), \quad k = 1, \ldots, n \tag{2.33}$$

Las soluciones de forma cerrada son preferibles pues en algunas aplicaciones, es necesario que las ecuaciones de cinemática inversa sean resueltas rapidamente. Además, en general éstas ecuaciones tienen multiples soluciones, las soluciones de forma cerrada nos permiten desarrollar reglas para la selección de una solución en particular. Una vez resuelto el problema es necesario checar que se satisfagan las restricciones de movimiento de las articulaciones.

2.8.1. Desacoplamiento cinemático

El problema general de las cinemática inversa resulta complicado.Pero en manipuladores de 6 grados de libertad, con las 3 últimas articulaciones intersectandose, éste se puede separar en dos problemas:

- Cinemática inversa de posición (Centro de la muñeca).
- Cinemática inversa de orientación (Orientación de la muñeca).

La ecuación (2.30) se representa como un conjunto de dos ecuaciones:

$$Q_0^6(q_1, \ldots, q_6) = Q \tag{2.34}$$

$$\mathbf{d}_0^6(q_1, \ldots, q_6) = \mathbf{d} \tag{2.35}$$

$\mathbf{d}$ y Q son respectivamente, la posición y orientación deseadas del sistema de coordenadas de la herramienta.

La consideración de una muñeca esférica implica que z_4, z_5 y z_6 se intersectan en O_c y por lo tanto O_4 y O_5 están siempre en el centro de la muñeca. El origen del sistema coordenado de la herramienta es obtenido mediante una traslación b_6 a lo largo de z_5 a partir de o_c.

$$\mathbf{d}_0^6 - \mathbf{d}_0^c = b_6 Q\mathbf{k} \tag{2.36}$$

Si $\mathbf{p}_c = (p_x, p_y, p_z)^T$ representa al vector que va del origen del S_0 al centro de la muñeca, y los componentes de la posición del efector final son denotados por $\mathbf{d} = (d_x, d_y, d_z)^T$.

$$\mathbf{p}_c = \mathbf{d} - b_6 Q\mathbf{k} \tag{2.37}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_x - b_6 q_{13} \\ d_y - b_6 q_{23} \\ d_z - b_6 q_{33} \end{bmatrix} \tag{2.38}$$

Con la aplicación de (2.38) se pueden obtener los valores de las tres primeras variables de articulación $q_i(q_1, q_2, q_3)$, esto determina Q_0^3, así:

$$Q = Q_0^3 Q_3^6 \tag{2.39}$$

De manera que

$$Q_3^6 = \left[Q_0^3\right]^T Q \tag{2.40}$$

Los últimos tres angulos de articulación se determinan como un conjunto de ángulos de Euler que corresponden a Q_3^6.

2.8.2. Cinemática inversa de posición: método geométrico.

Para la mayoria de los manipuladores es posible utilizar el método geométrico para encontrar las variables q_1, q_2, q_3, correspondientes al centro de la muñeca. Este método es muy relevante por dos razones. Primeramente, la mayoría de los manipuladores actuales son cinemáticamente simples. En segundo lugar, existen algunas técnicas que pueden manejar el problema general de cinemática inversa para configuraciones arbitrarias.

En general, la complejidad del problema de la cinemática inversa se incrementa en cuanto se tiene un número mayor de parámetros DH diferentes de cero, en estos casos el método geométrico resulta más simple. La idea general del método geométrico es obtener una solución para la variable q_i, mediante la proyección del manipulador en un plano y resolución de un problema de trigonometría simple.

2.9. Cinemática de velocidad

El objetivo de la cinemática de velocidad es relacionar las velocidades lineales y angulares del efector final con las velocidades de articulación. Las relaciones de velocidad son determinadas por el jacobiano de la función que se obtiene en el análisis de cinemática directa.

2.10. Matrices antisimétricas

Es necesario definir el concepto de matriz antisimétrica. Puesto que esta herramienta permite simplificar la mayoría de los cálculos relacionados con la cinemática de velocidad. Una matriz A se dice que es antisimétrica si y sólo si:

$$A^T + A = 0 \tag{2.41}$$

dicho de otra manera los componentes de A cumplen con:

$$a_{ij} + a_{ji} = 0: \quad i, j = 1, 2, 3$$

Se observa que $a_{ii} = 0$. $i \neq j$ satisfacen la condición $a_{ij} = -a_{ji}$. La matriz A será de la forma:

$$A = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.42}$$

2.10.1. La derivada de una matriz de rotación.

Suponga que una matriz de rotación Q es función de una sola variable θ. Puesto que Q al ser una matriz de rotación es ortogonal:

$$Q(\theta) Q(\theta)^T = I \tag{2.43}$$

Derivando con respecto a θ, la ecuación(2.43) obtenemos:

$$\frac{dQ(\theta)}{d\theta} Q(\theta)^T + Q(\theta) \frac{dQ(\theta)^T}{d\theta} = \mathbf{0} \tag{2.44}$$

Si se define la matriz antisimétrica A como:

$$A = \frac{dQ(\theta)}{d\theta} Q(\theta)^T \tag{2.45}$$

Entonces la transpuesta de A es:

$$A^T = \left(\frac{dQ(\theta)}{d\theta} Q(\theta)^T \right)^T = Q(\theta) \frac{dQ(\theta)^T}{d\theta} \tag{2.46}$$

Así, la ecuación (2.44) nos dice que:

$$A + A^T = 0 \tag{2.47}$$

Por lo tanto Q es antisimétrica si (2.45) se multiplica por $Q(\theta)$

$$AQ(\theta) = \frac{dQ(\theta)}{d\theta} \tag{2.48}$$

Lo que dice 2.48 es muy importante, dice que la derivada de una matriz Q es igual a la misma matriz premultiplicada por una matriz antisimétrica A. Las derivadas de las matrices de rotación básica son:

$$\frac{dQ_x(\theta)}{d\theta} = A(\mathbf{i})\, Q_x(\theta) \tag{2.49}$$

$$\frac{dQ_y(\theta)}{d\theta} = A(\mathbf{j})\, Q_y(\theta) \tag{2.50}$$

$$\frac{dQ_z(\theta)}{d\theta} = A(\mathbf{k})\, Q_z(\theta) \tag{2.51}$$

Donde:

$$\mathbf{i} = \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \mathbf{j} = \begin{bmatrix}0\\1\\0\end{bmatrix} \quad \mathbf{k} = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{2.52}$$

Y las matrices antisimétricas A $(\mathbf{i})$, $A(\mathbf{j})$ y $A(\mathbf{k})$ son:

$$A(\mathbf{i}) = \begin{bmatrix}0 & 0 & 0\\0 & 0 & -1\\0 & 1 & 0\end{bmatrix} \quad A(\mathbf{j}) = \begin{bmatrix}0 & 0 & 1\\0 & 0 & 0\\-1 & 0 & 0\end{bmatrix} \quad A(\mathbf{k}) = \begin{bmatrix}0 & -1 & 0\\1 & 0 & 0\\0 & 0 & 0\end{bmatrix} \tag{2.53}$$

2.10.2. El jacobiano

El jacobiano o la matriz jacobiana relaciona las velocidades del efector final con las velocidades de las articulaciones. Es una matriz de $6 \times n$. Se tiene un manipulador de n eslabones con variables de articulación $q_1.q_2, \ldots, q_n$ entonces:

$$T_0^n(\mathbf{q}) = \begin{bmatrix} Q_0^n(\mathbf{q}) & \mathbf{d}_0^n(\mathbf{q}) \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.54}$$

T_0^n representa la transformación desde el sistema de referencia del efector final S_n, al sistema de referencia de la base S_0, donde:

$$\mathbf{q} = (q_1, q_2, \ldots, q_n)^T \tag{2.55}$$

$\mathbf{q}$ es el vector de las variables de articulación. El objetivo es relacionar las velocidades lineal y angular del efector final con el vector de las variables de articulación $\dot{\mathbf{q}}(t)$. El vector de velocidad angular $\vec{\omega}_0^n$ del efector final, se define como:

$$\dot{Q}_0^n = A(\vec{\omega}_0^n)\, Q_0^n \tag{2.56}$$

$$A(\vec{\omega}_0^n) = \dot{Q}_0^n {Q_0^n}^T \tag{2.57}$$

mientras que la velocidad lineal del efector final corresponde a:

$$\mathbf{V_0^n} = \dot{\mathbf{d}}_0^n \tag{2.58}$$

De acuerdo al objetivo anteriormente descrito, se busca obtener expresiones de la forma

$$\begin{aligned} \mathbf{V_0^n} &= J_v \dot{\mathbf{q}} \\ \vec{\omega}_0^n &= J_\omega \dot{\mathbf{q}} \end{aligned} \tag{2.59}$$

donde J_v y J_ω son matrices $3 \times n$, en donde n es el número de eslabones. Las ecuaciones (2.59) las podemos escribir de la forma:

$$\begin{bmatrix} \mathbf{V_0^n} \\ \vec{\omega}_0^n \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{\mathbf{q}} = J_0^n \dot{\mathbf{q}} \tag{2.60}$$

A la matriz J_0^n de $6 \times n$ se le denomina jacobiano del manipulador o simplemente jacobiano. El Jacobiano es una de las cantidades más importantes en el análisis y el control del movimiento de un robot. Surge en cada aspecto, en la planeación y ejecución de trayectorias suaves, en la determinación de singularidades, y en las ecuaciones dinámicas de movimiento.

2.10.3. Velocidad lineal y angular del jacobiano

En la sección anterior se vio que el Jacobiano se divide en dos partes. La parte superior J_v nos permite relacionar la velocidad de las articulaciones con la velocidad lineal de efector final. J_v es de la forma:

$$J_v = [J_{v1}, \ldots, J_{vn}] \tag{2.61}$$

donde la i-ésima columna de J_v es

$$J_{vi} = \begin{cases} \mathbf{e}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1}) & \text{si la articulación } i \text{ es revoluta} \\ \mathbf{e}_{i-1} & \text{si la articulación } i \text{ es prismática} \end{cases}$$

la parte inferior J_ω de manera similar relaciona la velocidad de las articulaciones con la velocidad angular de efector final. J_ω es de la forma:

$$J_\omega = [J_{\omega 1}, \ldots, J_{\omega n}] \tag{2.62}$$

la i-ésima columna de j_ω es

$$J_{\omega i} = \begin{cases} \mathbf{e}_{i-1} & \text{si la articulación } i \text{ es revoluta} \\ \mathbf{0} & \text{si la articulación } i \text{ es prismática} \end{cases}$$

Donde $\mathbf{e}_{i-1} = Q_0^{i-1}\mathbf{k}$ representa al vector unitario del eje z_{i-1} representado en S_0. $\mathbf{o}_{i-1}$ es el vector de coordenadas del origen del marco $i-1$ y $\mathbf{o}_n$ es el vector de coordenadas del origen del marco n. Si unimos la parte superior e inferior del Jacobiano, se obtiene:

$$J = [J_1, J_2, \ldots, J_n] \tag{2.63}$$

Para este Jacobiano, si la articulación es revoluta, la i-ésima columna estará dada por:

$$J_i = \begin{bmatrix} \mathbf{e}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1}) \\ \mathbf{e}_{i-1} \end{bmatrix} \tag{2.64}$$

Si la articulación es prismática:

$$J_i = \begin{bmatrix} \mathbf{e}_{i-1} \\ \mathbf{0} \end{bmatrix} \tag{2.65}$$

2.11. Singularidades

El rango de una matriz es el número de columnas o renglones linealmente independientes. Éste rango no necesariamente es constante ya que depende del vector de variables de articulación $\mathbf{q}$. Aquellas configuraciones para las cuales el rango de J decrece, se llaman singulares. Son de de interés en nuestro análisis por varias razones.

1. Representan configuraciones para las que ciertas direcciones del movimiento no se pueden alcanzar.
2. Cerca de una singularidad no existirá una solución única al problema de la cinemática inversa. En tales casos no habrá solución o habrá un número infinito de soluciones.
3. Las singularidades corresponden a puntos en el espacio de trabajo del manipulador que pueden ser inalcanzables bajo pequeñas perturbaciones de los parámetros de los eslabones.

2.11.1. Desacoplamiento de singularidades

Un método para determinar las singularidades del Jacobiano es el desacoplamiento de singularidades. En este método:

- Se determinan las singularidades del brazo, que resultan del movimiento de las tres primeras articulaciones.
- Se determinan las singularidades de la muñeca que resultan del movimiento de las articulaciones de esta.

Se parte del hecho de que se tienen 3GDL del brazo y 3GDL de la muñeca, así, J es de dimensión 6×6 y una configuración singular se da si y solo si

$$\det(J(\mathbf{q})) = 0$$

Si se parte J en bloques de 3×3, se tiene:

$$J = \begin{bmatrix} J_p & J_o \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{2.66}$$

Como se puede observar J_p representa el Jacobiano de posición, referente al brazo del manipulador, y J_o representa el Jacobiano de orientación, referente a la muñeca. Cada bloque es una matriz de 3×3 donde $\det(J_{11})$ define las singularidades del brazo, mientras que $\det(J_{22})$ determina las singularidades de la muñeca.

Capítulo 3

Adquisición de datos

Para los propósitos de nuestro proyecto es necesario realizar la captura de la trayectoria seguida por el manipulador. Esto es, conocer la posición en x, y y z del efector final, así como el tiempo en que fue alcanzada esta posición. En otras palabras se realiza la captura de cada uno de los puntos que componen la trayectoria cumplido un periodo de tiempo. La captura de la trayectoria nos permitirá obtener los datos necesarios para conocer los aspectos cinemáticos del manipulador utilizado, para posteriormente hacer la comparativa con las arquitecturas propuestas.

Para la captura de la trayectoria se hace uso de un prototipo genérico de manipulador serial de tres grados de libertad [7], equipado con encoders en cada una de sus tres articulaciones que permiten la medición del ángulo de articulación.

Como se mencionó en un principio, al trabajar en la plataforma ADEFID se pueden generar aplicaciones desde cero, o bien tomar una aplicación ya desarrollada la cual será tomada como plataforma para lograr un nivel más alto de especialización. En nuestro caso se partirá de un paquete ya desarrollado, SnAM [8], este paquete nos provee de herramientas que simplificarán la adquisición de la trayectoria, SnAM cuenta con una interfaz de comunicación que recibe y procesa la señal de los encoders. SnAM también permitirá en secciones posteriores realizar la simulación de la trayectoria adquirida. A continuación se da una breve descripción del prototipo génerico y del paquete SnAM.

3.1. Prototipo genérico

El protótipo genérico de manipulador serial de tres grados de libertad, mostrado en la imagen 3.1, fue construido por el M.I. Erick Alejandro González Barbosa. Este prototipo es del tipo 3R. Las características de este manipulador son las siguientes [7]:

- Cuenta con 3 grados de libertad que permiten el posicionamiento de un objeto en el espacio.

- Cada articulación cuenta con un encoder para medir el ángulo θ_i. Los encoders son del tipo rotatorio incremental, con una resolución de 2000 pulsos/revolución, cada cuadratura de pulso está compuesta por 4 posibles combinaciones de registro de cambios en la señal detectada; por lo que se registra una resolución de 8000 pulsos/revolución. Todos los encoder se conectan a un tablero de control que a su vez se conecta con el paquete SnAM.
- Es versátil, ya que los parámetros D-H, b, a y α, pueden ajustarse de manera sencilla a las dimensiones requeridas.
- El manejo es de forma manual, no cuenta con motores necesarios para su movimiento automático.

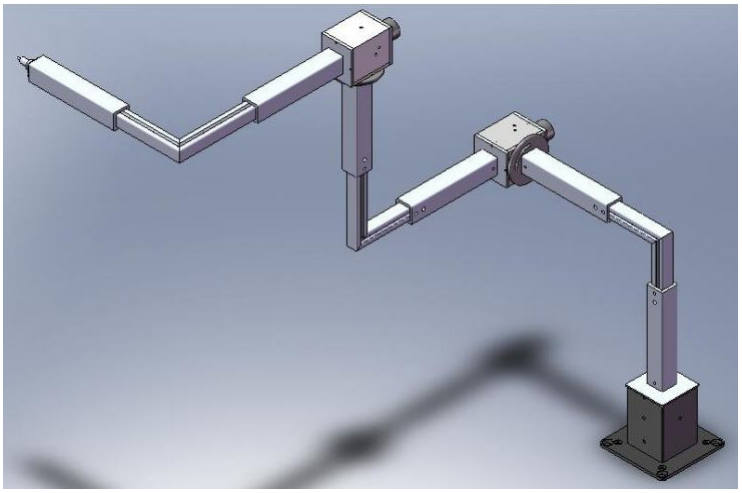

Figura 3.1: Conficuración general del prototipo genérico [7].

Como ya se mencionó, los encoders irán conectados a un tablero de control. Dicho tablero consta de un sistema SNAP PAC de Opto22, que integra software y hardware, para aplicaciones de control industrial, monitoreo remoto y adquisición de datos [10]. El sistema SNAP PAC se divide en tres partes principales:

- Un módulo de entradas y sálidas, en nuestro caso serán módulos de cuadratura, que cuentan los pulsos generados por los encoders.
- Un cerebro (Brain), encargado del procesamiento de las entradas y salidas, también provee de control al sistema y permite la comunicación vía ethernet.
- Una computadora con el software adecuado para controlar el sistema. El paquete de software SNAM, integra la clase CIOBRAIN que sirve para manejar las funciones del sistema SNAP PAC.

3.2. SnAM

La adquisición de datos se desarrollará en SnAM puesto que este paquete tiene la capacidad de controlar el sistema SNAP. Desarrollado bajo la plataforma ADEFID, SnAM es un paquete de software interactivo para el análisis, síntesis y simulación de manipuladores seriales [8].

Los componentes principales de este software son: la interfaz gráfica y la interfaz de comunicación. La interfaz gráfica está basada en herramientas de OpenGL escritas en C++, esto incluye a los eslabones, articulaciones (prismáticos o revoluta), el ambiente, y la interfaz de usuario, todo esto con la ayuda de Microsoft Visual Studio C++ y MFC. La interfaz de comunicación recibe y procesa las señales de los encoders. Esto se logra mediante los dispositivos de Opto22.

La simulación de SnAM es renderizada en tiempo real, e incluye el análisis cinemático de todo tipo de manipuladores seriales. Para el análisis cinemático se incluye la solución de la cinemática directa para manipuladodes seriales de n-grados de libertad. También se incluye la solución de cinemática inversa, pero esta se encuentra limitada a configuraciones de hasta 6 grados de libertad.

3.3. Metodología

Debido a que la adquisición se hará mediante periodo, es de vital importancia conocer en que momento se cumple el periodo para adquirir la posición del efector final. Para lo cual se sigue el algoritmo descrito en el diagrama de la figura 3.2.
La función `clock()` y `difftime(t2,t1)`, son propias de la biblioteca `time.h`, esta biblioteca contiene funciones para obtener y manipular la información de tiempo [11]. `clock()` devuelve el tiempo consumido por el programa, el valor es expresado en ciclos de reloj. Para convertir a segundos los ciclos de reloj, los dividimos entre el macro `CLOCKS_PER_SEC`, como su nombre lo sugiere, este macro representa el número de ciclos de reloj por segundo.

La función `difftime(t2,t1)` calcula la diferencia entre el tiempo final `t2` y el tiempo inicial `t1`. Las variables `t1` y `t2` son del tipo `time_t`, representan tiempo, y también son propias de la biblioteca `time.h`.

La variable `StartRec` es booleana, el valor de esta nos permite realizar el control de la adquisición y se modifica a partir de un diálogo en la interfaz del usuario.

El guardado de los valores de θ_1, θ_2 y θ_3 se hace mediante el manejo de ficheros. Para ello primero creamos un apuntador del tipo `FILE *`. Después abrimos el archivo donde almacenaremos los datos, utilizando la función `fopen` y asignamos el resultado de la llamada a nuestro apuntador. A continuación realizamos en este caso las operaciones de escritura en el archivo abierto mediante la función `fprint`. Por último cerramos el archivo utilizando la función `fclose`.

Recordando el proceso principal de un proyecto de ADEFID, descrito en el diagrama de flujo de la figura 1.8. Dentro del proceso principal se leerán las

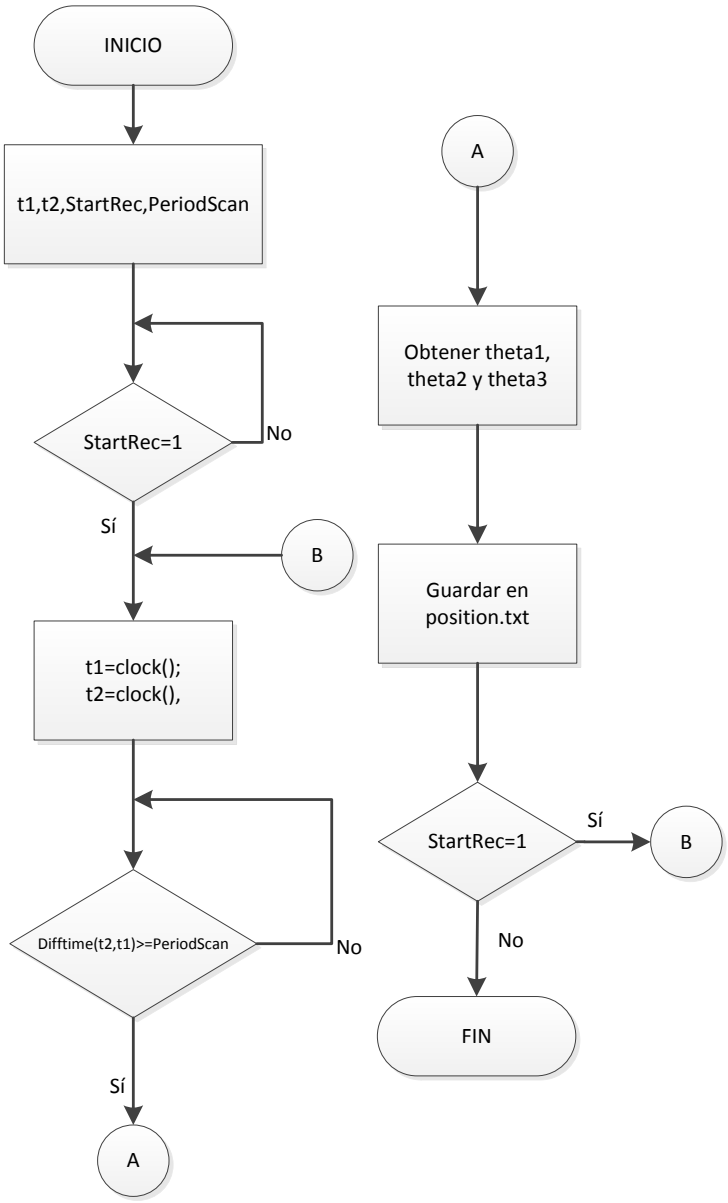

Figura 3.2: Diagrama de flujo de adquisición de datos.

entradas procesadas mediante el Opto 22, en esté caso son las mediciones de los encoders. Se revisan las entradas de seguridad y se actualizan los estados de las máquinas. Por último se tiene acceso a cada máquina mediante su miembro de función "`Algorithm()`", en esta parte es donde se programa el algoritmo previamente descrito. El código implementado es mostrado en 3.1.

Código 3.1: Código implementado para la adquisición de datos

```
void CManipulator::Algorithm(void)
{
...
....
.....
switch(GetPickPointMode())
{
        case MANUALLY:
        if(pADEFIDDoc->m_Brain.ReadOneInput(m_iSSetPoint)==1 && false == m_IsPointSet)
                {
                        SetPoint();
                        m_IsPointSet=true;
                }
        if(pADEFIDDoc->m_Brain.ReadOneInput(m_iSSetPoint)==0)
                m_IsPointSet=false;
        break;
        case GIVENPERIOD:
        //Codigo implementado
        //Se ajustan t2 y t1
        if(setclks==0)
        {
                t1=clock();
                t2=clock();
        }
        //Se revisa la diferencia entre t1 y t2 para comprobar si se ha cumplido el periodo
        //periodscan es un macro definido al principio
        if(difftime(t2,t1)>=periodscan*CLOCKS_PER_SEC)
        {
                SetPoint();
                setclks=0;
                //Se comprueba si se ha iniciado la adquisicion
                if(pADEFIDDoc->RecordScan.StartRec==1)
                {
                //Se guardan los datos adquiridos en position.txt
                 if((fopen_s(&posicion,"position.txt", "a+"))==TRUE);
                 else
                 {
                        long var1,var2,var3;
                        float tetha1, tetha2, tetha3;
                        pADEFIDDoc->m_Brain.GetDigPtCounts(4,&var1);
                        pADEFIDDoc->m_Brain.GetDigPtCounts(6,&var2);
                        pADEFIDDoc->m_Brain.GetDigPtCounts(8,&var3);

                        tetha1 = m_thetaHome[0]*(180./PI)+(float)(var1*(360/8000.));
                        tetha2 = m_thetaHome[1]*(180./PI)+(float)(var2*(360/8000.));
                        tetha3 = m_thetaHome[2]*(180./PI)-(float)(var3*(360/8000.));

                        fprintf_s(posicion,"%.4f, %.4f, %.4f\n",tetha1, tetha2, tetha3);
                        m_TransfMatrix = FullForwardKinematics(&Dh4);
                        fclose(posicion);
                 }
                }
        }
        else
        {
                t2=clock();
                setclks=1;
        }
        break;
        case GIVENLENGTH:
        if(IsPenDown==TRUE)
        {
                m_pos.x = m_TransfMatrix.q[0][3];
                m_pos.y = m_TransfMatrix.q[1][3];
                m_pos.z = m_TransfMatrix.q[2][3];
                if(GetLength(m_firstPoint,m_pos)>pADEFIDDoc->PathGenDlg.m_scan_length)
                //Change 2 for the parametric unit
                {
                        SetPoint();
                        pADEFIDDoc->PathGen.SetPoint(m_pos.x,m_pos.y,m_pos.z);
                        m_firstPoint = m_pos;
                }
        }
        else
        {
                m_firstPoint = m_pos;
```

```
77                  m_point_counter=0;
78          }
79          break;
80  }
81  .....
82  ....
83  ...
84  }
```

Como se puede ver, el algoritmo de adquisición mediante periodo, se encuentra dentro de una serie de casos en la función "`Algorithm()`", esto es debido a que SnAM permite realizar el escaneo de posición mediante tres métodos. "`Manually`", permite realizar el escaneo de manera manual donde el usuario establece los puntos escaneados. "`GIVENPERIOD`", el caso implementado en este proyecto, escanea automáticamente los puntos cumplido un periodo de tiempo definido, si la adquisición de datos se encuentra activa (`StartRec==1`) los datos se guardan en el archivo y formato indicado. "`GIVENLENGHT`" por su parte realiza el escaneo también de manera automática, pero en este caso el escaneo se realiza cumplida una distancia determinada. La función "`GetPickPointMode()`" obtiene el modo en el cual se encuentra el escaneo, este se modifica dentro de la interfaz gráfica en el menú "Scan Mode".

3.3.1. Menú y diálogo "Record Scan By Period"

Como ya fue mencionado, la adquisición de datos será controlada por el usuario mediante la interfaz gráfica, para ello se creará un diálogo que será asociado a un menú. El proceso completo para la creación y asociación de diálogos y menús se describe en [9]. SnAM cuenta con una serie de menús preprogramados, podemos agregar menús u opciones a los menús ya existentes.

Dentro del menú "Tasks" se agregó la opción "RecordScanByPeriod", para ello dentro de Microsoft Visual Studio se accede mediante la vista de recursos a ADEFID.rc\Menu\IDR_MAINFRAME. Al seguir esta ruta podremos editar y agregar nuevos menús, como se muestra en la figura 3.3.

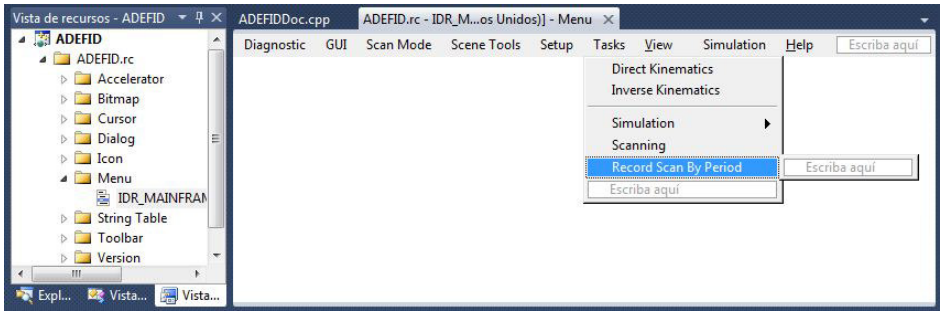

Figura 3.3: Creación del menú Record Scan By Period.

Por otra parte, tenemos que agregar un nuevo diálogo, el cual será muy sencillo ya que constará únicamente de dos botones, "Start" y "Stop", que indican

el inicio y término de la adquisición. Se podría simplificar el diálogo aún más utilizando un sólo botón, pero de ser así se correría el riesgo de dar click accidentalmente dos veces seguidas al botón y no adquirir los datos. Para insertar un diálogo se hace click derecho sobre la carpeta Dialog de la vista de recursos y se selecciona insertar diálogo. Este proceso generará una plantilla con los botones "Aceptar" y "Cancelar", los eliminamos. Mediante el cuadro de herramientas agregamos los botones "Start" y "Stop", el diálogo generado es el mostrado en 3.4.

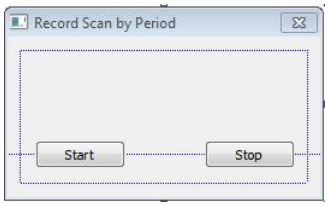

Figura 3.4: Creación del diálogo Record Scan By Period.

El siguiente paso es asociar una clase al diálogo creado. Para esto, teniendo abierto el diálogo se hace click derecho sobre él y se selecciona la opción "Add Class", ahora se abre el asistente para agregar clases, mostrado en la figura 3.5. La nueva clase es nombrada `CRecordScan`. En Base Class se selecciona la clase `CDialog`, para que la nueva clase herede su contenido.

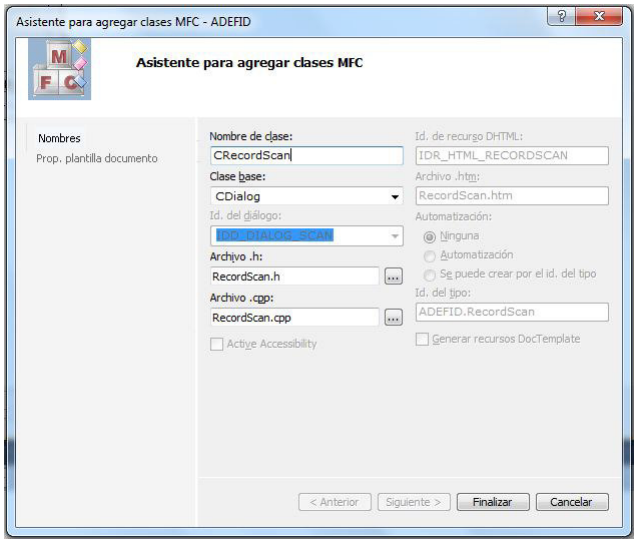

Figura 3.5: Creación de la clase RecordScan.

En este punto, el siguiente paso es activar el menú y asociar el diálogo. El manejador de eventos permite establecer las acciones que realiza el menú. Para

acceder a ello, teniendo abierto IDR_MAINFRAME se da click derecho en el menú a activar y se selecciona la opción "Agregar Controlador de eventos", se abrirá el asistente para controladores de eventos mostrado en la figura 3.6, en el tipo de mensaje se selecciona COMMAND y en la lista de clases se seleccionará CADEFIDDoc. Después de esto se presiona el botón "Agr.Editar". Después de

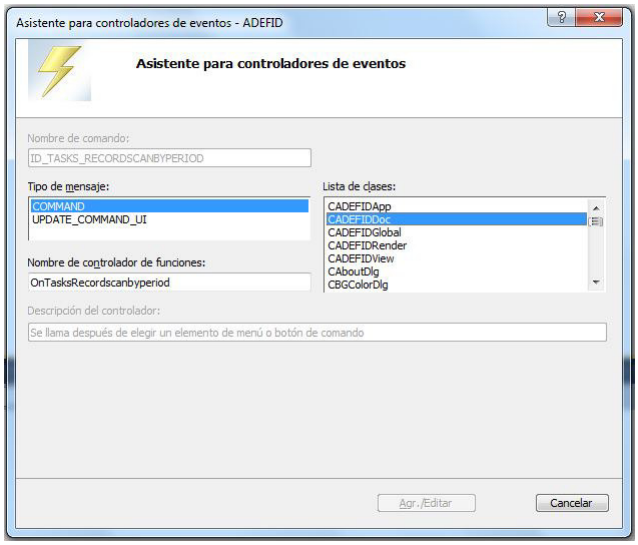

Figura 3.6: Controlador de eventos para un menú.

realizado éste paso en `ADEFIDDoc.cpp` se agregará el código correspondiente al manejador de eventos creado, preparado para agregar las instruciones que se necesiten. En este caso la asociación del diálogo. Para la asociación del diálogo primero se crea un objeto de la clase `CRecordScan`, en el archivo `ADEFIDDoc.h` se agregan las líneas `#include "RecordScan.h"` y `CRecordScan RecordScan`. Después se agrega el código `RecordScan.DoModal();` en el manejador de eventos previamente creado, como se muestra en el código 3.2.

Código 3.2: Código para el controlador de eventos del menú Record Scan by Period

```
1 void CADEFIDDoc::OnTasksRecordscanbyperiod()
2 {
3 // TODO: Agregue aqui su codigo de controlador de comandos
4         RecordScan.DoModal();
5 }
```

Ahora al presionar la opción "Record Scan By Period" del menú "Tasks" aparecerá el diálogo correspondiente. Pero hasta el momento no han sido asignadas funciones a los botones del diálogo, para realizar esta asignación se abre el diálogo y dando click derecho sobre el botón a activar, se selecciona "Agre-

gar controlador de eventos", y se agrega el controlador de eventos tal como se muestra en la figura 3.7.

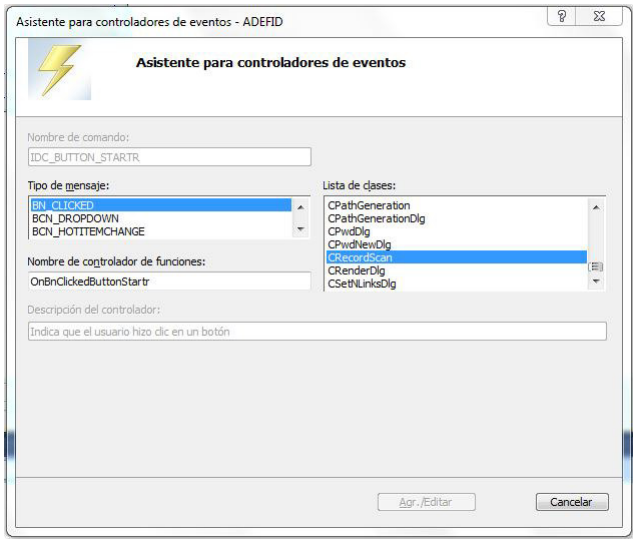

Figura 3.7: Controlador de eventos para un botón.

Al Agregar los controladores de eventos para los botones "Start" y "Stop", se agregará el código preparado para las instrucciones de control que se requieran. Para este caso, se controla el valor de la variable `StartRec`, mediante los códigos 3.3 y 3.4.

Código 3.3: Código para el controlador de eventos del botón Start

```
1 void CRecordScan::OnBnClickedButtonStartr()
2 {
3         // TODO: Agregue aqui su codigo de controlador de
      notificacion de control
4         StartRec=1;
5 }
```

Código 3.4: Código para el controlador de eventos del botón Stop

```
1 void CRecordScan::OnBnClickedButtonStopr()
2 {
3         // TODO: Agregue aqui su codigo de controlador de
      notificacion de control
4         StartRec=0;
5 }
```

Al controlar el valor de `StartRec` mediante estos botones, se controla el inicio y fin de la adquisición de datos.

3.4. Metodología de empleo

Para realizar la adquisición de datos implementada en el proyecto es necesario seguir los pasos descritos a continuación.

- Revisar que los encoders del prototipo genérico estén conectados al tablero del Opto 22 y se enciende el tablero.
- Se conectan a la misma red el tablero y la computadora, para realizar esta conexión se puede usar un router o conectar la computadora directamente al tablero.
- Puesto que el prototipo genérico cuenta con encoders del tipo incremental, mismos que solo indican la distancia angular recorrida, pero no asignan a cada posición un valor, es necesario colocar el prototipo en una posición conocida la cual se nombra "Home". En selección de la posición Home debe considerarse que sea fácil de alcanzar, en este caso se selecciona la posición mostrada en la imagen 3.8, donde $\theta_2 + \theta_3 = 90$.

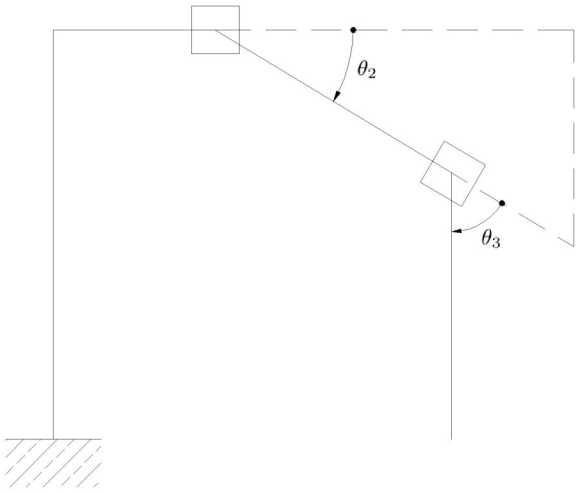

Figura 3.8: Posición Home del prototipo genérico.

- Se ejecuta el programa y se espera a que en el diálogo de la imagen 3.9, muestre que inicialización del sistema es exitosa , esto es que exista comunicación entre el software SnAM y el Opto22. En seguida se abre la ventana principal de SnAM, mostrada en la imagen 3.10.
- Para activar la adquisición por periodo, en el menú “Scan Mode” se selecciona “By Period”. Para conectar el prototipo a SnAM, en el menú “Tasks” se selecciona la opción “Direct Kinematics”, se abre automáticamente el diálogo de la imagen 3.11, y se presiona el botón “connect” para conectar

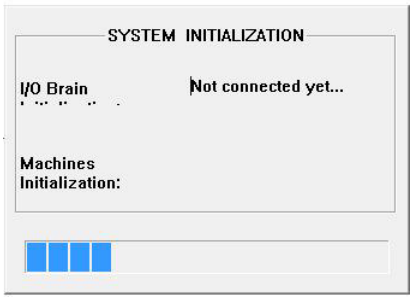

Figura 3.9: Inicialización del sistema.

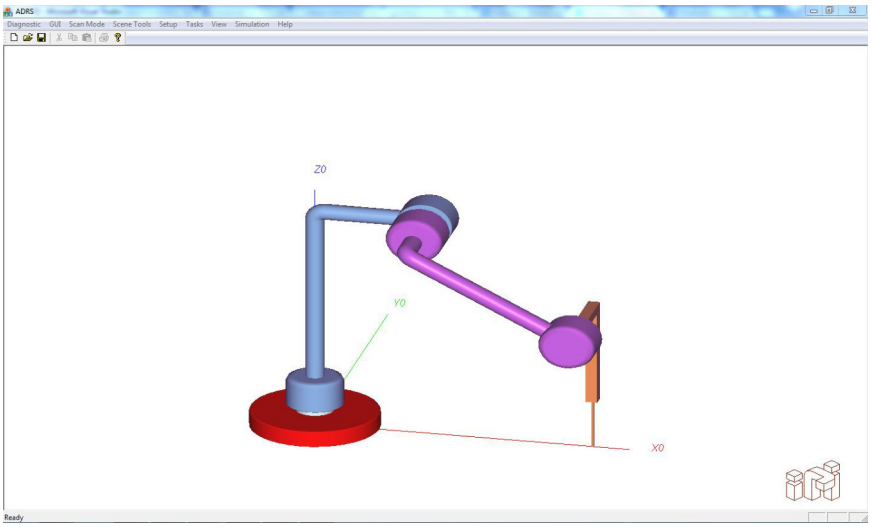

Figura 3.10: Ventana principal de SnAM.

el prototipo genérico a SnAM, de este modo el manipulador representado en SnAM seguirá como esclavo los movimientos del prototipo. Es importante mencionar que antes de conectar el prototipo genérico a SnAM, es necesario posicionar el manipulador representado en SnAM en la posición Home, tal como se muestra en la imagen 3.12.

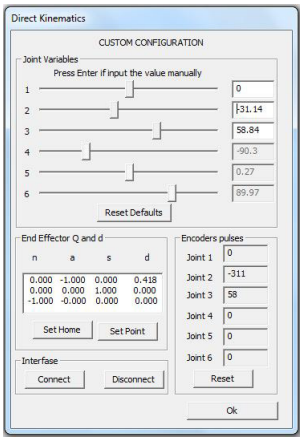

Figura 3.11: Diálogo de cinemática directa.

Figura 3.12: Prototipo en posición Home.

- Se posiciona el prototipo al inicio de la trayectoria a adquirir. Se abre el diálogo "Record Scan By Period", de la imagen 3.13, se presiona el botón "Start" y se sigue la trayectoria. Al término del seguimiento de la

trayectoria se presiona el botón “Stop” con lo que se da por terminada la adquisición de datos.

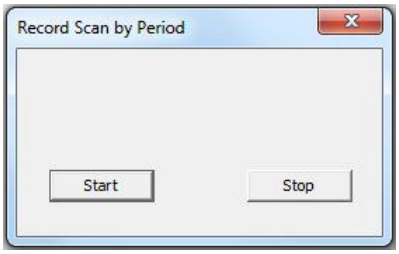

Figura 3.13: Diálogo de adquisición de trayectoria.

- Los datos adquiridos estarán guardados en el archivo `position.txt`, en el formato que se indicó en el código 3.1.

En el presente capítulo se ha explicado la metodología que se uso en el proyecto para realizar la adquisición de trayectorias mediante el prototipo genérico y el paquete de software SnAM. La información adquirida mediante este proceso, será utilizada, como se muestra en los siguientes capítulos, para realizar la simulación del prototipo genérico y realizar la comparativa con arquitecturas distintas a la usada en la adquisición. Esta información también permite conocer la velocidad y aceleración de las articulaciones del prototipo genérico.

Capítulo 4

Simulación

Puesto que el paquete SnAM ha sido desarrollado bajo la plataforma ADEFID, éste contiene todas las herramientas necesarias para generar la simulación, así como el estudio comprensivo de cualquier configuración de manipuladores seriales [8].

Una vez realizada la adquisición de la trayectoria mediante el prototipo genérico, utilizando la metodología descrita en el capítulo anterior, se puede simular la trayectoria adquirida. En este capitulo se muestra cómo se realiza la simulación a partir de los datos adquiridos.

4.1. Control

ADEFID se deriva de una plantilla de MFC, que cuenta con una arquitectura Vista/Documento. En esta arquitectura, el Documento almacena los datos, administra la impresión de los mismos y coordina la actualización de varias vistas de los datos. Mientras que la vista muestra los datos y administra la interacción de los usuarios con éstos, incluyendo la selección y la edición.

Teniendo en cuenta la operación de la arquitectura Vista/Documento, para los propósitos de nuestro proyecto, es necesario controlar como el Documento realiza la coordinación de la actualización de la Vista. En otras palabras se controlará como el Documento envia los datos de las poses a impresión, esto tomando en cuenta el periodo de tiempo utilizado en la adquisición de los datos utilizados. Para realizar dicho control se sigue la metodología mostrada en el diagrama de flujo de la imagen 4.1.

Como se puede observar, primero se revisa si se ha indicado el inicio de la simulación. Enseguida se cargan los datos previamente adquiridos, que representan la trayectoria a simular. Cumplido esto se hace el control de la actualización de la pose mostrada en la vista. Para ello se definen tres posiciones, la posición Home, la misma posición que fue utilizada en el capítulo anterior para la adquisición, la posición A, que representa la posición de inicio de la trayectoria adquirida, y la posición B, que es la última posición de la trayectoria adquirida.

También quedan definidos tres estados para el manipulador. "RESET", el manipulador se encuentra en una posición arbitraria y seguirá una trayectoria para llegar a la posición HOME. "PIN", en este estado el manipulador se situa en la posición HOME y seguirá una trayectoria para alcanzar la posición A. "POUT", el manipulador se encuentra en A y seguirá una trayectoria para alcanzar la posición B. El definir estos tres estados es de utilidad cuando se quiere implementar en código el algoritmo representado en el diagrama de flujo.

Para la simulación se hace uso de las funciones `SetInvariants` y `GetPos`, estas funciones son propias de la clase `CLinearPath`. `SetInvariants(IPos,FPos)` que permite generar trayectorias en línea recta entre dos puntos, para ello se envían las matrices de pose del manipulador en la posición inicial y final de la trayectoria a generar. La función GetPos devuelve la matriz de una pose intermedia entre la posición inicial y final de la trayectoria previamente generada mediante `SetInvariants`, enviando un valor entre 0 y 1. Donde 0 representa la posición inicial de la trayectoria y 1 la posición final.

Ya cargados los datos se revisa, que el manipulador esté en la posición Home, de no estar en la posición Home, se cambia el estado del manipulador a RESET, de este modo se genera una trayectoria a seguir entre la posición actual y la posicion HOME mediante la función `LinePath.SetInvariants`.

Se inicializa el temporizador `Time`, éste indica el tiempo que tendrá que transcurrir para ir de la posición inicial a la posición final. También se inicia el contador `PC`(Poses Counter), el cual lleva la cuenta de las poses intermedias. Enseguida se revisa que el tiempo transcurrido sea igual al tiempo neccesario para alcanzar la siguiente pose intermedia, la variable `PN` queda definida como un macro el cual representa el número total de poses intermedias.

Después mediante `GetPos` se obtiene la pose intermedia correspondiente a la proporción entre `PC` y `PN`. Obtenida esta posición intermedia se genera la solución de cinemática inversa mediante la función `SetInversePosition`, esto para conocer las variables de articulación del manipulador en la nueva posición alcanzada.

Mediante `RefreshDisplay()` se muestra en pantalla el manipulador con la pose actualizada, este proceso se repite hasta alcanzar la posición deseada. Por último se revisa que el tiempo transcurrido sea igual al temporizador Time, definido para ir de la posición inicial arbitraria a HOME.

El proceso que se sigue para ir de la posición arbitraria a HOME, es el mismpo para ir de la posición HOME a la posición A, y de la posición A a la posición B, claro esta, el estado del manipulador (RESET, PIN y POUT) se modificará de acuerdo a la posición a alcanzar.

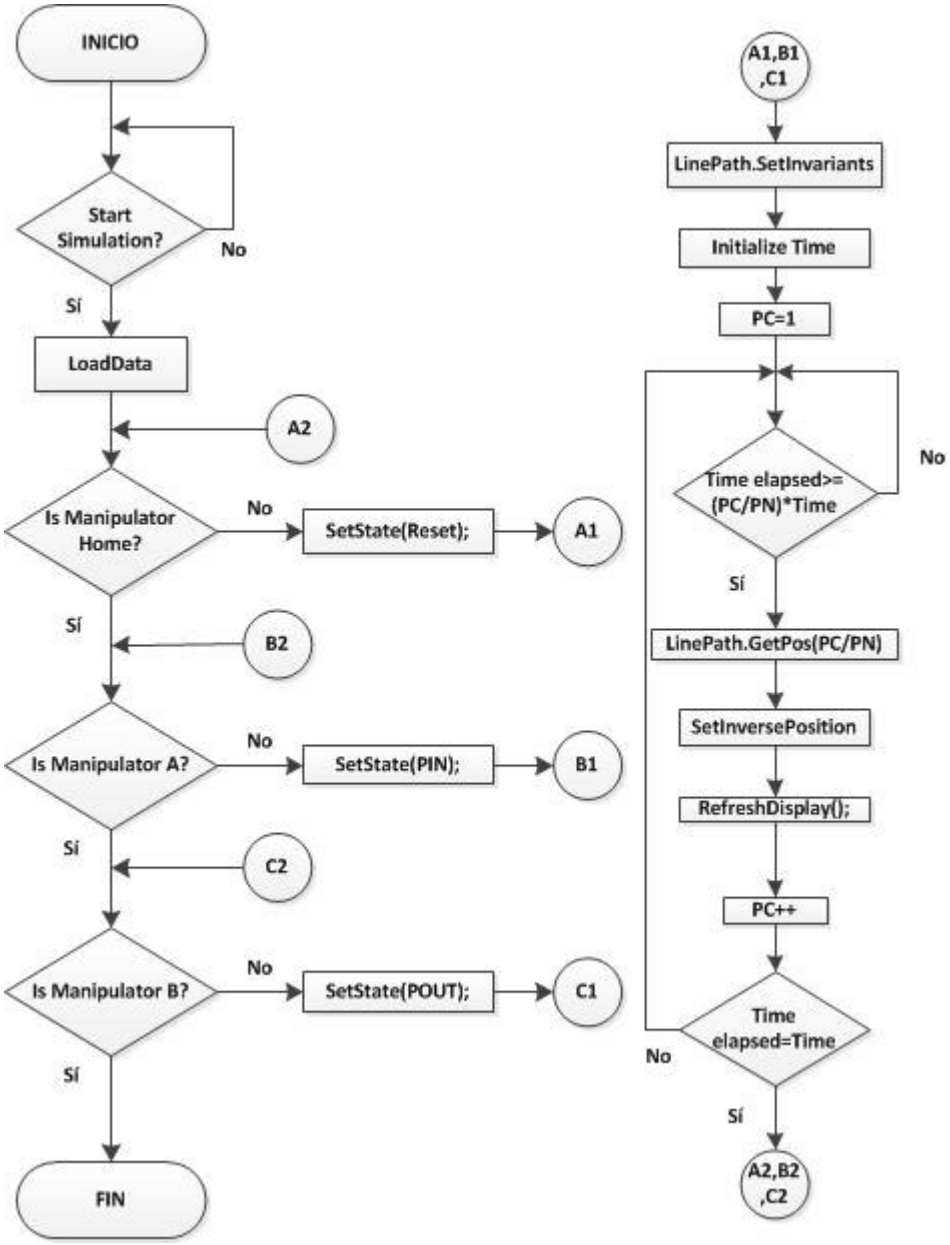

Figura 4.1: Diagrama de flujo del control de la simulación.

4.2. Código implementado

El código 4.1 que realiza las operaciones de control representadas en el diagrama de flujo, trabaja de la misma manera, pero se organiza mediante casos. La mayor variación entre el diagrama de flujo y el código implementado es cuando el manipulador esta en el estado `POUT` puesto que en este punto del control es donde se envían todos los puntos de la trayectoria adquirida y se van representando uno a la vez.

Código 4.1: Control de la simulación.

```
if (m_State != SUSPEND)
                switch (m_IP)
                {
                        //-----------------------------------------
                        case ERROR_IP:
                                SetState(SUSPEND);
                                SetMode(ERRORSTOP);
                                break;

                        //---------------- reset procedure -------------
                        case RESET_IP:
                                m_posesCounter=0;
                                m_IP++;
                                break;

                        case RESET_IP+1:
                                Timer.InitTimer(2000);
                                ++m_posesCounter;
                                m_IP++;
                                break;

                        case RESET_IP+2:
                                if(Timer.GetTime()>= m_posesCounter*Timer.m_period/m_posesNumber)
                                {
                                Matrix pos=LinePath.GetPos((float)m_posesCounter/(float)m_posesNumber);
                                        m_TransfMatrix=pos;
                                        CartPoint p;
                                        p.x=pos.q[0][3];
                                        p.y=pos.q[1][3];
                                        p.z=pos.q[2][3];
                                        SetInversePosition(&Dh4,p);
                                        m_posesCounter++;

                                }
                                if(Timer.GetTime()>=Timer.m_period)
                                        m_IP++;
                                break;
                        case RESET_IP+3:
                                m_posesCounter=0;
                                IsHome=TRUE;
                                SetState(SUSPEND);
                                if(GetMode()==AUTOMATIC)
                                pADEFIDDoc->IsHtoAOn=TRUE;
                                break;
                        //------------ transfer in procedure -----------------
                        case IN_IP:
                                k=0;
                                SetLinePath(HomePose,NPose,pADEFIDDoc->Points[k]);
                                k++;
                                m_IP++;
                                break;
                        case IN_IP+1:
                                Timer.InitTimer(1500);
                                ++m_posesCounter;
                                m_IP++;
                                break;
                        case IN_IP+2:
                                if(Timer.GetTime()>= m_posesCounter*Timer.m_period/m_posesNumber)
                                {

                                Matrix pos=LinePath.GetPos((float)m_posesCounter/(float)m_posesNumber);
                                        CartPoint p;
                                        p.x=pos.q[0][3];
                                        p.y=pos.q[1][3];
                                        p.z=pos.q[2][3];
                                        SetInversePosition(&Dh4,p);
                                        m_posesCounter++;
                                }
                                if(Timer.GetTime()>=Timer.m_period)
                                        m_IP++;
```

```
71                              break;
72
73                      case IN_IP + 3:
74                              SetDelay(300,m_IP+1);
75                              break;
76
77                      case IN_IP + 4:
78                              m_posesCounter=0;
79                              IsHome=FALSE;
80                              SetState(SUSPEND);
81                              if(GetMode()==AUTOMATIC)
82                                      pADEFIDDoc->IsAtoBOn=TRUE;
83                              break;
84
85                      //------------ transfer out procedure ---------
86                      case OUT_IP:
87                              SetLinePath(APose,NPose,pADEFIDDoc->Points[k]);
88                              k++;
89                              m_IP++;
90                              break;
91                      case OUT_IP + 1:
92                              Timer.InitTimer(41);
93                              ++m_posesCounter;
94                              m_IP++;
95                              break;
96                      case OUT_IP+2:
97                              if(Timer.GetTime()>=Timer.m_period)
98                              {
99                                      Matrix pos=LinePath.GetPos((float)1);
100                                     m_TransfMatrix=pos;
101                                     CartPoint p;
102                                     p.x=pos.q[0][3];
103                                     p.y=pos.q[1][3];
104                                     p.z=pos.q[2][3];
105                                     SetInversePosition(&Dh4,p);
106                                     m_posesCounter++;
107                             }
108                             if(Timer.GetTime()>=Timer.m_period)
109                                     m_IP++;
110                             break;
111
112                     case OUT_IP+3:
113                             m_IP++;
114                             break;
115
116                     case OUT_IP+4:
117                             m_posesCounter=0;
118                             SetState(SUSPEND);
119
120                             if(k>=pADEFIDDoc->j)
121                             {
122                                     if(GetMode()==AUTOMATIC)
123                                     pADEFIDDoc->IsResetOn=TRUE;
124                             }
125                             else
126                             {
127                                     pADEFIDDoc->IsAtoBOn=TRUE;
128                             }
129                             break;
130                     //-------------------- delay-------------------------
131                     case DELAY_IP:
132                             Delay();
133                             break;
134
135                     //-------------------- scanning ---------------------
136                     case SCANNING_IP:
137                             ScanSensor();
138                             break;
139
140                     //------------ default ----------------
141                     default:
142                             break;
143             }
```

4.3. Declaración de nuevas máquinas

Hasta este punto se ha descrito el algoritmo desarrollado para la simulación del seguimiento de la trayectoria del prototipo genérico, pero para realizar la comparativa entre distintos manipuladores es necesario declarar nuevas máquinas, en esta sección se describe cómo es que se declaran estas máquinas dentro del proyecto.

De nueva cuenta, puesto que ADEFID se deriva de una plantilla de una aplicación MFC, será necesario realizar modificaciones en ADEFIDDoc y ADEFIDRender.

Los primero que se tiene que hacer es declarar una nueva clase que definirá a la nueva máquina (nuevo manipulador), esta clase tendrá como clase base `CManipulator`, tal y como se muestra en la imagen 4.2, para la declaración de la clase para el manipulador SCARA.

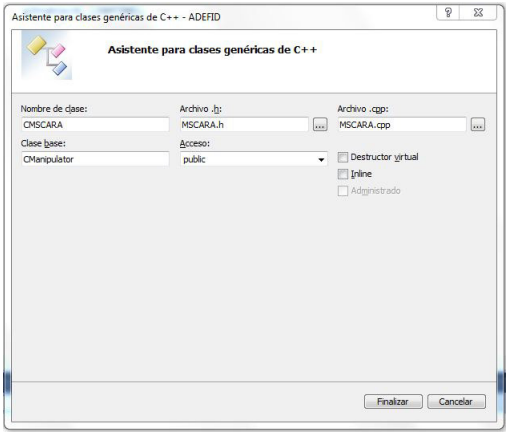

Figura 4.2: Creación de la clase `CMSCARA`.

En la nueva clase se agregarán las funciones `void SetDefaults(void)` y `BOOL SetInversePosition(D_H* pDH, CartPoint point)`. Esto puede hacerse de manera manual o utilizando el asistente como se muestra en la imagen 4.3, para agregar la función `SetInversePosition`.

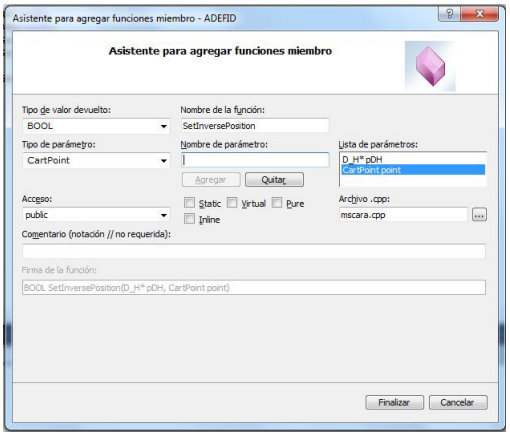

Figura 4.3: Creación de la función `CMSCARA`.

A continuación se agregan las líneas de código correspondientes a cada una de las funciones. Como su nombre lo indica en `SetDefaults`, se incluirán los valores predeterminados para el manipulador. Para la función `SetInversePosition` el código implementado surge del análisis de cinemática inversa del manipulador, dicho análisis se encuentra en el apéndice A, para todos los manipuladores propuestos. En el código 4.2 se muestra el código implementado para el manipulador SCARA.

Código 4.2: Código de la función `SetInversePosition` para el manipulador SCARA.

```
BOOL CMSCARA::SetInversePosition(D_H* pDH, CartPoint point)
{
        double r, c, d;
        pDH->b[2]=pDH->b[0]-point.z;
        c=sqrt(point.x*point.x+point.y*point.y);
        d=(c*c-pDH->a[0]*pDH->a[0]-pDH->a[1]*pDH->a[1])/(2*pDH->a[0]*pDH->a[1]);
        pDH->theta[1]=atan2(sqrt(1-d*d),d);
        pDH->theta[0]=atan2(point.y,point.x)-atan2(pDH->a[1]*sin(pDH->theta[1]),pDH->a[0]+pDH->a[1]*cos(pDH
       ->theta[1]));
        return TRUE;
}
```

Los códigos correspondientes a los demás manipuladores propuestos se encuentran en el Apéndice B.

Hasta el momento se ha generado la clase para el nuevo manipulador, el siguiente paso es declarar el objeto dentro de `ADEFIDDoc`, para ello agregamos en `ADEFIDDoc.h` la cabecera recién creada, en el caso del manipulador SCARA se incluye `#include "MSCARA.h"`. El nuevo objeto es declarado agregando `CMSCARA MSCARA;` en la sección `public`.

Ahora ya se tiene la nueva máquina declarada, el siguiente paso es inicializar dicha máquina, para ello modificamos la función `CADEFIDDoc::OnInitMachines` como se muestra en el código 4.3.

Código 4.3: Código `OnInitMachines`

```
BOOL CADEFIDDoc::OnInitMachines(void)
{
        if(!ADEFIDGlobal.Initialization(m_pDoc, "System") ||
                !CustomManip.Initialization(m_pDoc,"Custom")||
                !MSCARA.Initialization(m_pDoc,"SCARA",ROBOT1))
                return false;

        pManipulator = new CManipulator;
        pManipulator = &CustomManip;
        return true;
}
```

Como se puede observar, para inicializar una nueva máquina sólo es necesario agregar el código `|| !MSCARA.Initialization(m_pDoc,"SCARA",ROBOT1)`, para ello se modifica de que objeto proviene la inicialización, así como el nombre y el número de robot. También es necesario incluir el Algoritmo del nuevo manipulador, esto se hace dentro de la función `MaterialHandlingProcess`, para el manipulador SCARA se agrega `MSCARA.Algorithm();`, después de la línea `pManipulator->Algorithm();`.

Con esto se terminan las modificaciones necesarias en el Documento para la inicialización de una nueva máquina, ahora es necesario realizar las modificaciones pertinentes en Vista para que se muestre el nuevo manipulador.

Las modificaciones de Vista se realizan en `ADEFIDRender.cpp`. Primero se tiene que agregar el código para que el manipulador sea dibujado y se muestre en pantalla cuando el programa es inicializado, esto mediante la función `Draw_Links`, para ello agregamos la línea correspondiente en la sección nombrada `//New Manipulators`, tal como se muestra en el código 4.4.

Código 4.4: Código `SetUpScene`

```
void CADEFIDRender::SetupScene(void)
{
        pDoc->pManipulator->DefineGripper();
        pDoc->pManipulator->Draw_Links(&pDoc->pManipulator->Dh4);

        //New Manipulators
        pDoc->MSCARA.Draw_Links(&pDoc->MSCARA.Dh4);

        DrawTarget();
}
```

También dentro de la función `RenderUScene`, se agregará la función `Draw_Chain`, esto como se muestra en el código

Código 4.5: Código `RenderUScene`

```
BOOL CADEFIDRender::RenderUScene(void)
{
        ...
        pDoc->pManipulator->Draw_Chain(&pDoc->pManipulator->Dh4);
        pDoc->MSCARA.Draw_Chain(&pDoc->MSCARA.Dh4);
        ...
}
```

La función `RenderUScene` contiene más elementos que permiten modificar la escena en pantalla, tales como mostrar mensajes, dibujar la trayectoria seguida por el manipulador, mostrar ejes coordenados y planos cartesianos.

Hasta aquí se ha mostrado como crear un nuevo manipulador, inicializarlo y mostrarlo en pantalla. Ahora es necesario crear condicionales para la impresión y simulación en pantalla del nuevo manipulador. Estas condicionales serán controladas de manera interna en los menús de la simulación.

4.4. Modificación de parámetros del manipulador.

Ya que se tiene la capacidad de inicializar nuevas máquinas y realizar su simulación, resulta necesario modificar los parámetros de los manipuladores propuestos, para ello se crean diálogos. A continuación se muestra cómo se generan y funcionan dichos diálogos.

El procedimiento para la creación, activación y asociación diálogos es el mismo que fue descrito en la sección 3.3.1. Lo primero que se hace es agregar el nuevo diálogo, de este modo se tendrá una nueva plantilla. A esta nueva plantilla se le cambia el nombre y se le agrega una clase.

Es necesario crear un manejador de eventos para asociar el menú al diálogo recién creado, los diálogos de modificación de parámetros de los manipuladores son del tipo modal, lo que significa que no prodrán realizarse otras acciones hasta cerrar el diálogo.

Los diálogos para la modificación de parámetros se asocian al menú "SetUp", tal y como se muestra en la imagen 4.4.

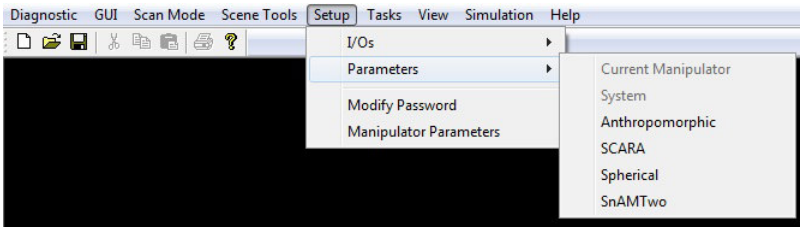

Figura 4.4: Menú Parameters.

Hasta este punto se dió un breve resumen de el procedimiento previamente descrito en 3.3.1. Ahora se desean establecer las funciones del diálogo, que el diálogo modifique los parámetros de los manipuladores.

Para los diálogos de la modificación de parámetros se agregan cuadros de edición y barras deslizantes, así como un botón para aplicar los cambios, como el diálogo mostrado en la imagen 4.5 para los parámetros del manipulador SCARA. Los diálogos creados para los demás manipuladores son similares.

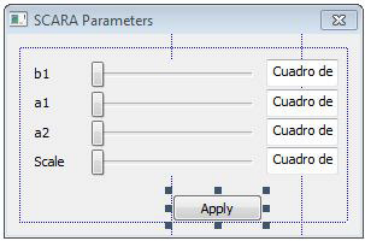

Figura 4.5: Diálogo de parámetros SCARA.

Al diálogo creado para los parámetros del manipulador SCARA fueron agregados 9 elementos, a los cuales se les asigna un nombre que permita identificarlos de manera sencilla. Por ejemplo para el parámetro B1 se eligió `IDC_SLIDER_B1` para la barra deslizante y `IDC_EDIT_B1` para el cuadro de edición.

También es necesario agregar un controlador para cada elemento, todos con el tipo de mensaje `NM_CUSTOMDRAW` y con la clase correspondiente a la creada

para el diálogo. Enseguida se agregan variables a cada elemento, nuevamente, el nombre de las variables se selecciona buscando que permita identificarlos de manera sencilla.

Para que los cuadros de edición muestren el valor correspondiente a la posición de la barra deslizante es necesario agregar el código 4.6 en el controlador de eventos previamente creado. También se agrega el código necesario para que se actualice el parámetro HD modificado.

Código 4.6: Código de manejador de eventos de barra deslizante de b1

```
void CParamSCARA::OnNMCustomdrawSliderB1(NMHDR *pNMHDR, LRESULT *
    pResult)
{
        LPNMCUSTOMDRAW pNMCD = reinterpret_cast<LPNMCUSTOMDRAW>(
    pNMHDR);
        //Codigo agregado para que el cuadro de edicion muestre el
        //valor de la barra deslizante
        B1 = (float)(slider_b1.GetPos()/100.);
        //Codigo para que el valor presentado en el dialogo
        //se actualice en los parametros HD de la maquina
        pDoc->MSCARA.Dh4.b[0]=B1;
        pDoc->MSCARA.Link[0].m_b=B1;
        pDoc->MSCARA.Link[0].DrawLink(pDoc->MSCARA.m_BaseDiam);
        UpdateData(FALSE);
        *pResult = 0;
}
```

De manera inversa, para que la barra deslizante se posicione en el lugar correspondiente al valor ingresado en el cuadro de edición se agrega el código 4.7 al controlador de eventos.

Código 4.7: Código de manejador de eventos de cuadro de edición de b1

```
void CParamSCARA::OnEnChangeEditB1()
{
        UpdateData(TRUE);
        //Codigo agregado para que la barra deslizante
        //se posicione en el lugar correspondiente al
        //valor ingresado en el cuadro de edicion
        slider_b1.SetPos((int)(B1*100.));
        //Codigo para que el valor presentado en el cuadro de
        //edicion se actualice en los parametros HD de
        //la maquina
        pDoc->MSCARA.Dh4.b[0]=B1;
        pDoc->MSCARA.Link[0].m_b=B1;
        pDoc->MSCARA.Link[0].DrawLink(pDoc->MSCARA.m_BaseDiam);
}
```

Para que al inicializar el diálogo las barras deslizantes y los cuadros de edición muestren los parámetros HD del manipulador se debe de ir al la vista de clases y en la ventana de propiedades, en la sección de invalidaciones se busca la opción "`OnInitDialog`", como se muestra en la imagen 4.6. Al activar la opción se

agregará un código donde se dan los valores iniciales para los elementos del diálogo como se muestra en el código 4.8.

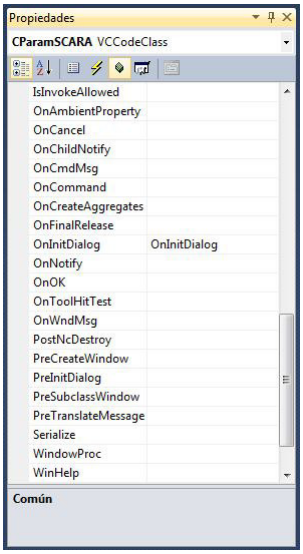

Figura 4.6: Activación de la opción `OnInitDialog`.

Código 4.8: Código para la inicialización de diálogo de parámetros

```
BOOL CParamSCARA::OnInitDialog()
{
        CDialog::OnInitDialog();

        // TODO:  Agregue aqui la inicializacion adicional
        B1=pDoc->MSCARA.Dh4.b[0];
        A1=pDoc->MSCARA.Dh4.a[0];
        A2=pDoc->MSCARA.Dh4.a[1];
        SCALE=pDoc->MSCARA.m_ScaleFactor;
        slider_b1.SetPos((int)(B1*100.));
        slider_a1.SetPos((int)(A1*100.));
        slider_a2.SetPos((int)(A2*100.));
        slider_scale.SetPos((int)(SCALE*100.));
        UpdateData(FALSE);

        return TRUE;
}
```

Ahora bien para que los cambios realizados sean conservados después de cerrar la aplicación se creo el botón "Apply", el código 4.9 implementado en el controlador de eventos de dicho elemento permite guardar los cambios para la próxima ejecución de la aplicación.

Código 4.9: Código para guardar los cambios después del cierre de la aplicación

```
void CParamSCARA::OnBnClickedButtonApply()
{
        // TODO: Agregue aqui su codigo de controlador
        //de notificacion de control
        pDoc->pMainView->SetupScene();
        pDoc->MSCARA.Link[0].SaveGeometryValues();
        pDoc->MSCARA.Link[1].SaveGeometryValues();
        pDoc->MSCARA.Link[2].SaveGeometryValues();
        pDoc->MSCARA.SetBaseLength(&pDoc->MSCARA.Dh4);
        pDoc->MSCARA.SaveGeometryValues();
}
```

Por último, para que los cambios realizados en el diálogo se muestren de manera instantánea, se agrega un temporizador de la clase `CTimer TimerRefresh;` y se agrega el código 4.10 al final de la función `CADEFIDDoc::MaterialHandlingProcess()`.

Código 4.10: Código para la actualización de impresión en pantalla

```
if(TimerRefresh.IsExpired())
        {
                pMainView->refreshDisplay();
                //m_running_time.Format(_T("  T=%d ms"),
    m_cycleTimer.GetTime());
                TimerRefresh.InitTimer(41);
        }
```

Ahora el diálogo es completamente funcional, modifica los parámetros HD del manipulador y los cambios son presentados de manera instantánea.

4.5. Metodología de empleo

Para ejecutar la simulación, ya inicializado el programa se va al menú "Simulation" y se elige una de las opciones. "Reset", para que el manipulador se posicione en HOME. "SnAM" para que se simule únicamente el prototipo genérico. "Anthropomorphic" para que se simule el manipulador antropomórfico además del prototipo genérico. Y así para cada opción.

Al presionar la opción "simulation" la simulación empezará a correr, y la trayectoria se marcará por una serie de puntos como la mostrada en la imagen 4.8, para el prototipo genérico.

Cuando se presiona "simulation" de cualquiera de los otros manipuladores disponibles, se mostrará de manera simultánea, la simulación del manipulador seleccionado y la del prototipo genérico. Como se muestra en la imagen 4.9, para los manipuladores Antropomórfico, SCARA y Esférico.

El ejecutar la simulación es muy sencillo. Resulta obvio que para ejecutar la simulación es necesario previamente haber adquirido la trayectoria, tal y como se describió en el capítulo anterior y contar con el archivo position.txt.

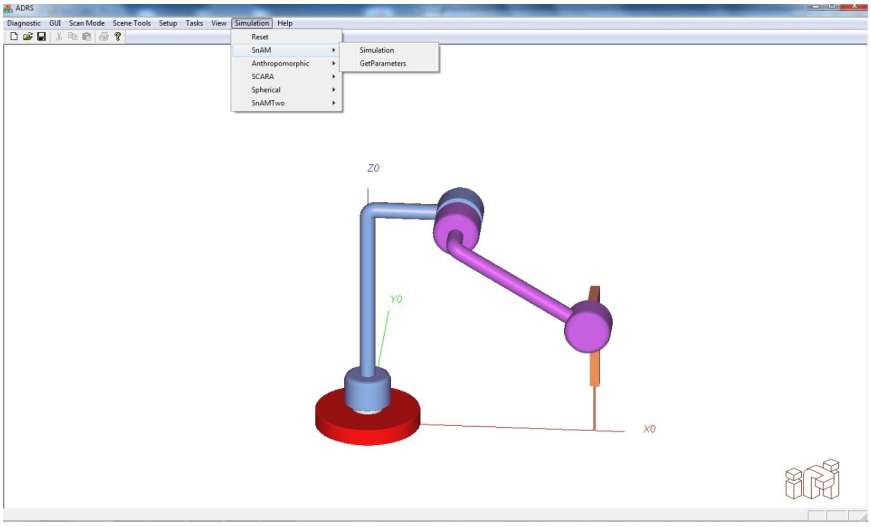

Figura 4.7: Inicialización de la simulación.

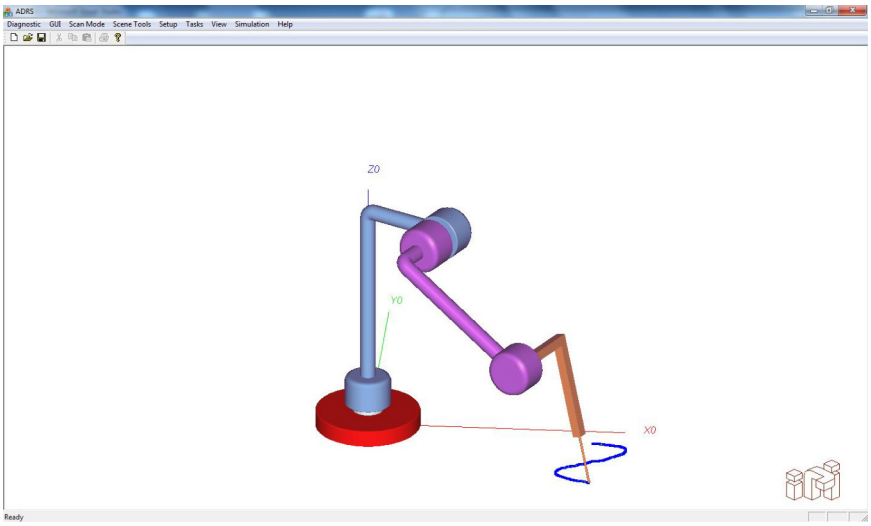

Figura 4.8: Simulación de una trayectoria sinusoidal mediante prototipo genérico.

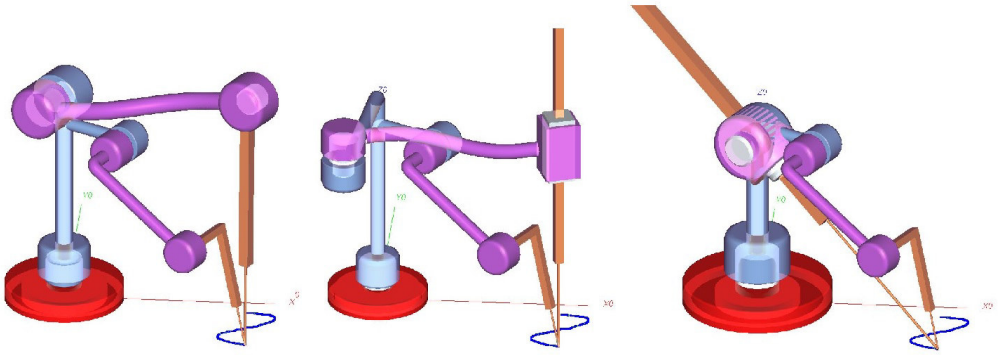

Figura 4.9: Simulación de una trayectoria sinusoidal mediante distintos manipuladores.

Si bien, la simulación de las trayectorias mediante ADEFID, validan los resultados obtenidos. Resulta interesante hacer una comparación con los resultados que se pueden obtener mediante un software comercial, para ello en el apéndice C, se modelan los manipuladores propuestos en Adams.

Capítulo 5

Cálculo de velocidad y aceleración

En los capítulos anteriores se ha visto como adquirir la trayectoria seguida por el prototipo genérico, y cómo simular en pantalla dicha trayectoria. Pero a través de los cálculos de velocidad y aceleración se tendrá la posibilidad de realizar comparativas entre el comportamiento de distintos manipuladores. En este capítulo se analiza la metodología llevada a cabo para realizar dichos cálculos así como las herramientas que fueron utilizadas para la resolución de los problemas que se presentaron en el desarrollo.

5.1. Obtención de parámetros de velocidad y aceleración

Lo primero que se tiene que hacer es el análisis cinemático de velocidad y aceleración de los manipuladores propuestos, dicho análisis se encuentra en el Apéndice A. Con el análisis de cinemática y velocidad realizado, haciendo usos de los datos obtenidos en la adquisición se puede calcular la velocidad y aceleración de las articulaciones del manipulador.

Se recuerda que la cinemática de velocidad busca relacionar las velocidades lineales y angulares del efector final con las velocidades de articulación mediante la ecuación (5.1).

$$\dot{X} = J\dot{\mathbf{q}} \tag{5.1}$$

Derivando la ecuación (5.1), se obtiene la ecuación de aceleración:

$$\ddot{X} = J\ddot{\mathbf{q}} + \frac{d(J(\mathbf{q}))}{dt}\dot{\mathbf{q}} \tag{5.2}$$

En esta ocasión se realiza el proceso inverso, conocida la velocidad del efector final se busca obtener la velocidad de las articulaciones. Para ello se hace un

reacomodo de la ecuación (5.1), despejando $\dot{\mathbf{q}}$:

$$\boxed{\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{X}} \tag{5.3}$$

Despejando $\ddot{\mathbf{q}}$ de la ecuación (5.2), se obtiene:

$$\boxed{\ddot{\mathbf{q}} = J^{-1}(\mathbf{q})\mathbf{b}} \tag{5.4}$$

Donde $\mathbf{b}$ equivale a:

$$\mathbf{b} = \ddot{X} - \frac{d(J(\mathbf{q}))}{dt}\dot{\mathbf{q}} \tag{5.5}$$

Donde $\dot{\mathbf{q}}$ y $\ddot{\mathbf{q}}$ representan al vector velocidad y aceleración de las articulaciones del manipulador. Mientras $\dot{X}$ y $\ddot{X}$ representan a la velocidad y aceleración del efector final.

Para hacer uso de las ecuaciones (5.3) y (5.4) es necesario calcular la velocidad y aceleración lineal del efector final. Para realizar estos cálculos se cuenta con los datos que representan la trayectoria seguida por el manipulador, esto es la serie de puntos que caracterizan a la trayectoria y el tiempo en que fueron alcanzados. A partir de esta información se pueden utilizar diferencias finitas, para obtener los datos deseados.

5.2. Derivación numérica

La derivación numérica permite calcular la aproximación de la derivada de una función en un punto utilizando los valores de la misma. En este caso dichos valores son los datos que representan a la trayectoria adquirida. Al tratarse de una aproximación el hacer uso de éste método implicará un error numérico.

Existen tres tipos de diferencia, conocidas como diferencia hacia adelante, hacia atrás y central. Reciben su nombre debido a los datos que usan. Aquí se aplicará la diferencia central que utiliza los datos $i-1$ e $i+1$. La aproximación a la primera y segunda derivada con diferencias centradas se muestra en las siguientes ecuaciones:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \tag{5.6}$$

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} \tag{5.7}$$

Donde $f(x_{i+1})$ es el valor del siguiente punto y $f(x_{i-1})$ el valor del punto previo. h es llamado tamaño de paso o incremento, en otras palabras es el espaciado que hay entre cada punto.

Resulta simple implementar este método cómo se muestra en el código 5.1. Se utiliza el mismo ciclo para el cálculo de la velocidad y aceleracion en x, y y z. Donde j representa al número de datos que representan a la trayectoria. Los arreglos bidimensionales `pos[][]`, `vel[][]` y `acel[][]` comparten el mismo formato en donde el primer índice respresenta el número de punto. El segundo

refiere a 0 para los valores de x, 1 para los valores de y y 2 para los valores de z. h queda definido como el periodo de adquisición.

Código 5.1: Código para el cálculo de velocidad y aceleración del efector final

```
for(i=0;i<j;i++)
{
                    //Diferencia finita centrada
          vel[i][0]=(pos[i+1][0]-pos[i-1][0])/(2*h);
          vel[i][1]=(pos[i+1][1]-pos[i-1][1])/(2*h);
          vel[i][2]=(pos[i+1][2]-pos[i-1][2])/(2*h);

          acel[i][0]=(pos[i+1][0]-2*pos[i][0]+pos[i-1][0])/(h*h);
          acel[i][1]=(pos[i+1][1]-2*pos[i][1]+pos[i-1][1])/(h*h);
          acel[i][2]=(pos[i+1][2]-2*pos[i][2]+pos[i-1][2])/(h*h);
}
```

Se podría creer que a partir de este punto se esta listo para hacer uso de las ecuaciones (5.3) y (5.4). Pero debido a las propiedades de este método numérico es imposible como se muestra a continuación.

Como se mencionó al principio de esta sección, este procedimiento está diseñado para determinar la derivada de una función dada, pero para nuestro caso, los datos han sido adquiridos manualmente y no representan a ninguna función.

Otro requerimiento de este método es que los datos están igualmente espaciados. Aunque el periodo de adquisición es el mismo para todos los puntos, al ser datos empíricos (datos adquiridos manualmente), se presentan errores de medición los cuales tienden a amplificarse.

Por lo tanto, los datos adquiridos que representan a la trayectoria del manipulador no son adecuados para la aplicación del método de diferencias finitas, previo a la aplicación de este método es necesario ajustar los datos a una función suave y diferenciable, esto mediante regresión de mínimos cuadrados [12].

5.3. Regresión por mínimos cuadrados

En esta técnica, se intenta encontrar la función que mejor se aproxime a los datos, de acuerdo al mínimo error cuadrático.

Se busca que una serie de m datos se aproxime a una función de grado n. Consideré para ello la función 5.8.

$$y = a_0 + a_1 x + \ldots + a_n x^n + e \tag{5.8}$$

Donde e representa el error entre los datos y la función. Ahora si se hace la suma del cuadrado de los errores.

$$S_r = \sum_{i=1}^{n} (y - a_0 - a_1 x - \ldots - a_n x^n)^2 \tag{5.9}$$

Para determinar los valores de $a_o, \ldots, a_n$ es necesario derivar con respecto a cada uno de los coeficientes.

$$\begin{aligned}
\frac{\partial S_r}{\partial a_0} &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n) \\
\frac{\partial S_r}{\partial a_1} &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n)x \\
&\vdots \\
\frac{\partial S_r}{\partial a_n} &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n)x^n
\end{aligned}$$

Se igualan las derivadas a cero para tener un error S_r mínimo.

$$\begin{aligned}
0 &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n) \\
0 &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n)x \\
&\vdots \\
0 &= -2\sum_{i=1}^{n}(y - a_0 - a_1x - \ldots - a_nx^n)x^n
\end{aligned}$$

Si se observa $\sum a_0 = ma_0$ y se hace un reacomodo, se tiene:

$$\begin{aligned}
ma_0 + \left(\sum x\right)a_1 + \ldots + \left(\sum x^n\right)a_n &= \sum y \\
\left(\sum x\right)a_0 + \left(\sum x^2\right)a_1 + \ldots + \left(\sum x^{n+1}\right)a_n &= \sum xy \\
&\vdots \\
\left(\sum x^{n+1}\right)a_0 + \left(\sum x^{n+2}\right)a_1 + \ldots + \left(\sum x^{n+n}\right)a_n &= \sum x^n y
\end{aligned}$$

Este sistema de ecuaciones lineales se puede escribir matricialmente como:

$$\begin{bmatrix}
m & \sum x & \sum x^2 & \ldots & \sum x^n \\
\sum x & \sum x^2 & \sum x^3 & \ldots & \sum x^{n+1} \\
\sum x^2 & \sum x^3 & \sum x^4 & \ldots & \sum x^{n+2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\sum x^{n+1} & \sum x^{n+2} & \sum x^{n+3} & \ldots & \sum x^{n+n}
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} =
\begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2y \\ \vdots \\ \sum x^n y \end{bmatrix} \tag{5.10}$$

Este sistema puede ser resuelto por el método de Gauss-Jordan. Planteado el problema se busca aproximar la serie de puntos adquiridos a una función diferenciable. Para esté propósito se creó la clase `CLSquaresFit`, esta clase cuenta

con una función que devuelve los coeficientes a de la función aproximada a la serie de puntos. Su funcionamiento es el mostrado en el digrama de flujo de la imagen 5.1.

Para la regresión por mínimos cuadrados, primero se recibe la serie de datos que representan a la trayectoria, esto es la posición en x, y o z, así como el tiempo t en que dicha posición fue alcanzada. A partir de esta serie de puntos se genera el sistema de ecuaciones de acuerdo al grado indicado. Después se resuelve el sistema mediante Gauss-Jordan. Por último los coeficientes obtenidos se guardan en un vector, para que posteriormente los valores trayectoria sean ajustados a la función.

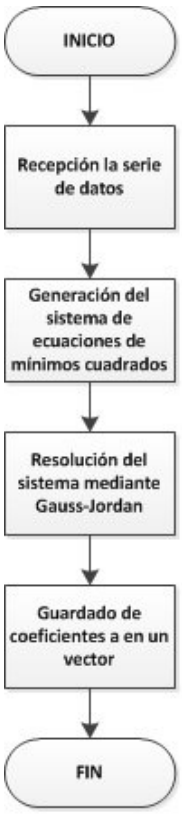

Figura 5.1: Funcionamento de la clase `CLSquaresFit::Operation`.

El código 5.2 muestra el mismo funcionamiento que el diagrama de flujo y como se indica, se divide en tres bloques que son la construcción del sistema de ecuaciones, la solución del sistema mediante Gauss-Jordan y el guardado de los coeficientes.

Código 5.2: Código para el ajuste por mínimos cuadrados

```
void CLSquaresFit::operation(double x[300],double y[300],int m, int
     n)
{

        //Construccion del sistema de ecuaciones

         for(i=0;i<=n;i++)
      {
               for(j=0;j<=n+1;j++)
               {
                                if(j>=0 && j<=n)
                                {
                                      for(k=0;k<m;k++)
                                      {
                                       mat[i][j]+=pow(x[k],i+j);
                                      }
                                }
                                else if(i==0 && j==0)
                                {
                                         mat[i][j]=m;
                                }
                                else if(j==n+1)
                                {
                                      for(k=0;k<m;k++)
                                      {
                                       mat[i][j]+=y[k]*pow(x[k],i);
                                      }
                                }
               }
      }

      //Solucion del sistema de ecuaciones mediante Gauss-Jordan

      for(i=0; i<=n; i++)
      {
                        aux=mat[i][i];
                        for(j=0;j<=n+1;j++)
                        {
                                              mat[i][j]/=aux;
                        }
                        for(k=0;k<=n;k++)
                        {
                              aux2=mat[k][i];
                              if(k==i)
                              {
                              }
                              else
                              {
                              for(l=0;l<=n+1;l++)
                              {
                               mat[k][l]=mat[k][l]-(aux2*mat[i][l]);
                              }
                              }
                        }
      }
```

```
55
56          //Guardado de coeficientes
57          for(j=0;j<=n+1;j++)
58                                  {
59                                                          a[j]=mat[j][n+1];
60                                  }
61  }
```

Con la técnica de regresión por mínimos cuadrados ahora se puede aproximar a una función la serie de puntos adquiridos, para así ajustarlos a una función diferenciable y poder aplicar el método de diferencias finitas, para que posteriormente se realice el cálculo de velocidad y aceleración de las articulaciones.

5.4. Creación de la función "GetPoVeAcParameters"

Resueltos los problemas de acondicionamiento de la serie de puntos que representan la trayectoria y obtenidas la velocidad y aceleración del efector final, ahora sí es posible implementar las ecuaciones (5.3) y (5.4).

A partir de este punto con el análisis cinématico de los manipuladores propuestos en el Apéndice A resulta relativamente sencillo el generar una función que realice los cálculos de velocidad y acerleración para las articulaciones del manipulador. La función creada, es `GetPoVeAcParameters`, propia de cada máquina, a partir de la serie de puntos, esta función desarrolla por completo el análisis cinemático de los manipuladores propuestos. Los datos calculados por la función son entregados en un archivo de texto para su tratamiento posterior.

El funcionamiento de `GetPoVeAcParameters` como se muestra en el diagrama de flujo de la imagen 5.2, se parte de la lectura de la serie de puntos que representan a la trayectoria y a estos puntos se les da el tratamiento que se describió en las secciones previas, para acondicionar los datos, estos se ajustan a una función mediante mínimos cuadrados.

Enseguida a partir de los datos ya ajustados se aplica el método de diferencias finitas, esto para calcular la velocidad y aceleración del efector final. Y con el propósito de implementar las ecuaciones (5.3) y (5.4) se calcula el Jacobiano y Jacobiano inverso. Completado este proceso se calculan la velocidad y aceleración de las articulaciones. Por último para darle un tratamiento posterior fuera de ADEFID, los resultados son impresos en formato de texto.

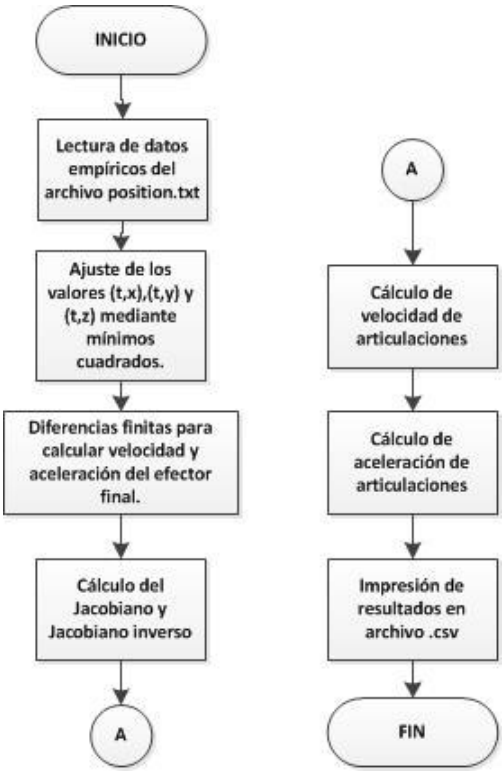

Figura 5.2: Diagrama de flujo de la función `GetPoVeAcParameters`.

Las funciones `GetPoVeAcParameters` para cada uno de los manipuladores, aunque simples en funcionamiento, resultan extensas por la cantidad de elementos con los que cuentan los Jacobianos como se puede observar en el Apéndice A, por tal motivo los códigos se encuentran en el Apéndice B.

5.4.1. Metodología de empleo

El funcionamiento del `GetPoVeAcParameters` a través de la interfaz del usuario es con base en menús, dentro del menú "Simulation" en cada uno de los manipuladores de la lista se encuentra activa la opción "GetParameters" como se muestra en la imagen 5.3, al presionarla, internamente se activará la función `GetPoVeAcParameters` para el manipulador correspondiente y también para el prototipo genérico, esto es por que se pretende realizar una comparativa entre el manipulador propuesto y el manipulador con el que se realizó la captura de la trayectoria.

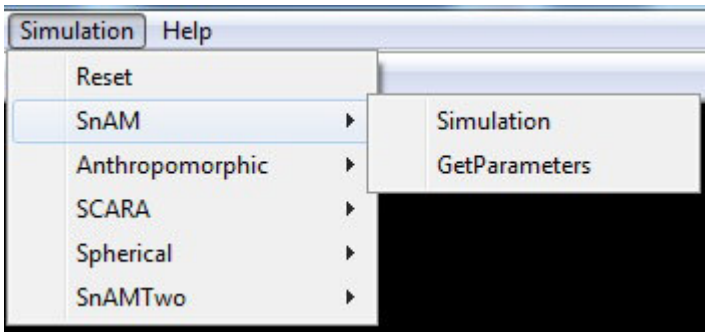

Figura 5.3: Opción `GetParameters` del menú `Simulation`

Una vez ejecutada la función, se generarán dos archivos de texto, los cuales tienen guardados todos los valores referentes a la trayectoria adquirida, correspondientes a los manipuladores a comparar. Los archivos son de la extensión "*.csv", este tipo de archivo se encuentra separado por comas, y es reconocido por excel. Cada conjunto de datos es separado por un cambio de renglón. Los archivos tienen el siguiente formato de guardado.

```
n, px, py, pz, th1, th2, th3, thp1, thp2, thp3, th2p1, th2p2, th2p3
```

Donde:

`n:` Número de conjunto de datos.

`px:` Posición del efector final en x.

`py:` Posición del efector final en y.

`pz:` Posición del efector final en z.

`th1:` Ángulo de la articulación 1.

`th2:` Ángulo de la articulación 2.

`th3,b3:` Ángulo o desplazamiento de la articulación 3.

`thp1:` Velocidad de la articulación 1.

`thp2:` Velocidad de la articulación 2.

`thp3,bp3:` Velocidad de la articulación 3.

`th2p1:` Aceleración de la articulación 1.

`th2p2:` Aceleración de la articulación 2.

`th2p3,b2p3:` Aceleración de la articulación 3.

5.5. Graficación de los datos obtenidos

Una forma de analizar cualitativamente los datos obtenidos por la función `GetPoVeAcParameters` es graficando, de este modo se podrá realizar una comparativa visual entre los comportamientos de los distintos parámetros de los manipuladores. Para graficar los datos se hace uso de matlab, el cual cuenta con las herramientas necesarias y son fáciles de usar.

Para gráficar lo primero que se tiene que hacer es cambiar la extensión del archivo de resultados, fue guardado con extensión `.csv` y ahora tendrá que ser guardado como `.dat` puesto que esta extensión es de la que hace uso matlab.

Ya en matlab se usa la función `csvread`, esta función lee los valores en formato de separación por comas (CSV) y los guarda en una matriz [13]. Para separar los valores son guardados en vectores, después estos vectores son graficados mediante la función `subplot`. El código 5.3 muestra la implementación para la graficación de las curvas de los parámetros de los manipuladores antropomórfico y SnAM.

Código 5.3: Código de graficación en matlab a partir de archivos `.csv`

```
clear all
clc
M1=csvread('resultados.dat');
M2=csvread('resultadosantropomorfico.dat');

th1=M1(:,5);
th2=M1(:,6);
th3=M1(:,7);
thp1=M1(:,8);
thp2=M1(:,9);
thp3=M1(:,10);
th2p1=M1(:,11);
th2p2=M1(:,12);
th2p3=M1(:,13);

th1a=M2(:,5);
th2a=M2(:,6);
th3a=M2(:,7);
thp1a=M2(:,8);
thp2a=M2(:,9);
thp3a=M2(:,10);
th2p1a=M2(:,11);
th2p2a=M2(:,12);
th2p3a=M2(:,13);

l=length(th1);

t=[0:0.04:(l-1)*0.04];
subplot(3,1,1);
plot(t,th3,'r-',t,th3a,'r:')
grid on
subplot(3,1,2);
plot(t,thp3,'g-',t,thp3a,'g.')
grid on
```

```
35 subplot(3,1,3);
36 plot(t,th2p3,'b-',t,th2p3a,'b.');
37 grid on
```

La imagen 5.4 es un ejemplo del funcionamiento del código 5.3.

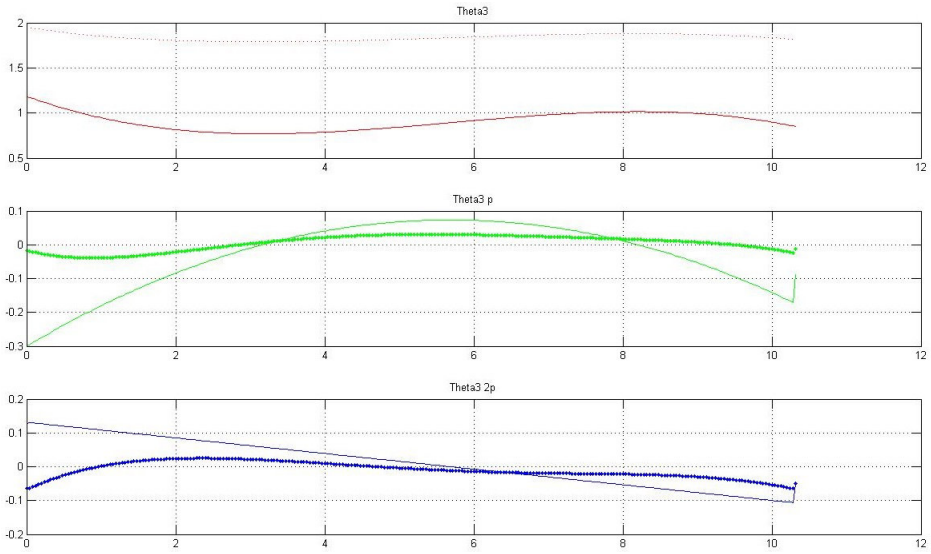

Figura 5.4: Graficación en matlab de parámetros de articulación.

Hasta este punto se ha mostrado el desarrollo de todas las herramientas para este proyecto así como el funcionamiento de las mismas. El siguiente paso es probar la operación del programa, en el siguiente capítulo se proponen trayectorias a seguir y se muestra el análisis desarrollado para las mismas.

Capítulo 6

Pruebas

Desarrolladas todas las herramientas necesarias para el análisis cinemático de los manipuladores propuestos, se está en condiciones para realizar pruebas del funcionamiento del sistema. Para la realización de estas pruebas se proponen varias trayectorias simples, una trayectoria circular, una trayectoria en rampa y una trayectoria sinusoidal.

Se recuerda que el manejo prototipo genérico utilizado para la adquisición es de carácter manual, por lo que seguir una trayectoria más de una vez con los mismos parámetros resulta imposible.

Para tener un patrón de adquisición en la trayectoria recta y circular, se hace uso de un manipulador cartesiano [9]. Dicho robot cuenta con un software propio, proporcionado por el fabricante que permite la generación de trayectorias simples (trayectorias rectas y circulares), haciendo uso de instrucciones sencillas.

A continuación se presentan los resultados de las pruebas realizadas con las distintas trayectorias y manipuladores propuestos y se hace un análisis cualitativo de los resultados. Cabe resaltar que se realizaron varias adquisiciones de la misma trayectoria, buscando que en lo posible éstas fueran lo más similares posible. Aunque aquí solo se presenta una serie de resultados, los demás se incluyen en el Apéndice D. Se da validez a los resultados de posicionamiento en el apéndice C, haciendo uso de Adams.

6.1. Parámetros H-D de los manipuladores propuestos.

Como ya se mencionó anteriormente, se proponen distintas arquitecturas para la comparativa con el prototipo genérico. En los cuadros 6.1-6.4, se muestran los distintos paramétros H-D de dichos manipuladores. Se recuerda que el análisis cinemático de estos manipuladores se encuentra en el Apéndice A.

i	θ_i	b_i	a_i	α_i
1	θ_1^*	0,5m	0	90
2	θ_2^*	0	0,4m	0
3	θ_3^*	0	0,3m	0

Cuadro 6.1: Parámetros H-D del manipulador antropomórfico

i	θ_i	b_i	a_i	α_i
1	θ_1^*	0,5m	0,4m	0
2	θ_2^*	0	0,4m	180
3	0	b_3^*	0	0

Cuadro 6.2: Parámetros H-D del manipulador SCARA

i	θ_i	b_i	a_i	α_i
1	θ_1^*	0,4m	0	90
2	θ_2^*	0	0	90
3	0	b_3^*	0	0

Cuadro 6.3: Parámetros H-D del manipulador esférico

i	θ_i	b_i	a_i	α_i
1	θ_1^*	0,43m	0,17m	90
2	θ_2^*	0,15m	0,28m	180
3	θ_3^*	0,15m	0,28m	0

Cuadro 6.4: Parámetros H-D del manipulador SnAM

6.2. Trayectoria de rampa.

La primer trayectoria a analizar será una rampa. Para esto se hace uso de un plano inclinado especialmente diseñado para este uso. Se hace la comparativa del prototipo genérico contra un manipulador antropormórfico, y después se compara con un manipulador esférico.

6.2.1. Prototipo genérico Vs. Manipulador antropomórfico.

Primeramente se observa en la figura 6.1 una captura de pantalla de la simulación de la trayectoria adquirida.

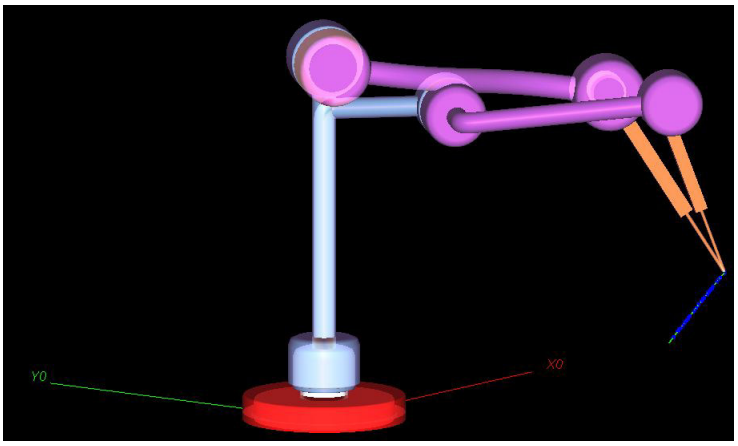

Figura 6.1: Simulación prototipo genérico Vs. Manipulador antropomórfico.

Ahora se observa la graficación de los valores de la primera articulación θ_1, en la figura 6.2, la variación del ángulo respecto al tiempo para seguir la trayectoria rampa propuesta para el prototipo genérico y el manipulador antropomórfico es la misma. Aunque la velocidad del prototipo genérico se aprecia que es ligeramente mayor pero sigue el mismo comportamiento. Del mismo modo para la aceleración, la variación es mínima, no representa mayor diferencia.

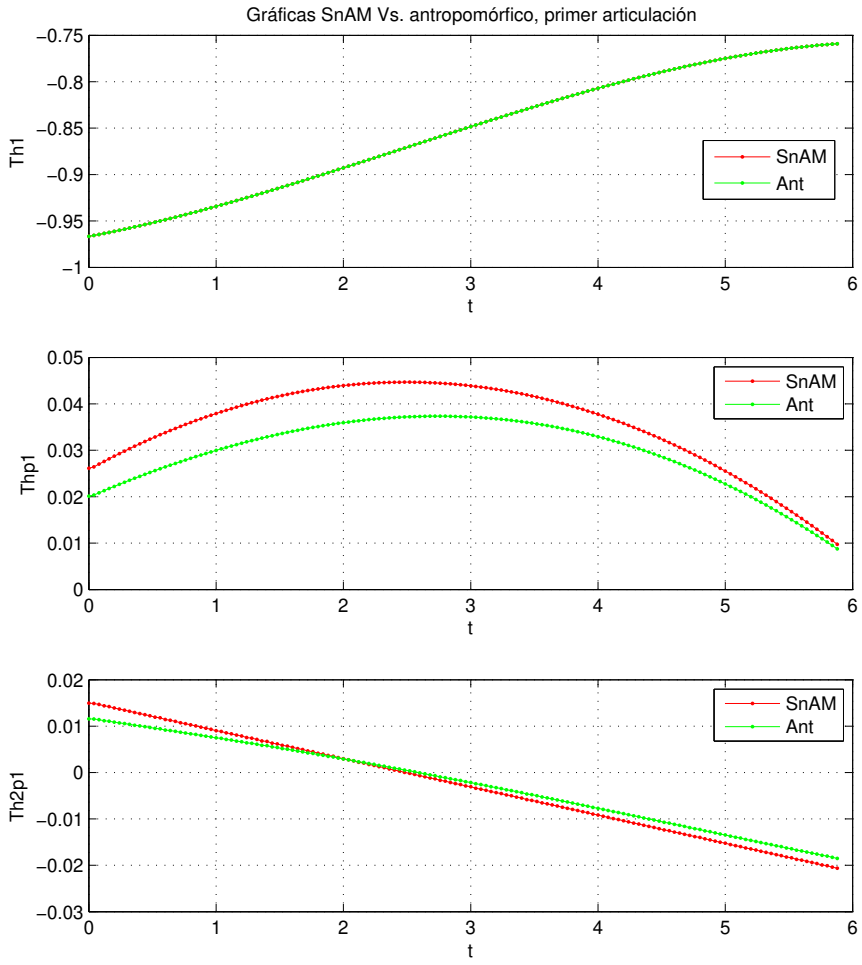

Figura 6.2: Graficación de valores de la primer articulación.

Ahora en la figura 6.3 se observan los valores de θ_2 para la segunda articulación. El ángulo para el manipulador SnAM es mayor que el del manipulador antropomórfico, pero se puede observar que la variación con respecto del tiempo para ambas articulaciones es similar. Para el caso de la velocidad la articulacion del manipulador antropomórfico es mayor, además de que presenta mayor variación a través del tiempo. Como es de esperar la aceleración también es mayor en el manipulador antropomórfico.

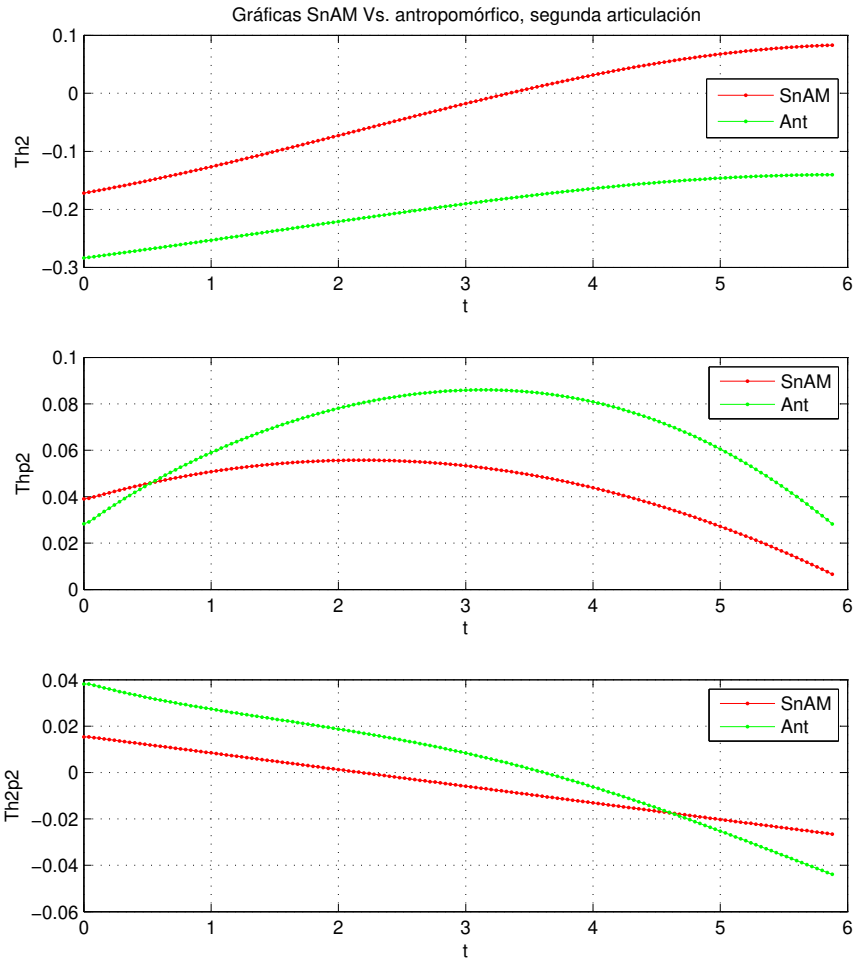

Figura 6.3: Graficación de valores de la segunda articulación.

Por último en la figura 6.4 se presentan los valores de θ_3 para la tercer articulación. De nueva cuenta como en la segunda articulación el ángulo del manipulador antropomófico es mayor, pero la variación que presentan las articulaciones de ambos manipuladores es simular. Para la velocidad, la variación es mayor en el caso del manipulador antropomórfico, lo mismo ocurre para la aceleración de esta articulación.

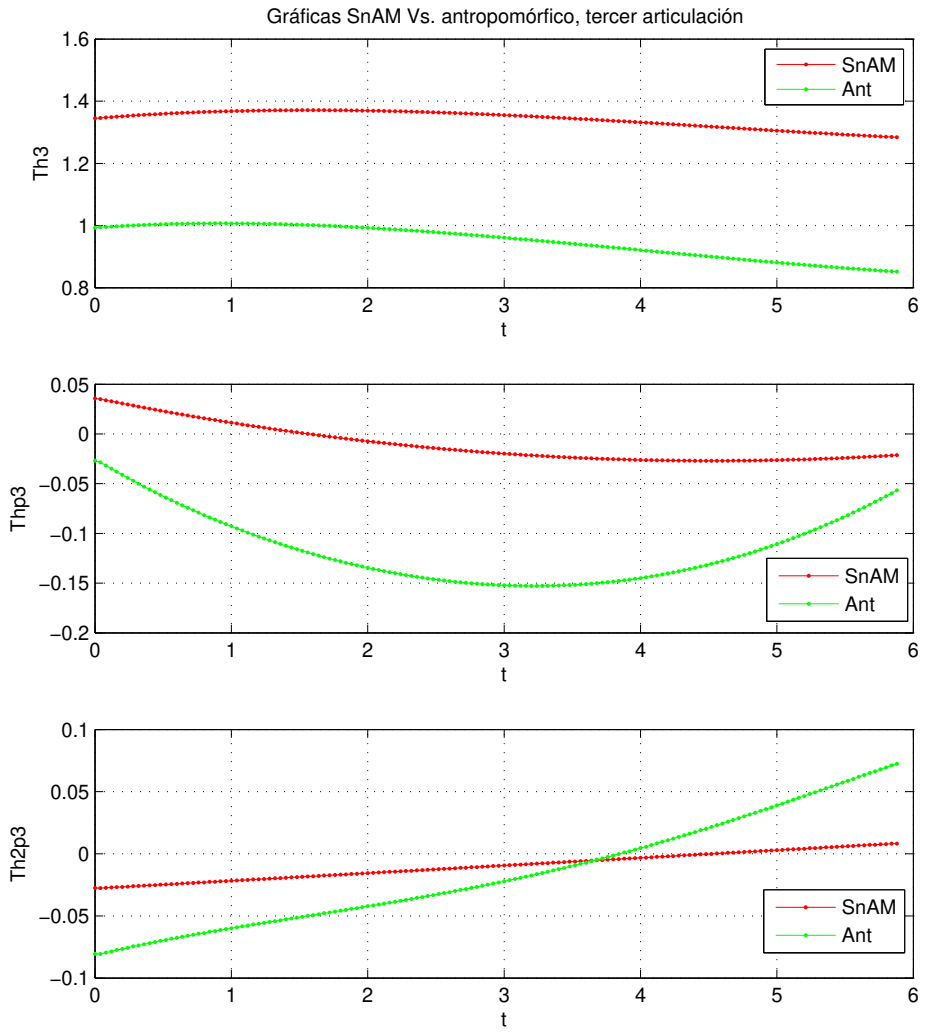

Figura 6.4: Graficación de valores de la tercer articulación.

Se puede decir que para el seguimiento de la trayectoria propuesta, el prototipo genérico (SnAM) es el que presenta mejor comportamiento, puesto que presenta valores más estables de velocidad y aceleración en su tercer articulación, en comparativa con las mismas articulaciones del manipulador antropomórfico.

6.2.2. Prototipo genérico Vs. Manipulador esférico.

En este caso, se considera la misma trayectoria que en la sección anterior, pero se realiza la comparativa con un manipulador esférico.

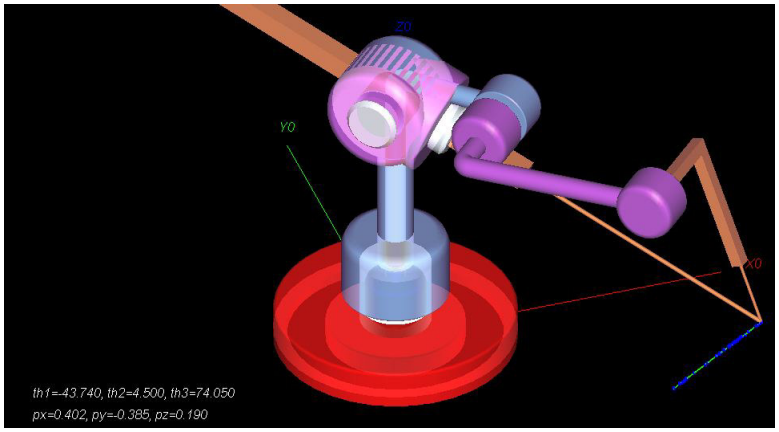

Figura 6.5: Simulación prototipo genérico Vs. Manipulador esférico.

En la figura 6.6 se presentan los valores para la primer articulación. En este caso no hay mucho que analizar puesto que en todos los aspectos para ambos manipuladores, las respectivas articulaciones presentan el mismo comportamiento. Es por este motivo que en las gráficas se aparenta que solo se mostraran los valores del manipulador esférico.

Para el caso de la figura 6.7 se observa que la variación del ángulo con respecto del tiempo para la segunda articulación es mayor en el caso del manipulador esférico. Mientras que en el caso de la velocidad, es mayor para el SnAM. Aunque la aceleración del manipulador SnAm es mayor, esta variación no representa una diferencia significativa.

Ahora, para el caso de la tercer articulación de la imagen 6.8, estas son de diferente clase, mientras que para el manipulador SnAm la tercer articulación es una revoluta, la articulación del manipulador esférico es prismática. En este caso, no es conveniente graficar los valores de dichas articulaciones en un mismo plano puesto que las unidades son distintas. En la figura 6.8 se presentan ambas series de gráficas, que si bien no pueden ser graficadas en el mismo plano cartesiano, resulta interesante ver su graficación a la par. Como se puede observar en el caso de la aceleración, el manipulador esférico presenta un comportamiento más estable.

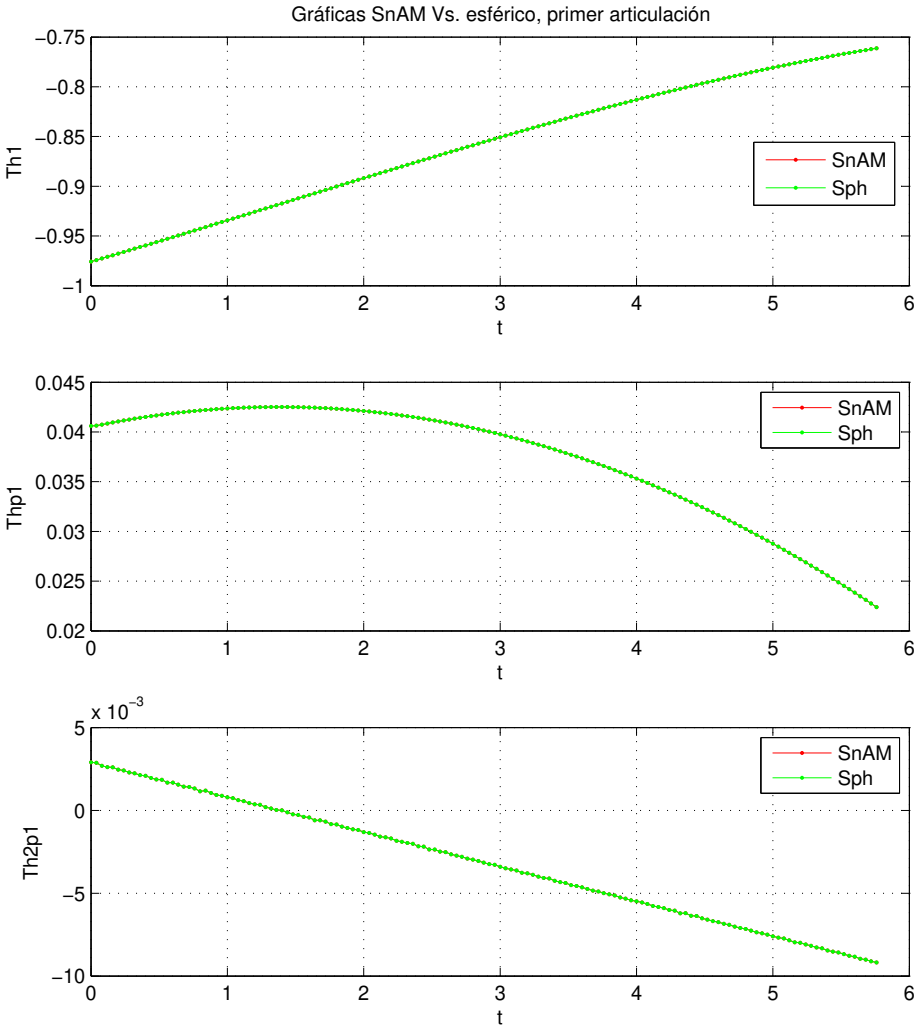

Figura 6.6: Graficación de valores de la primer articulación.

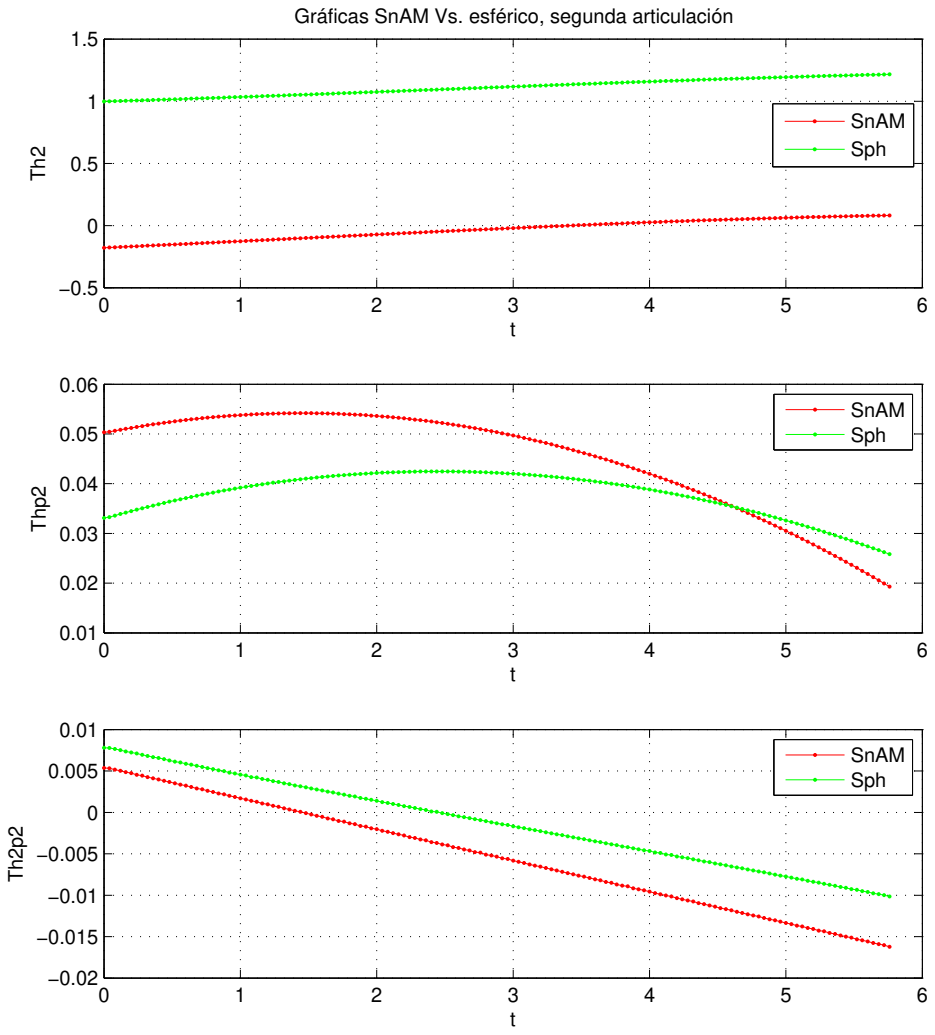

Figura 6.7: Graficación de valores de la segunda articulación.

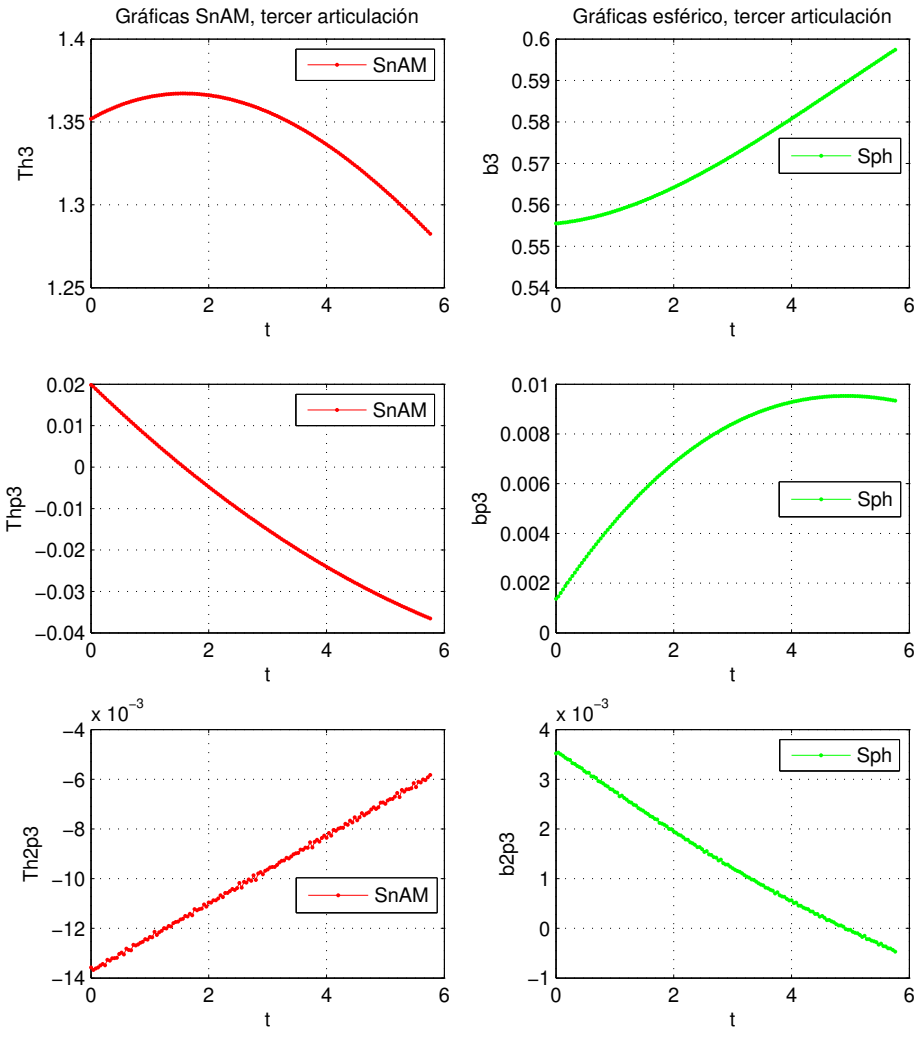

Figura 6.8: Graficación de valores de la tercer articulación.

6.3. Trayectoria circular.

La siguiente, es una trayectoria circular, igual que en la trayectoria anterior, se preparó una plantilla que permita el seguimiento de la trayectoria. Además, se hace uso del manipulador cartesiano[9], para tener un patrón de adquisición y que la trayectoria resulte mas uniforme. Para este caso se hace la comparativa con los manipuladores antropomórfico, SCARA y esférico.

6.3.1. Prototipo genérico Vs. Manipulador antropomórfico.

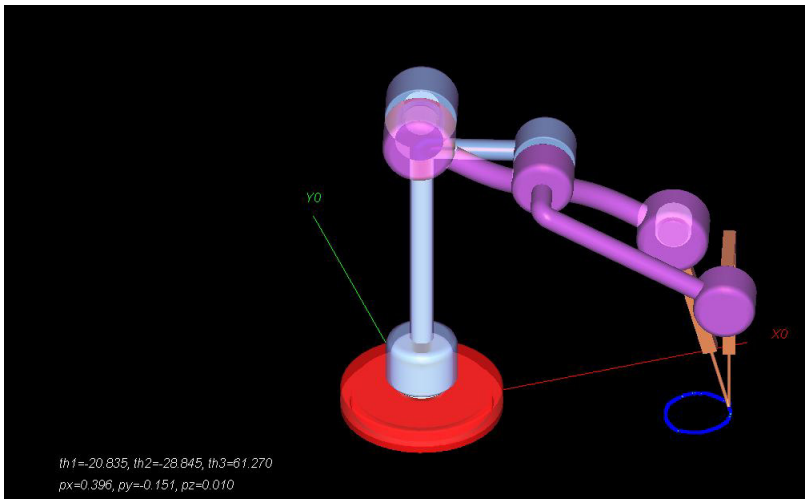

Figura 6.9: Simulación prototipo genérico Vs. Manipulador antropomórfico.

Para la primer articulación, los valores de θ_1 se muestran en la gráfica 6.10, aquí se puede observar que la variación del ángulo para las dos arquitecturas es la misma. En el caso de la velocidad $\dot{\theta}_1$ (Thp1) en la gráfica se observa que la variación es ligeramente mayor en el caso del manipulador SnAM. Para la aceleración $\ddot{\theta}_1$ (Th2p1) aunque el comportamiento de ambas arquitecturas es similar, para SnAM se observa una pendiente más pronunciada.

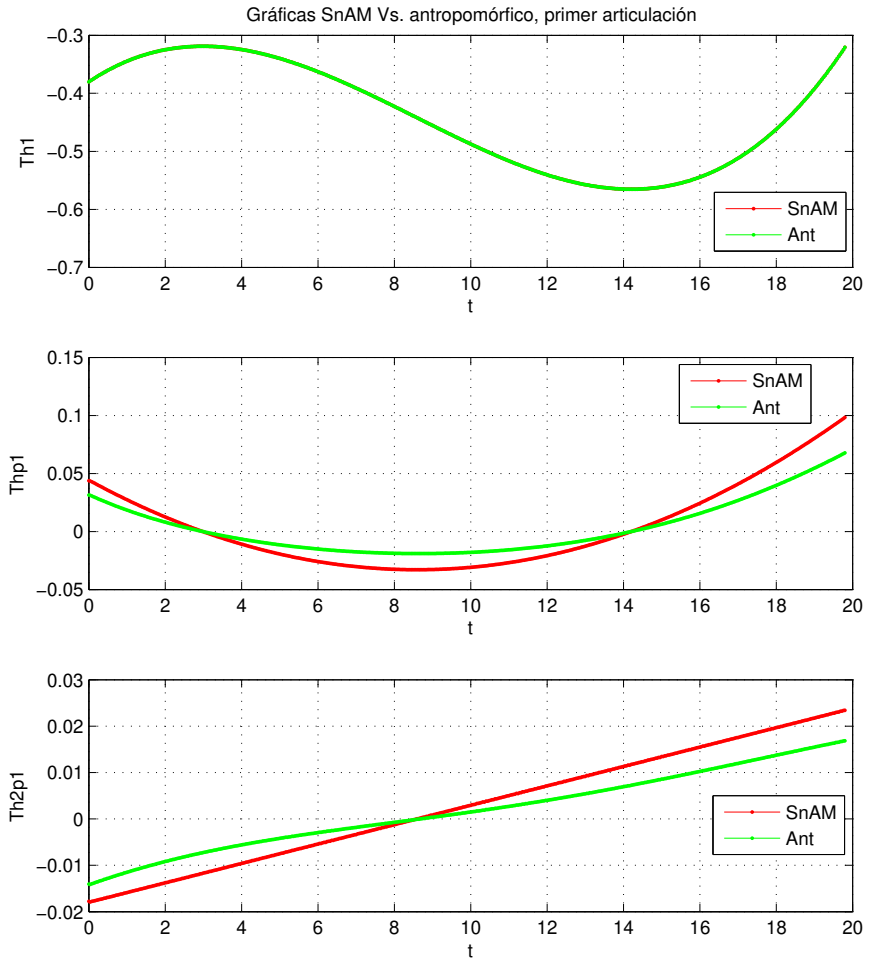

Figura 6.10: Graficación de valores de la primer articulación.

En la gráfica 6.11 se observan los valores de la segunda articulación. En este caso notamos que los ángulos θ_2 (Th2) son muy distintos para las arquitecturas, por esta razón, observamos que la variación de velocidad $\dot{\theta_2}$ (Thp2) es mínima en el caso del manipulador SnAM. Del mismo modo, en caso de la aceleración $\ddot{\theta_2}$ (Th2p2), el que presenta mejor comportamiento es el manipulador SnAM. En este caso específico el manipulador SnAM presenta ventajas.

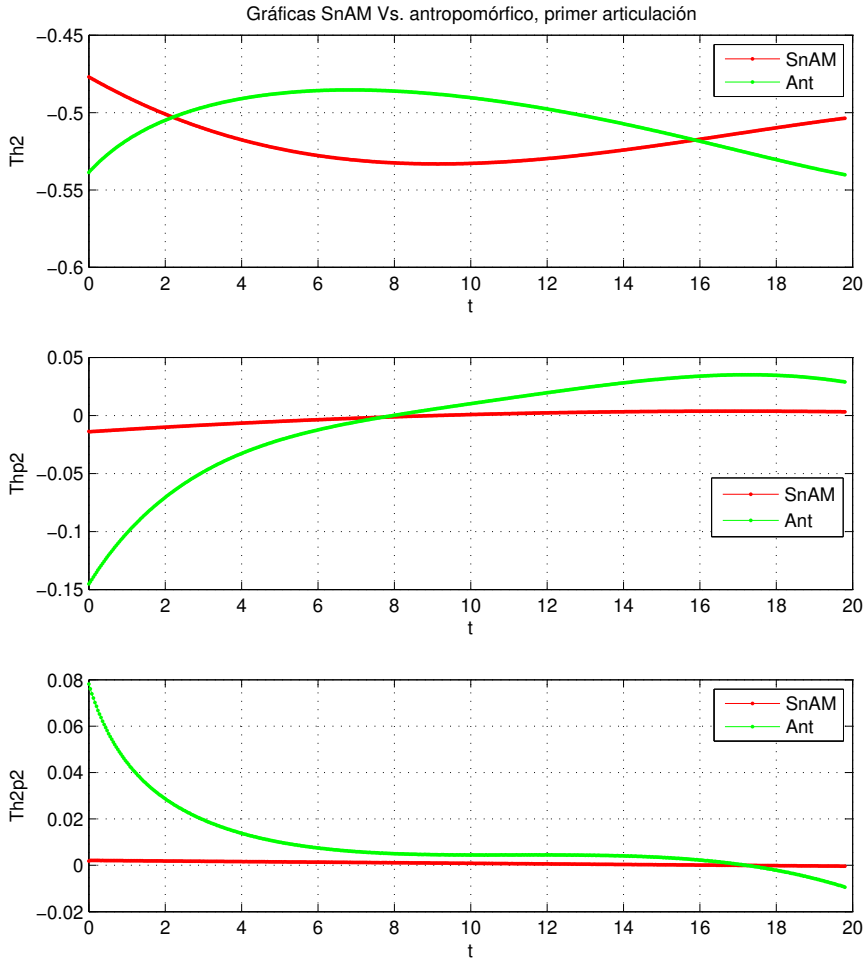

Figura 6.11: Graficación de valores de la segunda articulación.

Para los valores de la tercer articulación, mostrados en la gráfica 6.12 se observa que el ángulo θ_3 (Th3) es mayor para el manipulador SnAM, pero presentan una variación similar. En el caso de la velocidad y aceleración, de nueva cuenta el manipulador SnAM presenta un mejor comportamiento, siendo menor la variación que presenta.

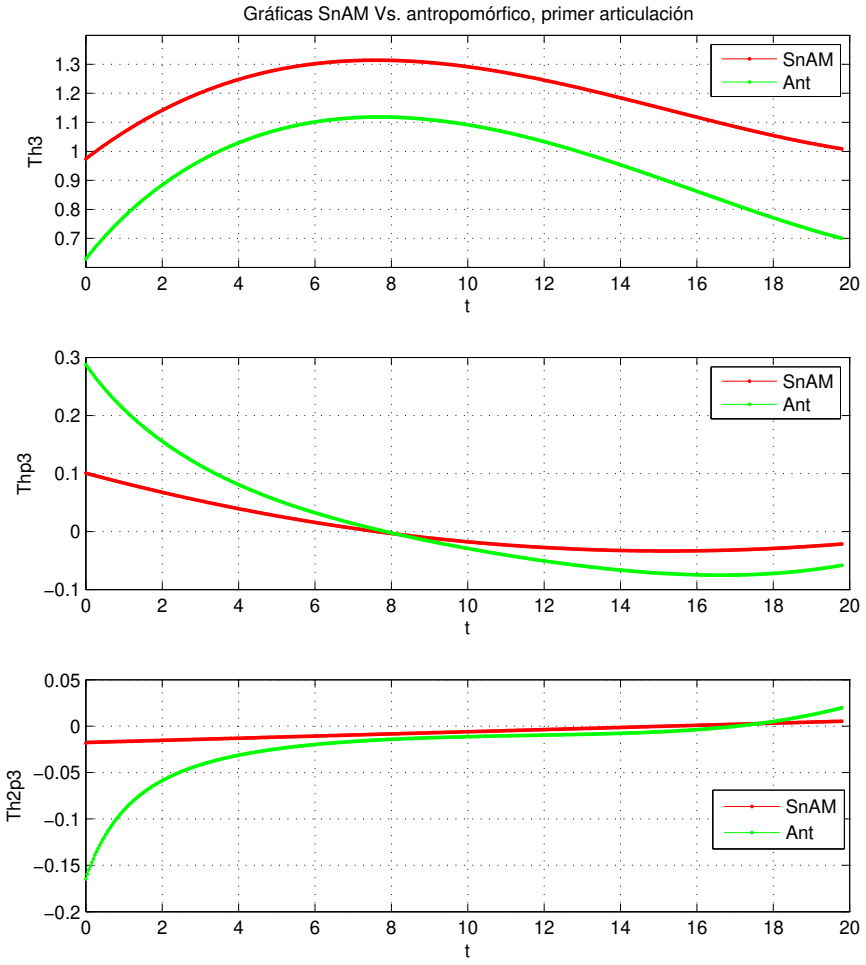

Figura 6.12: Graficación de valores de la tercer articulación.

Como se pudo observar, las articulaciones del manipulador SnAM presentan un mejor comportamiento para el seguimiento de la trayectoria circular. Los valores de velocidad y aceleración en las articulaciones muestran una variación más estable. Por esta razón sería apropiado optar por el manipulador SnAM.

6.3.2. Proptotipo genérico Vs. Manipulador esférico.

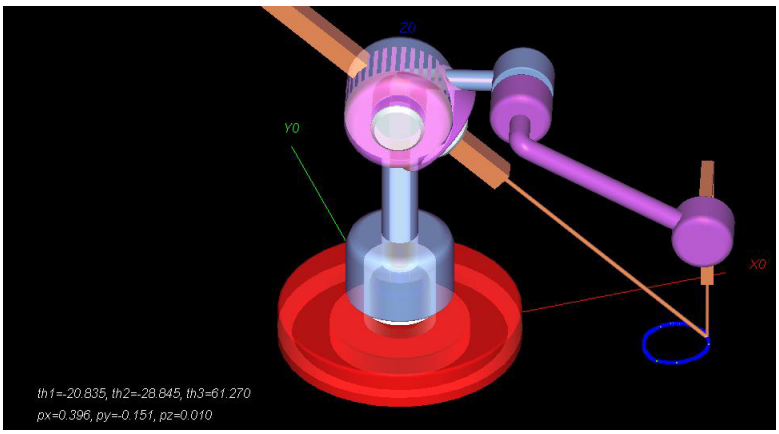

Figura 6.13: Simulación prototipo genérico Vs. Manipulador esférico.

Para los valores de la primer articulación, mostrados en la figura 6.14, se observa que presentan el mismo comportamiento para el ángulo θ_1, velocidad angular $\dot{\theta_1}$ y aceleración angular $\ddot{\theta_1}$ en ambas articulaciones. De nueva cuenta las gráficas del manipulador esférico quedan sobrepuestas en las del manipulador SnAM, por lo que se aparenta que solo se graficaran los valores del manipulador esférico.

Para la segunda articulación, gráfica 6.15, el manipulador esférico tiene un ángulo θ_2 (Th2) mayor, aunque la variación presentada en ambos casos es similar. En el caso de la velocidad angular, $\dot{\theta_2}$ (Thp2), se puede ver que el manipulador SnAM presenta una variación menor a la variacón del manipulador esférico. Para la aceleración $\ddot{\theta_2}$ (Th2p2), es la misma situación, el manipulador SnAM presenta menor variación.

En la tercer articulación, gráfica 6.16, se encuentra que son de distinta clase, se grafican en distintos planos cartesianos a la par, se observa que la variación de θ_3 en el manipulador SnAM es más suave que la variación del desplazamiento b_3 en el manipulador esférico. En cuanto a la variación de velocidad para ambos manipuladores no hay mucho que decir, con sus diferencias ambas presentan un comportamiento estable. En cuanto a la aceleración de la articulación del manipulador SnAM, la variación de la aceleración angular $\ddot{\theta_3}$ es estable y lineal. Si bien la información presentada en este análisis no es concluyente, permite observar el comportamiento de cada manipulador. Para optar por una u otra arquitectura será necesario tomar en cuenta otros factores. Resultará interesante observar el comportamiento del manipulador SCARA en el seguimiento de esta misma trayectoria.

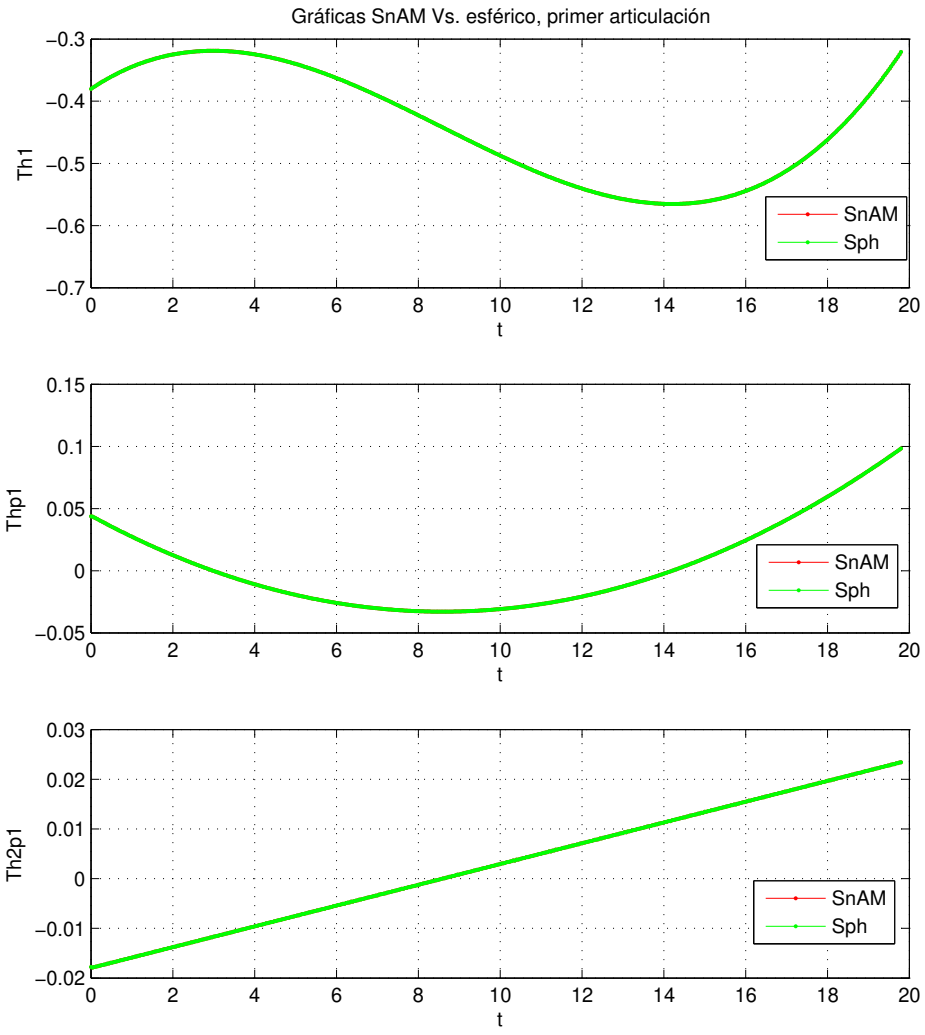

Figura 6.14: Graficación de valores de la primer articulación.

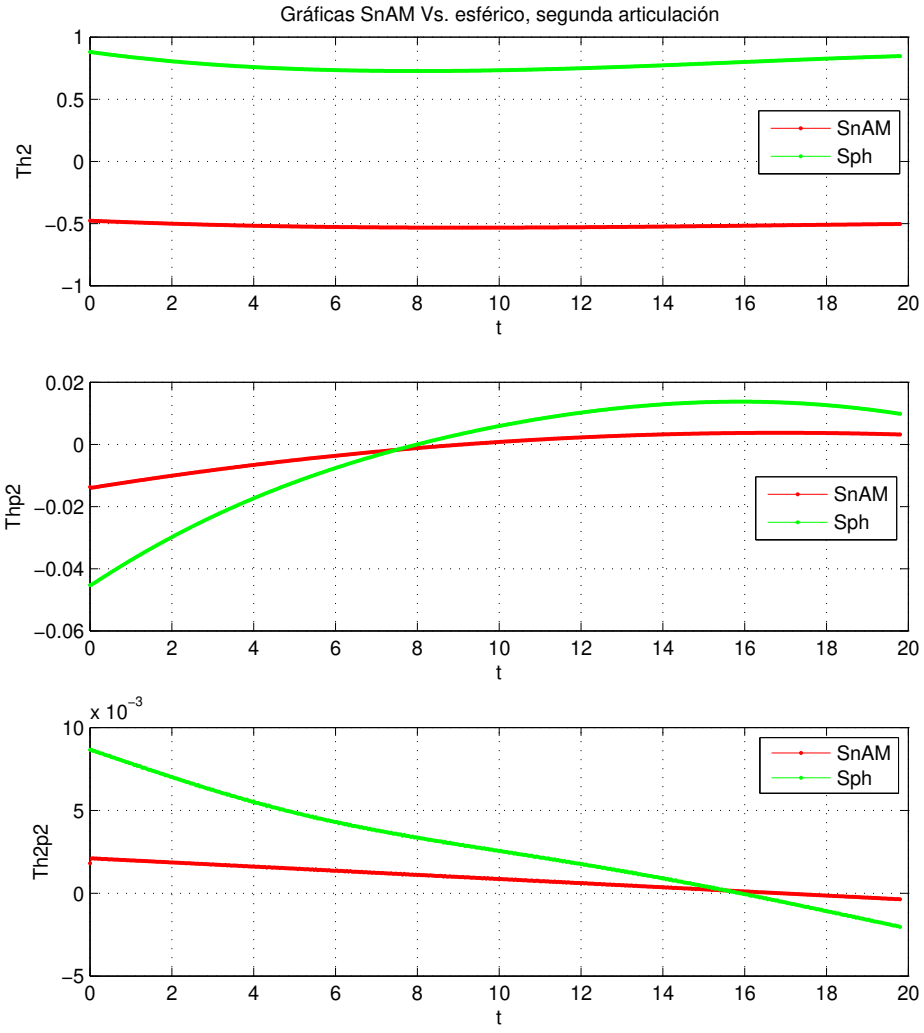

Figura 6.15: Graficación de valores de la segunda articulación.

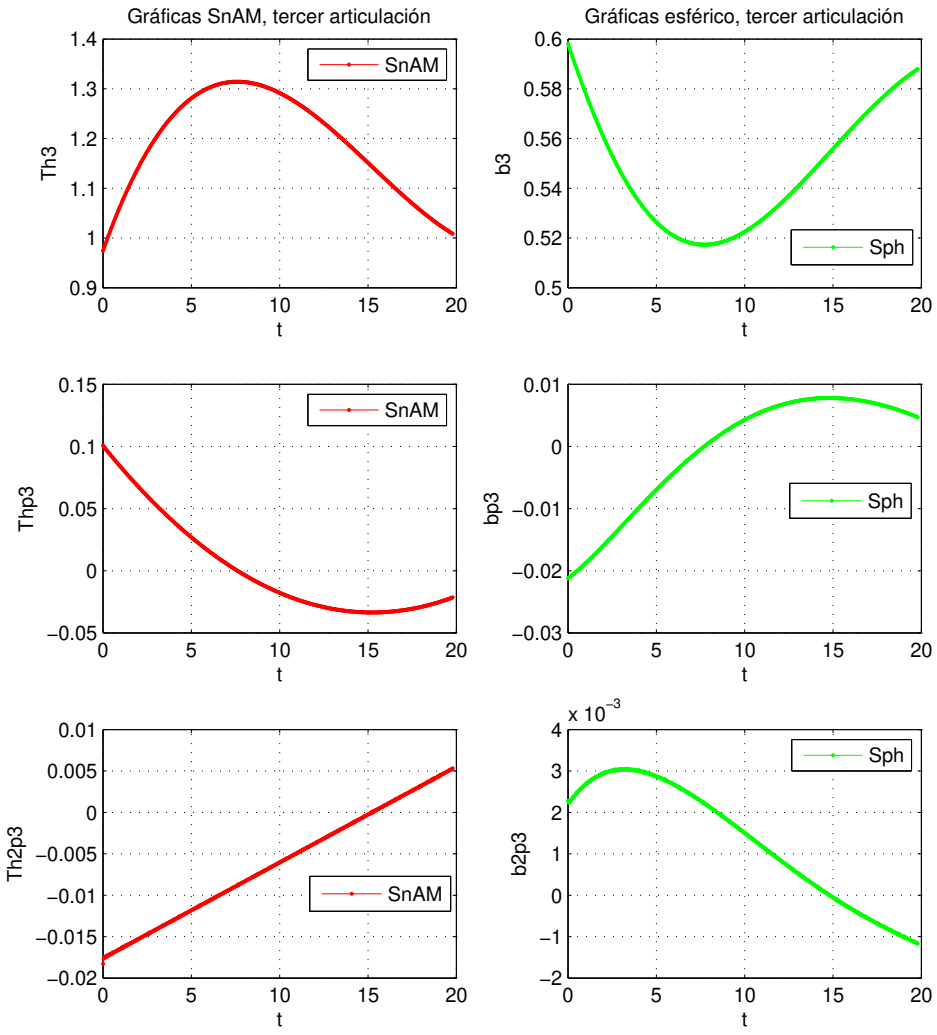

Figura 6.16: Graficación de valores de la tercer articulación.

6.3.3. Prototipo genérico Vs. Manipulador SCARA.

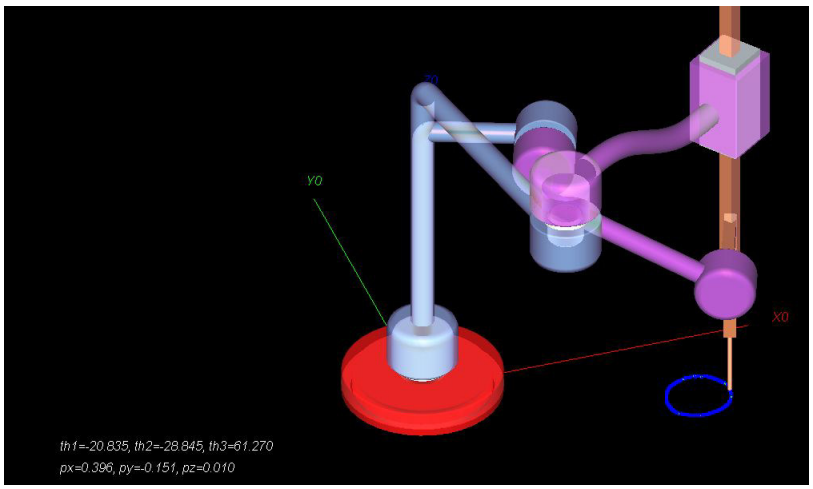

Figura 6.17: Simulación prototipo genérico Vs. Manipulador scara.

Para la gráfica 6.18, en la primera articulación se nota que el ángulo θ_1 (Th1) para el manipulador SnAM es mayor, pero el comportamiento en ambas arquitecturas es muy similar. En el caso de la velocidad angular $\dot{\theta_1}$ la variación es mínima. Para la aceleración angular $\ddot{\theta_1}$, la variación que se presenta también es pequeña.

Ahora en la segunda articulación, gráfica 6.19, se nota que el ángulo θ_2 del manipulador SnAM es más pequeño y con menor variación. Para la velocidad $\dot{\theta_2}$ y aceleración $\ddot{\theta_2}$, el comportamiento es mejor en el caso del manipulador SnAM, ya que también la variación que estos presentan es menor que en el caso del manipulador SCARA.

Para la tercer articulación, gráfica 6.20. también son de distinta clase, por lo cual las gráficas se presentan a la par aunque en distintos planos cartesianos. En este caso en especial resulta interesante analizar los valores del manipulador SCARA, si bien en el caso de b_3 se muestra una variación, en realidad esto no debería de ser así, tratándose de una trayectoria plana, esta variación es debida a que se trata de una adquisición manual, además del ajuste por mínimos cuadrados para obtener funciones adecuadas. A pesar de estas situaciones es importante remarcar que tanto la velocidad $\dot{b_3}$ como la aceleración $\ddot{b_3}$ se aproximan a cero, lo cual concuerda con la arquitectura del manipulador SCARA.

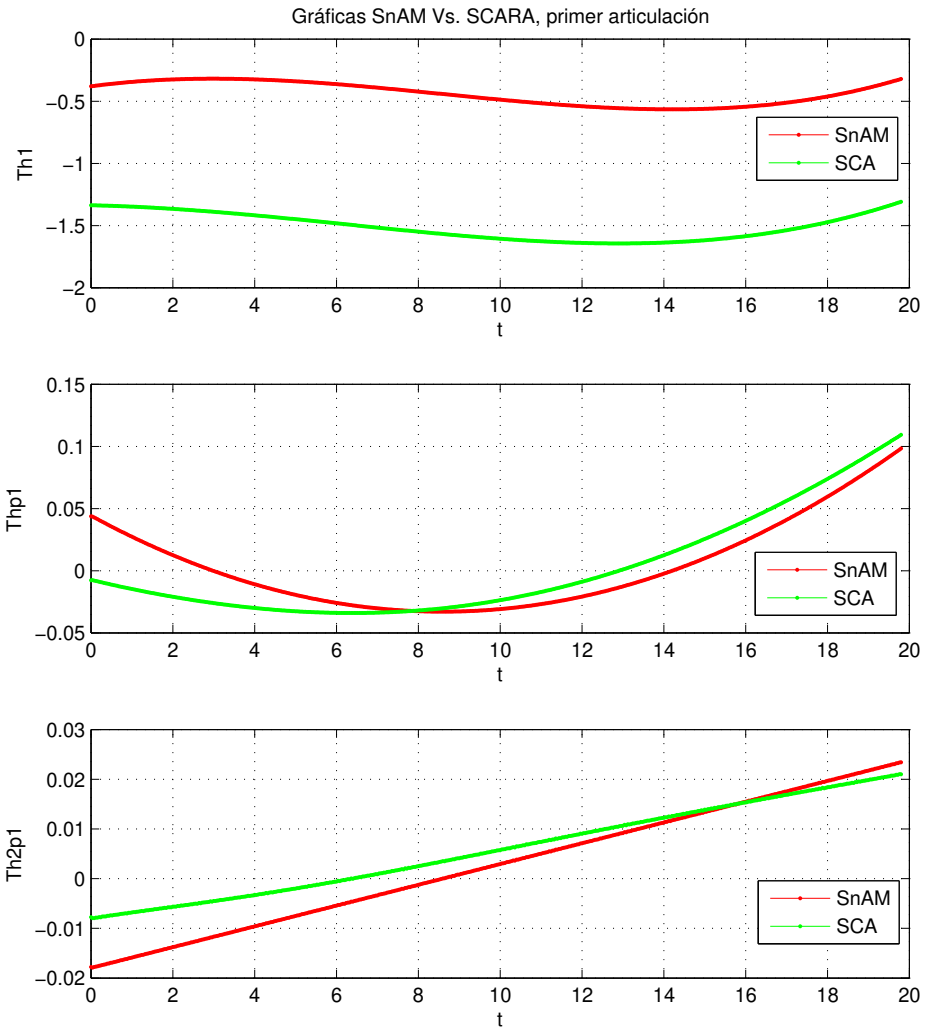

Figura 6.18: Graficación de valores de la primer articulación.

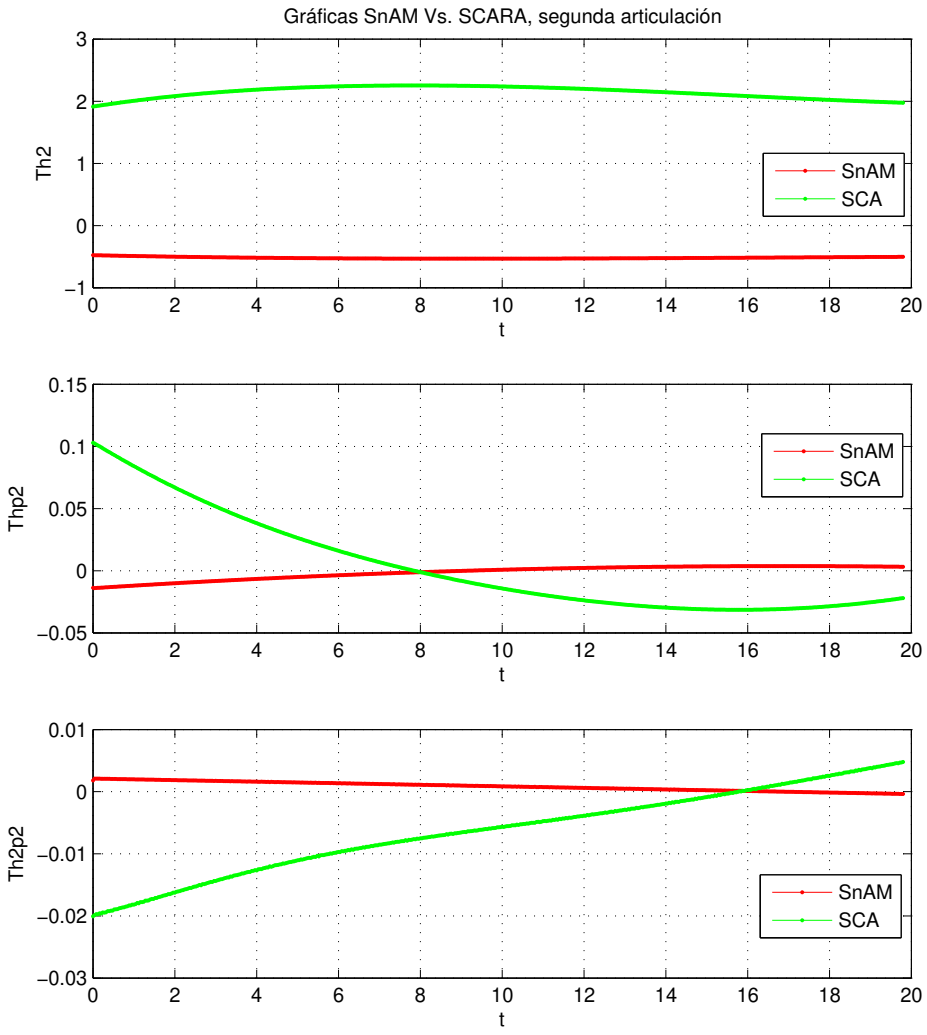

Figura 6.19: Graficación de valores de la segunda articulación.

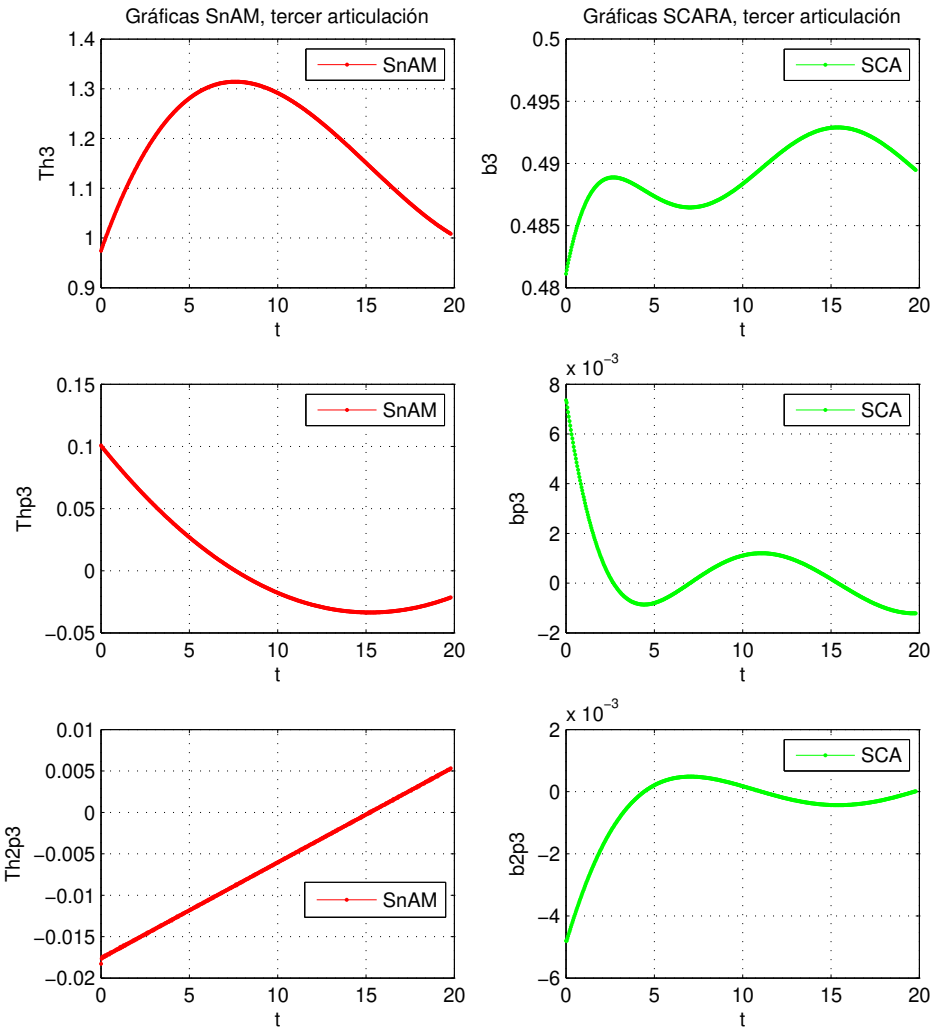

Figura 6.20: Graficación de valores de la tercer articulación.

Para este caso, se observa que el manipulador SCARA presenta ventaja en la tercer articulación al compararlo con el manipulador SnAM. También se puede notar cómo, a pesar de tener la misma clase de articulaciones el manipulador esférico y SCARA, la orientación distinta genera variaciones interesantes.

6.4. Trayectoria Sinusoidal

Por último se analiza una trayectoria sinusoidal inclinada, de igual modo se tiene una plantilla que facilita la adquisición. Para este caso, se comparan los valores del prototipo genérico con el manipulador antropormórfico y esférico.

6.4.1. Prototipo genérico Vs. Manipulador antropomórfico.

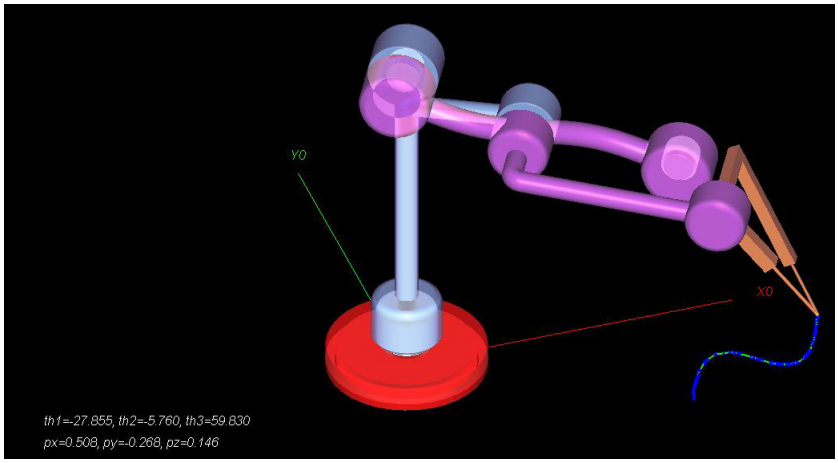

Figura 6.21: Simulación prototipo genérico Vs. Manipulador antropomórfico.

En la primer articulación, gráfica 6.22, el ángulo θ_1 (Th1), para ambas arquitecturas es el mismo. En cuanto a la velocidad angular $\dot{\theta}_1$ (Thp1) para el manipulador SnAM es ligeramente mayor, pero no es una variación importante. Lo mismo pasa en el caso de la aceleración angular $\ddot{\theta}_1$ (Th2p1), donde nuevamente para el manipulador SnAM es mayor.

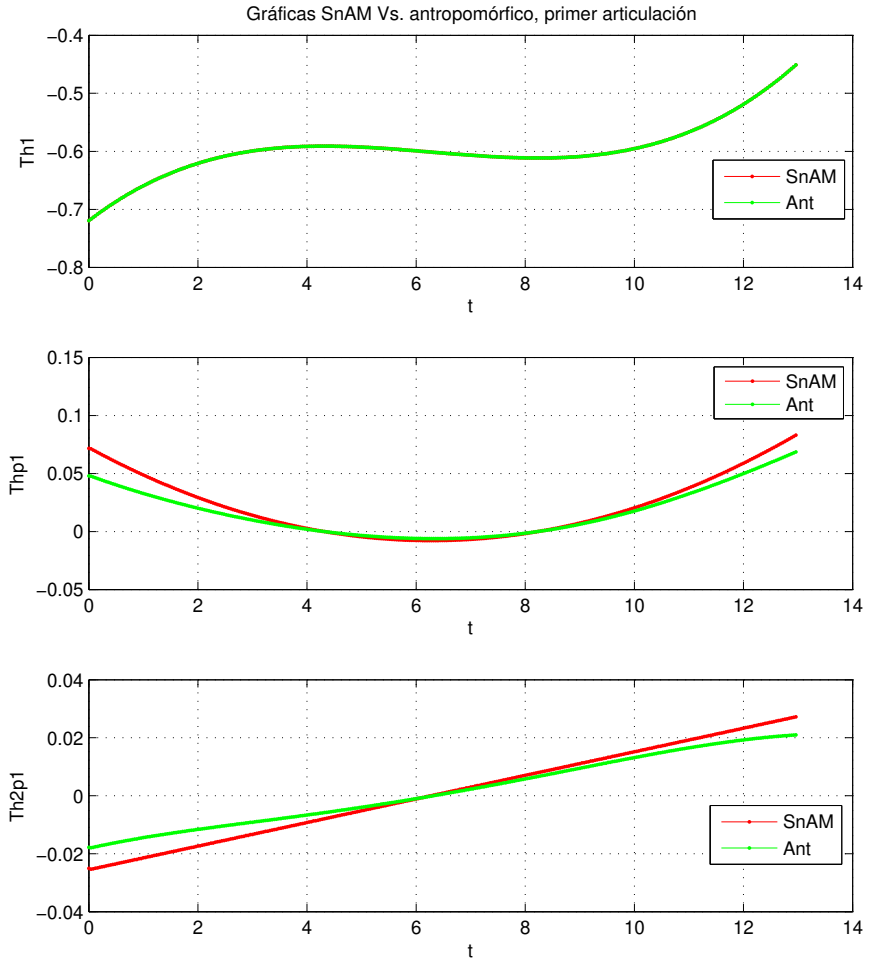

Figura 6.22: Graficación de valores de la primer articulación.

En la segunda articulación, gráfica 6.23, se observa que el ángulo θ_2 (Th2) es mayor en el manipulador SnAM, pero con una variación más estable comparado al manipulador antropomórfico. La velocidad angular $\dot{\theta_2}$ (Thp2) del manipulador SnAM es menor y estable. La aceleración angular $\ddot{\theta_2}$ (Th2p2) también es menor. En este caso es evidente que la segunda articulación del manipulador SnAM presenta un mejor comportamiento.

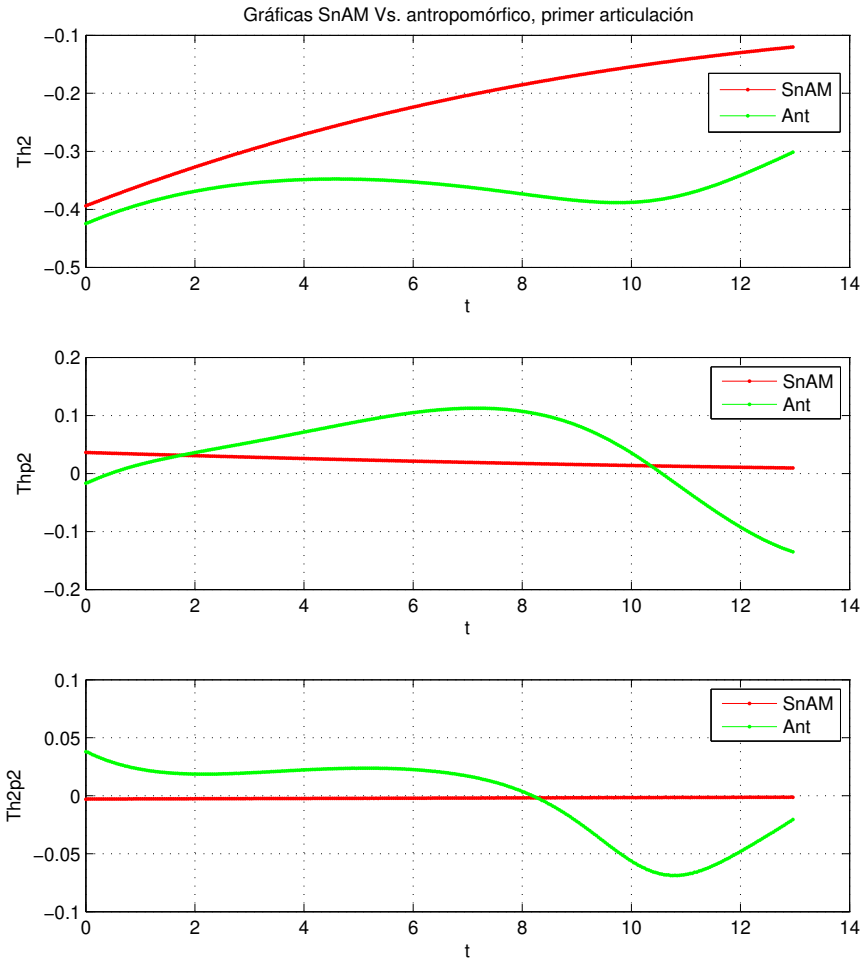

Figura 6.23: Graficación de valores de la segunda articulación.

Para la tercer articulación, gráfica 6.22, el ángulo θ_3 (Th3) es mayor en el manipulador SnAM. Para la velocidad angular $\dot{\theta}_3$ (Thp3), la articulación del manipulador SnAM presenta un mejor comportamiento, con variación menos pronunciada comparada con el manipulador antropomórfico. Del mismo modo la aceleración $\ddot{\theta}_3$ (Th2p3) la articulación del manipulador SnAM presenta un comportamiento más estable. En conclusión, el manipulador SnAM tiene un mejor comportamiento para seguir la trayectoria sinusoidal.

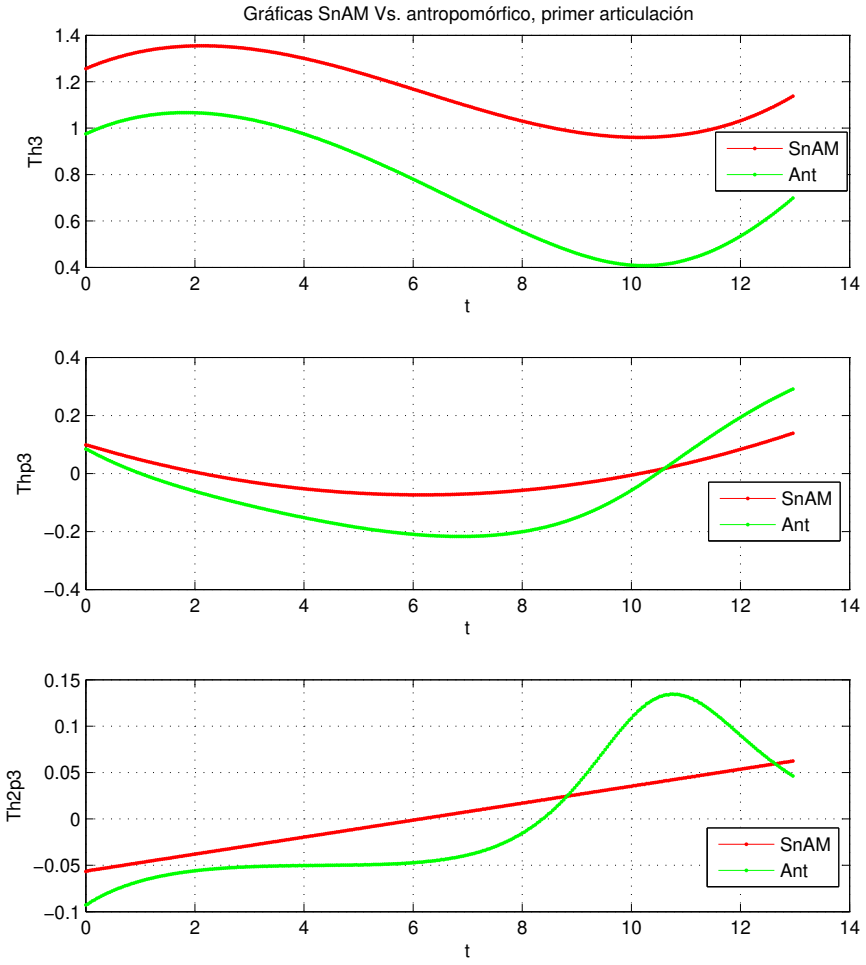

Figura 6.24: Graficación de valores de la tercer articulación.

6.4.2. Proptotipo genérico Vs. Manipulador esférico.

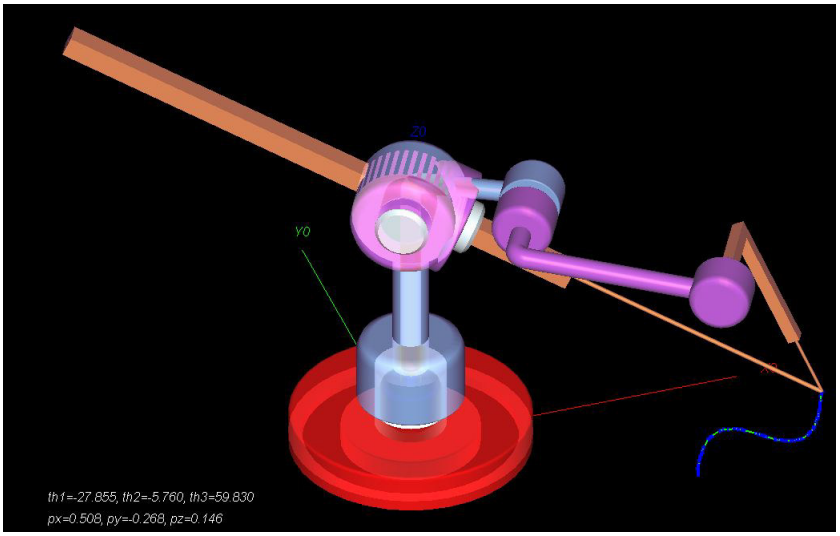

Figura 6.25: Simulación prototipo genérico Vs. Manipulador esférico.

En la gráfica 6.26 se observa que el ángulo θ_1 (Th1) es igual en ambas arquitecturas. Lo mismo ocurre con el comportamiento de la velocidad $\dot{\theta_1}$ y aceleración $\ddot{\theta_1}$, no hay diferencia de comportamiento entre las articulaciones de ambas arquitecturas.

Para la segunda articulación, gráfica 6.27, el ángulo θ_2 (Th2)es mayor en el manipulador esférico, pero presenta un comportamiento similar al manipulador SnAM. En el caso de la velocidad $\dot{\theta_2}$ (thp2), el manipulador SnAM presenta un comportamiento más estable, con menor variación. Lo mismo ocurre con la aceleración $\ddot{\theta_2}$, siendo el manipulador SnAM la mejor opción.

Para la tercer articulación, gráfica 6.28, siendo de diferente clase en las arquitecturas, se nota que el ángulo θ_3 tiene un comportamiento suave y estable, lo mismo ocurre en el caso de la extensión b_3. En cuanto a la velocidad, la velocidad angular $\dot{\theta_3}$ del manipulador SnAM presenta una variación estable, mientras que para el caso de $\dot{b_3}$ aunque también es estable, presenta una variación más pronunciada. En el caso de la aceleración, en el manipulador SnAM la aceleración angular $\ddot{\theta_3}$ tiene un mejor comportamiento, puesto que en el manipulador esférico la aceleración $\ddot{b_3}$ tiene variaciones más pronunciadas.

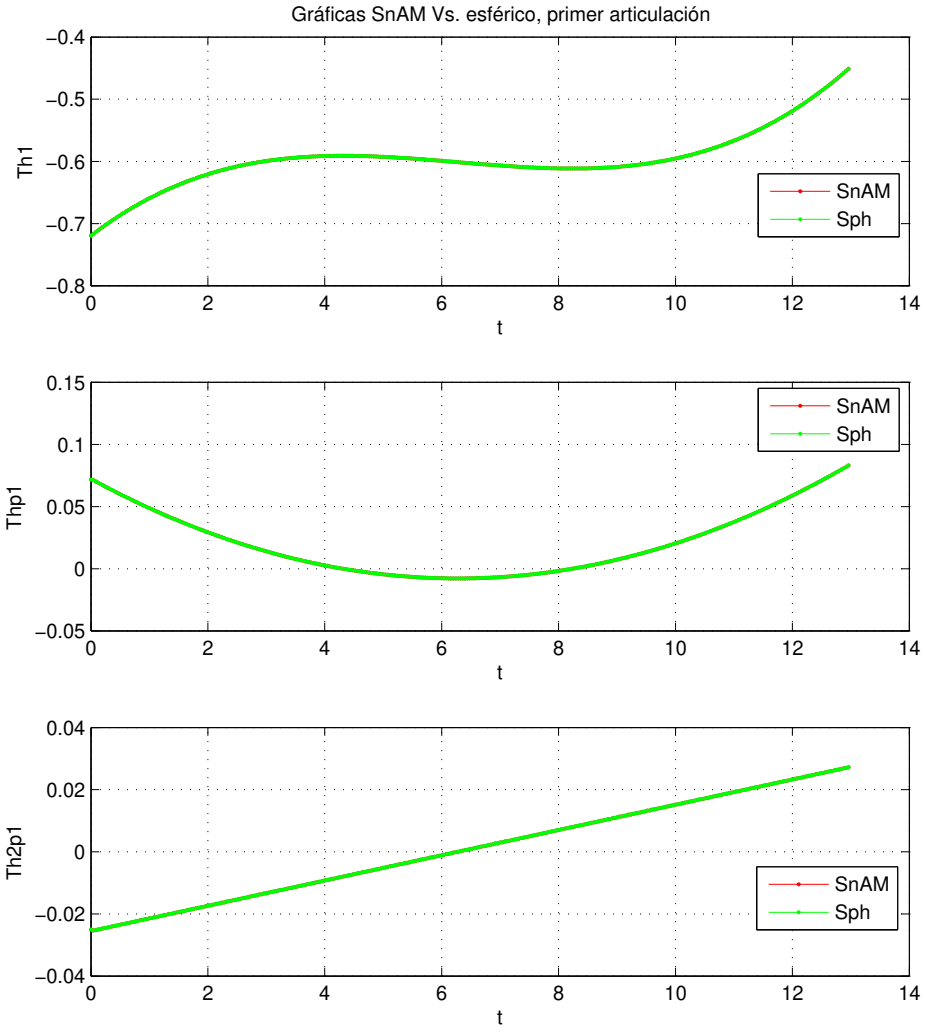

Figura 6.26: Graficación de valores de la primer articulación.

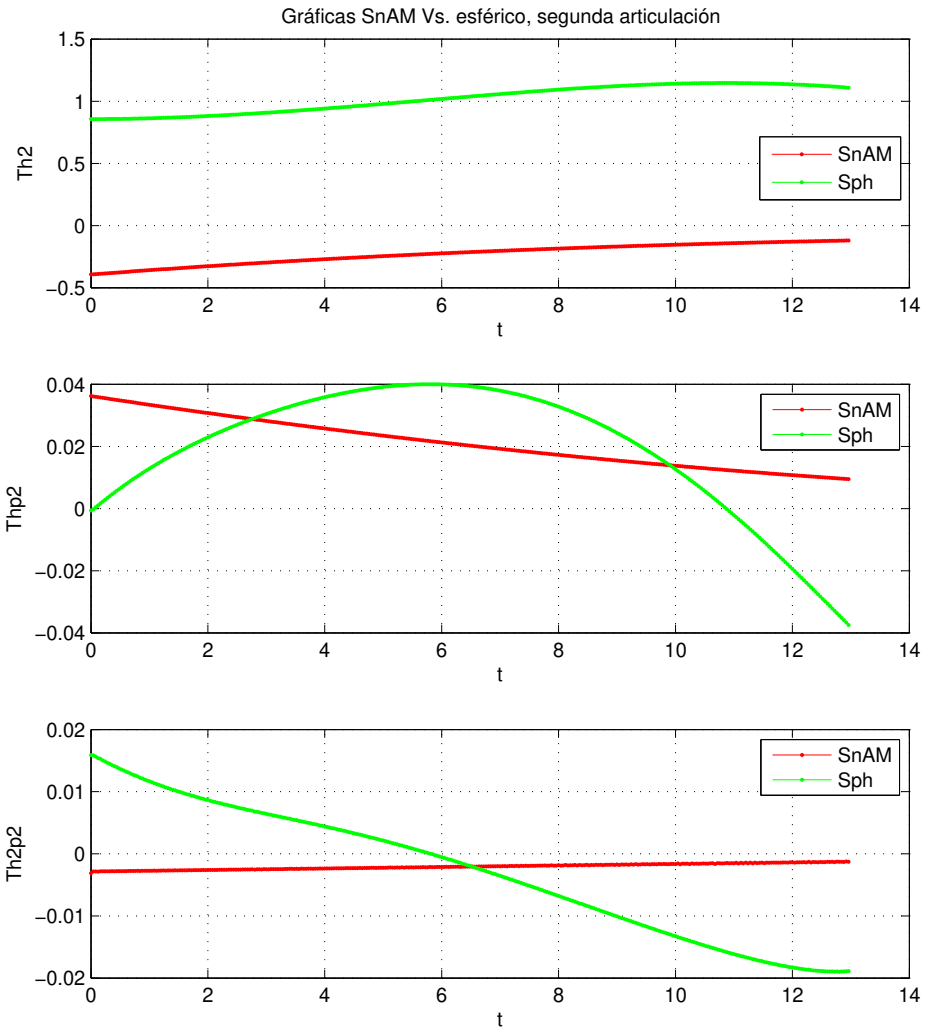

Figura 6.27: Graficación de valores de la segunda articulación.

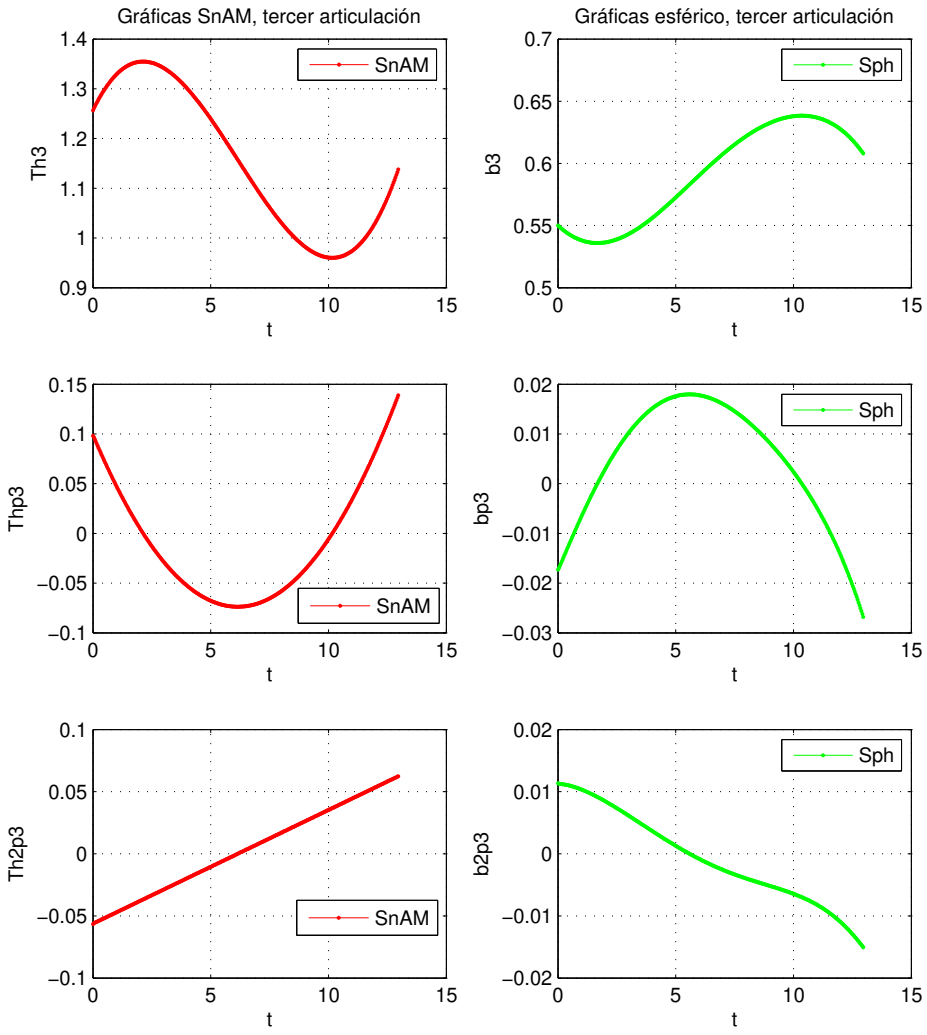

Figura 6.28: Graficación de valores de la tercer articulación.

Capítulo 7

Conclusiones y trabajos a futuro

La elaboración de este proyecto fue posible gracias a la plataforma de desarrollo ADEFID, ya que cuenta con todas las herramientas necesarias para la vinculación de dispositivos de entrada, adquisición, procesamiento y simulación de las trayectorias, presentando la ventaja de ser escalable, pudiendo ser este proyecto la base para un futuro proyecto con un grado más alto de especialización.

El prototipo genérico, equipado con encoders en cada una de sus articulaciones, conectadas al sistema SNAP PAC de Opto 22, permite la adquisición de los datos necesarios para el análisis de seguimiento de trayectorias. La trayectoria es adquirida punto a punto, capturado un punto se espera un lapso de tiempo determinado hasta capturar el siguiente punto. De este modo se obtiene la información necesaria para el análisis de la trayectoria.

Además del prototipo genérico utilizado para la adquisición, se propusieron manipuladores con arquitecturas distintas. Revisadas las herramientas necesarias para el análisis cinemático, se dio solución a la cinematica directa e inversa de cada manipulador así como la cinemática de velocidad y aceleración de los mismos.

Haciendo uso de los datos obtenidos en la adquisición, se presentó la simulación del seguimiento de la trayectoria, se exhibió la simulación de dos arquitecturas a la par, permitiendo así visualizar las variaciones de comportamiento de cada una de las articulaciones de diferentes arquitecturas ante una misma trayectoria.

Es importante resaltar el uso de métodos numéricos que auxilian en la solución a la cinemática de los manipuladores. En este caso para la obtención de la velocidad y aceleración del efector final se hizo uso de diferencias finitas, pero un requerimiento para la aplicación de este método es que los datos utilizados correspondan a una función suave y diferenciable, debido a la naturaleza empírica de los datos obtenidos mediante el prototipo genérico, estos presentan errores y

es necesario adecuar los datos mediante el método de mínimos cuadrados. Estos métodos suelen utilizarse en conjunto, ya que proporcionan buenos resultados y la bibliografía revisada así lo recomienda.

Para el análisis de posición, velocidad y aceleración de cada una de las articulaciones de cada arquitectura, se creó la función `GetPoVeAcParameters`. Esta función cuenta con todo lo necesario para realizar el cálculo de los parámetros requeridos, y al tratarse de una gran cantidad de datos los resultados son entregados en un archivo de texto con un formato diseñado para facilitar su análisis a través de la graficación de los resultados mediante softwares especializados como es el caso matlab. La graficación de los resultados permitió analizar el comportamiento de las articulaciones de los manipuladores de una manera más sencilla.

En resumen, el presente proyecto de tesis tiene un desarrollo completo, donde se observa la aplicación interdisciplinaria de la mecatrónica. Se dio cumplimiento a los resultados esperados al inicio. También se da solución a los retos inesperados en el desarrollo del proyecto. Se comprobó la utilidad de las características de la plataforma flexible ADEFID, que permitieron el desarrollo integral de este proyecto.

Trabajos a futuro

Desarrollado este proyecto existen tareas a desarrollar en un futuro. Entre tales trabajos a futuro destaca la posibilidad de implementar la adquisición de las trayectorias mediante un prototipo genérico que presente mejores características. Debido al tamaño del manipulador genérico éste presenta flexión, lo cual modifica ligeramente las mediciones realizadas.

En cuanto a la adquisición, también se podría hacer análisis con manipuladores servocontrolados, esto permitiría eliminar los errores por la adquisición manual.

También resultaría interesante extender los conceptos aquí presentados a manipuladores de 6GDL, de este modo también se podria incluir en el análisis brazos robóticos industriales.

Se podrían desarrollar estrategias para que se pueda realizar un análisis cuantitativo, y no únicamente cualitativo como en el presente proyecto, esto permitiría tener mas información para dar un veredicto de cual arquitectura presenta mejor comportamiento frente a una trayectoria determinada.

Apéndice A

Manipuladores propuestos.

Se proponen tres manipuladores para realizar la comparativa con el prototipo genérico, los manipuladores propuestos son el manipulador antropomórfico, SCARA y esférico. A continuación se presenta el análisis cinemático para dichos manipuladores.

A.1. Manipulador antropomórfico

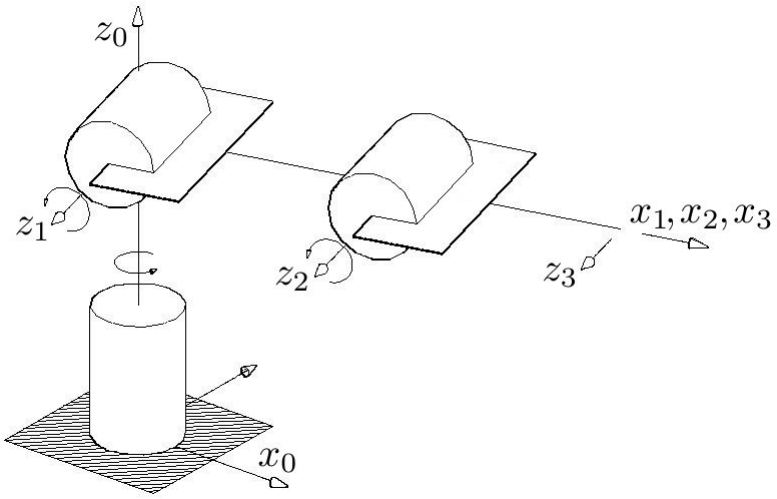

Figura A.1: Configuración del manipulador antropomórfico

i	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	0	90
2	θ_2^*	0	a_2	0
3	θ_3^*	0	a_3	0

Cuadro A.1: Parámetros H-D del manipulador antropomórfico

A.1.1. Parámetros D-H

$$T_0^3 = B_1 B_2 B_3 \tag{A.1}$$

$$T_0^3 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

$$T_0^3 = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & a_2C_1C_2 + a_3C_1C_{23} \\ S_1C_{23} & -S_1S_{23} & -C_1 & a_2S_1C_2 + a_3S_1C_{23} \\ S_{23} & C_{23} & 0 & b_1 + a_2S_2 + a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

A.1.2. Cinemática inversa.

$$\theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right) \tag{A.4}$$

$$\gamma = \pi - \theta_3 \tag{A.5}$$

$$c^2 = a_2^2 + a_3^2 - 2a_2a_3\cos\gamma \tag{A.6}$$

$$\cos\gamma = \frac{a_2^2 + a_3^2 - c^2}{2a_2a_3} \tag{A.7}$$

$$\cos\gamma = \cos(\pi - \theta_3) = -\cos\theta_3 \tag{A.8}$$

$$\cos\theta_3 = \frac{c^2 - a_2^2 - a_3^2}{2a_2a_3} = D \tag{A.9}$$

$$r = \sqrt{p_x^2 + p_y^2} \tag{A.10}$$

$$c^2 = r^2 + (p_z - b_1)^2 \tag{A.11}$$

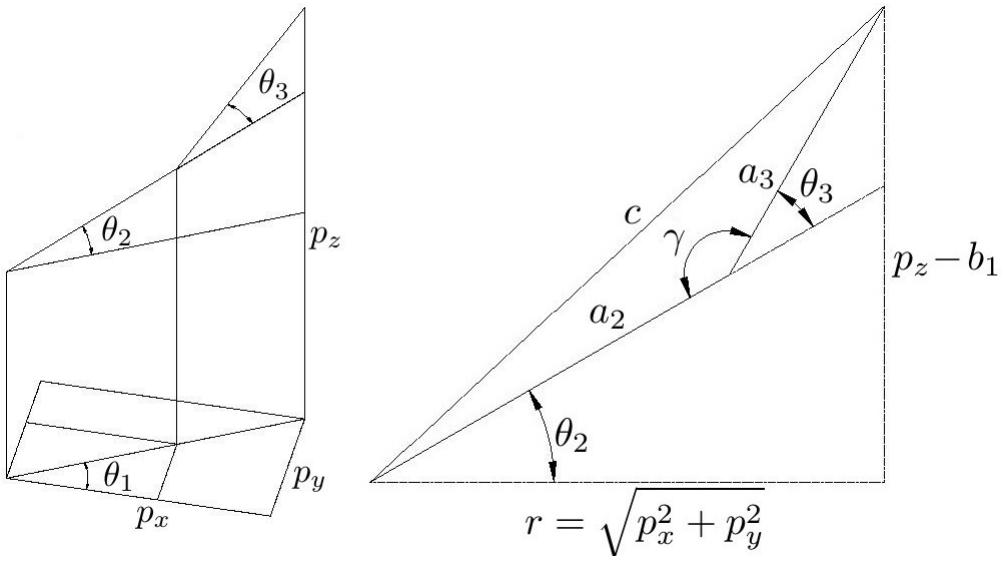

Figura A.2: Cinemática inversa de manipulador antropomórfico

$$\theta_3 = \tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D}\right) \tag{A.12}$$

$$\theta_2 = \tan^{-1}\left(\frac{p_z - b_1}{r}\right) - \tan^{-1}\left(\frac{a_3 \sin\theta_3}{a_2 + a_3\cos\theta_3}\right) \tag{A.13}$$

A.1.3. Jacobiano del manipulador antropomórfico

$$J = \begin{bmatrix} \mathbf{e}_0 \times (\mathbf{o}_3 - \mathbf{o}_0) & \mathbf{e}_1 \times (\mathbf{o}_3 - \mathbf{o}_1) & \mathbf{e}_2 \times (\mathbf{o}_3 - \mathbf{o}_2) \\ \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \tag{A.14}$$

Donde:

$$\mathbf{o}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.15}$$

$$\mathbf{e}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.16}$$

$$\mathbf{e}_1 = \mathbf{e}_2 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix} \tag{A.17}$$

De donde obtenemos:

$$J = \begin{bmatrix} -(a_2S_1C_2 + a_3S_1C_{23}) & -(a_2C_1S_2 + a_3C_1S_{23}) & -(a_3C_1S_{23}) \\ a_2C_1C_2 + a_3C_1C_{23} & -(a_2S_1S_2 + a_3S_1S_{23}) & -(a_3S_1S_{23}) \\ 0 & a_2C_2 + a_3C_{23} & a_3C_{23} \\ 0 & S_1 & S_1 \\ 0 & -C_1 & -C_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.18}$$

A.1.4. Cinemática de velocidad

$$\dot{X} = J(\mathbf{q})\dot{\mathbf{q}} \tag{A.19}$$

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{X} \tag{A.20}$$

$$J^{-1}(\mathbf{q}) = \frac{[AdjJ(\mathbf{q})]^T}{Det(J(\mathbf{q}))} \tag{A.21}$$

$$A = [AdjJ(\mathbf{q})]^T \tag{A.22}$$

Donde:

$$A_{11} = a_2a_3S_1S_3 \tag{A.23}$$

$$A_{12} = -a_2a_3C_1S_3 \tag{A.24}$$

$$A_{13} = 0 \tag{A.25}$$

$$A_{21} = a_2a_3C_1C_2C_{23} + a_3^2C_1C_{23}^2 \tag{A.26}$$

$$A_{22} = -a_2a_3S_1C_2C_{23} - a_3^2S_1C_{23}^2 \tag{A.27}$$

$$A_{23} = a_2S_{23}(a_2C_2 + a_3C_{23})(C_1^2 - S_1^2) \tag{A.28}$$

$$A_{31} = -C_1\left(a_2C_2 + a_3C_{23}\right)^2 \tag{A.29}$$

$$A_{32} = S_1\left(a_2C_2 + a_3C_{23}\right)^2 \tag{A.30}$$

$$A_{33} = (a_2^2S_2C_2 + a_2a_3S_{223} + a_3^2S_{23}C_{23})(S_1^2 - C_1^2) \tag{A.31}$$

$$\tag{A.32}$$

$$Det(J(\mathbf{q}) = (-a_2^2a_3C_2S_3 - a_2a_3^2S_3C_{23})(S_1^2 - C_1^2) \tag{A.33}$$

$$\ddot{X} = \dot{J}(\mathbf{q})\dot{\mathbf{q}} + J(\mathbf{q})\ddot{\mathbf{q}} \tag{A.34}$$

$$\ddot{\mathbf{q}} = J^{-1}(\mathbf{q})\left(\ddot{X} - \dot{J}(\mathbf{q})\dot{\mathbf{q}}\right) \tag{A.35}$$

$$\dot{J}_{11} = -a_2\dot{\theta}_1 C_1 C_2 + a_2\dot{\theta}_2 S_1 S_2 - a_3\dot{\theta}_1 C_1 C_{23} + a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) S_1 S_{23} \tag{A.36}$$

$$\dot{J}_{12} = a_2\dot{\theta}_1 S_1 S_2 - a_2\dot{\theta}_2 C_1 C_2 + a_3\dot{\theta}_1 S_1 S_{23} - a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) C_1 C_{23} \tag{A.37}$$

$$\dot{J}_{13} = a_3\dot{\theta}_1 S_1 S_{23} - a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) C_1 C_{23} \tag{A.38}$$

$$\dot{J}_{21} = -(a_2\dot{\theta}_1 S_1 C_2 + a_2\dot{\theta}_2 C_1 S_2 + a_3\dot{\theta}_1 S_1 C_{23} + a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) C_1 S_{23}) \tag{A.39}$$

$$\dot{J}_{22} = -a_2\dot{\theta}_1 C_1 S_2 - a_2\dot{\theta}_2 S_1 C_2 - a_3\dot{\theta}_1 C_1 S_{23} - a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) S_1 C_{23} \tag{A.40}$$

$$\dot{J}_{23} = -a_3\dot{\theta}_1 C_1 S_{23} - a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) S_1 C_{23} \tag{A.41}$$

$$\dot{J}_{31} = 0 \tag{A.42}$$

$$\dot{J}_{32} = -a_2\dot{\theta}_2 S_2 - a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) S_{23} \tag{A.43}$$

$$\dot{J}_{33} = -a_3\left(\dot{\theta}_2 + \dot{\theta}_3\right) S_{23} \tag{A.44}$$

$$\dot{J}_{41} = \dot{J}_{51} = \dot{J}_{61} = \dot{J}_{62} = \dot{J}_{63} = 0 \tag{A.45}$$

$$\dot{J}_{42} = \dot{J}_{43} = \dot{\theta}_1 C_1 \tag{A.46}$$

$$\dot{J}_{52} = \dot{J}_{53} = \dot{\theta}_1 S_1 \tag{A.47}$$

$$\tag{A.48}$$

A.2. Manipulador SCARA

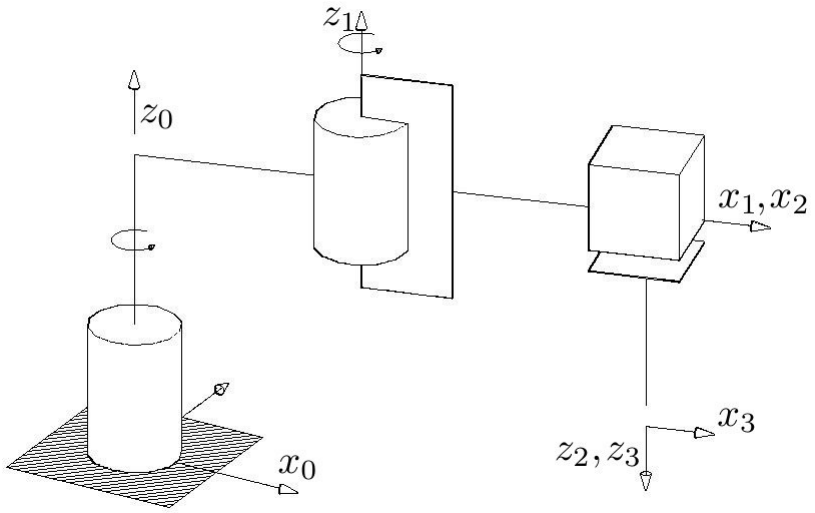

Figura A.3: Configuración del manipulador SCARA

A.2.1. Parámetros D-H

i	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	a_1	0
2	θ_2^*	0	a_2	180
3	0	b_3^*	0	0

Cuadro A.2: Parámetros H-D del manipulador SCARA

$$T_0^3 = B_1 B_2 B_3 \tag{A.49}$$

$$T_0^3 = \begin{bmatrix} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 & a_2 C_2 \\ S_2 & -C_2 & 0 & a_2 S_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.50}$$

$$T_0^3 = \begin{bmatrix} C_{12} & S_{12} & 0 & a_1 C_1 + a_2 C_{12} \\ S_{12} & -C_{12} & 0 & a_1 S_1 + a_2 S_{12} \\ 0 & 0 & -1 & b_1 - b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.51}$$

A.2.2. Cinemática inversa

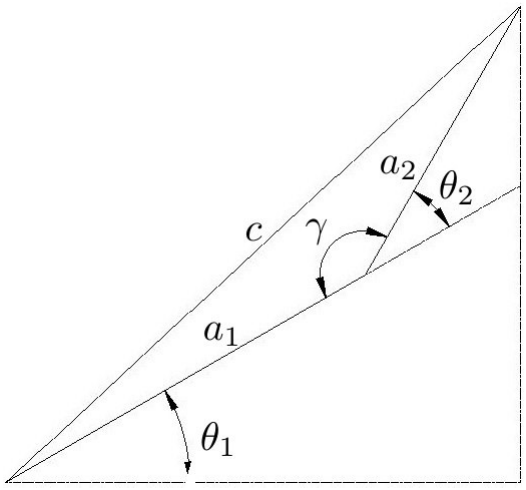

Figura A.4: Cinemática inversa de manipulador SCARA

$$b_3 = b_1 - p_z \tag{A.52}$$

$$c^2 = a_1^2 + a_2^2 - 2a_1a_2\cos\gamma \tag{A.53}$$

$$c^2 = p_x^2 + p_y^2 \tag{A.54}$$

$$\cos\gamma = \cos(\pi - \theta_2) = -\cos\theta_2 \tag{A.55}$$

$$\cos\theta_2 = \frac{c^2 - a_1^2 - a_2^2}{2a_1a_2} = D \tag{A.56}$$

$$\theta_2 = \tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D}\right) \tag{A.57}$$

$$\theta_1 = \tan^{-1}\left(\frac{p_x}{p_y}\right) - \tan^{-1}\left(\frac{a_2\sin\theta_2}{a_1 + a_2\cos\theta_2}\right) \tag{A.58}$$

A.2.3. Jacobiano del manipulador SCARA

$$J = \begin{bmatrix} \mathbf{e}_0 \times (\mathbf{o}_3 - \mathbf{o}_0) & \mathbf{e}_1 \times (\mathbf{o}_3 - \mathbf{o}_1) & \mathbf{e}_2 \\ \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{0} \end{bmatrix} \tag{A.59}$$

Donde:

$$\mathbf{o}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.60}$$

$$\mathbf{e}_0 = \mathbf{e}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.61}$$

$$\mathbf{e}_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{A.62}$$

De donde obtenemos:

$$J = \begin{bmatrix} -(a_1S_1 + a_2S_{12}) & -a_2S_{12} & 0 \\ a_1C_1 + a_2C_{12} & a_2C_{12} & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{A.63}$$

A.2.4. Cinemática de velocidad

$$\dot{X} = J(\mathbf{q})\dot{\mathbf{q}} \tag{A.64}$$

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{X} \tag{A.65}$$

$$J^{-1}(\mathbf{q}) = \frac{[AdjJ(\mathbf{q})]^T}{Det(J(\mathbf{q}))} \tag{A.66}$$

$$A = [AdjJ(\mathbf{q})]^T \tag{A.67}$$

Donde:

$$A_{11} = -a_2 C_{12} \tag{A.68}$$

$$A_{12} = -a_2 S_{12} \tag{A.69}$$

$$A_{13} = 0 \tag{A.70}$$

$$A_{21} = a_1 C_1 + a_2 C_{12} \tag{A.71}$$

$$A_{22} = a_1 S_1 + a_2 S_{12} \tag{A.72}$$

$$A_{23} = 0 \tag{A.73}$$

$$A_{31} = 0 \tag{A.74}$$

$$A_{32} = 0 \tag{A.75}$$

$$A_{33} = -a_1 a_2 S_2 \tag{A.76}$$

$$\tag{A.77}$$

$$Det(J(\mathbf{q}) = a_1 a_2 S_2 \tag{A.78}$$

$$\ddot{X} = \dot{J}(\mathbf{q})\dot{\mathbf{q}} + J(\mathbf{q})\ddot{\mathbf{q}} \tag{A.79}$$

$$\ddot{\mathbf{q}} = J^{-1}(\mathbf{q}) \left(\ddot{X} - \dot{J}(\mathbf{q})\dot{\mathbf{q}} \right) \tag{A.80}$$

$$\dot{J}_{11} = -a_1 \dot{\theta}_1 C_1 - a_2(\dot{\theta}_1 + \dot{\theta}_2) C_{12} \tag{A.81}$$

$$\dot{J}_{12} = -a_2(\dot{\theta}_1 + \dot{\theta}_2) C_{12} \tag{A.82}$$

$$\dot{J}_{21} = -a_1 \dot{\theta}_1 S_1 - a_2(\dot{\theta}_1 + \dot{\theta}_2) S_{12} \tag{A.83}$$

$$\dot{J}_{22} = -a_2(\dot{\theta}_1 + \dot{\theta}_2) S_{12} \tag{A.84}$$

$$\dot{J}_{13} = \dot{J}_{23} = \dot{J}_{31} = \dot{J}_{32} = \dot{J}_{33} = 0 \tag{A.85}$$

A.3. Manipulador esférico

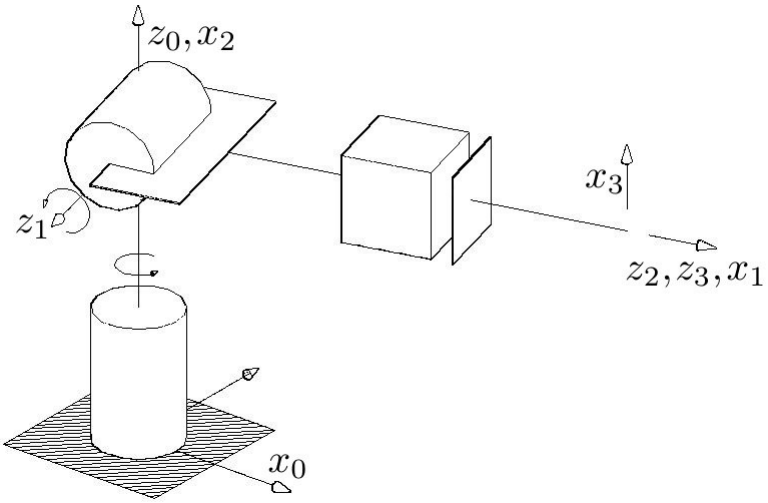

Figura A.5: Configuración del manipulador esférico

A.3.1. Parámetros D-H

i	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	0	90
2	θ_2^*	0	0	90
3	0	b_3^*	0	0

Cuadro A.3: Parámetros H-D del manipulador esférico

$$T_0^3 = B_1 B_2 B_3 \tag{A.86}$$

$$T_0^3 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.87}$$

$$T_0^3 = \begin{bmatrix} C_1C_2 & S_1 & C_1S_2 & b_3C_1S_2 \\ S_1C_2 & -C_1 & S_1S_2 & b_3S_1S_2 \\ S_2 & 0 & -C_2 & b_1 - b_3C_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.88}$$

A.3.2. Cinemática inversa

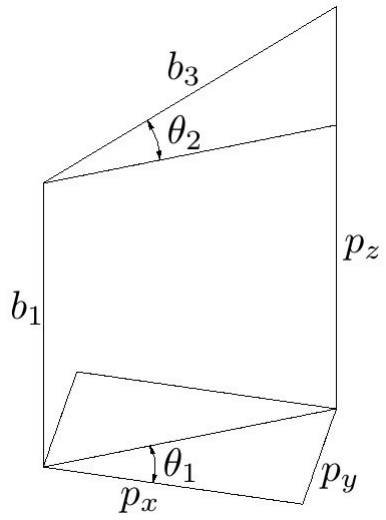

Figura A.6: Cinemática inversa de manipulador esférico

$$\theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right) \tag{A.89}$$

$$r = \sqrt{P_x^2 + P_y^2} \tag{A.90}$$

$$\theta_2 = \tan^{-1}\left(\frac{P_z - b_1}{r}\right) \tag{A.91}$$

$$b_3 = \sqrt{r^2 + (P_z - b_1)^2} \tag{A.92}$$

A.3.3. Jacobiano del manipulador esférico

$$J = \begin{bmatrix} \mathbf{e}_0 \times (\mathbf{o}_3 - \mathbf{o}_0) & \mathbf{e}_1 \times (\mathbf{o}_3 - \mathbf{o}_1) & \mathbf{e}_2 \\ \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{0} \end{bmatrix} \tag{A.93}$$

Donde:

$$\mathbf{o}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.94}$$

$$\mathbf{o}_1 = \begin{bmatrix} 0 \\ 0 \\ b_1 \end{bmatrix} \tag{A.95}$$

$$\mathbf{e}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.96}$$

$$\mathbf{e}_1 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix} \tag{A.97}$$

$$\mathbf{e}_2 = \begin{bmatrix} C_1 S_2 \\ S_1 S_2 \\ -C_2 \end{bmatrix} \tag{A.98}$$

De donde obtenemos:

$$J = \begin{bmatrix} -b_3 S_1 S_2 & b_3 C_1 C_2 & C_1 S_2 \\ b_3 C_1 S_2 & b_3 S_1 C_2 & S_1 S_2 \\ 0 & b_3 S_2 & -C_2 \\ 0 & S_1 & 0 \\ 0 & -C_1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.99}$$

A.3.4. Cinemática de velocidad

$$\dot{X} = J(\mathbf{q})\dot{\mathbf{q}} \tag{A.100}$$

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{X} \tag{A.101}$$

$$J^{-1}(\mathbf{q}) = \frac{[AdjJ(\mathbf{q})]^T}{Det(J(\mathbf{q}))} \tag{A.102}$$

$$A = [AdjJ(\mathbf{q})]^T \tag{A.103}$$

Donde:

$$A_{11} = -b_3 S_1 \tag{A.104}$$

$$A_{12} = b_3 C_1 \tag{A.105}$$

$$A_{13} = 0 \tag{A.106}$$

$$A_{21} = b_3 C_1 S_2 C_2 \tag{A.107}$$

$$A_{22} = b_3 S_1 S_2 C_2 \tag{A.108}$$

$$A_{23} = b_3 S_2^2 \tag{A.109}$$

$$A_{31} = b_3^2 C_1 S_2^2 \tag{A.110}$$

$$A_{32} = b_3^2 S_1 S_2^2 \tag{A.111}$$

$$A_{33} = -b_3^2 S_2 C_2 \tag{A.112}$$

$$Det(J(\mathbf{q}) = b_3^2 S_2 \tag{A.113}$$

$$\ddot{X} = \dot{J}(\mathbf{q})\dot{\mathbf{q}} + J(\mathbf{q})\ddot{\mathbf{q}} \tag{A.114}$$

$$\ddot{\mathbf{q}} = J^{-1}(\mathbf{q})\left(\ddot{X} - \dot{J}(\mathbf{q})\dot{\mathbf{q}}\right) \tag{A.115}$$

$$\dot{J}_{11} = -\dot{b}_3 S_1 S_2 - b_3\dot{\theta}_1 C_1 S_2 - b_3\dot{\theta}_2 S_1 C_2 \tag{A.116}$$

$$\dot{J}_{12} = \dot{b}_3 C_1 C_2 - b_3\dot{\theta}_1 S_1 C_2 - b_3\dot{\theta}_2 C_1 S_2 \tag{A.117}$$

$$\dot{J}_{13} = -\dot{\theta}_1 S_1 S_2 + \dot{\theta}_2 C_1 C_2 \tag{A.118}$$

$$\dot{J}_{21} = \dot{b}_3 C_1 S_2 - b_2\dot{\theta}_1 S_1 S_2 + b_3\dot{\theta}_2 C_1 C_2 \tag{A.119}$$

$$\dot{J}_{22} = \dot{b}_3 S_1 C_2 + b_3\dot{\theta}_1 C_1 C_2 - b_3\dot{\theta}_2 S_1 S_2 \tag{A.120}$$

$$\dot{J}_{23} = \dot{\theta}_1 C_1 S_2 + \dot{\theta}_2 S_1 C_2 \tag{A.121}$$

$$\dot{J}_{31} = 0 \tag{A.122}$$

$$\dot{J}_{32} = \dot{b}_3 S_2 + b_3\dot{\theta}_2 C_2 \tag{A.123}$$

$$\dot{J}_{33} = \dot{\theta}_2 S_2 \tag{A.124}$$

$$\dot{J}_{42} = \dot{\theta}_1 C_1 \tag{A.125}$$

$$\dot{J}_{52} = \dot{\theta}_1 S_1 \tag{A.126}$$

$$\dot{J}_{41} = \dot{J}_{43} = \dot{J}_{51} = \dot{J}_{53} = \dot{J}_{61} = \dot{J}_{62} = \dot{J}_{63} = 0 \tag{A.127}$$

A.4. Manipulador SnAM

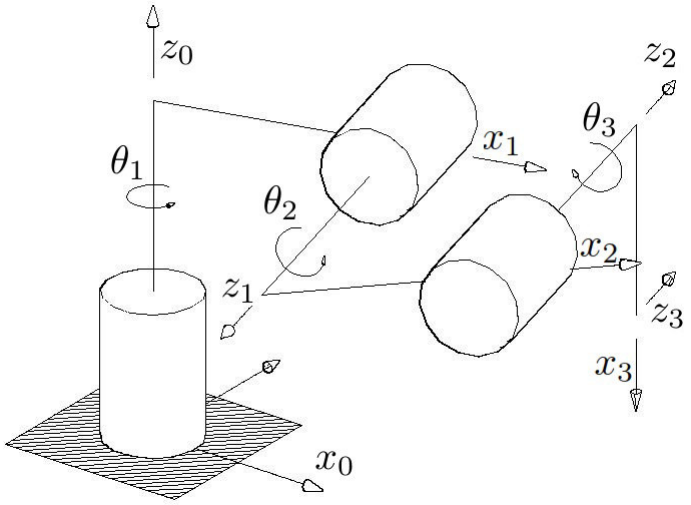

Figura A.7: Configuración del manipulador SnAM

A.4.1. Parámetros D-H

i	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	a_1	90
2	θ_2^*	b_2	a_2	180
3	θ_3^*	b_3	a_3	0

Cuadro A.4: Parámetros H-D del manipulador SnAM

$$T_0^3 = B_1 B_2 B_3 \tag{A.128}$$

$$T_0^3 = \begin{bmatrix} C_1 & 0 & S_1 & a_1 C_1 \\ S_1 & 0 & -C_1 & a_1 S_1 \\ 0 & 1 & 0 & b_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_2 & 0 & a_2 C_2 \\ S_2 & -C_2 & 0 & a_2 S_2 \\ 0 & 0 & -1 & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.129}$$

$$T_0^3 = \begin{bmatrix} C_1C_{2-3} & C_1S_{2-3} & -S_1 & a_1C_1 + a_2C_1C_2 + b_2S_1 - b_3S_1 + a_3C_1C_{2-3} \\ S_1C_{2-3} & S_1S_{2-3} & C_1 & a_1S_1 + a_2S_1C_2 - b_2C_1 + b_3C_1 + a_3S_1C_{2-3} \\ S_{2-3} & -C_{2-3} & 0 & b_1 + a_2S_2 + a_3S_{2-3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.130}$$

A.4.2. Cinemática inversa

La cinemática inversa para el prototipo genérico fue desarrollada en [14] previo a este proyecto.

A.4.3. Jacobiano del manipulador SnAM

$$J = \begin{bmatrix} \mathbf{e}_0 \times (\mathbf{o}_3 - \mathbf{o}_0) & \mathbf{e}_1 \times (\mathbf{o}_3 - \mathbf{o}_1) & \mathbf{e}_2 \times (\mathbf{o}_3 - \mathbf{o}_2) \\ \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \tag{A.131}$$

Donde:

$$\mathbf{o}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.132}$$

$$\mathbf{e}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.133}$$

$$\mathbf{e}_1 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix} \tag{A.134}$$

$$\mathbf{e}_2 = \begin{bmatrix} -S_1 \\ C_1 \\ 0 \end{bmatrix} \tag{A.135}$$

De donde obtenemos:

$$J = \begin{bmatrix} -(a_1S_1 + a_2S_1C_2 - b_2C_1 + b_3C_1 + a_3S_1C_{2-3}) & -(a_2C_1S_2 + a_3C_1S_{2-3}) & a_3C_1S_{2-3} \\ a_1C_1 + a_2C_1C_2 + b_2S_1 - b_3S_1 + a_3C_1C_{2-3} & -(a_2S_1S_2 + a_3S_1S_{2-3}) & a_3S_1S_{2-3} \\ 0 & a_2C_2 + a_3C_{2-3} & -a_3C_{2-3} \\ 0 & S_1 & -C_1 \\ 0 & -C_1 & S_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{A.136}$$

A.4.4. Cinemática de velocidad

$$\dot{X} = J(\mathbf{q})\dot{\mathbf{q}} \tag{A.137}$$

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{X} \tag{A.138}$$

$$J^{-1}(\mathbf{q}) = \frac{[AdjJ(\mathbf{q})]^T}{Det(J(\mathbf{q}))} \tag{A.139}$$

$$A = [AdjJ(\mathbf{q})]^T \tag{A.140}$$

Donde:

$$A_{11} = a_2 a_3 S_1 S_3 \tag{A.141}$$

$$A_{12} = -a_2 a_3 C_1 S_3 \tag{A.142}$$

$$A_{13} = 0 \tag{A.143}$$

$$A_{21} = a_1 a_3 C_1 C_{2-3} + a_2 a_3 C_1 C_2 C_{2-3} + a_3 b_2 S_1 C_{2-3} - a_3 b_3 S_1 C_{2-3} + a_3^2 C_1 C_{2-3}^2 \tag{A.144}$$

$$A_{22} = a_1 a_3 S_1 C_{2-3} + a_2 a_3 S_1 C_2 C_{2-3} - a_3 b_2 C_1 C_{2-3} + a_3 b_3 C_1 C_{2-3} + a_3^2 S_1 C_{2-3}^2 \tag{A.145}$$

$$A_{23} = a_1 a_3 S_{2-3} + a_2 a_3 C_2 S_{2-3} + a_3^2 S_{2-3} C_{2-3} \tag{A.146}$$

$$A_{31} = a_1 a_2 C_1 C_2 + a_1 a_3 C_1 C_{2-3} + a_2^2 C_1 C_2^2 + a_2 a_3 C_1 C_2 C_{2-3} + a_2 b_2 S_1 C_2 + a_3 b_2 S_1 C_{2-3} - a_2 b_3 S_1 C_2 - a_3 b_3 S_1 C_{2-3} + a_2 a_3 C_1 C_2 C_{2-3} + a_3^2 C_1 C_{2-3}^2 \tag{A.147}$$

$$A_{32} = a_1 a_2 S_1 C_2 + a_1 a_3 S_1 C_{2-3} + a_2^2 S_1 C_2^2 + a_2 a_3 S_1 C_2 C_{2-3} - a_2 b_2 C_1 C_2 - a_3 b_2 C_1 C_{2-3} + a_2 b_3 C_1 C_2 + a_3 b_3 C_1 C_{2-3} + a_2 a_3 S_1 C_2 C_{2-3} + a_3^2 S_1 C_{2-3}^2 \tag{A.148}$$

$$A_{33} = a_1 a_2 S_2 + a_1 a_3 S_{2-3} + a_2^2 S_2 C_2 + a_2 a_3 S_{22-3} + a_3 S_{2-3} C_{2-3} \tag{A.149}$$

$$Det(J(\mathbf{q})) = -a_2 a_3 S_3 (a_1 + a_2 C_2 + a_3 C_{2-3}) \tag{A.150}$$

$$\ddot{X} = \dot{J}(\mathbf{q})\dot{\mathbf{q}} + J(\mathbf{q})\ddot{\mathbf{q}} \tag{A.151}$$

$$\ddot{\mathbf{q}} = J^{-1}(\mathbf{q})\left(\ddot{X} - \dot{J}(\mathbf{q})\dot{\mathbf{q}}\right) \tag{A.152}$$

$$\begin{aligned}\dot{J}_{11} = &-a_1\dot{\theta}_1 C_1 - a_2\dot{\theta}_1 C_1 C_2 + a_2\dot{\theta}_2 S_1 S_2 - b_2\dot{\theta}_1 S_1 + b_3\dot{\theta}_1 S_1 \\ &- a_3\dot{\theta}_1 C_1 C_{2-3} + a_3(\dot{\theta}_2 - \dot{\theta}_3) S_1 S_{2-3}\end{aligned} \tag{A.153}$$

$$\dot{J}_{12} = a_2\dot{\theta}_1 S_1 S_2 - a_2\dot{\theta}_2 C_1 C_2 + a_3\dot{\theta}_1 S_1 S_{2-3} - a_3(\dot{\theta}_2 - \dot{\theta}_3) C_1 C_{2-3} \tag{A.154}$$

$$\dot{J}_{13} = -a_3\dot{\theta}_1 S_1 S_{2-3} + a_3(\dot{\theta}_2 - \dot{\theta}_3) C_1 C_{2-3} \tag{A.155}$$

$$\begin{aligned}\dot{J}_{21} = &-a_1\dot{\theta}_1 S_1 - a_2\dot{\theta}_1 S_1 C_2 - a_2\dot{\theta}_2 C_1 S_2 + b_2\dot{\theta}_1 C_1 - b_3\dot{\theta}_1 C_1 \\ &- a_3\dot{\theta}_1 S_1 C_{2-3} - a_3(\dot{\theta}_2 - \dot{\theta}_3) C_1 S_{2-3}\end{aligned} \tag{A.156}$$

$$\dot{J}_{22} = -a_2\dot{\theta}_1 C_1 S_2 - a_2\dot{\theta}_2 S_1 C_2 - a_3\dot{\theta}_1 C_1 S_{2-3} - a_3(\dot{\theta}_2 - \dot{\theta}_3) S_1 C_{2-3} \tag{A.157}$$

$$\dot{J}_{23} = a_3\dot{\theta}_1 C_1 S_{2-3} + a_3(\dot{\theta}_2 - \dot{\theta}_3) S_1 C_{2-3} \tag{A.158}$$

$$\dot{J}_{31} = 0 \tag{A.159}$$

$$\dot{J}_{32} = -a_2\dot{\theta}_2 S_2 - a_3(\dot{\theta}_2 - \dot{\theta}_3) S_{2-3} \tag{A.160}$$

$$\dot{J}_{33} = a_3(\dot{\theta}_2 - \dot{\theta}_3) S_{2-3} \tag{A.161}$$

Apéndice B

Códigos

B.1. Códigos para cinemática inversa de los manipuladores propuestos.

A continuación se presentan los códigos implementados para la solución de la cinemática inversa de los manipuladores propuestos. Los códigos implementados son basados en el análisis mostrado en el apéndice A.

B.1.1. Manipulador Antropomórfico.

Código B.1: Código de cinemática inversa del manipulador Antropomórfico

```
BOOL CMAnthropomorphic::SetInversePosition(D_H* pDH, CartPoint
    point)
{
        double r, c, d;
        pDH->theta[0]=atan2(point.y,point.x);
        r=sqrt(point.x*point.x+point.y*point.y);
        c=sqrt(r*r+(point.z-pDH->b[0])*(point.z-pDH->b[0]));
        d=(c*c-pDH->a[1]*pDH->a[1]-pDH->a[2]*pDH->a[2])/(2*pDH->a
    [1]*pDH->a[2]);
        pDH->theta[2]=atan2(sqrt(1-d*d),d);
        pDH->theta[1]=atan2(point.z-pDH->b[0],r)+atan2(pDH->a[2]*
    sin(pDH->theta[2]),pDH->a[1]+pDH->a[2]*cos(pDH->theta[2]));
        return TRUE;
}
```

B.1.2. Manipulador SCARA.

Código B.2: Código de cinemática inversa del manipulador SCARA

```
BOOL CMSCARA::SetInversePosition(D_H* pDH, CartPoint point)
```

```
{
        double r, c, d;
        pDH->b[2]=pDH->b[0]-point.z;
        c=sqrt(point.x*point.x+point.y*point.y);
        d=(c*c-pDH->a[0]*pDH->a[0]-pDH->a[1]*pDH->a[1])/(2*pDH->a
    [0]*pDH->a[1]);
        pDH->theta[1]=atan2(sqrt(1-d*d),d);
        pDH->theta[0]=atan2(point.y,point.x)-atan2(pDH->a[1]*sin(
    pDH->theta[1]),pDH->a[0]+pDH->a[1]*cos(pDH->theta[1]));
        return TRUE;
}
```

B.1.3. Manipulador esférico

Código B.3: Código de cinemática inversa del manipulador esférico

```
BOOL CMSpherical::SetInversePosition(D_H* pDH, CartPoint point)
{
        double r;
        pDH->theta[0]=atan2(point.y,point.x);
        r=sqrt(point.x*point.x+point.y*point.y);
        pDH->theta[1]=PI/2+atan2(point.z-pDH->b[0],r);
        pDH->b[2]=sqrt(r*r+((point.z-pDH->b[0])*(point.z-pDH->b[0])
    ));
        return TRUE;
}
```

B.1.4. Manipulador SnAM

Código B.4: Código de cinemática inversa del manipulador SnAM

```
BOOL CMCustom::SetInversePosition(D_H* pDH, CartPoint point)
{
        return SetInvPos3R90_0_90(pDH,point);
}
```

La función `SetInvPos3R90_0_90(pDH,point);` fue desarrollada previamente a este proyecto, se puede ver una explicación de su funcionamiento en [14].

B.2. Códigos de la función GetPoVeAcParameters

Se presentan las funciones `GetPoVeAcParameters` para cada uno de los manipuladores propuestos. Como se puede observar, primero se hace la lectura de los datos que representan a la trayectoria y el acondicionamiento de los mismos. Después se calculan los jacobianos, la velocidad y aceleración de las articulaciones. Por último se imprimen los resultados en un archivo `.csv`.

B.2.1. Manipulador antropomórfico.

Código B.5: Código de cinemática inversa del manipulador SCARA

```
void CMAnthropomorphic::GetPoVeAcParameters(void)
{
        float s1[300], c1[300], s2[300], c2[300];
        float s23[300],c23[300];
        double a1=pDH->a[0];
        double a2=pDH->a[1];
        double a3=pDH->a[2];
        double b1=pDH->b[0];
        double b2=pDH->b[1];
        double b3=pDH->b[2];
        double c;
        int j,k;
        FILE *points;
        FILE *posicion;
        CartPoint P[300];
        float Angles[300][3];
        float           theta1[300],theta2[300], theta3[300];
        Matrix J[300],Jinv[300],Jp[300];
        double vel[300][3],acel[300][3];
        Vector  qp[300],q2p[300];

        CString str;
        if((fopen_s(&points,"points.csv", "r"))==TRUE)
                AfxMessageBox("Imposible abrir archivo");
    else
    {
        do{
       fscanf(points,"%lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf"
    ,&P[j].x,&P[j].y,&P[j].z,&vel[j][0],&vel[j][1],&vel[j][2],&acel
    [j][0],&acel[j][1],&acel[j][2]);
           j++;
         }while(!feof(points));
        j--;
        }
        fclose(points);


        //Jacobian for anthropomorphic
        for(k=0;k<j;k++)
        {
                SetInversePosition(pDH,P[k]);
                theta1[k]=pDH->theta[0];
                theta2[k]=pDH->theta[1];
                theta3[k]=pDH->theta[2];

                s1[k]=sin(theta1[k]);
                c1[k]=cos(theta1[k]);
                s2[k]=sin(theta2[k]);
                c2[k]=cos(theta2[k]);
                s23[k]=sin(theta2[k]+theta3[k]);
                c23[k]=cos(theta2[k]+theta3[k]);
        }
```

```
        for(k=0;k<j;k++)
        {
                J[k].col=3;
                J[k].row=3;

                J[k].q[0][0]=-(a2*s1[k]*c2[k]+a3*s1[k]*c23[k]);
                J[k].q[0][1]=-(a2*c1[k]*s2[k]+a3*c1[k]*s23[k]);
                J[k].q[0][2]=-a3*c1[k]*s23[k];

                J[k].q[1][0]=-(a2*c1[k]*c2[k]+a3*c1[k]*c23[k]);
                J[k].q[1][1]=-(a2*s1[k]*s2[k]+a3*s1[k]*s23[k]);
                J[k].q[1][2]=-a3*s1[k]*s23[k];

                J[k].q[2][0]=0;
                J[k].q[2][1]=a2*c2[k]+a3*c23[k];
                J[k].q[2][2]=a3*c23[k];

                J[k].q[3][0]=0;
                J[k].q[3][1]=s1[k];
                J[k].q[3][2]=s1[k];

                J[k].q[4][0]=0;
                J[k].q[4][1]=-c1[k];
                J[k].q[4][2]=-c1[k];

                J[k].q[5][0]=1;
                J[k].q[5][1]=0;
                J[k].q[5][2]=0;
        }
        for(k=0;k<j;k++)
        {
                Jinv[k].col=3;
                Jinv[k].row=3;
                Jinv[k]=GetInverse(J[k]);
        }
        //dtheta calculation
        for(k=0; k<j; k++)
        {
                qp[k].row=3;
                qp[k]=MatVecProd(Jinv[k],vel[k]);
        }

        //Compute Aceleration Data
        double thp1[300],thp2[300],thp3[300];
        for(k=0;k<j;k++)
        {
                thp1[k]=qp[k].v[0];
                thp2[k]=qp[k].v[1];
                thp3[k]=qp[k].v[2];

                Jp[k].col=3;
                Jp[k].row=3;

                Jp[k].q[0][0]=-a2*thp1[k]*c1[k]*c2[k]+a2*thp2[k]*s1
[k]*s2[k]-a3*thp1[k]*c1[k]*c23[k]+a3*(thp2[k]+thp3[k])*s1[k]*
s23[k];
```

```
105                 Jp[k].q[0][1]=a2*thp1[k]*s1[k]*s2[k]-a2*thp2[k]*c1[
    k]*c2[k]+a3*thp1[k]*s1[k]*s23[k]-a3*(thp2[k]+thp3[k])*c1[k]*c23
    [k];
106                 Jp[k].q[0][2]=a3*thp1[k]*s1[k]*s23[k]-a3*(thp2[k]+
    thp3[k])*c1[k]*c23[k];
107
108                 Jp[k].q[1][0]=a2*thp1[k]*s1[k]*c2[k]+a2*thp2[k]*c1[
    k]*s2[k]+a3*thp1[k]*s1[k]*c23[k]+a3*(thp2[k]+thp3[k])*c1[k]*s23
    [k];
109                 Jp[k].q[1][1]=-a2*thp1[k]*c1[k]*s2[k]-a2*thp2[k]*s1
    [k]*c2[k]-a3*thp1[k]*c1[k]*s23[k]-a3*(thp2[k]+thp3[k])*s1[k]*
    c23[k];
110                 Jp[k].q[1][2]=-a3*thp1[k]*c1[k]*s23[k]-a3*(thp2[k]+
    thp3[k])*s1[k]*c23[k];
111
112                 Jp[k].q[2][0]=0;
113                 Jp[k].q[2][1]=-a2*thp2[k]*s2[k]-a3*(thp2[k]+thp3[k
    ])*s23[k];
114                 Jp[k].q[2][2]=-a3*(thp2[k]+thp3[k])*s23[k];
115
116                 Jp[k].q[3][0]=0;
117                 Jp[k].q[3][1]=thp1[k]*c1[k];
118                 Jp[k].q[3][2]=thp1[k]*c1[k];;
119
120                 Jp[k].q[4][0]=0;
121                 Jp[k].q[4][1]=thp1[k]*s1[k];;
122                 Jp[k].q[4][2]=thp1[k]*s1[k];;
123
124                 Jp[k].q[5][0]=0;
125                 Jp[k].q[5][1]=0;
126                 Jp[k].q[5][2]=0;
127         }
128         //This is the final operation for d2theta calculation
129         for(k=0; k<j; k++)
130         {
131                 double aux[3],aux2[3];
132                 q2p[k].row=3;
133
134                 aux[0]=qp[k].v[0];
135                 aux[1]=qp[k].v[1];
136                 aux[2]=qp[k].v[2];
137
138                 Vector arg;
139                 arg.row=3;
140                 arg=MatVecProd(Jp[k],aux);
141                 aux2[0]=acel[k][0]-arg.v[0];
142                 aux2[1]=acel[k][1]-arg.v[1];
143                 aux2[2]=acel[k][2]-arg.v[2];
144                 q2p[k]=MatVecProd(Jinv[k],aux2);
145         }
146
147         for(k=0;k<j;k++)
148         {
149         if((fopen_s(&posicion,"resultadosantropomorfico.csv", "a+")
    )==TRUE);
150         else
151         {
```

```
152         if(k==0)
153         {
154         fprintf_s(posicion,"px, py, pz, th1, th2, th3, thp1, thp2,
    thp3, th2p1, th2p2, th2p3\n");
155         }
156         fprintf_s(posicion,"%d, %f, %f, %f, %f, %f, %f, %lf, %lf, %
    lf, %lf, %lf, %lf\n",k,P[k].x,P[k].y,P[k].z,theta1[k],theta2[k
    ],theta3[k],qp[k].v[0],qp[k].v[1],qp[k].v[2],q2p[k].v[0],q2p[k
    ].v[1],q2p[k].v[2]);
157         fclose(posicion);
158     }
159         }
160 }
```

B.2.2. Manipulador SCARA.

Código B.6: Código de cinemática inversa del manipulador SCARA

```
1
2  void CMSCARA::GetPoVeAcParameters(void)
3  {
4          float s1[300], c1[300], s2[300], c2[300];
5          float s12[300],c12[300];
6          double a1=pDH->a[0];
7          double a2=pDH->a[1];
8          double a3=pDH->a[2];
9          double b1=pDH->b[0];
10         double b2=pDH->b[1];
11         double b3[300];
12         double c;
13         int j,k;
14         FILE *points;
15         FILE *posicion;
16         CartPoint P[300];
17         float Angles[300][3];
18         float theta1[300],theta2[300];
19         Matrix J[300],Jinv[300],Jp[300];
20         double vel[300][3],acel[300][3];
21         Vector  qp[300],q2p[300];
22
23         CString str;
24         if((fopen_s(&points,"points.csv", "r"))==TRUE)
25         AfxMessageBox("Imposible abrir archivo");
26     else
27     {
28         do{
29        fscanf(points,"%lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf"
    ,&P[j].x,&P[j].y,&P[j].z,&vel[j][0],&vel[j][1],&vel[j][2],&acel
    [j][0],&acel[j][1],&acel[j][2]);
30                 j++;
31           }while(!feof(points));
32           j--;
33           }
34         fclose(points);
```

```
35          //Jacobian for SCARA
36          for(k=0;k<j;k++)
37          {
38                  SetInversePosition(pDH,P[k]);
39                  theta1[k]=pDH->theta[0];
40                  theta2[k]=pDH->theta[1];
41                  //theta3[k]=pDH->theta[2];
42                  b3[k]=pDH->b[2];
43
44                  s1[k]=sin(theta1[k]);
45                  c1[k]=cos(theta1[k]);
46                  s2[k]=sin(theta2[k]);
47                  c2[k]=cos(theta2[k]);
48                  s12[k]=sin(theta1[k]+theta2[k]);
49                  c12[k]=cos(theta1[k]+theta2[k]);
50          }
51          for(k=0;k<j;k++)
52          {
53                  J[k].col=3;
54                  J[k].row=3;
55
56                  J[k].q[0][0]=-(a1*s1[k]+a2*s12[k]);
57                  J[k].q[0][1]=-a2*s12[k];
58                  J[k].q[0][2]=-0;
59
60                  J[k].q[1][0]=(a1*c1[k]+a2*c12[k]);
61                  J[k].q[1][1]=a2*c12[k];
62                  J[k].q[1][2]=0;
63
64                  J[k].q[2][0]=0;
65                  J[k].q[2][1]=0;
66                  J[k].q[2][2]=-1;
67
68                  J[k].q[3][0]=0;
69                  J[k].q[3][1]=0;
70                  J[k].q[3][2]=0;
71
72                  J[k].q[4][0]=0;
73                  J[k].q[4][1]=0;
74                  J[k].q[4][2]=0;
75
76                  J[k].q[5][0]=1;
77                  J[k].q[5][1]=1;
78                  J[k].q[5][2]=0;
79          }
80          for(k=0;k<j;k++)
81          {
82                  Jinv[k].col=3;
83                  Jinv[k].row=3;
84                  Jinv[k]=GetInverse(J[k]);
85          }
86          //dtheta calculation
87          for(k=0; k<j; k++)
88          {
89                  qp[k].row=3;
90                  qp[k]=MatVecProd(Jinv[k],vel[k]);
91          }
```

```
92
93          //Compute Aceleration Data
94          double thp1[300],thp2[300],thp3[300];
95          for(k=0;k<j;k++)
96          {
97
98                  thp1[k]=qp[k].v[0];
99                  thp2[k]=qp[k].v[1];
100                 Jp[k].col=3;
101                 Jp[k].row=3;
102
103                 Jp[k].q[0][0]=-a1*thp1[k]*c1[k]-a2*(thp1[k]+thp2[k
    ])*c12[k];
104                 Jp[k].q[0][1]=-a2*(thp1[k]+thp2[k])*c12[k];
105                 Jp[k].q[0][2]=0;
106
107                 Jp[k].q[1][0]=-a1*thp1[k]*s1[k]-a2*(thp1[k]+thp2[k
    ])*s12[k];
108                 Jp[k].q[1][1]=-a2*(thp1[k]+thp2[k])*s12[k];;
109                 Jp[k].q[1][2]=0;
110
111                 Jp[k].q[2][0]=0;
112                 Jp[k].q[2][1]=0;
113                 Jp[k].q[2][2]=0;
114
115                 Jp[k].q[3][0]=0;
116                 Jp[k].q[3][1]=0;
117                 Jp[k].q[3][2]=0;
118
119                 Jp[k].q[4][0]=0;
120                 Jp[k].q[4][1]=0;
121                 Jp[k].q[4][2]=0;
122
123                 Jp[k].q[5][0]=0;
124                 Jp[k].q[5][1]=0;
125                 Jp[k].q[5][2]=0;
126         }
127         //This is the final operation for d2theta calculation
128         for(k=0; k<j; k++)
129         {
130                 double aux[3],aux2[3];
131                 q2p[k].row=3;
132
133                 aux[0]=qp[k].v[0];
134                 aux[1]=qp[k].v[1];
135                 aux[2]=qp[k].v[2];
136
137                 Vector arg;
138                 arg.row=3;
139                 arg=MatVecProd(Jp[k],aux);
140                 aux2[0]=acel[k][0]-arg.v[0];
141                 aux2[1]=acel[k][1]-arg.v[1];
142                 aux2[2]=acel[k][2]-arg.v[2];
143                 q2p[k]=MatVecProd(Jinv[k],aux2);
144         }
145
146         for(k=0;k<j;k++)
```

```
147                 {
148                 if((fopen_s(&posicion,"resultadoSCARA.csv", "a+"))
    ==TRUE);
149                 else
150                 {
151                 if(k==0)
152                 {
153                 fprintf_s(posicion,"n, px, py, pz, th1, th2, b3,
    thp1, thp2, bp3, th2p1, th2p2, th2p3\n");
154                 }
155                 fprintf_s(posicion,"%d, %f, %f, %f, %f, %f, %f, %lf
    , %lf, %lf, %lf, %lf, %lf\n",k,P[k].x,P[k].y,P[k].z,theta1[k],
    theta2[k],b3[k],qp[k].v[0],qp[k].v[1],qp[k].v[2],q2p[k].v[0],
    q2p[k].v[1],q2p[k].v[2]);
156                 fclose(posicion);
157             }
158             }
159 }
```

B.2.3. Manipulador esférico.

Código B.7: Código de cinemática inversa del manipulador SCARA

```
1
2  void CMSpherical::GetPoVeAcParameters(void)
3  {
4           float s1[300], c1[300], s2[300], c2[300];
5           float s12[300],c12[300];
6           double a1=pDH->a[0];
7           double a2=pDH->a[1];
8           double a3=pDH->a[2];
9           double b1=pDH->b[0];
10          double b2=pDH->b[1];
11          double b3[300];
12          double c;
13          int j,k;
14          FILE *points;
15          FILE *posicion;
16          CartPoint P[300];
17          float Angles[300][3];
18          float theta1[300],theta2[300], theta3[300];
19          Matrix J[300],Jinv[300],Jp[300];
20          double vel[300][3],acel[300][3];
21          Vector  qp[300],q2p[300];
22
23          CString str;
24          if((fopen_s(&points,"points.csv", "r"))==TRUE)
25                  AfxMessageBox("Imposible abrir archivo");
26      else
27      {
28          do{
29         fscanf(points,"%lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf, %lf"
    ,&P[j].x,&P[j].y,&P[j].z,&vel[j][0],&vel[j][1],&vel[j][2],&acel
    [j][0],&acel[j][1],&acel[j][2]);
```

```
                j++;
                }while(!feof(points));
                j--;
           }
        fclose(points);

        //Jacobian for spherical
        for(k=0;k<j;k++)
        {
                SetInversePosition(pDH,P[k]);
                theta1[k]=pDH->theta[0];
                theta2[k]=pDH->theta[1];
                b3[k]=pDH->b[2];

                s1[k]=sin(theta1[k]);
                c1[k]=cos(theta1[k]);
                s2[k]=sin(theta2[k]);
                c2[k]=cos(theta2[k]);
        }
        for(k=0;k<j;k++)
        {
                J[k].col=3;
                J[k].row=3;

                J[k].q[0][0]=-b3[k]*s1[k]*s2[k];
                J[k].q[0][1]=b3[k]*c1[k]*c2[k];
                J[k].q[0][2]=c1[k]*s2[k];

                J[k].q[1][0]=b3[k]*c1[k]*s2[k];
                J[k].q[1][1]=b3[k]*s1[k]*c2[k];
                J[k].q[1][2]=s1[k]*s2[k];

                J[k].q[2][0]=0;
                J[k].q[2][1]=b3[k]*s2[k];
                J[k].q[2][2]=-c2[k];

                J[k].q[3][0]=0;
                J[k].q[3][1]=s1[k];
                J[k].q[3][2]=0;

                J[k].q[4][0]=0;
                J[k].q[4][1]=-c1[k];
                J[k].q[4][2]=0;

                J[k].q[5][0]=1;
                J[k].q[5][1]=0;
                J[k].q[5][2]=0;
        }
        for(k=0;k<j;k++)
        {
                Jinv[k].col=3;
                Jinv[k].row=3;
                Jinv[k]=GetInverse(J[k]);
        }
        //dtheta calculation
        for(k=0; k<j; k++)
        {
```

```
                qp[k].row=3;
                qp[k]=MatVecProd(Jinv[k],vel[k]);
        }

        //Compute Aceleration Data
        double thp1[300],thp2[300],bp3[300];
        for(k=0;k<j;k++)
        {

                thp1[k]=qp[k].v[0];
                thp2[k]=qp[k].v[1];
                bp3[k]=qp[k].v[2];

                Jp[k].col=3;
                Jp[k].row=3;

                Jp[k].q[0][0]=-bp3[k]*s1[k]*s2[k]-b3[k]*thp1[k]*c1[
    k]*s2[k]-b3[k]*thp2[k]*s1[k]*c2[k];
                Jp[k].q[0][1]=bp3[k]*c1[k]*c2[k]-b3[k]*thp1[k]*s1[k
    ]*c2[k]-b3[k]*thp2[k]*c1[k]*s2[k];
                Jp[k].q[0][2]=-thp1[k]*s1[k]*s2[k]+thp2[k]*c1[k]*c2
    [k];

                Jp[k].q[1][0]=bp3[k]*c1[k]*s2[k]-b3[k]*thp1[k]*s1[k
    ]*s2[k]+b3[k]*thp2[k]*c1[k]*c2[k];
                Jp[k].q[1][1]=bp3[k]*s1[k]*c2[k]+b3[k]*thp1[k]*c1[k
    ]*c2[k]-b3[k]*thp2[k]*s1[k]*s2[k];
                Jp[k].q[1][2]=thp1[k]*c1[k]*s2[k]+thp2[k]*s1[k]*c2[
    k];

                Jp[k].q[2][0]=0;
                Jp[k].q[2][1]=bp3[k]*s2[k]+b3[k]*thp2[k]*c2[k];
                Jp[k].q[2][2]=thp2[k]*s2[k];

                Jp[k].q[3][0]=0;
                Jp[k].q[3][1]=thp1[k]*c1[k];
                Jp[k].q[3][2]=0;

                Jp[k].q[4][0]=0;
                Jp[k].q[4][1]=thp1[k]*s1[k];
                Jp[k].q[4][2]=0;

                Jp[k].q[5][0]=0;
                Jp[k].q[5][1]=0;
                Jp[k].q[5][2]=0;
        }
        //This is the final operation for d2theta calculation
        for(k=0; k<j; k++)
        {
                double aux[3],aux2[3];
                q2p[k].row=3;

                aux[0]=qp[k].v[0];
                aux[1]=qp[k].v[1];
                aux[2]=qp[k].v[2];

                Vector arg;
```

```
                arg.row=3;
                arg=MatVecProd(Jp[k],aux);
                aux2[0]=acel[k][0]-arg.v[0];
                aux2[1]=acel[k][1]-arg.v[1];
                aux2[2]=acel[k][2]-arg.v[2];
                q2p[k]=MatVecProd(Jinv[k],aux2);
        }

        for(k=0;k<j;k++)
        {
        if((fopen_s(&posicion,"resultadoSpherical.csv", "a+"))==
    TRUE);
        else
        {
        if(k==0)
        {
        fprintf_s(posicion,"n, px, py, pz, th1, th2, b3, thp1, thp2
    , bp3, th2p1, th2p2, th2p3\n");
        }
        fprintf_s(posicion,"%d, %f, %f, %f, %f, %f, %f, %lf, %lf, %
    lf, %lf, %lf, %lf\n",k,P[k].x,P[k].y,P[k].z,theta1[k],theta2[k
    ],b3[k],qp[k].v[0],qp[k].v[1],qp[k].v[2],q2p[k].v[0],q2p[k].v
    [1],q2p[k].v[2]);
        fclose(posicion);
        }
        }

}
```

B.2.4. Manipulador SnAM

Código B.8: Código de la función GetPoVeAcParameters para el manipulador SnAM

```

void CMCustom::GetPoVeAcParameters(void)
{
        FILE *posicion;
        int i,j=0,k;
        CString str;
        float pos[300][3];
        double vel[300][3],acel[300][3];
        Vector  qp[300],q2p[300];
        float h=0.04;
        float SnAMAngRad[300][3];
        float SnAMAng[300][3];

        D_H Dh5;
        Matrix Bi[300];
        Matrix J[300],Jinv[300],Jp[300];

        float theta1[300],theta2[300],theta3[300],s1[300],c1[300],
    s2[300],c2[300],s2m3[300],c2m3[300];
        float a1,a2,a3,b1,b2,b3;
```

```
20
21          //Recover Data from txt file
22
23          if((fopen_s(&posicion,"position.txt", "r"))==TRUE)
24                  AfxMessageBox("Imposible abrir archivo");
25      else
26          {
27                  do{
28                  fscanf(posicion,"%f, %f, %f",&SnAMAng[j][0],&
        SnAMAng[j][1],&SnAMAng[j][2]);
29                          SnAMAng[j][0]=SnAMAng[j][0]*PI/180;
30                          SnAMAng[j][1]=SnAMAng[j][1]*PI/180;
31                          SnAMAng[j][2]=SnAMAng[j][2]*PI/180;
32
33                          j++;
34                          }while(!feof(posicion));
35                          j--;
36
37          }
38          fclose(posicion);
39
40          //Adjust data by Least square method... apply three times
        for th1, th2 and th3
41          double time[3000],ang1[3000],ang2[3000],ang3[3000],ConEqth1
        [4],ConEqth2[4],ConEqth3[4];
42
43          for(i=0;i<j;i++)
44          {
45                  // this is an acquisition time of 40 ms, better if
        the time is defined like a macro
46                  time[i]=i*0.04;
47                  ang1[i]=SnAMAng[i][0];
48                  ang2[i]=SnAMAng[i][1];
49                  ang3[i]=SnAMAng[i][2];
50          }
51
52          LSquaresFit1.operation(time,ang1,j,3);
53          LSquaresFit2.operation(time,ang2,j,3);
54          LSquaresFit3.operation(time,ang3,j,3);
55
56          for(i=0;i<4;i++)
57          {
58                  ConEqth1[i]=LSquaresFit1.a[i];
59                  ConEqth2[i]=LSquaresFit2.a[i];
60                  ConEqth3[i]=LSquaresFit3.a[i];
61          }
62          ////////////////////////////////
63
64          //Compute Position data
65          for(i=0;i<j;i++)
66          {
67          Dh5=Dh4;
68          Dh5.theta[0]=ConEqth1[3]*pow(time[i],3)+ConEqth1[2]*pow(
        time[i],2)+ConEqth1[1]*time[i]+ConEqth1[0];
69          Dh5.theta[1]=ConEqth2[3]*pow(time[i],3)+ConEqth2[2]*pow(
        time[i],2)+ConEqth2[1]*time[i]+ConEqth2[0];
```

```
70          Dh5.theta[2]=ConEqth3[3]*pow(time[i],3)+ConEqth3[2]*pow(
    time[i],2)+ConEqth3[1]*time[i]+ConEqth3[0];
71
72          Bi[i]=FullForwardKinematics(&Dh5);
73          Bi[i]=FullForwardKinematics(&Dh5);
74          pos[i][0]=Bi[i].q[0][3];
75          pos[i][1]=Bi[i].q[1][3];
76          pos[i][2]=Bi[i].q[2][3];
77
78          theta1[i]=Dh5.theta[0];
79          theta2[i]=Dh5.theta[1];
80          theta3[i]=Dh5.theta[2];
81          }
82          //Compute Velocity data
83          ///***JACOBIAN***
84          a1=Dh4.a[0];
85          a2=Dh4.a[1];
86          a3=Dh4.a[2];
87          b1=Dh4.b[0];
88          b2=Dh4.b[1];
89          b3=Dh4.b[2];
90          for(k=0;k<j;k++)
91          {
92                   s1[k]=sin(theta1[k]);
93                   c1[k]=cos(theta1[k]);
94                   s2[k]=sin(theta2[k]);
95                   c2[k]=cos(theta2[k]);
96                   s2m3[k]=sin(theta2[k]-theta3[k]);
97                   c2m3[k]=cos(theta2[k]-theta3[k]);
98          }
99          for(k=0;k<j;k++)
100         {
101                  J[k].col=3;
102                  J[k].row=3;
103                  J[k].q[0][0]=-(a1*s1[k]+a2*s1[k]*c2[k]-b2*c1[k]+b3*
    c1[k]+a3*s1[k]*c2m3[k]);
104                  J[k].q[0][1]=-(a2*c1[k]*s2[k]+a3*c1[k]*s2m3[k]);
105                  J[k].q[0][2]=a3*c1[k]*s2m3[k];
106
107                  J[k].q[1][0]=a1*c1[k]+a2*c1[k]*c2[k]+b2*s1[k]-b3*s1
    [k]+a3*c1[k]*c2m3[k];
108                  J[k].q[1][1]=-(a2*s1[k]*s2[k]+a3*s1[k]*s2m3[k]);
109                  J[k].q[1][2]=a3*s1[k]*s2m3[k];
110
111                  J[k].q[2][0]=0;
112                  J[k].q[2][1]=a2*c2[k]+a3*c2m3[k];
113                  J[k].q[2][2]=-a3*c2m3[k];
114
115                  J[k].q[3][0]=0;
116                  J[k].q[3][1]=s1[k];
117                  J[k].q[3][2]=-s1[k];
118                  J[k].q[4][0]=0;
119                  J[k].q[4][1]=-c1[k];
120                  J[k].q[4][2]=c1[k];
121                  J[k].q[5][0]=1;
122                  J[k].q[5][1]=0;
123                  J[k].q[5][2]=0;
```

```
124         }
125         for(k=0;k<j;k++)
126         {
127                 Jinv[k].col=3;
128                 Jinv[k].row=3;
129                 Jinv[k]=GetInverse(J[k]);
130         }
131
132         for(i=0;i<j;i++)
133         {
134
135                         vel[i][0]=(pos[i+1][0]-pos[i-1][0])/(2*h);
136                         vel[i][1]=(pos[i+1][1]-pos[i-1][1])/(2*h);
137                         vel[i][2]=(pos[i+1][2]-pos[i-1][2])/(2*h);
138
139                         acel[i][0]=(pos[i+1][0]-2*pos[i][0]+pos[i
    -1][0])/(h*h);
140                         acel[i][1]=(pos[i+1][1]-2*pos[i][1]+pos[i
    -1][1])/(h*h);
141                         acel[i][2]=(pos[i+1][2]-2*pos[i][2]+pos[i
    -1][2])/(h*h);
142         }
143         //This is the final operation for dtheta calculation
144         for(k=0; k<j; k++)
145         {
146                 qp[k].row=3;
147                 qp[k]=MatVecProd(Jinv[k],vel[k]);
148         }
149
150         //Compute Aceleration Data
151         double thp1[300],thp2[300],thp3[300];
152         for(k=0;k<j;k++)
153         {
154                 thp1[k]=qp[k].v[0];
155                 thp2[k]=qp[k].v[1];
156                 thp3[k]=qp[k].v[2];
157
158                 Jp[k].col=3;
159                 Jp[k].row=3;
160
161                 Jp[k].q[0][0]=-a1*thp1[k]*c1[k]-a2*thp1[k]*c1[k]*c2
    [k]+a2*thp2[k]*s1[k]*s2[k]-b2*thp1[k]*s1[k]+b3*thp1[k]*s1[k]-a3
    *thp1[k]*c1[k]*c2m3[k]+a3*(thp2[k]-thp3[k])*s1[k]*s2m3[k];
162                 Jp[k].q[0][1]=a2*thp1[k]*s1[k]*s2[k]-a2*thp2[k]*c1[
    k]*c2[k]+a3*thp1[k]*s1[k]*s2m3[k]-a3*(thp2[k]-thp3[k])*c1[k]*
    c2m3[k];
163                 Jp[k].q[0][2]=-a3*thp1[k]*s1[k]*s2m3[k]+a3*(thp2[k
    ]-thp3[k])*c1[k]*c2m3[k];
164
165                 Jp[k].q[1][0]=-a1*thp1[k]*s1[k]-a2*thp1[k]*s1[k]*c2
    [k]-a2*thp2[k]*c1[k]*s2[k]+b2*thp1[k]*c1[k]-b3*thp1[k]*c1[k]-a3
    *thp1[k]*s1[k]*c2m3[k]-a3*(thp2[k]-thp3[k])*c1[k]*s2m3[k];
166                 Jp[k].q[1][1]=-a2*thp1[k]*c1[k]*s2[k]-a2*thp2[k]*s1
    [k]*c2[k]-a3*thp1[k]*c1[k]*s2m3[k]-a3*(thp2[k]-thp3[k])*s1[k]*
    c2m3[k];
167                 Jp[k].q[1][2]=a3*thp1[k]*c1[k]*s2m3[k]+a3*(thp2[k]-
    thp3[k])*s1[k]*c2m3[k];
```

```
168
169                 Jp[k].q[2][0]=0;
170                 Jp[k].q[2][1]=-a2*thp2[k]*s2[k]-a3*(thp2[k]-thp3[k
    ])*s2m3[k];
171                 Jp[k].q[2][2]=a3*(thp2[k]-thp3[k])*s2m3[k];
172
173                 Jp[k].q[3][0]=0;
174                 Jp[k].q[3][1]=thp1[k]*c1[k];
175                 Jp[k].q[3][2]=-thp1[k]*c1[k];;
176
177                 Jp[k].q[4][0]=0;
178                 Jp[k].q[4][1]=thp1[k]*s1[k];;
179                 Jp[k].q[4][2]=-thp1[k]*s1[k];;
180
181                 Jp[k].q[5][0]=0;
182                 Jp[k].q[5][1]=0;
183                 Jp[k].q[5][2]=0;
184
185
186         }
187         //This is the final operation for d2theta calculation
188         for(k=0; k<j; k++)
189         {
190                 double aux[3],aux2[3];
191                 q2p[k].row=3;
192
193                 aux[0]=qp[k].v[0];
194                 aux[1]=qp[k].v[1];
195                 aux[2]=qp[k].v[2];
196
197                 Vector arg;
198                 arg.row=3;
199                 arg=MatVecProd(Jp[k],aux);
200                 aux2[0]=acel[k][0]-arg.v[0];
201                 aux2[1]=acel[k][1]-arg.v[1];
202                 aux2[2]=acel[k][2]-arg.v[2];
203                 q2p[k]=MatVecProd(Jinv[k],aux2);
204         }
205
206         // writing results
207         for(k=0;k<j;k++)
208         {
209         if((fopen_s(&posicion,"resultados.csv", "a+"))==TRUE);
210         else
211         {
212         if(k==0)
213         {
214         fprintf_s(posicion,"n px, py, pz, th1, th2, th3, thp1, thp2
    , thp3, th2p1, th2p2, th2p3\n");
215         }
216         fprintf_s(posicion,"%d, %f, %f, %f, %f, %f, %f, %lf, %lf, %
    lf, %lf, %lf, %lf\n",k,pos[k][0],pos[k][1],pos[k][2],theta1[k],
    theta2[k],theta3[k],qp[k].v[0],qp[k].v[1],qp[k].v[2],q2p[k].v
    [0],q2p[k].v[1],q2p[k].v[2]);
217                 fclose(posicion);
218
219         }
```

```
220         }
221
222
223 for(k=0;k<j;k++)
224                     {
225                     if((fopen_s(&posicion,"points.csv", "a+"))==TRUE);
226                     else
227                     {
228                     fprintf_s(posicion,"%f, %f, %f, %f, %f, %f, %f, %f,
      %f\n",pos[k][0],pos[k][1],pos[k][2],vel[k][0],vel[k][1],vel[k
    ][2],acel[k][0],acel[k][1],acel[k][2]);
229                     fclose(posicion);
230
231                     }
232                     }
233 }
```

Apéndice C

Validación de los parámetros obtenidos mediante ADEFID

Se busca validar los parámetros de posicionamiento obtenidos mediante ADEFID. Para ello se hace uso de un software dedicado, de renombre mundial. Adams (***A**utomated **D**ynamic **A**nalysis of **M**echanical **S**ystems*), es un software que cuenta con todas las herramientas necesarias para modelar y simular sistemas mecánicos. Se hace la comparativa entre los resultados de posicionamiento obtenidos mediante ADEFID y Adams. Se observa que ambos softwares tienen el mismo comportamiento, validando así los resultados obtenidos en ADEFID.

C.1. Manipulador SnAM

Tomando un punto arbitrario en la trayectoria se obtienen los siguientes resultados mediante ADEFID:

```
px=0.4238, py=-0.1481, pz=0.0163, th1=-0.3362, th2=-0.5062, th3= 0.9423
```

Se modela el manipulador en Adams (Figura C.1) y se obtienen resultados mostrados en las figuras (C.2), (C.3) y (C.4). El valor de `px` corresponde a "Current" en la figura (C.2). Ahora para los valores `py` y `pz`, la correspondencia se ve invertida (`py` corresponde a la figura (C.4) y `pz` a (C.3)), esto debido a como son tomados los ejes de referencia dentro de Adams, pero presentan los valores esperados. También es importante resaltar que para el posicionamiento, las unidades utilizadas en ADEFID son metros, mientras que Adams utliza milímetros.

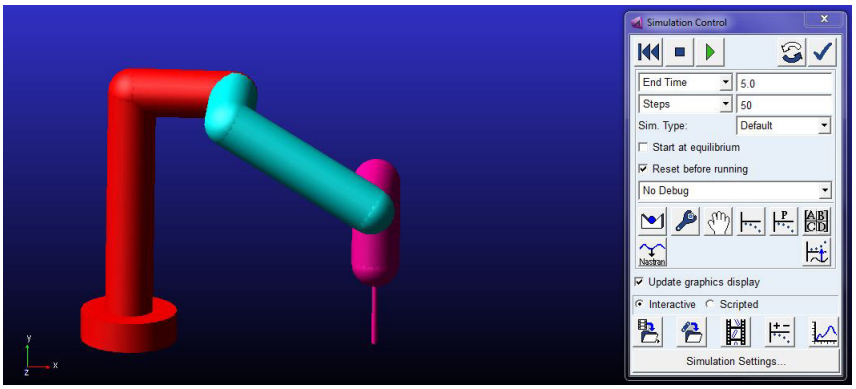

Figura C.1: Simulación prototipo SnAM mediante ADAMS.

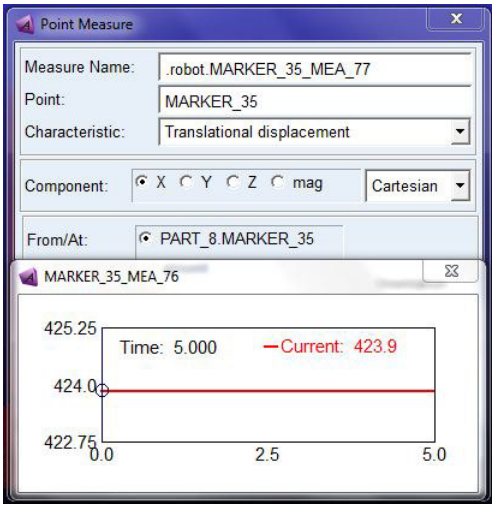

Figura C.2: Posición del efector final en X(px).

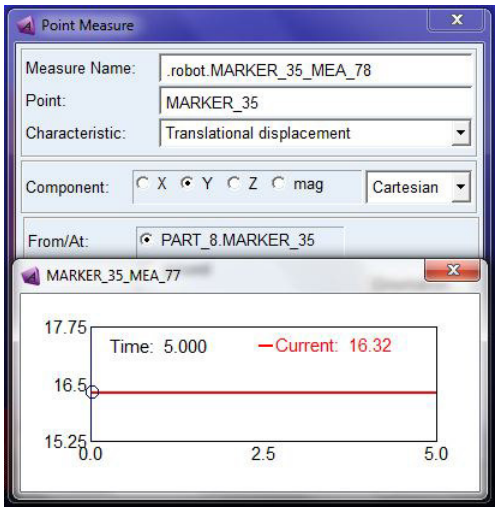

Figura C.3: Posición del efector final en Y(py).

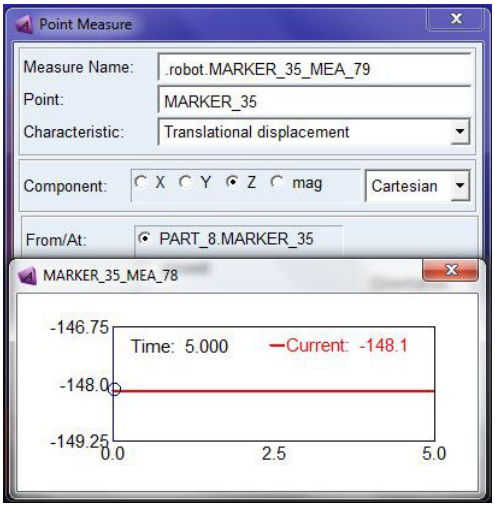

Figura C.4: Posición del efector final en Z(pz).

C.2. Antropomórfico

Ahora para el manipulador antropomófico se obtienen los siguientes valores mediante ADEFID:

```
px=0.4238, py=-0.1481, pz=0.0163, th1=-0.3362, th2=-0.5301, th3= 0.6869
```

En la figura (C.5) se presenta la simulación generada mediante Adams. Y en las figuras (C.6), (C.7) y (C.8), se presentan los valores de `px`, `py` y `pz`. Se observa que los valores son los mismos para ambos softwares.

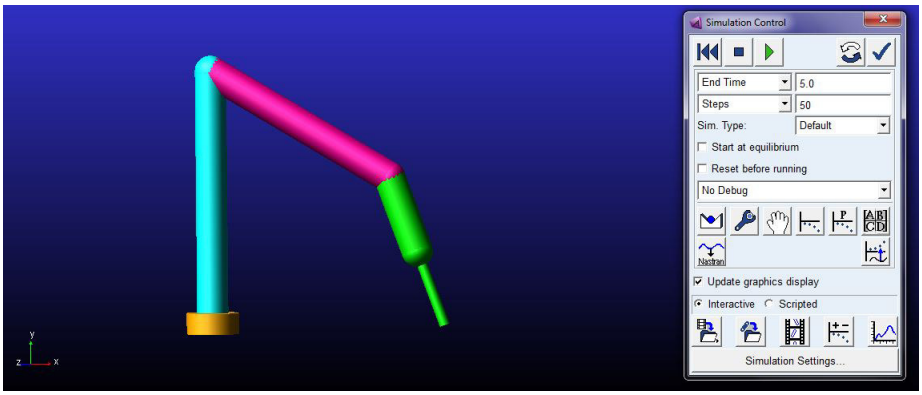

Figura C.5: Simulación prototipo Antropomórfico mediante ADAMS.

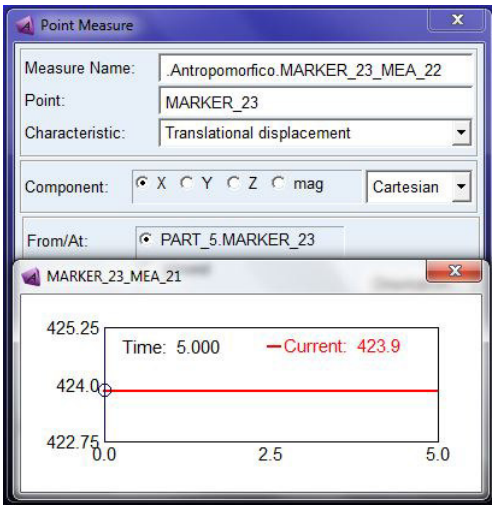

Figura C.6: Posición del efector final en X(px).

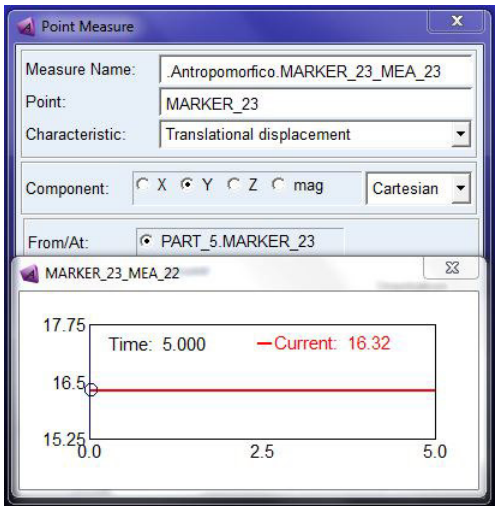

Figura C.7: Posición del efector final en Y(py).

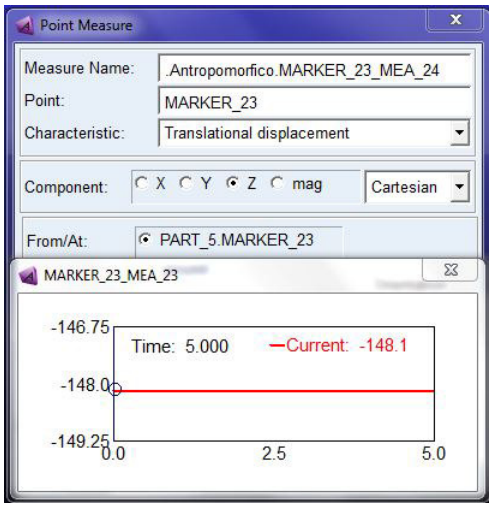

Figura C.8: Posición del efector final en Z(pz).

C.3. Esférico

De nueva cuenta, ahora para el manipulador esférico se obtienen los siguientes valores mediante ADEFID:

```
px=0.4238, py=-0.1481, pz=0.0163, th1=-0.336237, th2=0.863689, b3=0.5906
```

En la figura (C.9) se presenta la simulación generada mediante Adams. En las figuras (C.6) se presenta el valor de `px`. En (C.8) se presenta el valor de `py`. Y en (C.7), se presenta el valor de `pz`. En ambos softwares se tiene el mismo comportamiento.

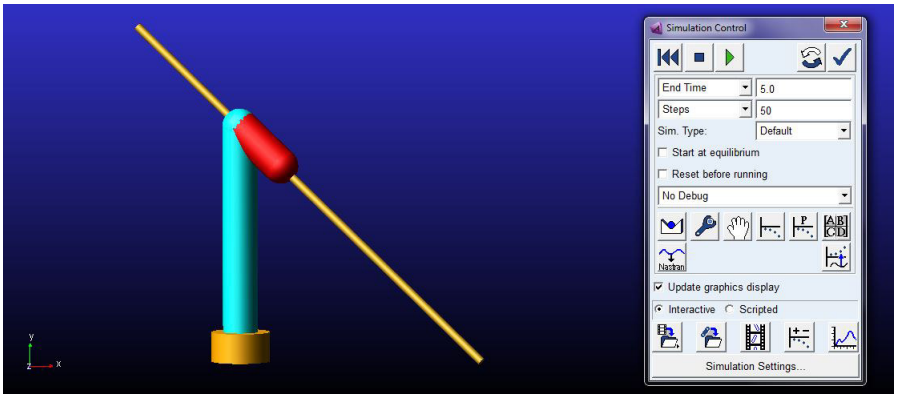

Figura C.9: Simulación prototipo esférico mediante ADAMS.

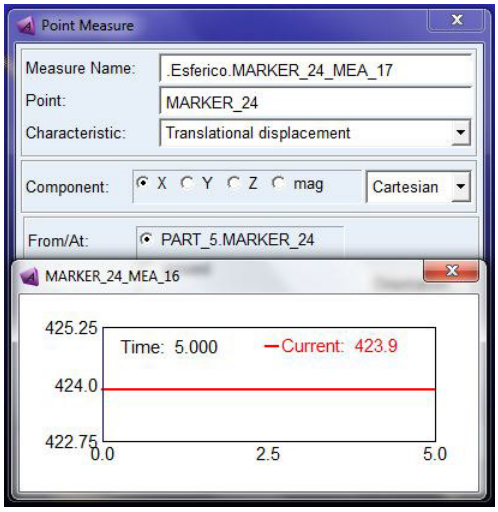

Figura C.10: Posición del efector final en X(px).

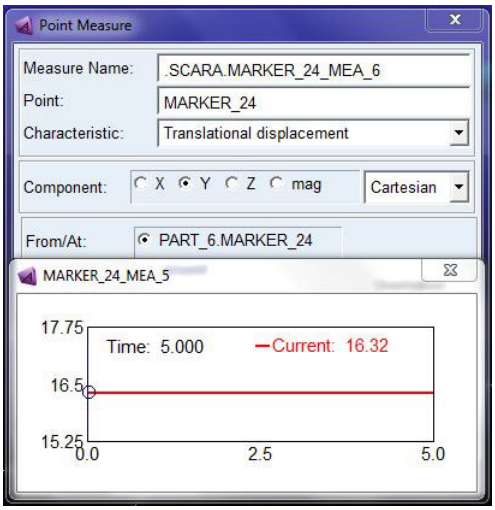

Figura C.11: Posición del efector final en Y(py).

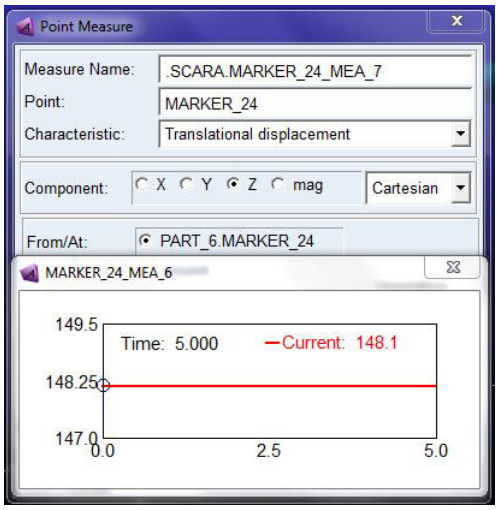

Figura C.12: Posición del efector final en Z(pz).

C.4. SCARA

Por último, para el manipulador SCARA se obtienen los siguientes valores mediante ADEFID:

```
px=0.4238, py=-0.1481, pz=0.0163, th1=-1.3111, th2=1.9497, b3=0.4836
```

En la figura (C.13) se presenta la simulación generada mediante Adams. En las figuras (C.14) se presenta el valor de `px`. En (C.16) se presenta el valor de `py`. Y en (C.15), se presenta el valor de `pz`. En ambos softwares se tiene el mismo comportamiento.

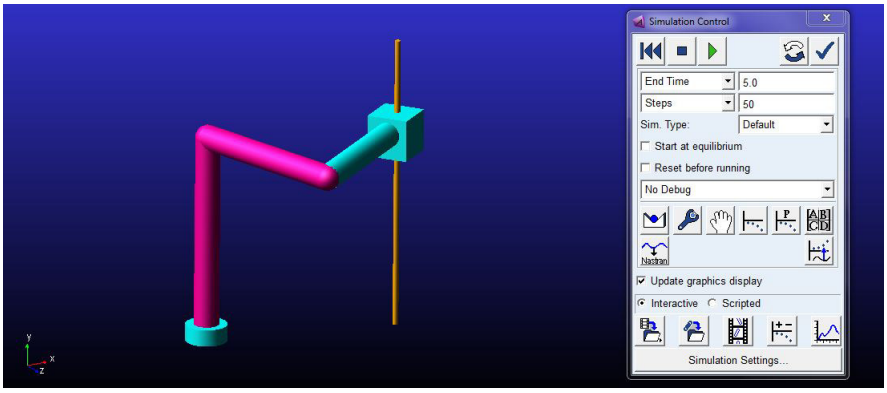

Figura C.13: Simulación prototipo SCARA mediante ADAMS.

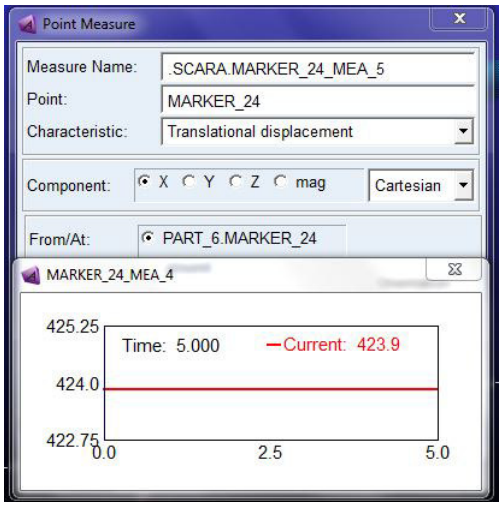

Figura C.14: Posición del efector final en X(px).

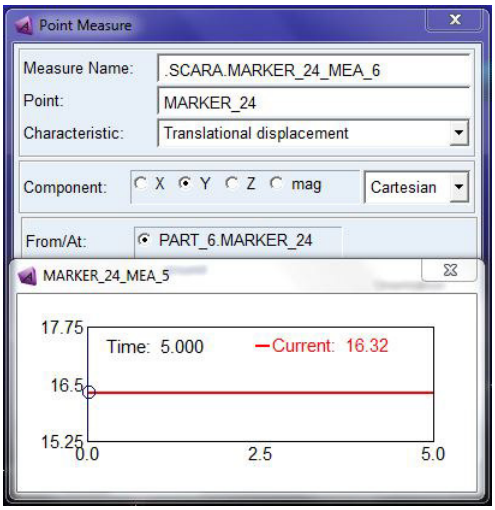

Figura C.15: Posición del efector final en Y(py).

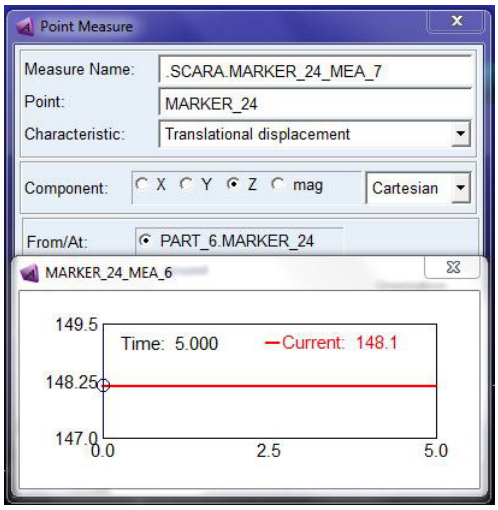

Figura C.16: Posición del efector final en Z(pz).

Apéndice D

Resultados

El presente Apéndice es un complemento al Capítulo 6, en este se presentan más pruebas características a las trayectorias propuestas, tratando en el mayor grado de lo posible, tener las mismas condiciones para la adquisición. Se observa que el comportamiento es el mismo, por lo que el análisis realizado en el Capítulo 6, también aplica para estas pruebas. Entiéndase como la primer prueba, la presentada en el Capítulo 6.

D.1. Trayectoria rampa, SnAM Vs. Antropomórfico, Segunda prueba

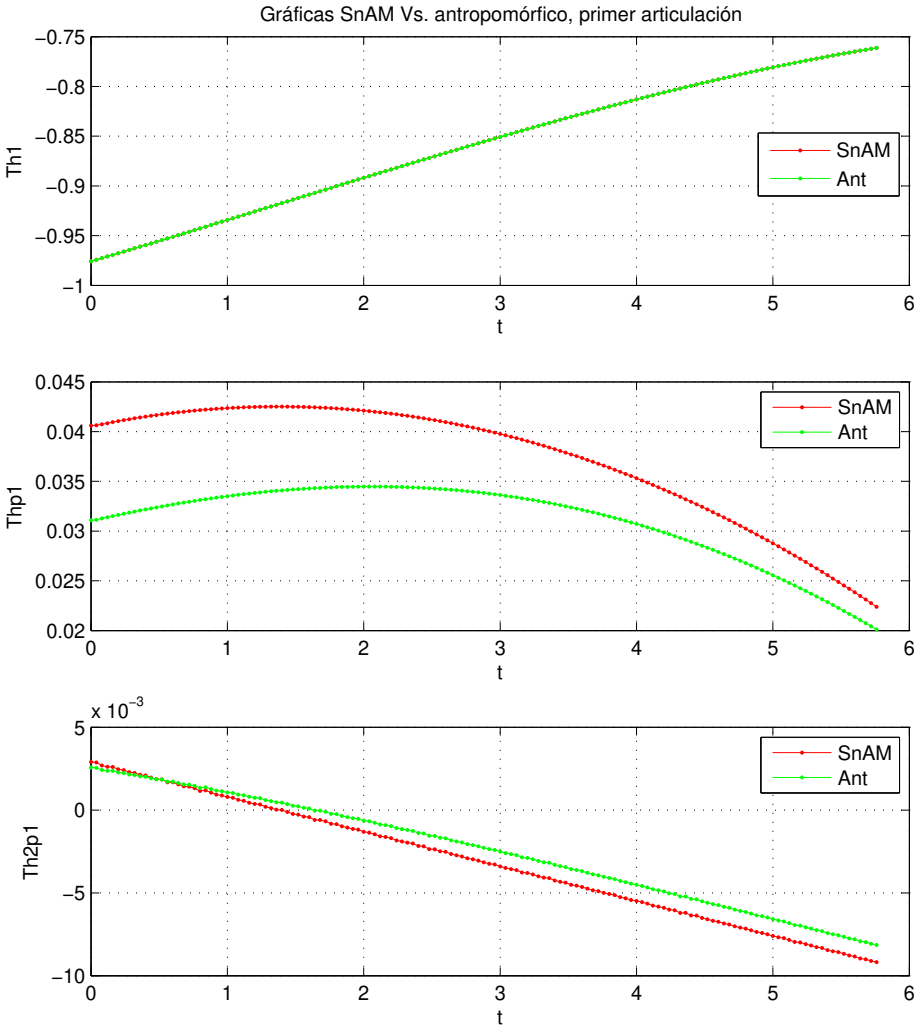

Figura D.1: Graficación de valores de la primer articulación.

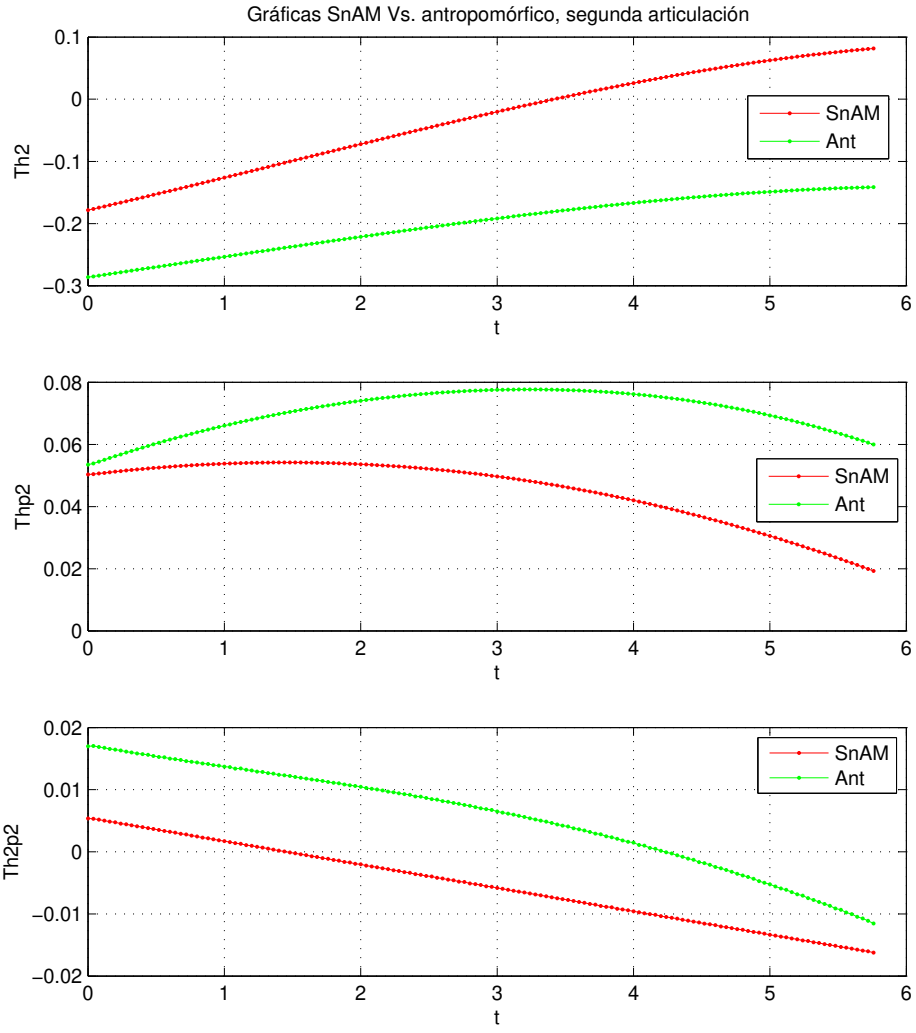

Figura D.2: Graficación de valores de la segunda articulación.

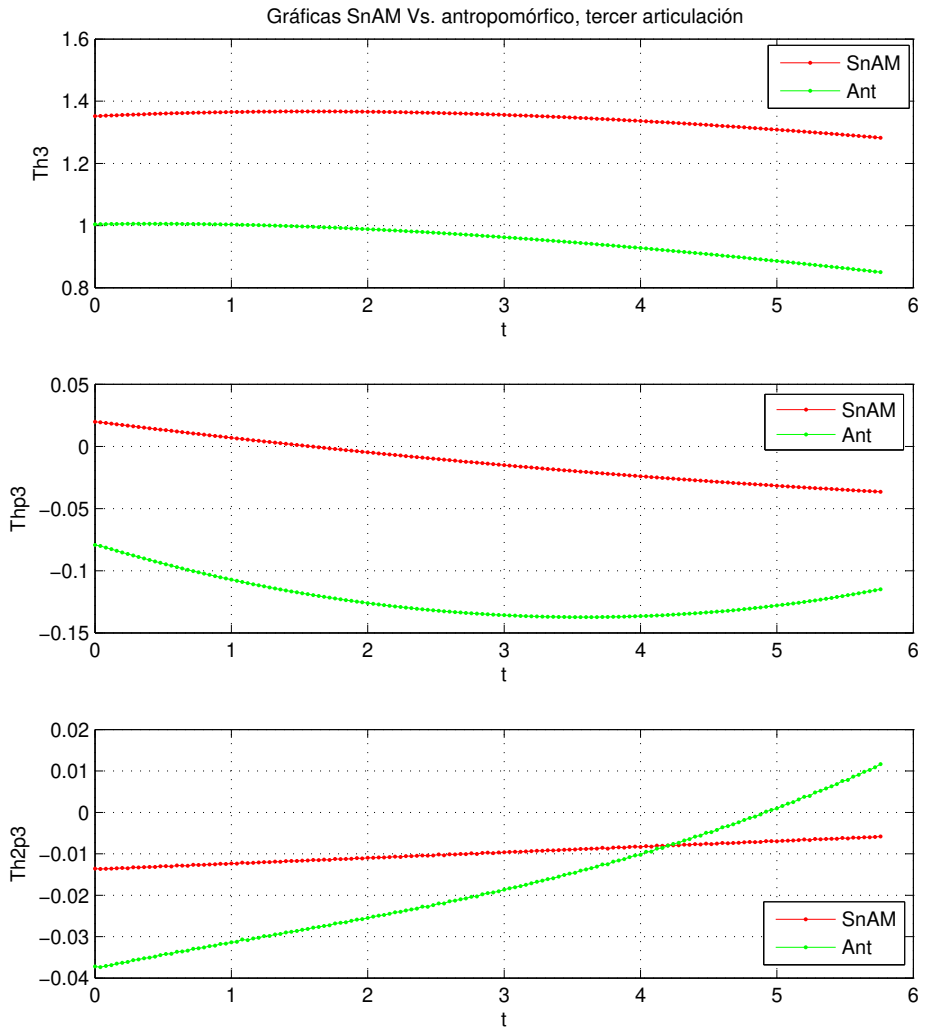

Figura D.3: Graficación de valores de la tercer articulación.

D.2. Trayectoria rampa, SnAM Vs. Antropomórfico, Tercer prueba

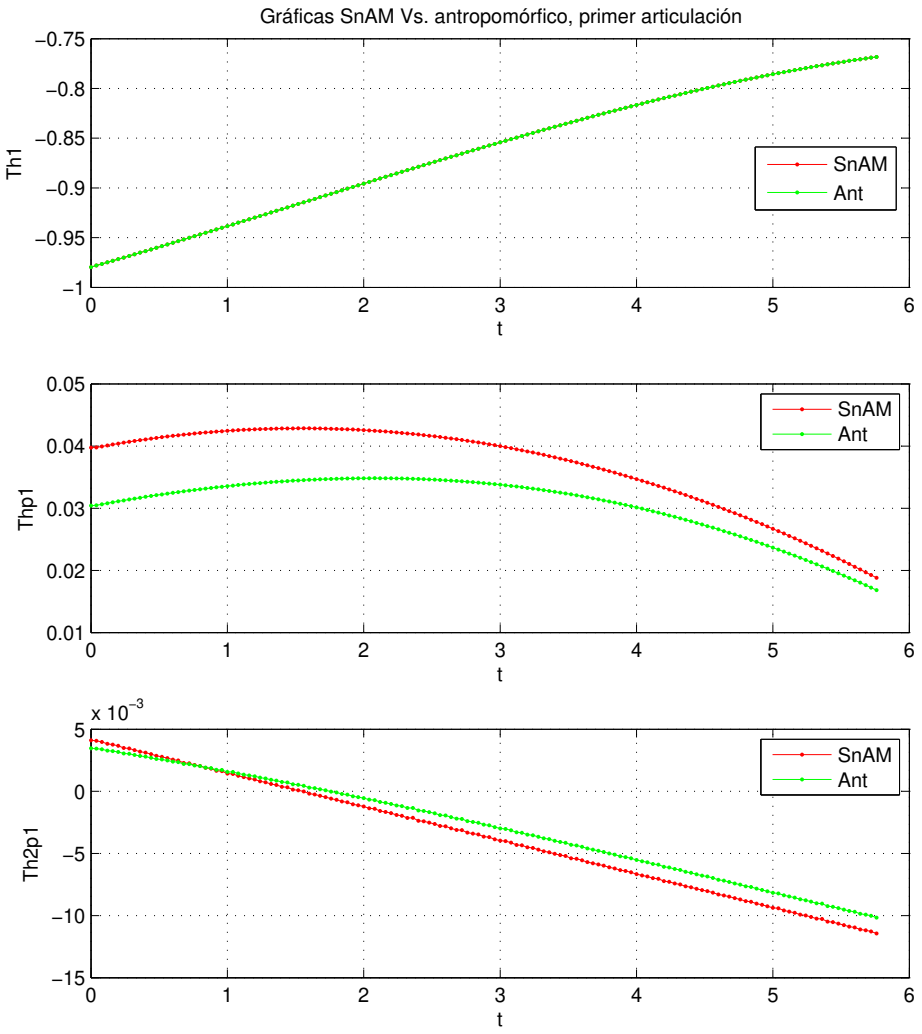

Figura D.4: Graficación de valores de la primer articulación.

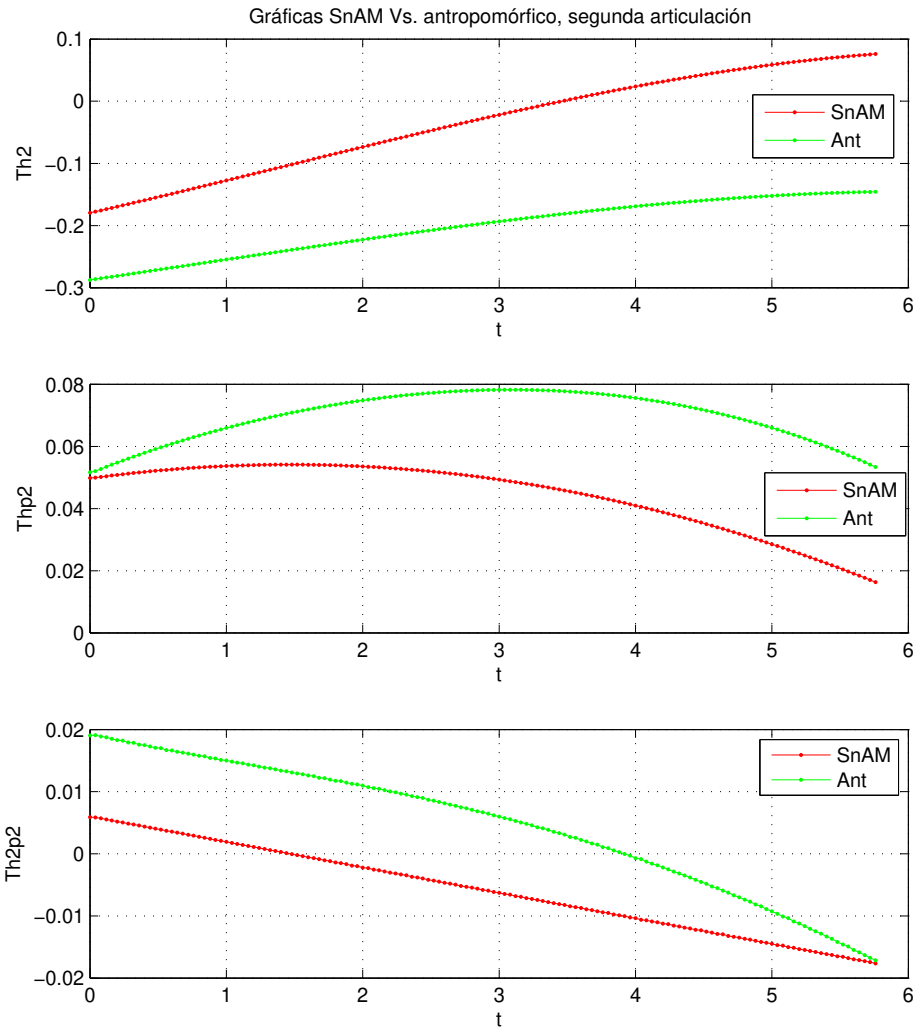

Figura D.5: Graficación de valores de la segunda articulación.

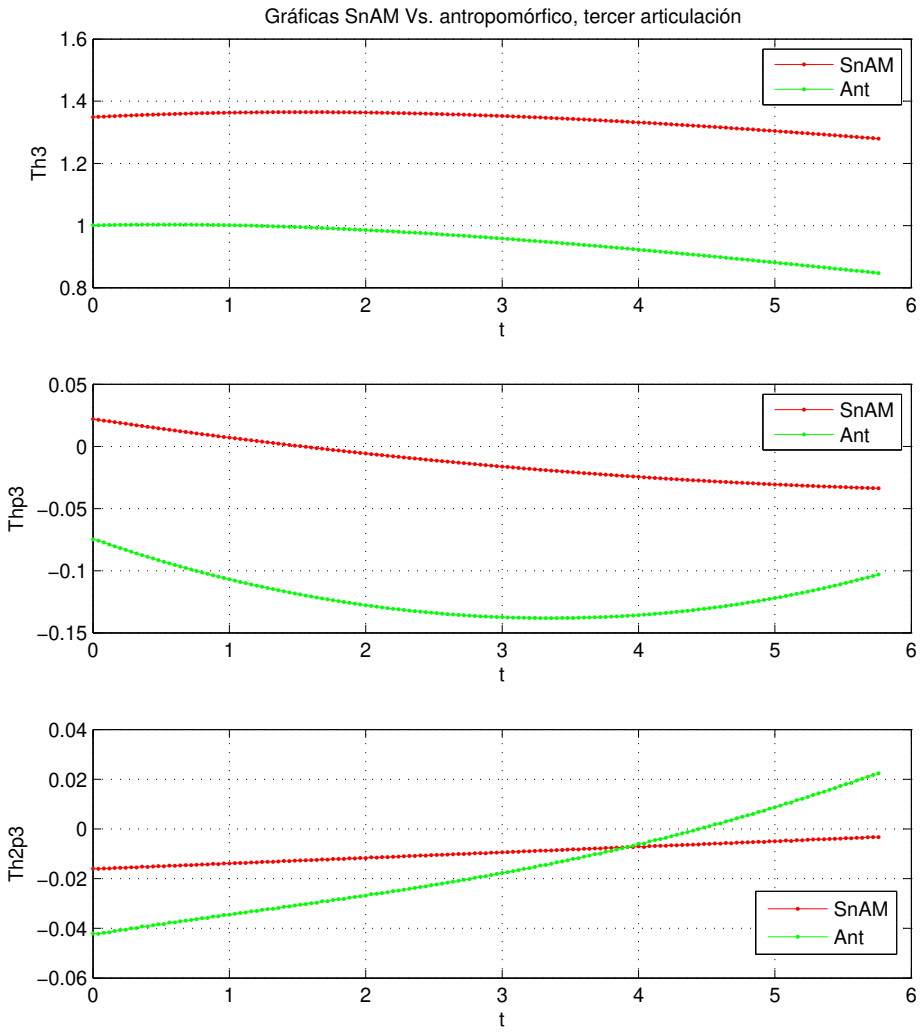

Figura D.6: Graficación de valores de la tercer articulación.

D.3. Trayectoria rampa, SnAM Vs. esférico, Segunda prueba

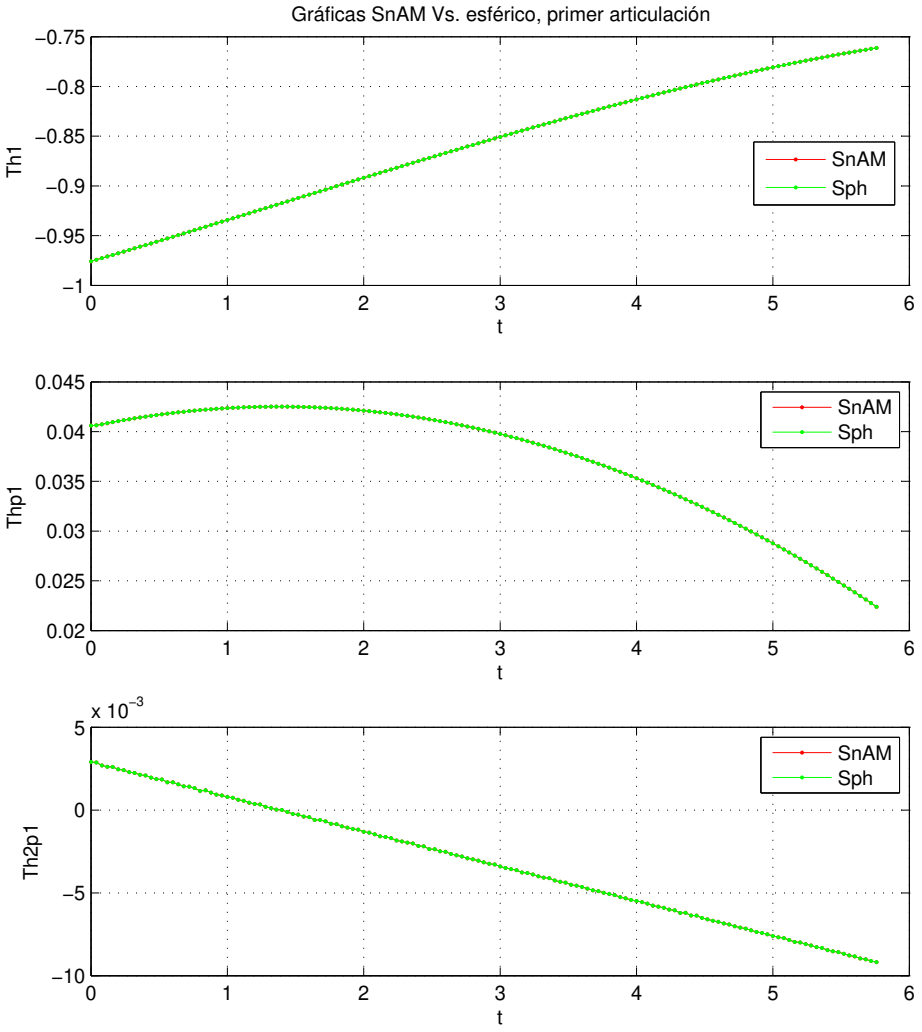

Figura D.7: Graficación de valores de la primer articulación.

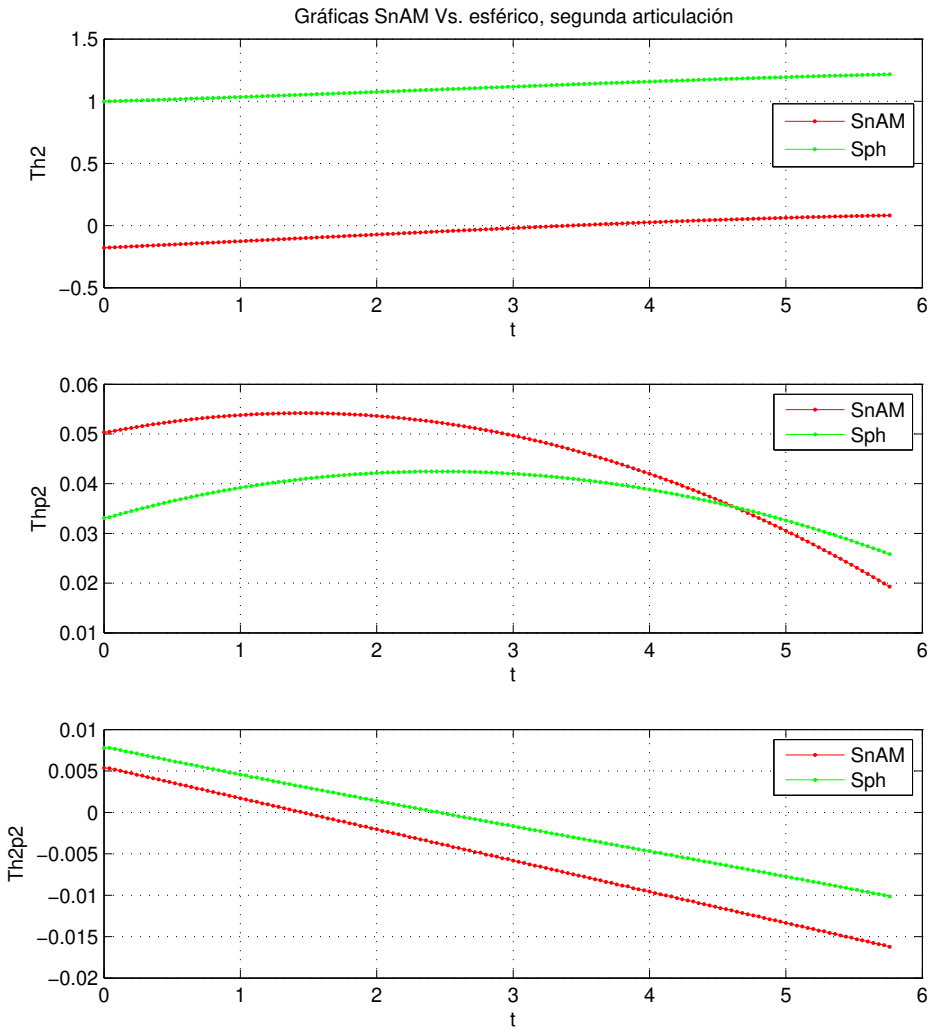

Figura D.8: Graficación de valores de la segunda articulación.

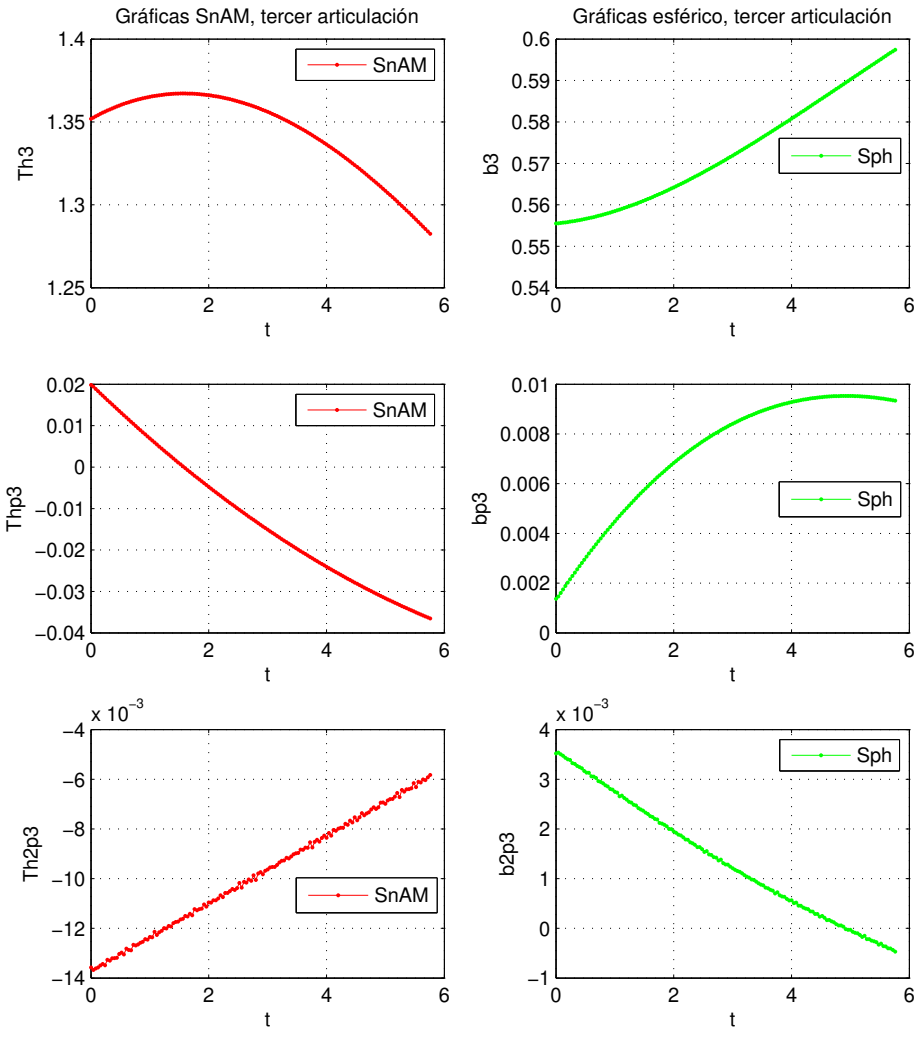

Figura D.9: Graficación de valores de la tercer articulación.

D.4. Trayectoria rampa, SnAM Vs. esférico, tercer prueba

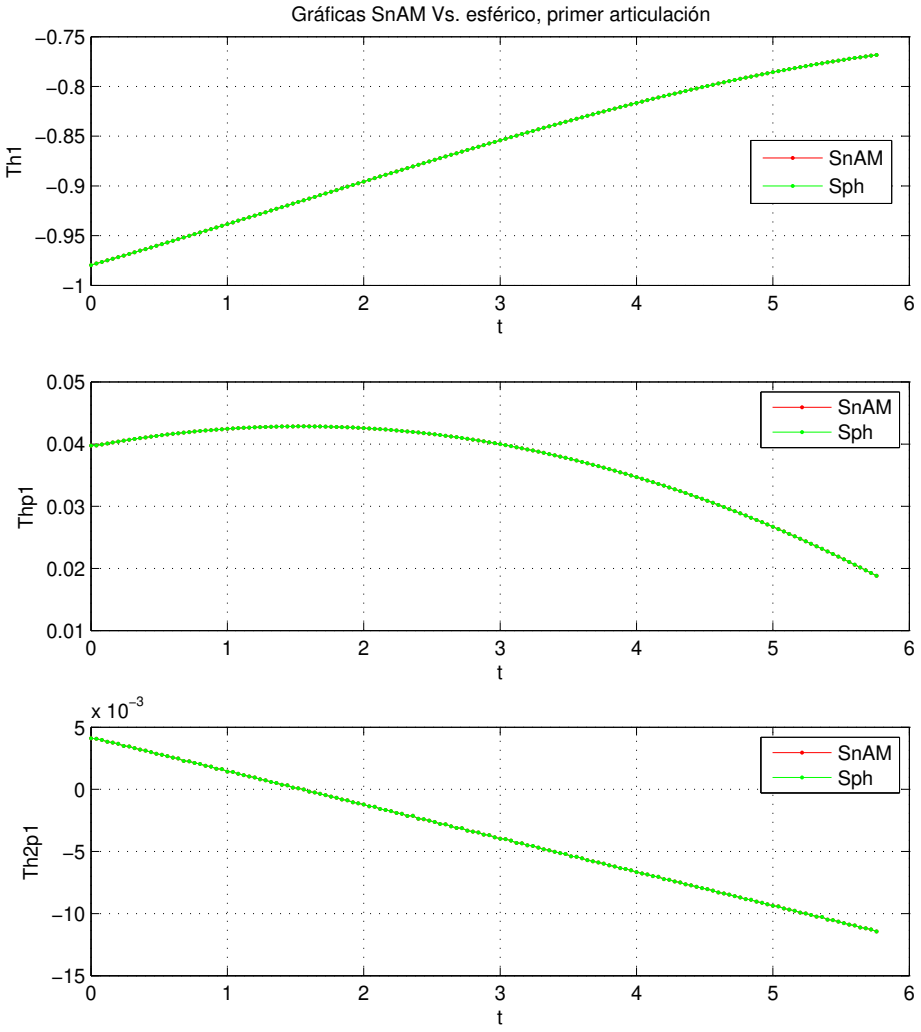

Figura D.10: Graficación de valores de la primer articulación.

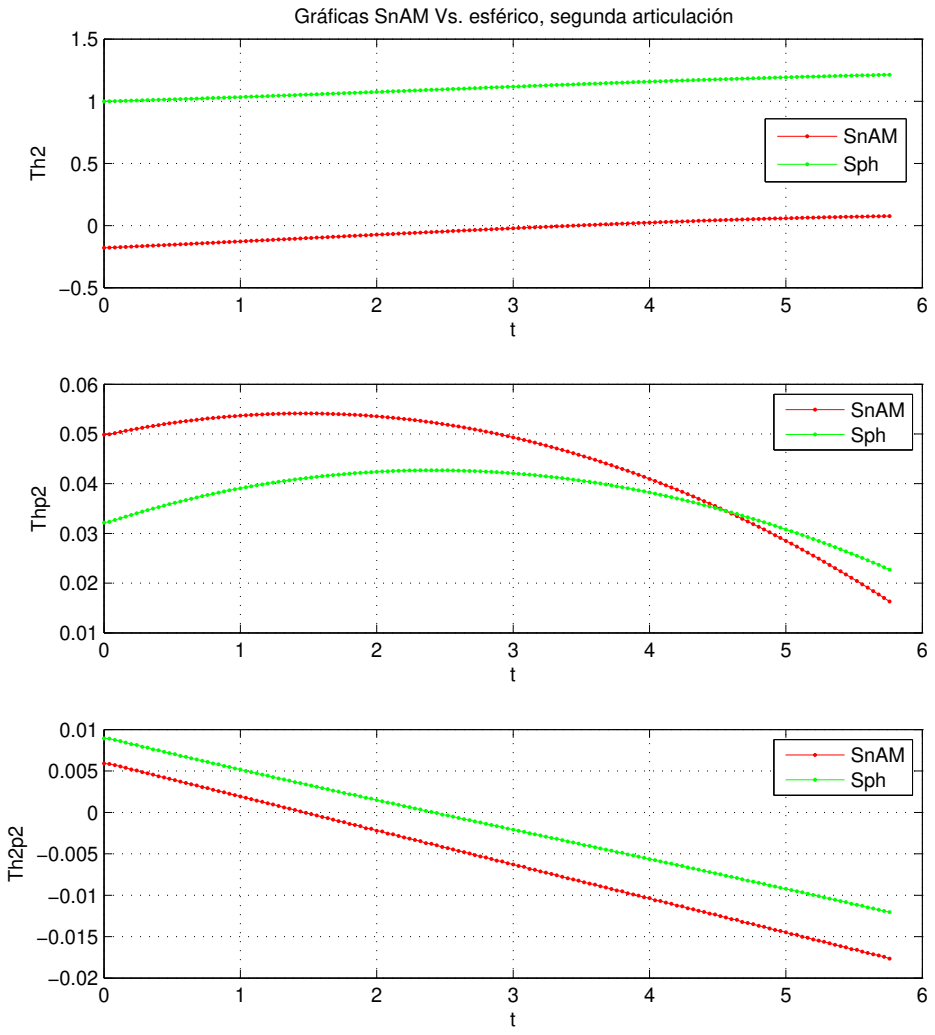

Figura D.11: Graficación de valores de la segunda articulación.

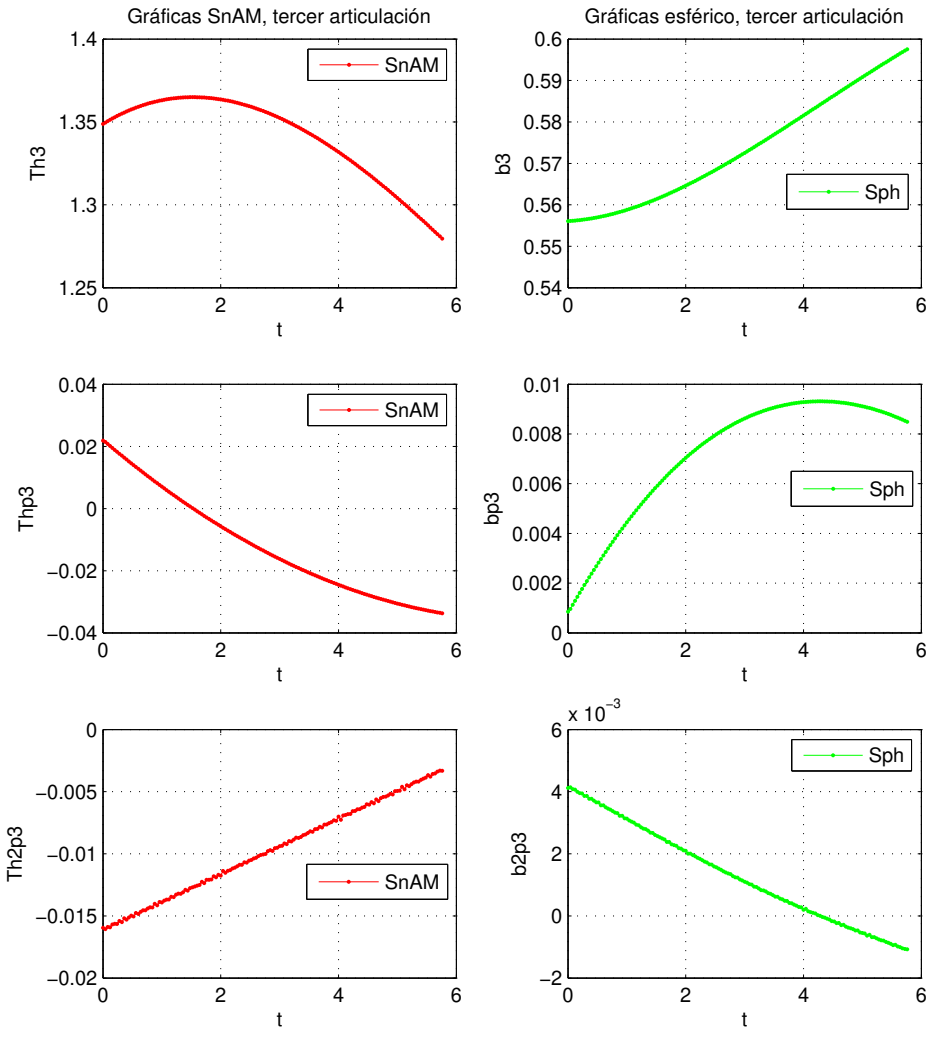

Figura D.12: Graficación de valores de la tercer articulación.

D.5. Trayectoria circular, SnAM Vs. antropomórfico, segunda prueba

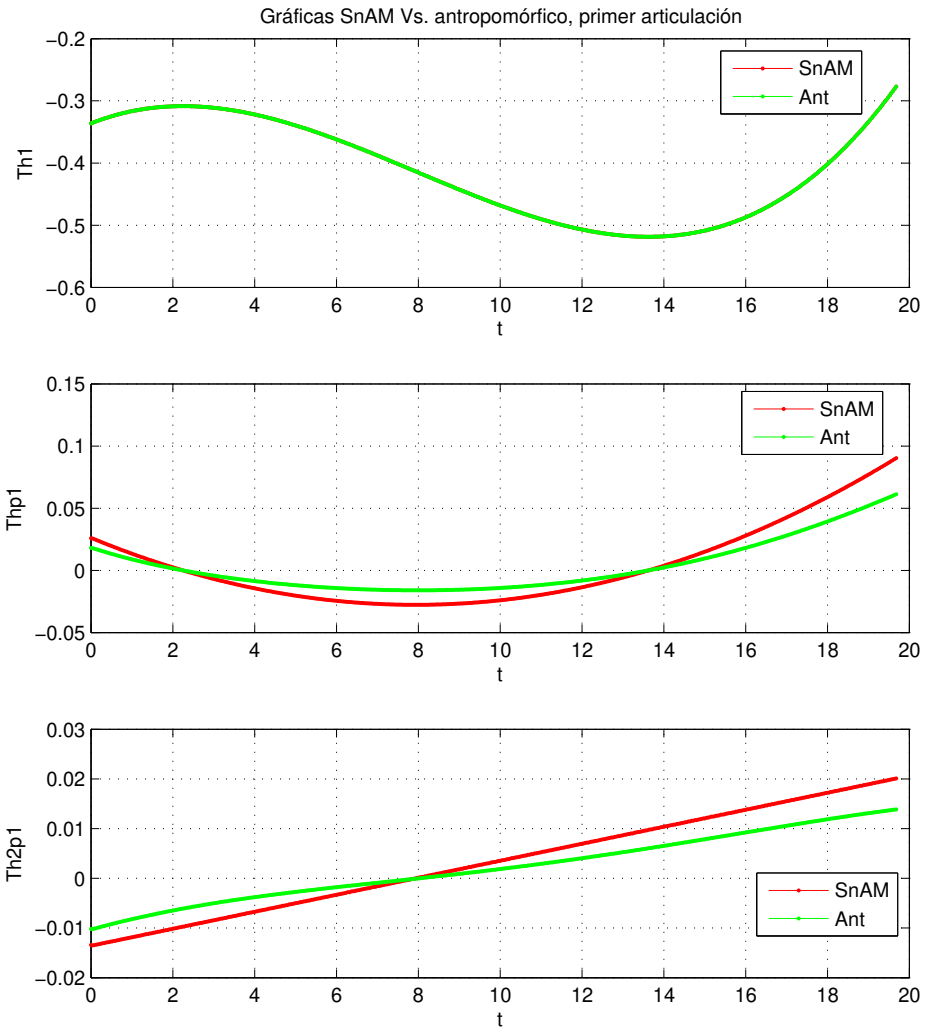

Figura D.13: Graficación de valores de la primer articulación.

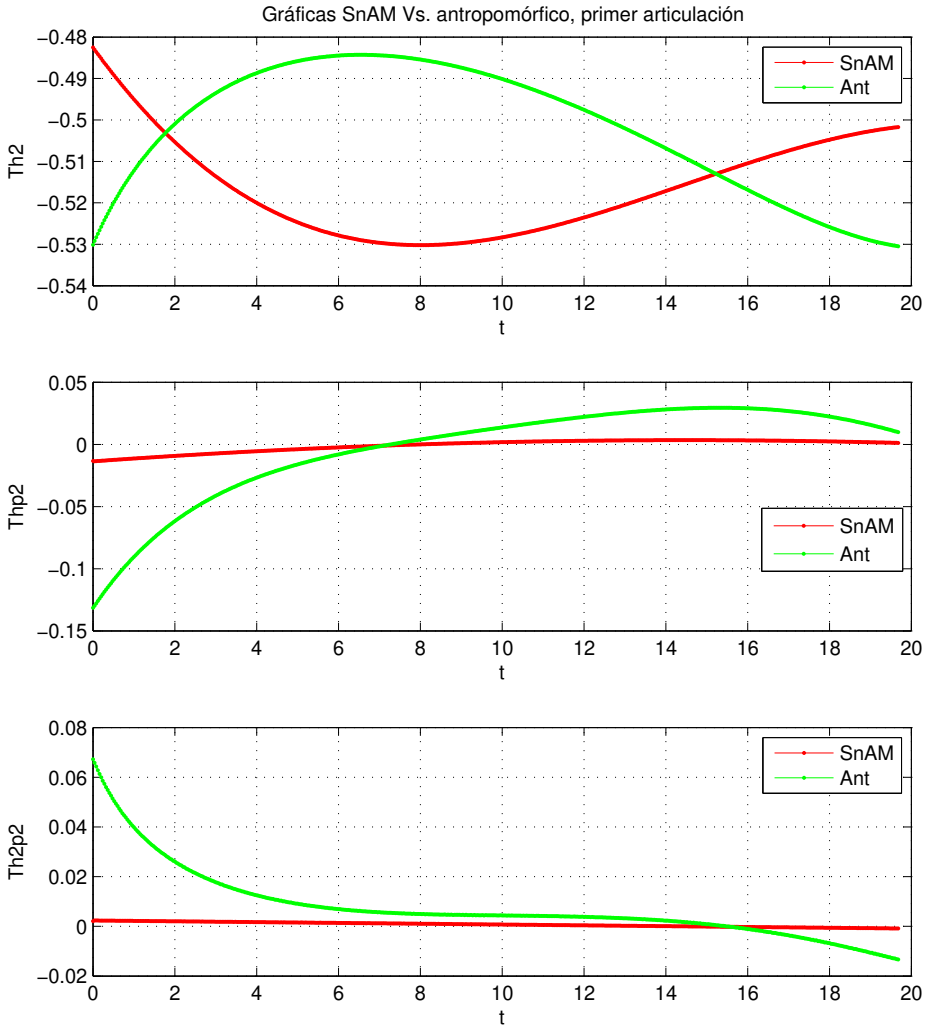

Figura D.14: Graficación de valores de la segunda articulación.

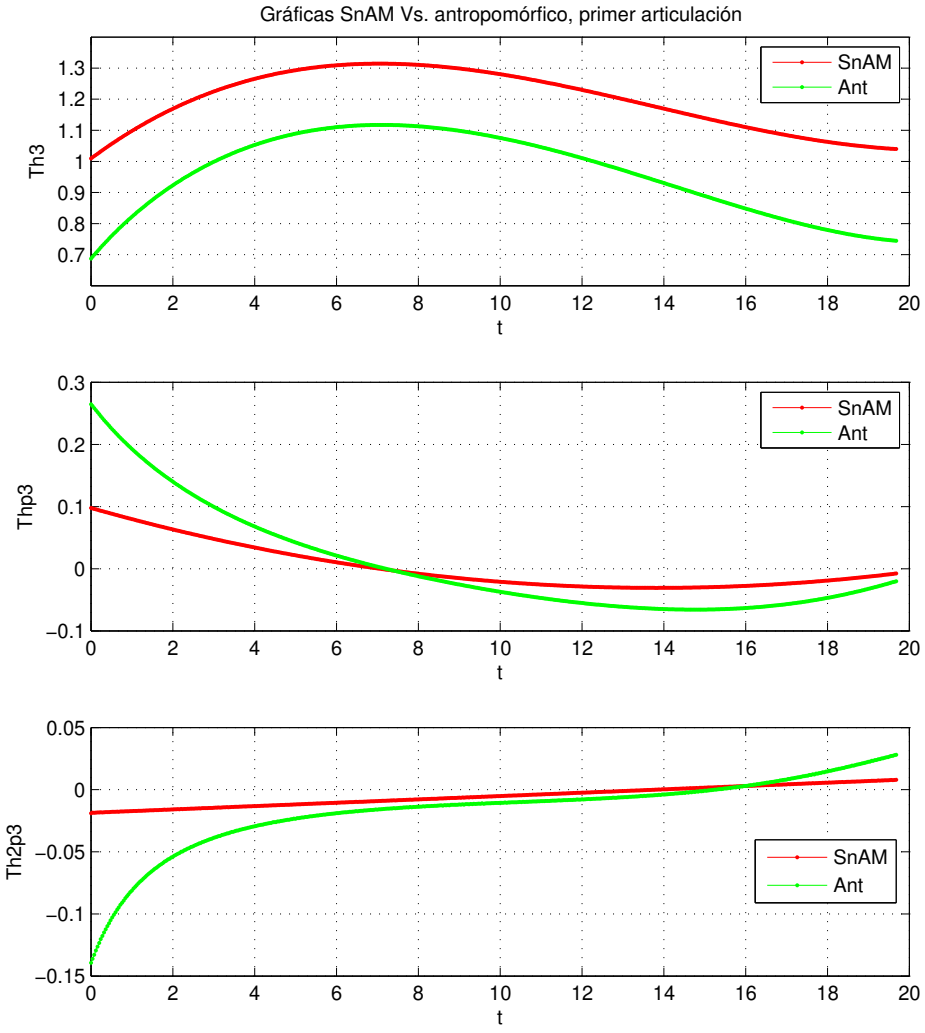

Figura D.15: Graficación de valores de la tercer articulación.

D.6. Trayectoria circular, SnAM Vs. antropomórfico, tercer prueba

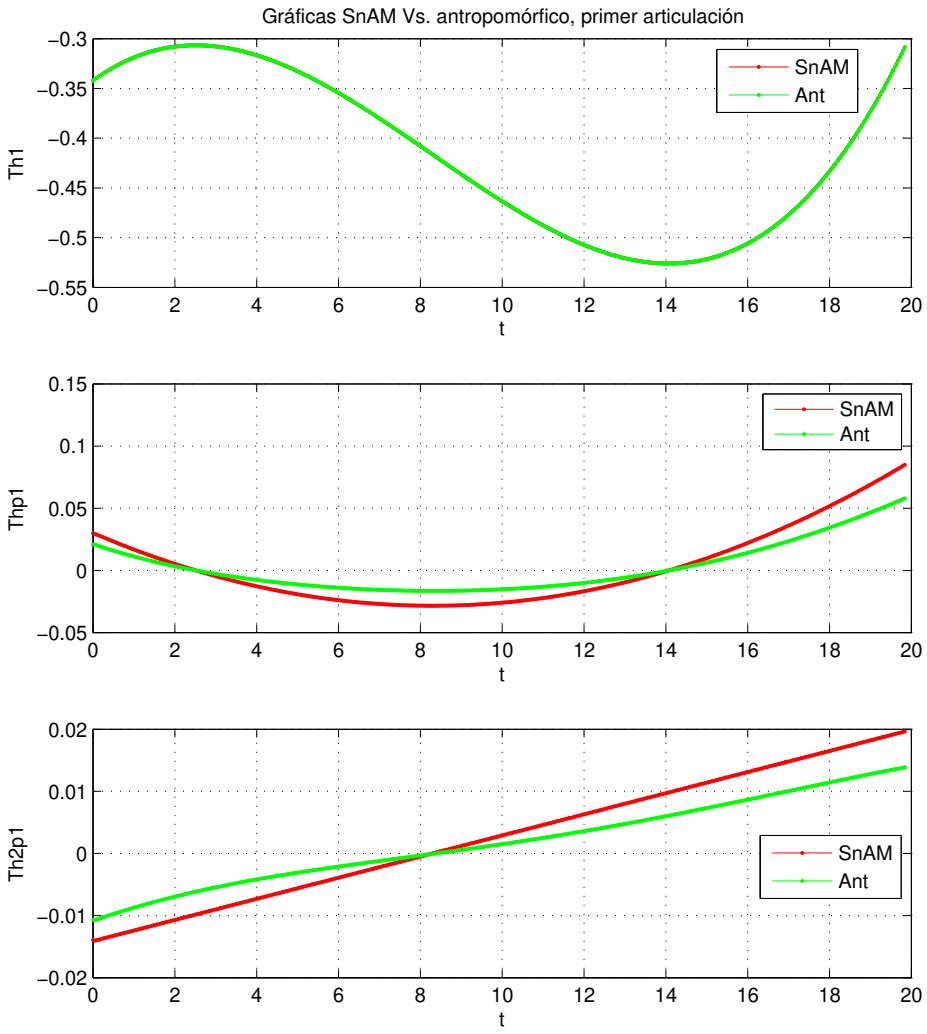

Figura D.16: Graficación de valores de la primer articulación.

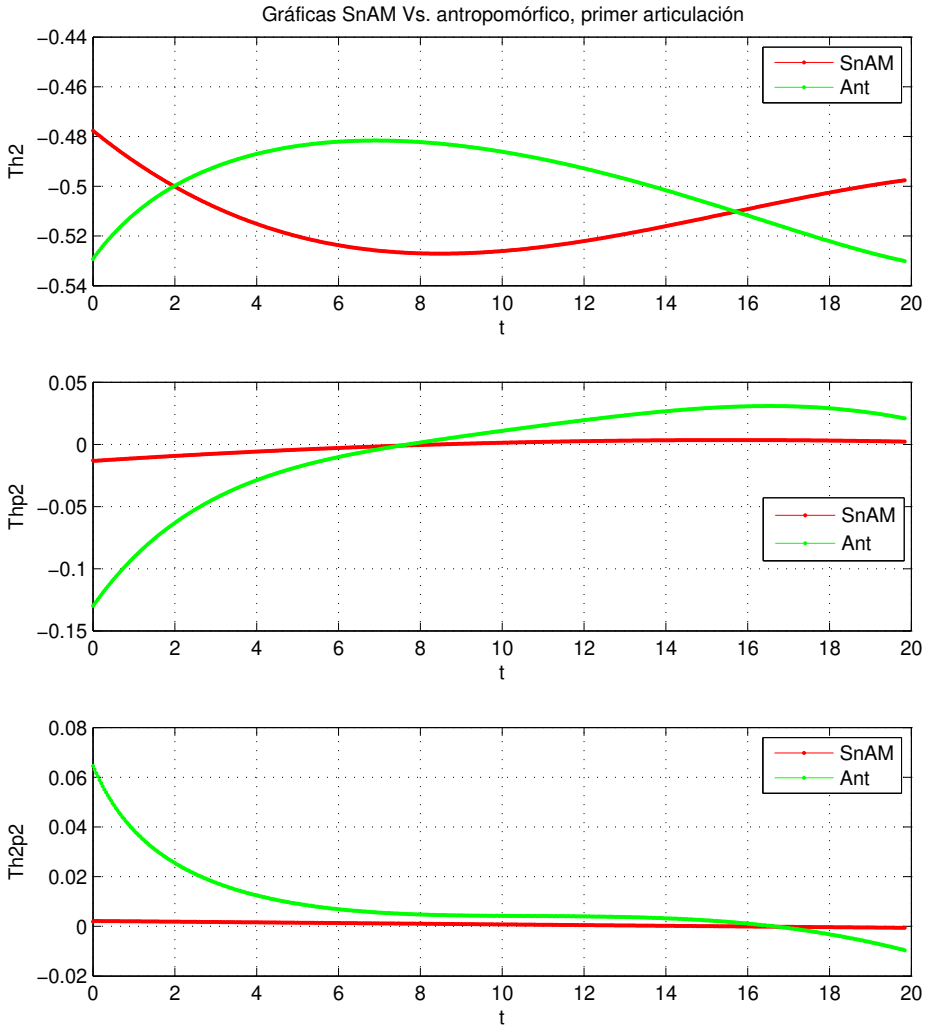

Figura D.17: Graficación de valores de la segunda articulación.

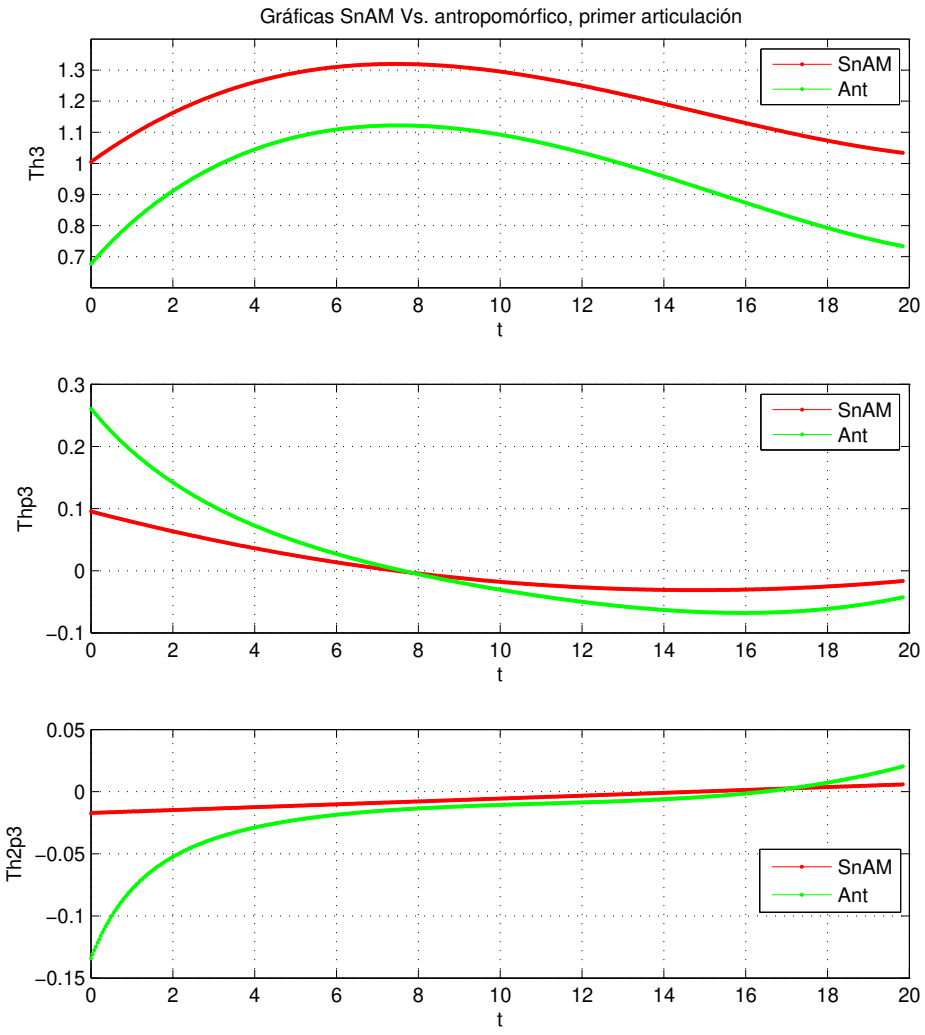

Figura D.18: Graficación de valores de la tercer articulación.

D.7. Trayectoria circular. SnAM Vs. esférico, segunda prueba

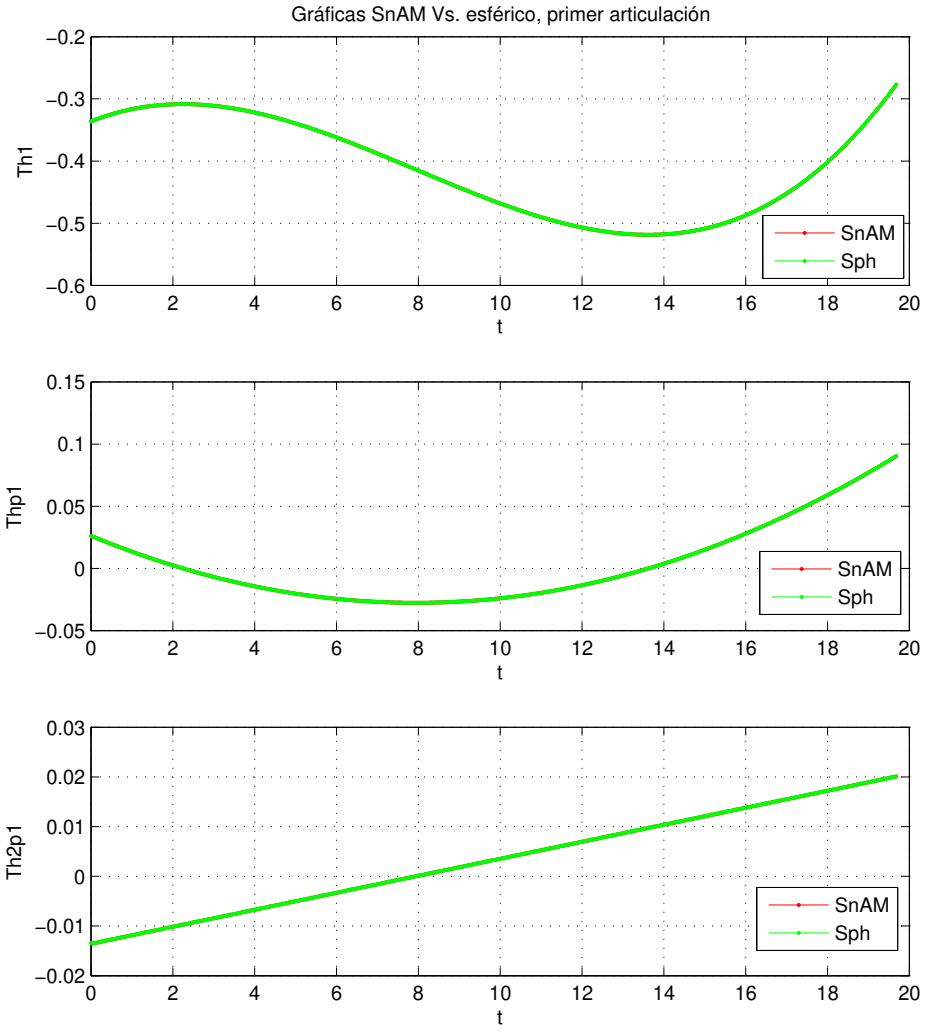

Figura D.19: Graficación de valores de la primer articulación.

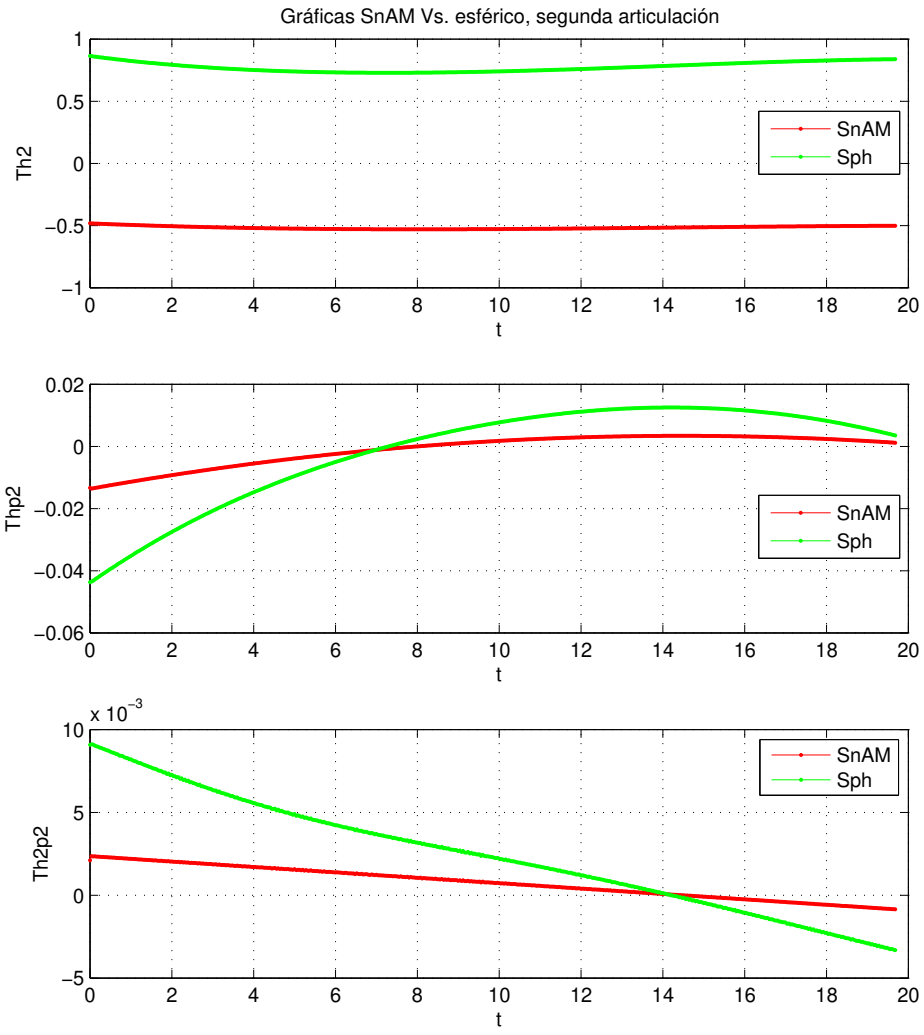

Figura D.20: Graficación de valores de la segunda articulación.

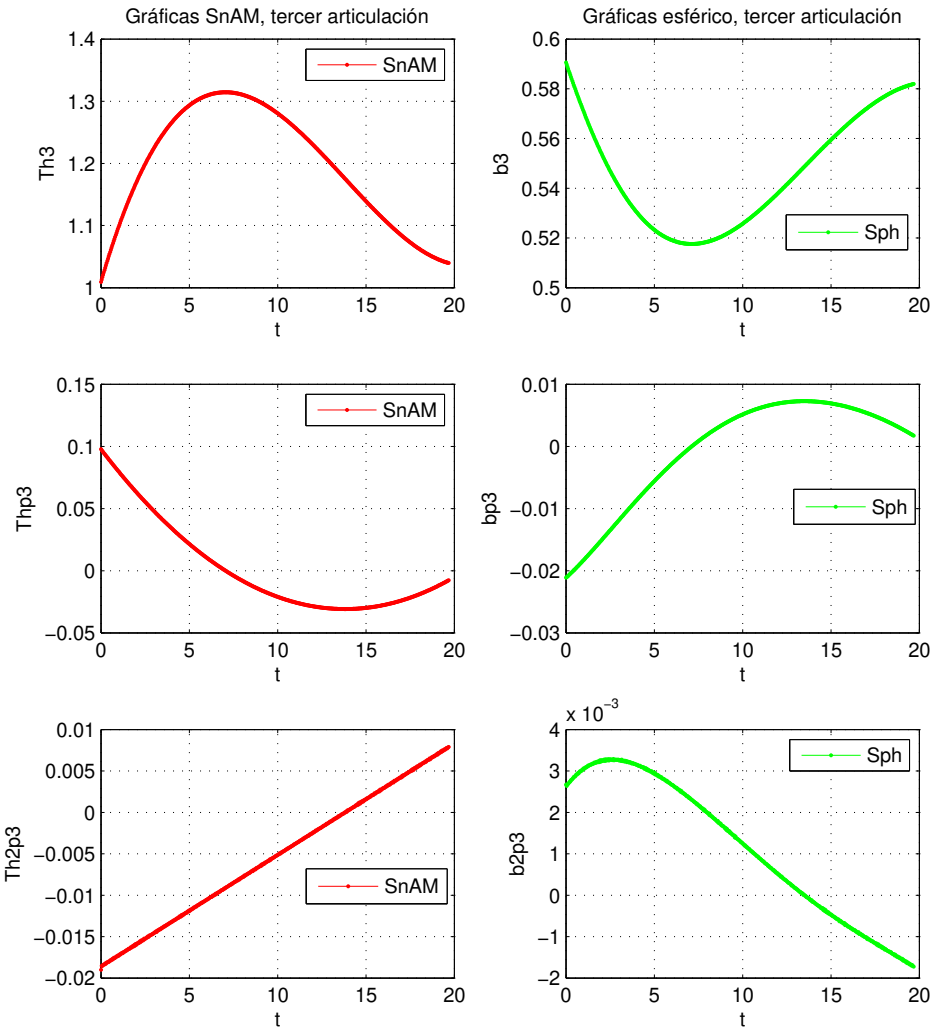

Figura D.21: Graficación de valores de la tercer articulación.

D.8. Trayectoria circular. SnAM Vs. esférico, tercer prueba

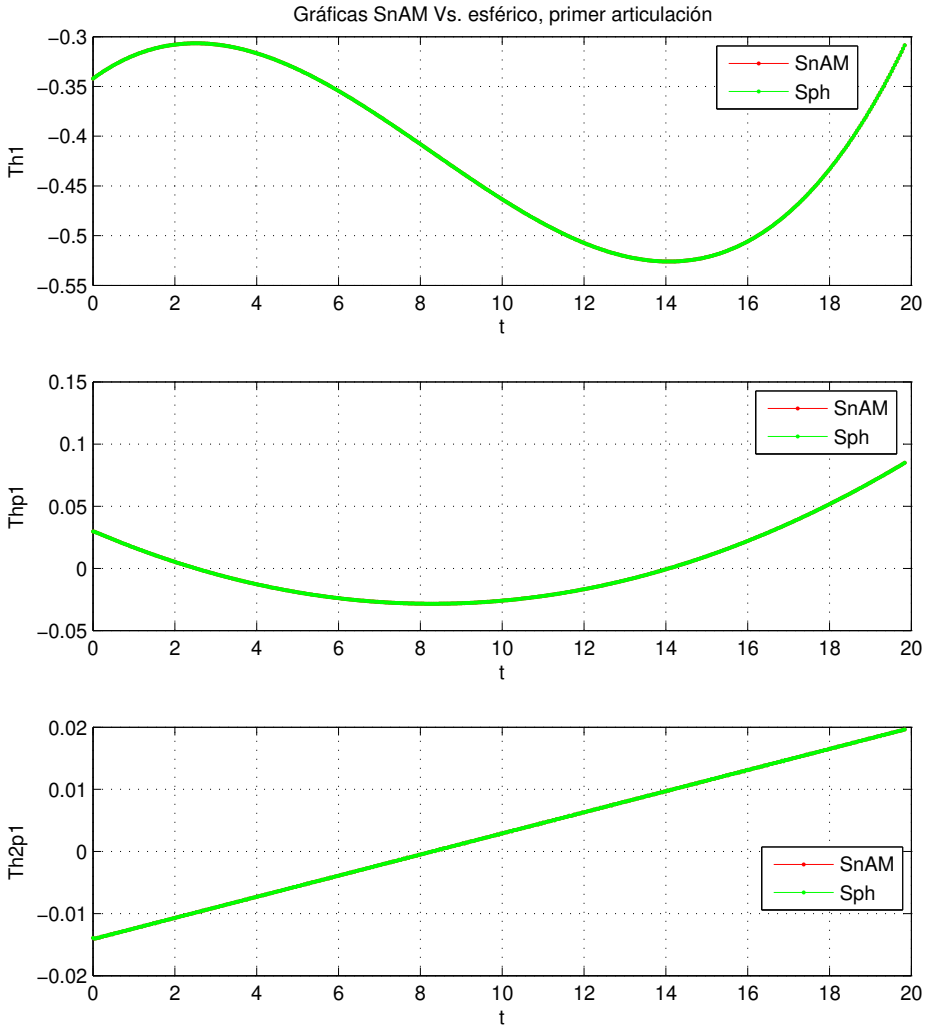

Figura D.22: Graficación de valores de la primer articulación.

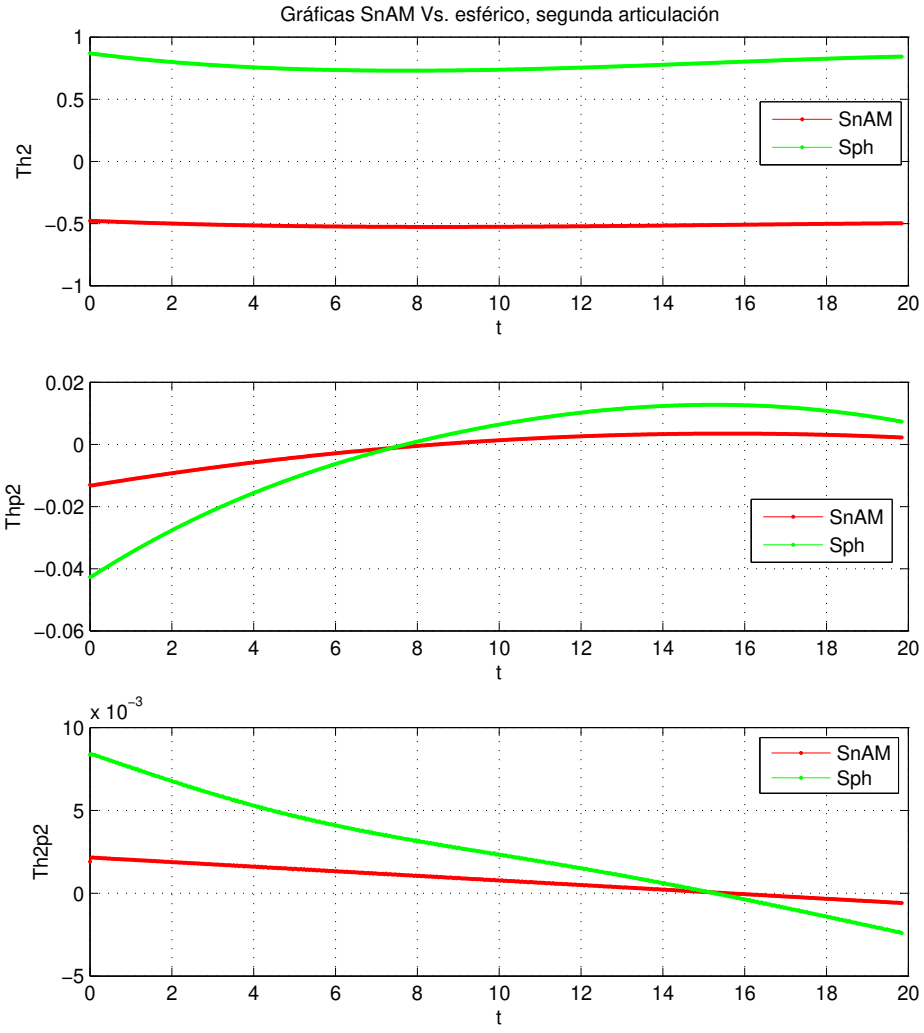

Figura D.23: Graficación de valores de la primer articulación.

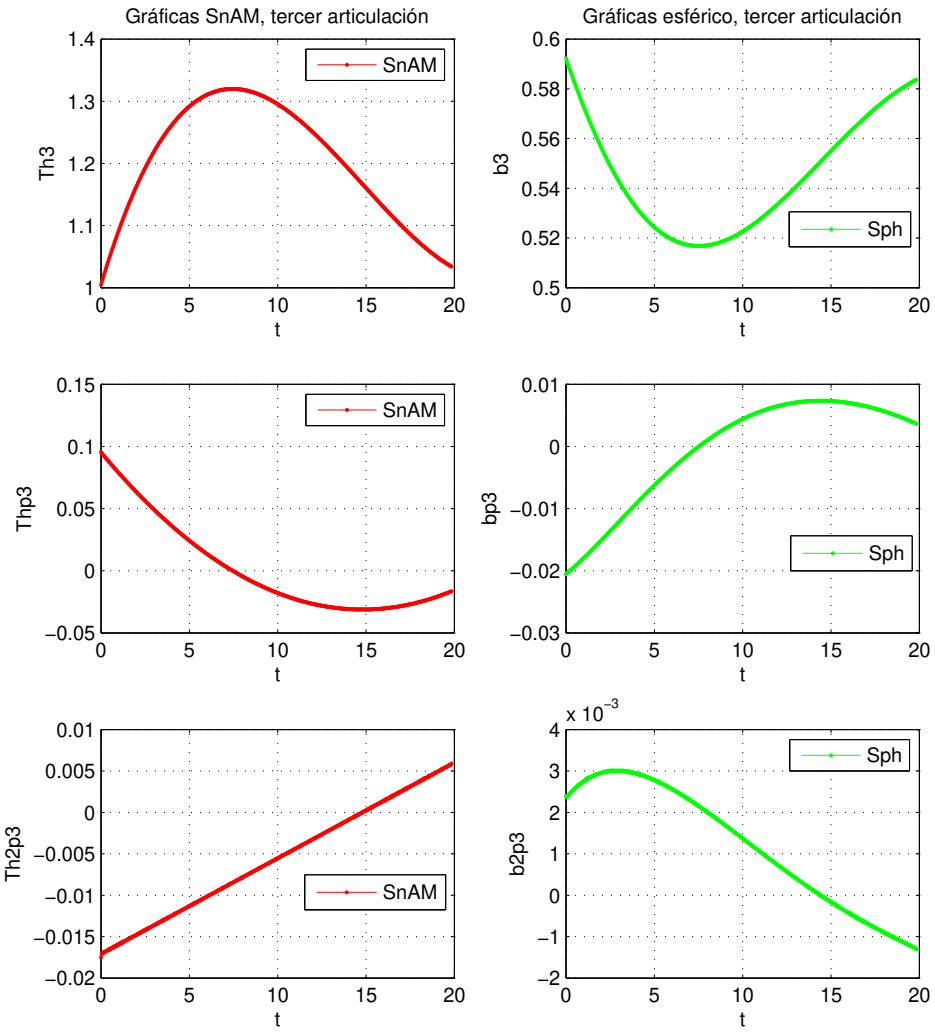

Figura D.24: Graficación de valores de la primer articulación.

D.9. Trayectoria circular. SnAM Vs. SCARA, segunda prueba

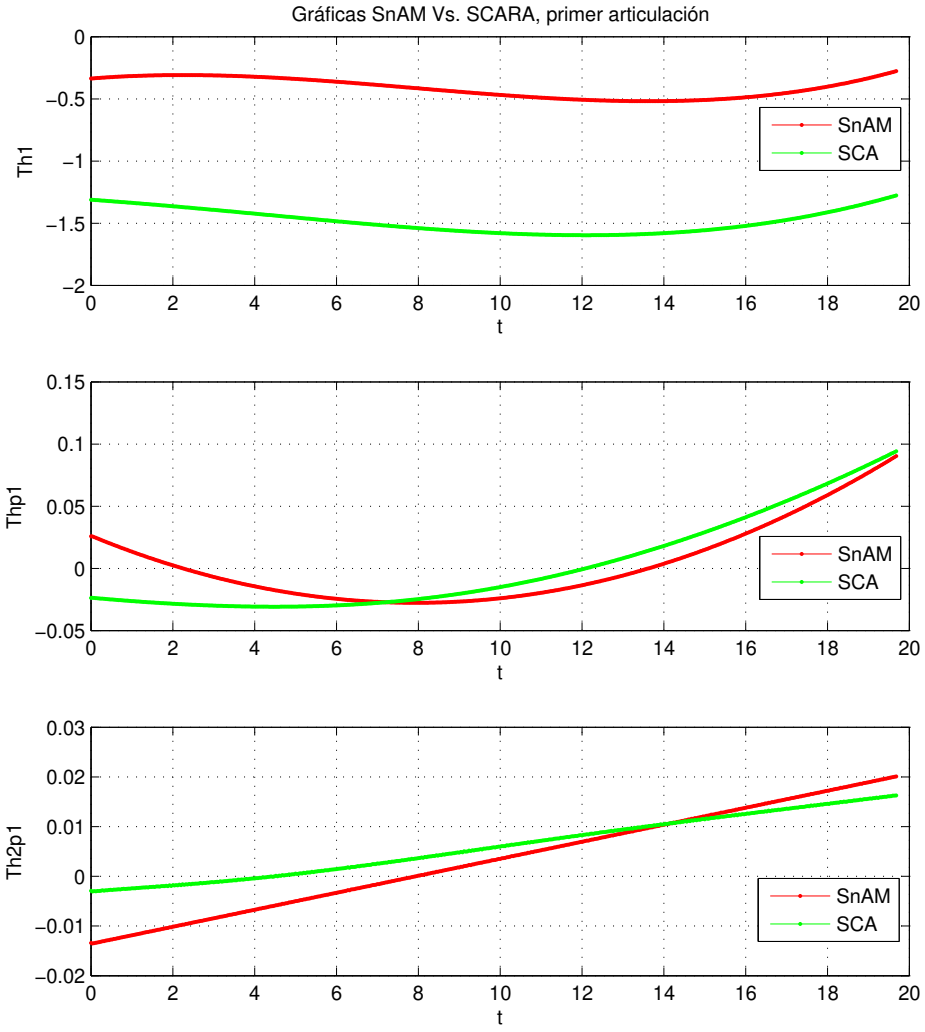

Figura D.25: Graficación de valores de la primer articulación.

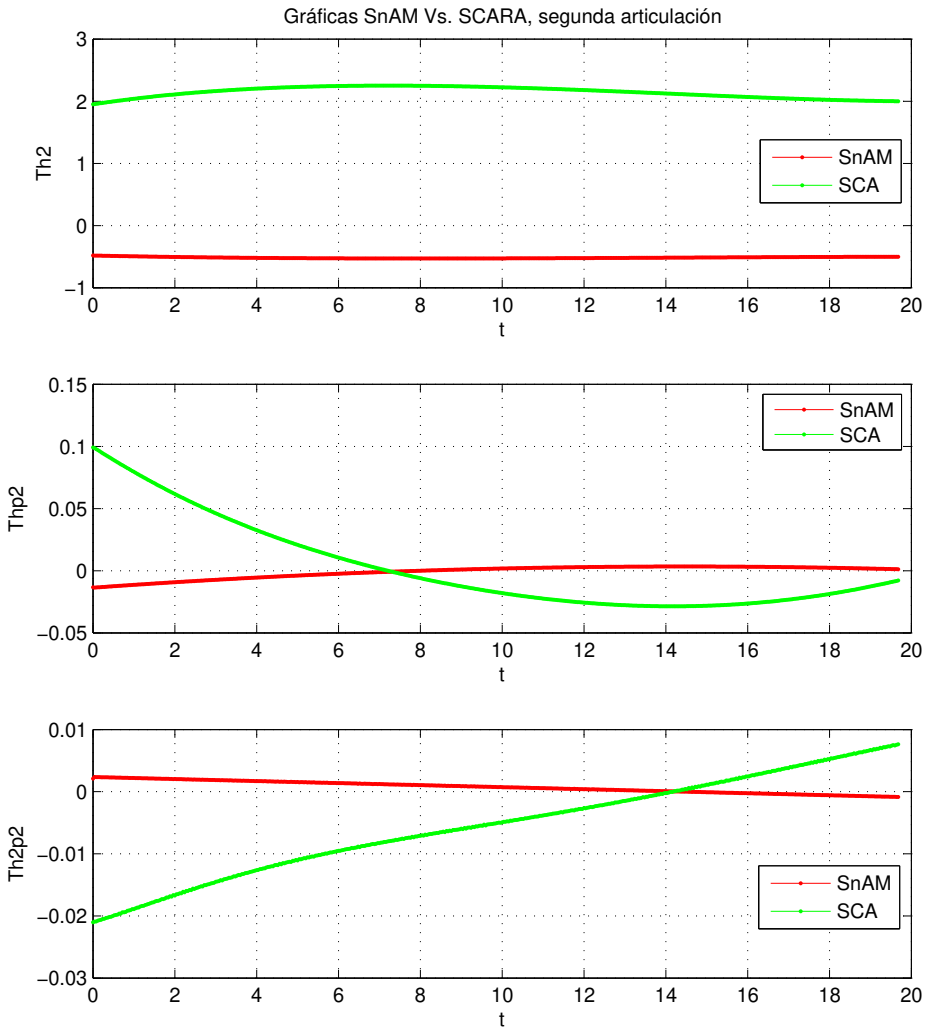

Figura D.26: Graficación de valores de la segunda articulación.

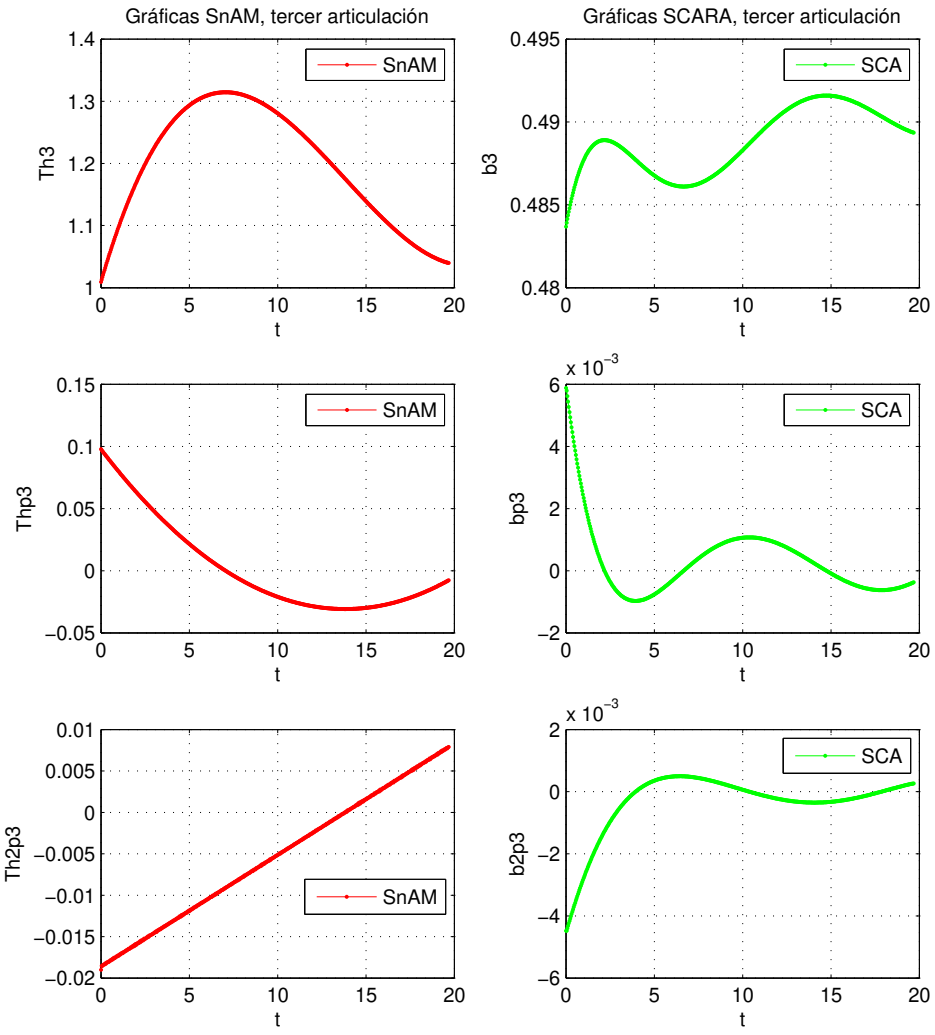

Figura D.27: Graficación de valores de la tercer articulación.

D.10. Trayectoria circular. SnAM Vs. SCARA, tercer prueba

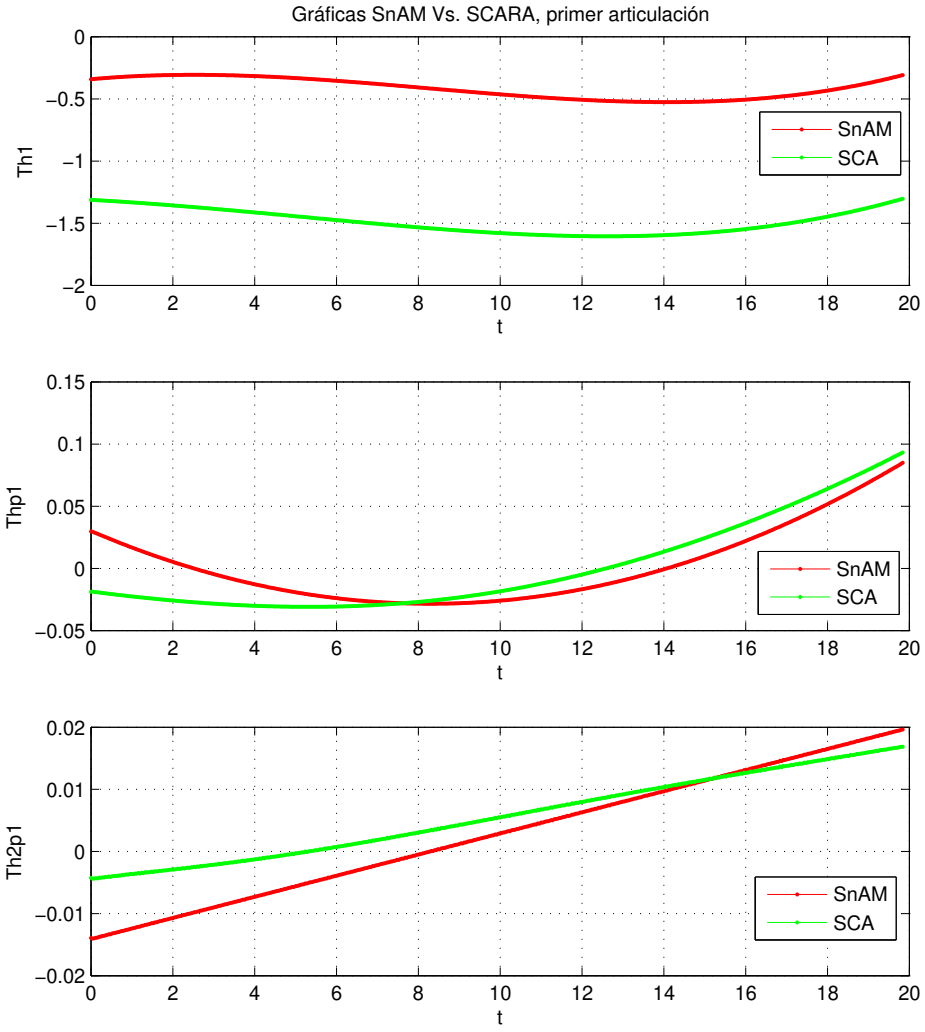

Figura D.28: Graficación de valores de la primer articulación.

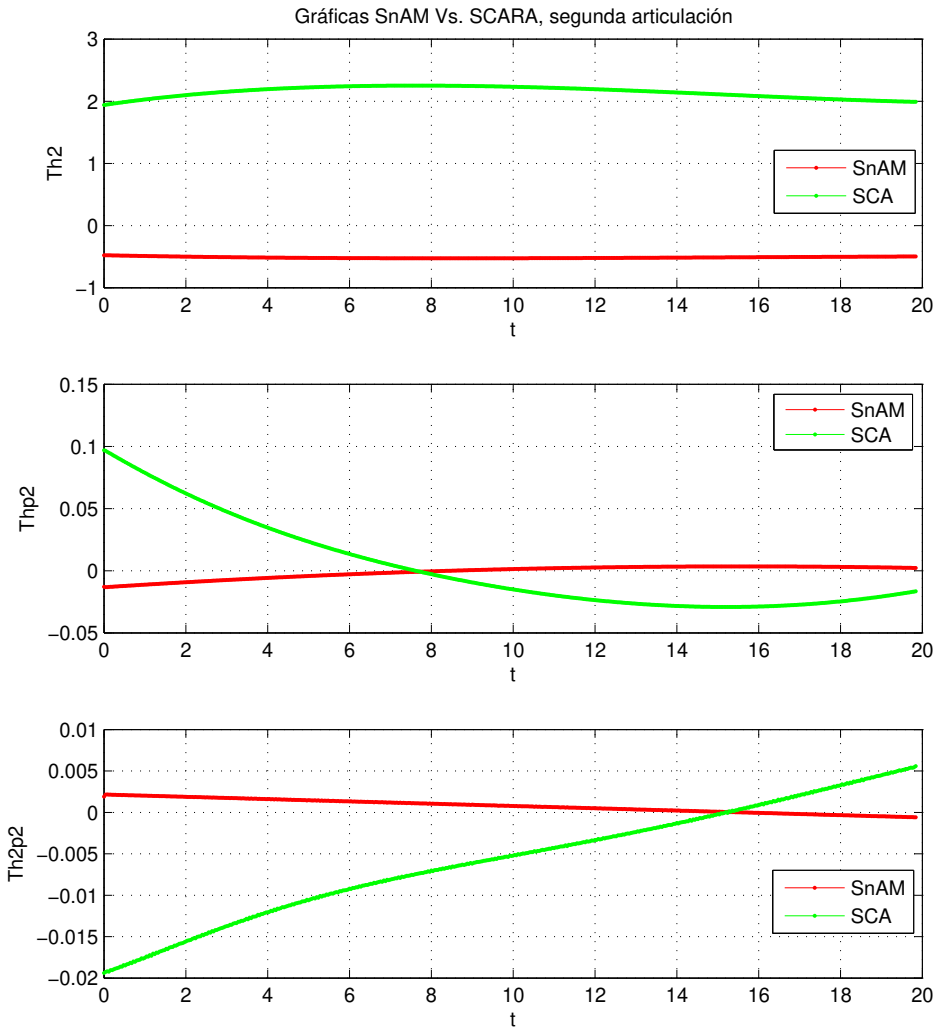

Figura D.29: Graficación de valores de la segunda articulación.

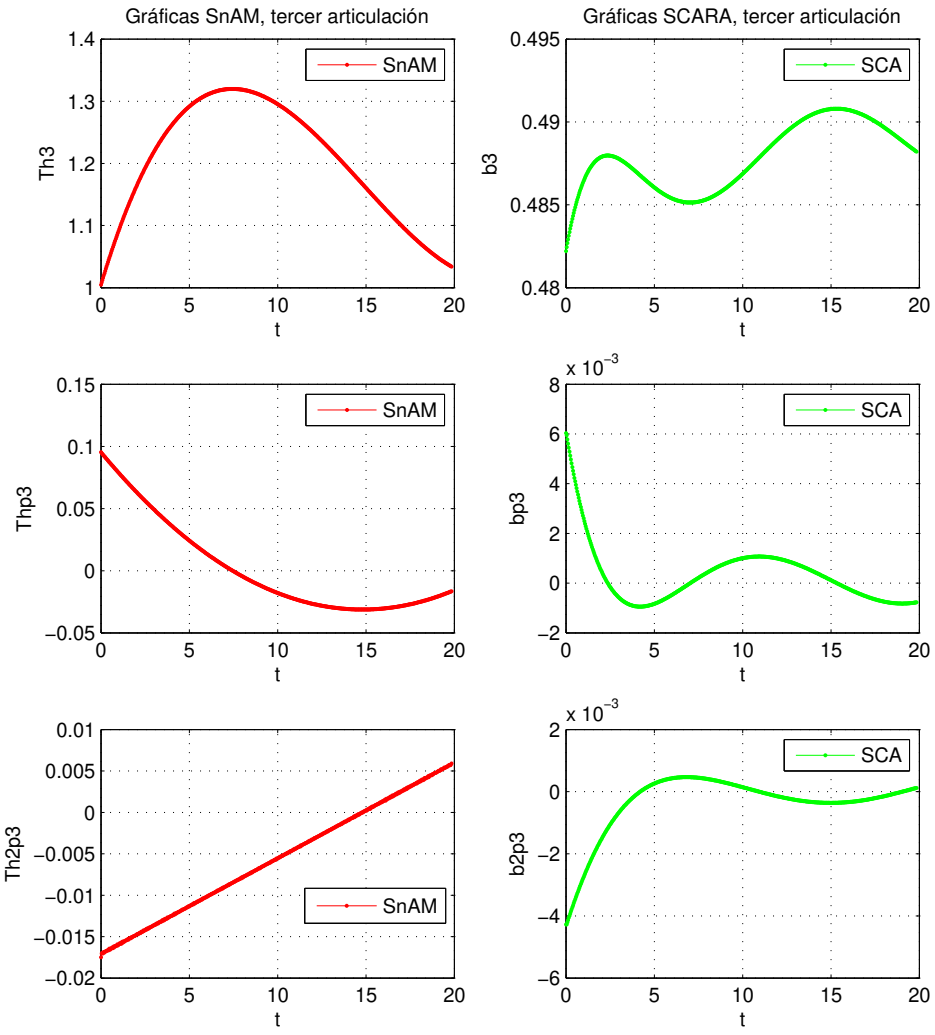

Figura D.30: Graficación de valores de la tercer articulación.

D.11. Trayectoria sinusoidal. SnAM Vs. antropomórfico, segunda prueba

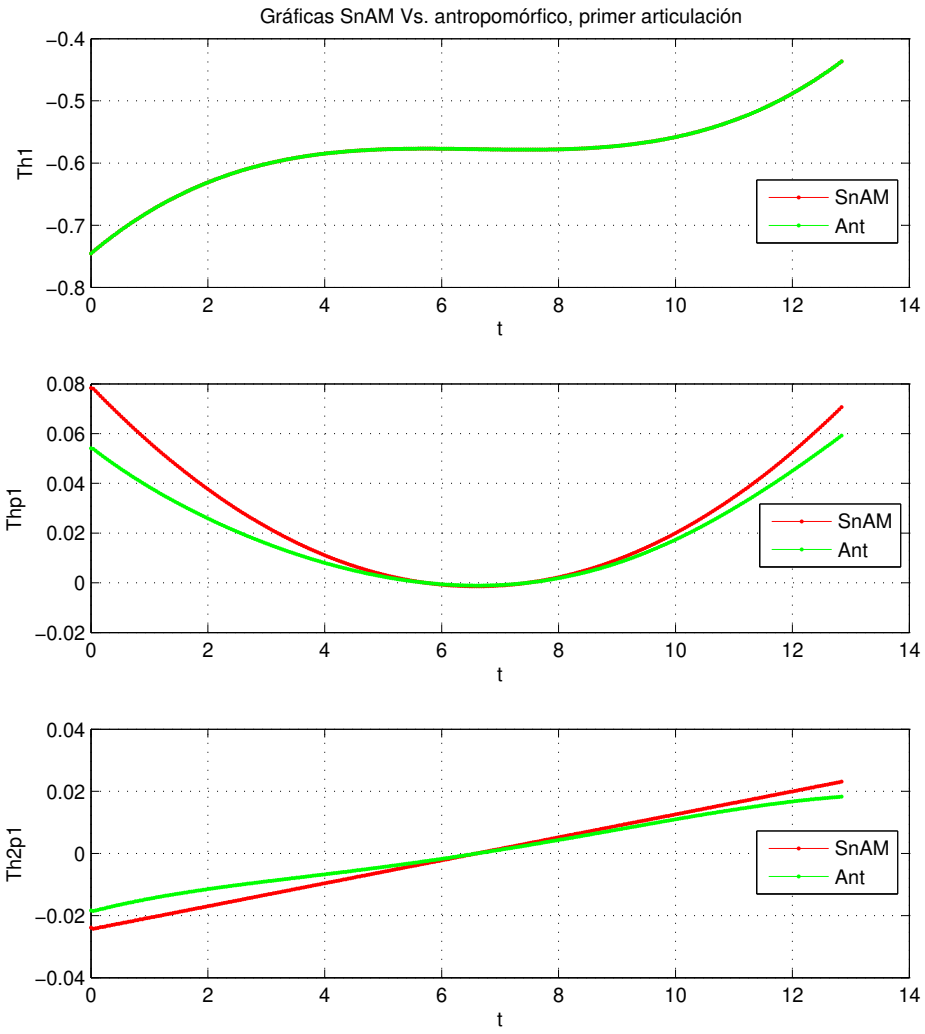

Figura D.31: Graficación de valores de la primer articulación.

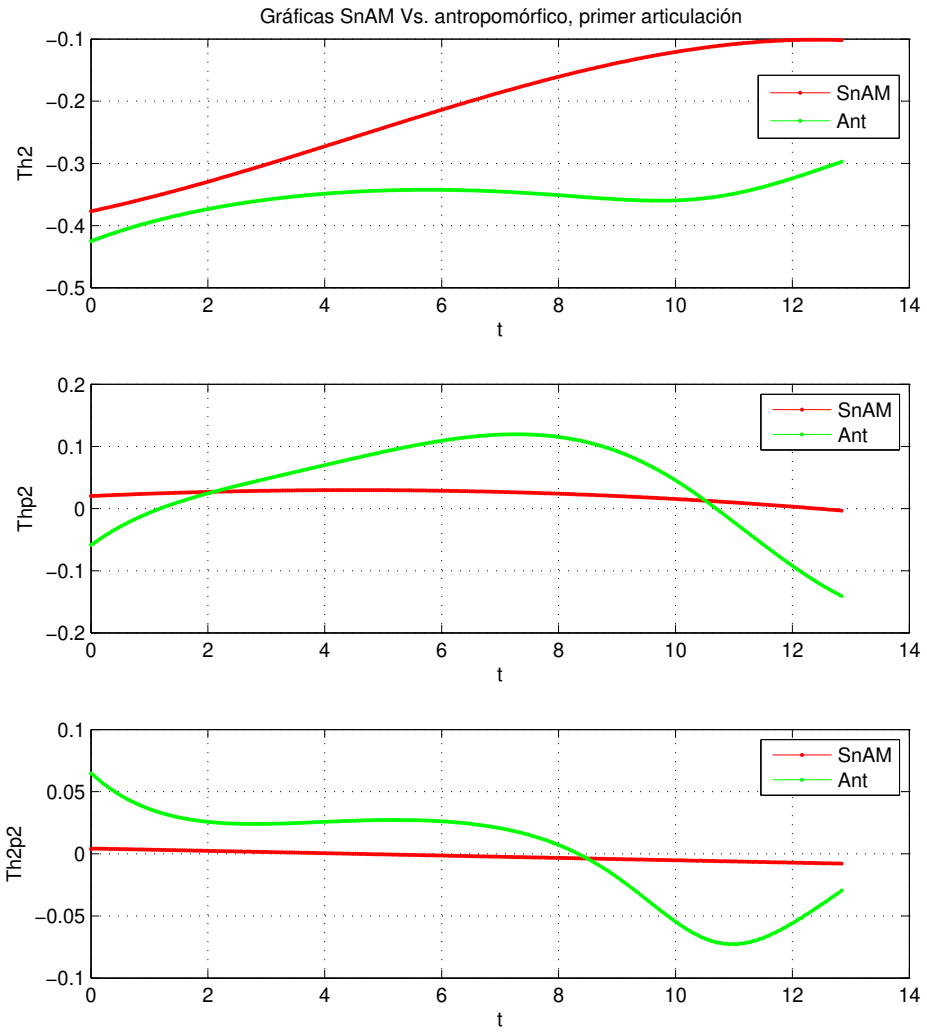

Figura D.32: Graficación de valores de la segunda articulación.

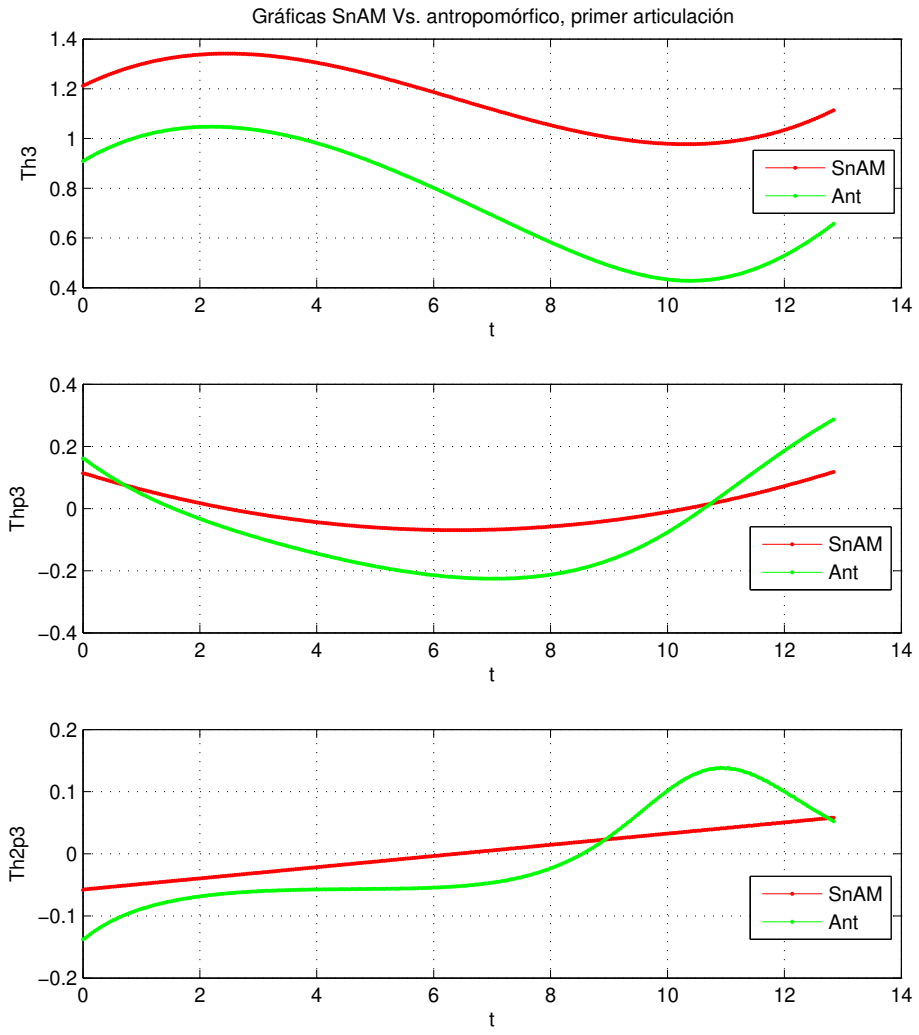

Figura D.33: Graficación de valores de la tercer articulación.

D.12. Trayectoria sinusoidal. SnAM Vs. antropomórfico, tercer prueba

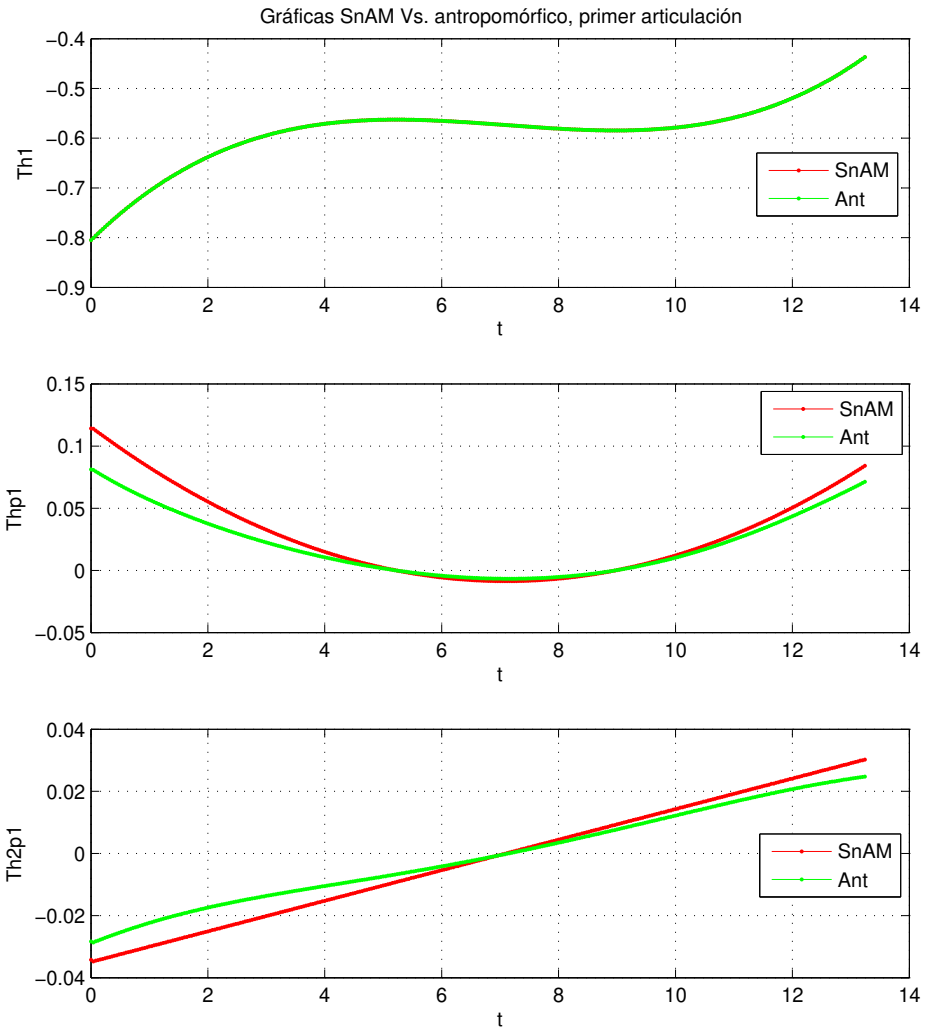

Figura D.34: Graficación de valores de la primer articulación.

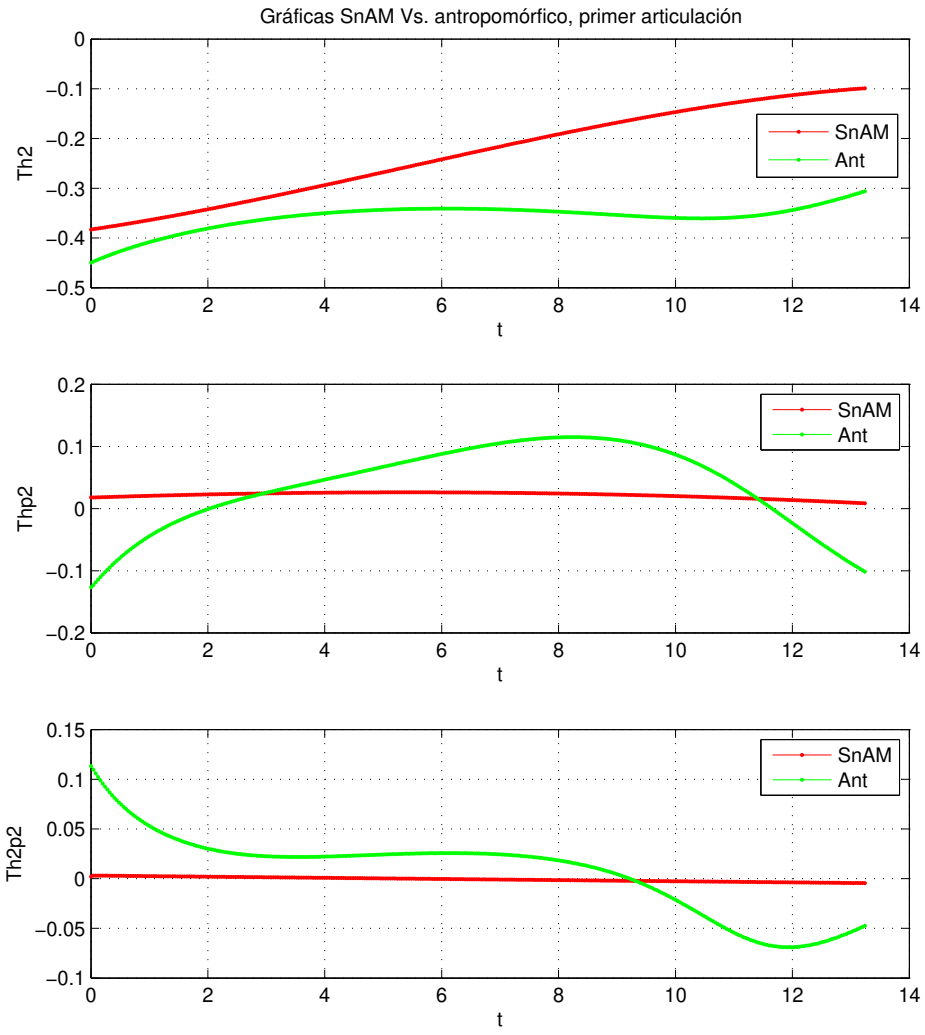

Figura D.35: Graficación de valores de la segunda articulación.

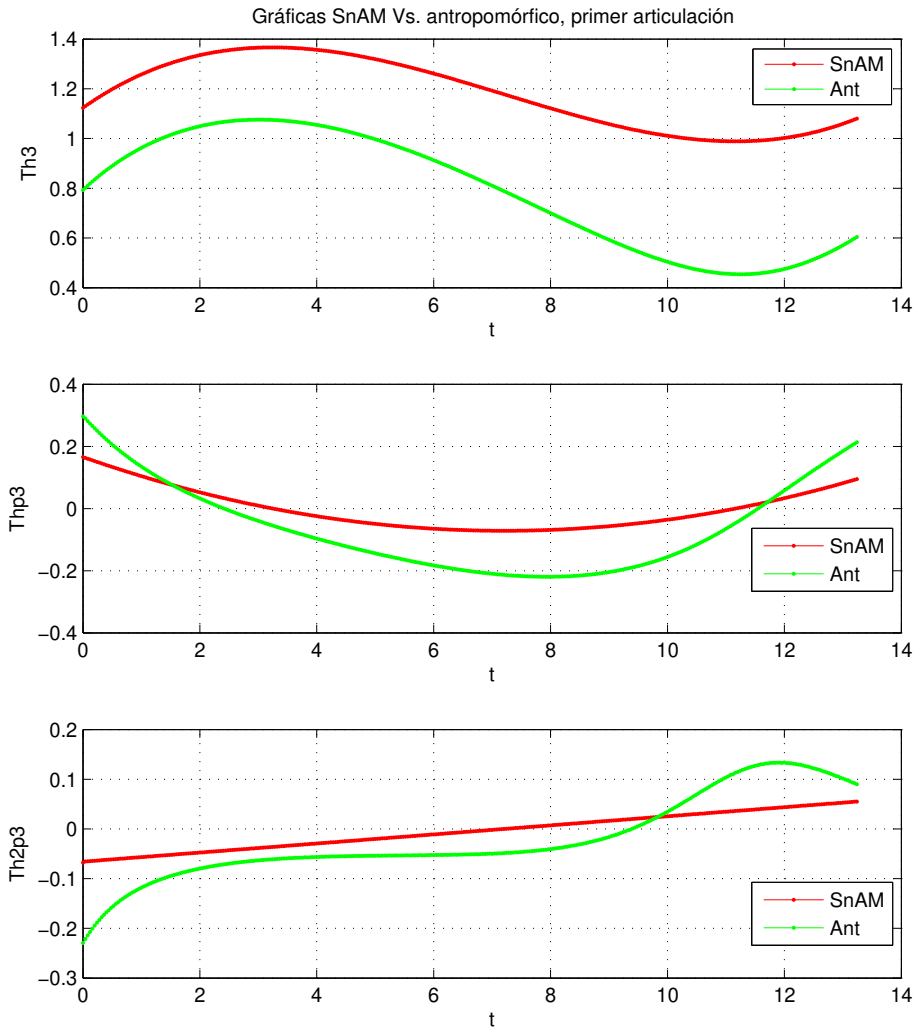

Figura D.36: Graficación de valores de la tercer articulación.

D.13. Trayectoria sinusoidal. SnAM Vs. esférico, segunda prueba

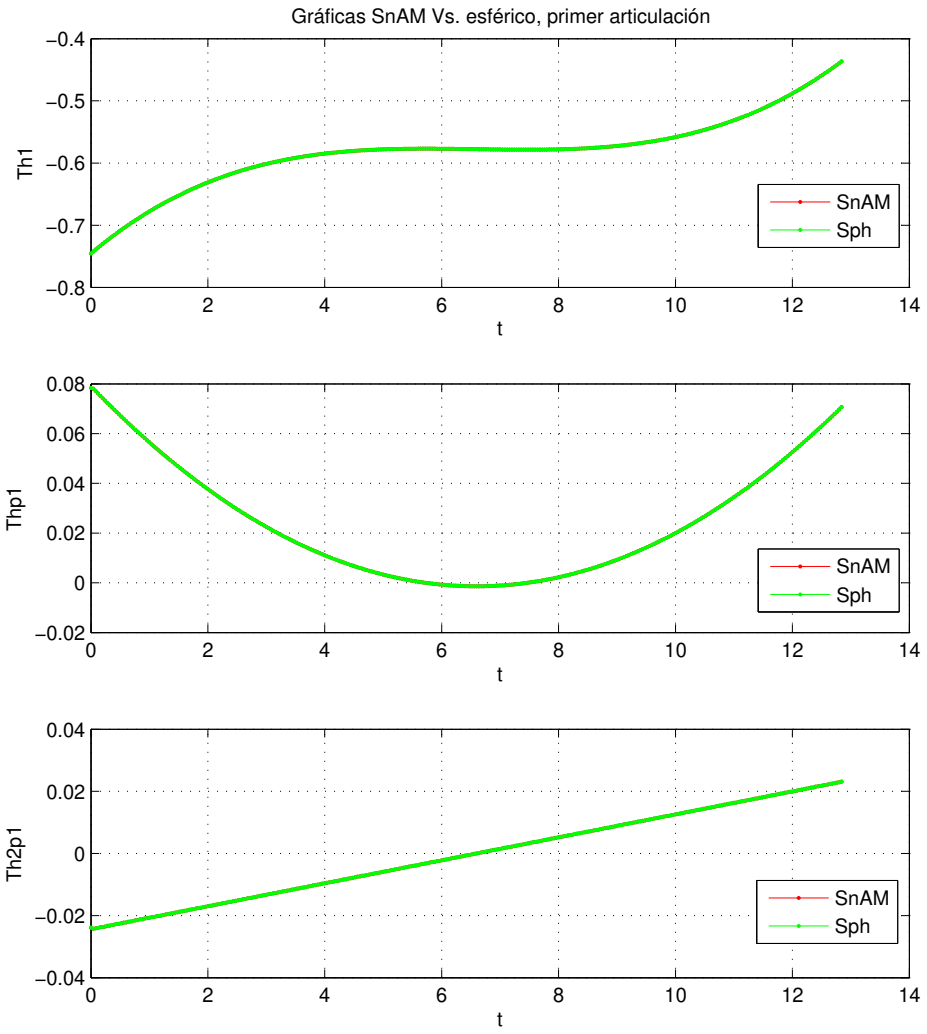

Figura D.37: Graficación de valores de la primer articulación.

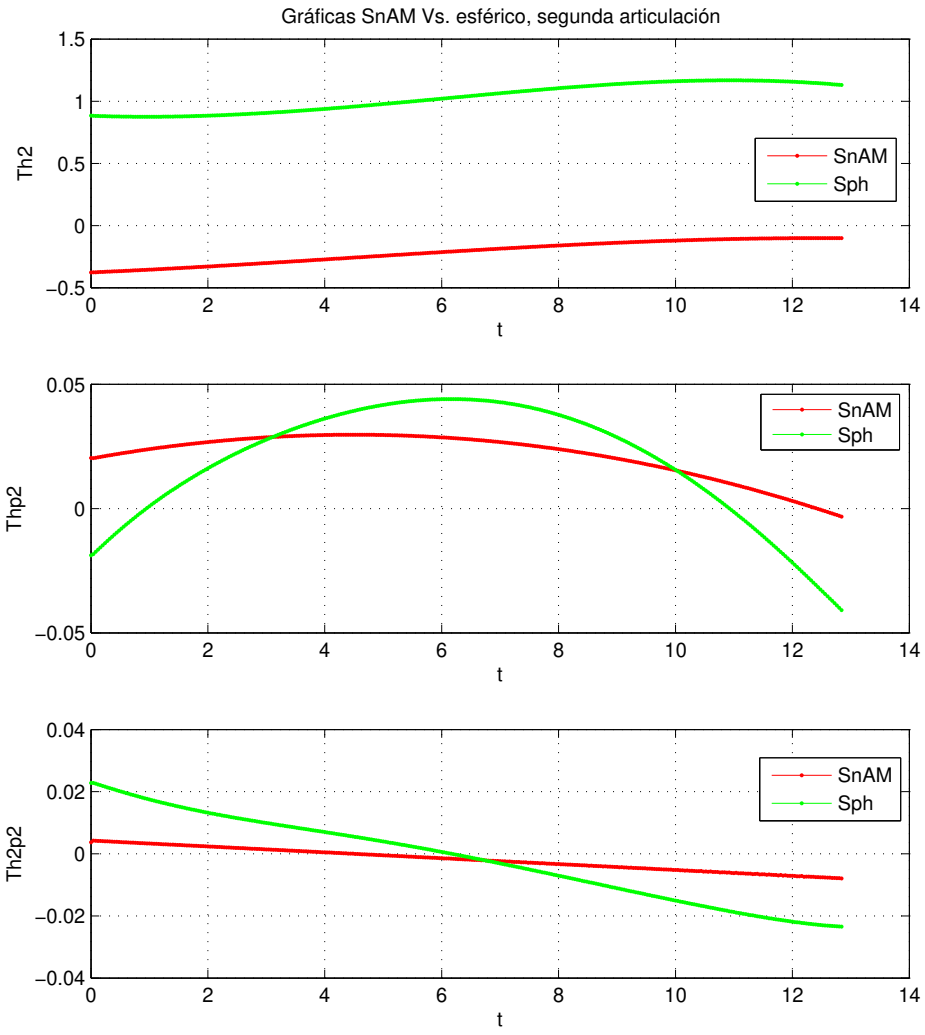

Figura D.38: Graficación de valores de la segunda articulación.

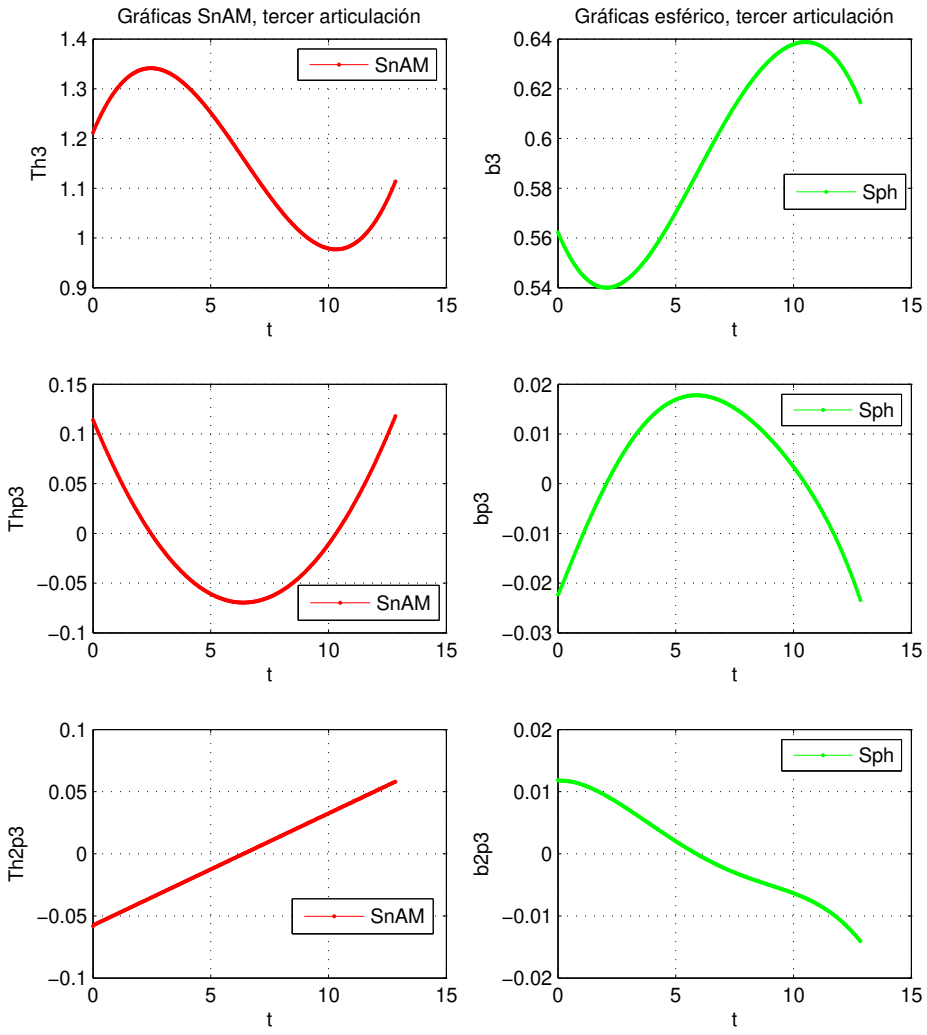

Figura D.39: Graficación de valores de la tercer articulación.

D.14. Trayectoria sinusoidal. SnAM Vs. esférico, tercer prueba

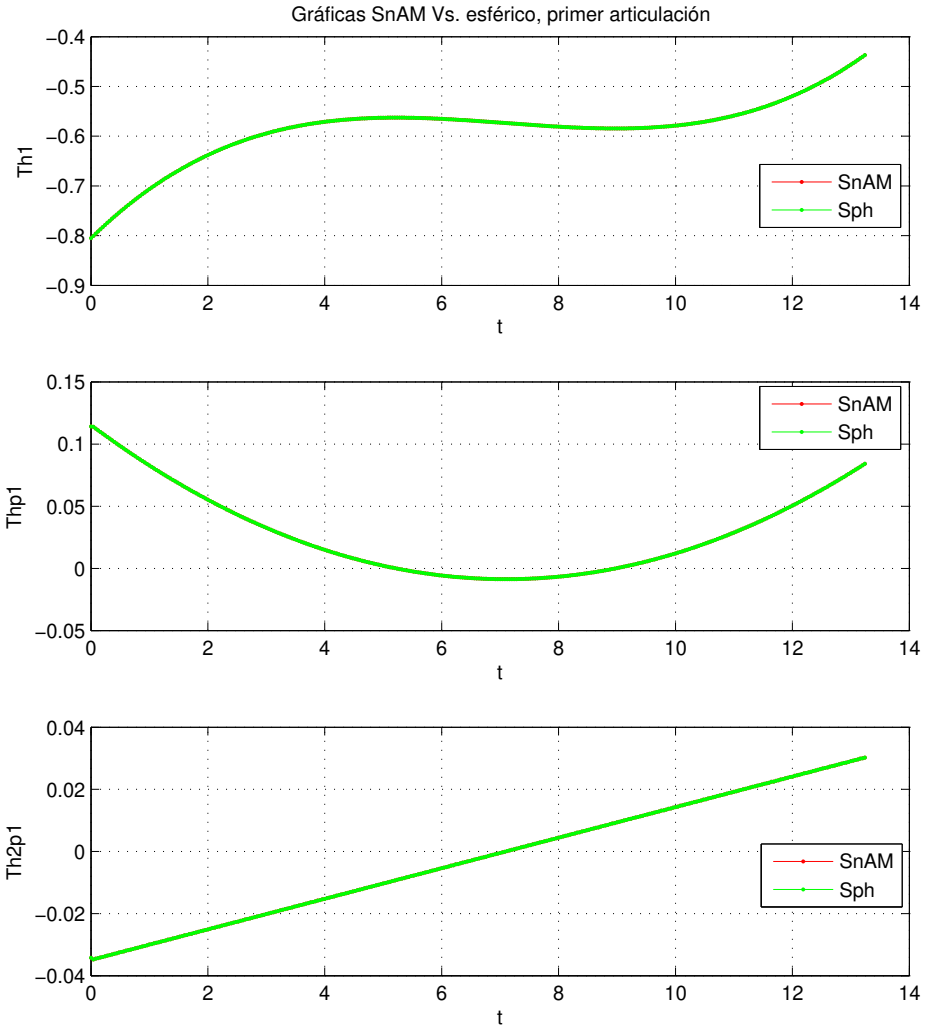

Figura D.40: Graficación de valores de la primer articulación.

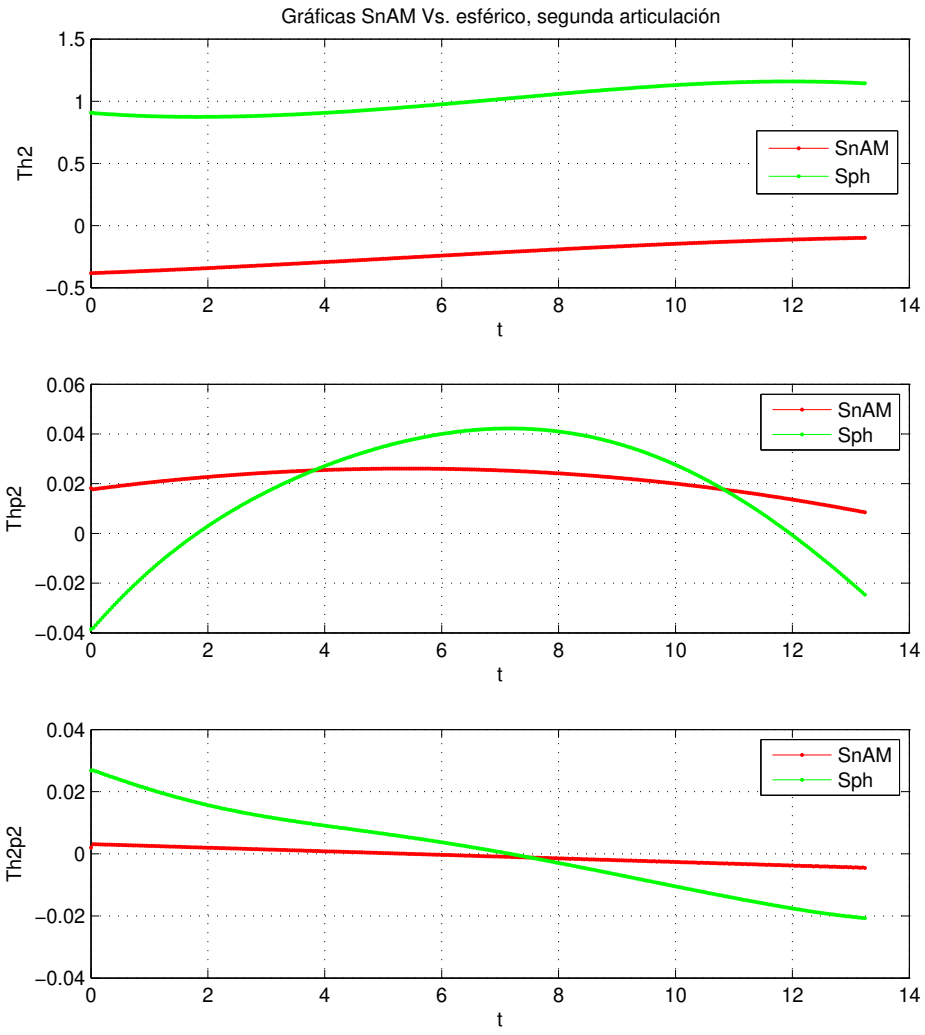

Figura D.41: Graficación de valores de la segunda articulación.

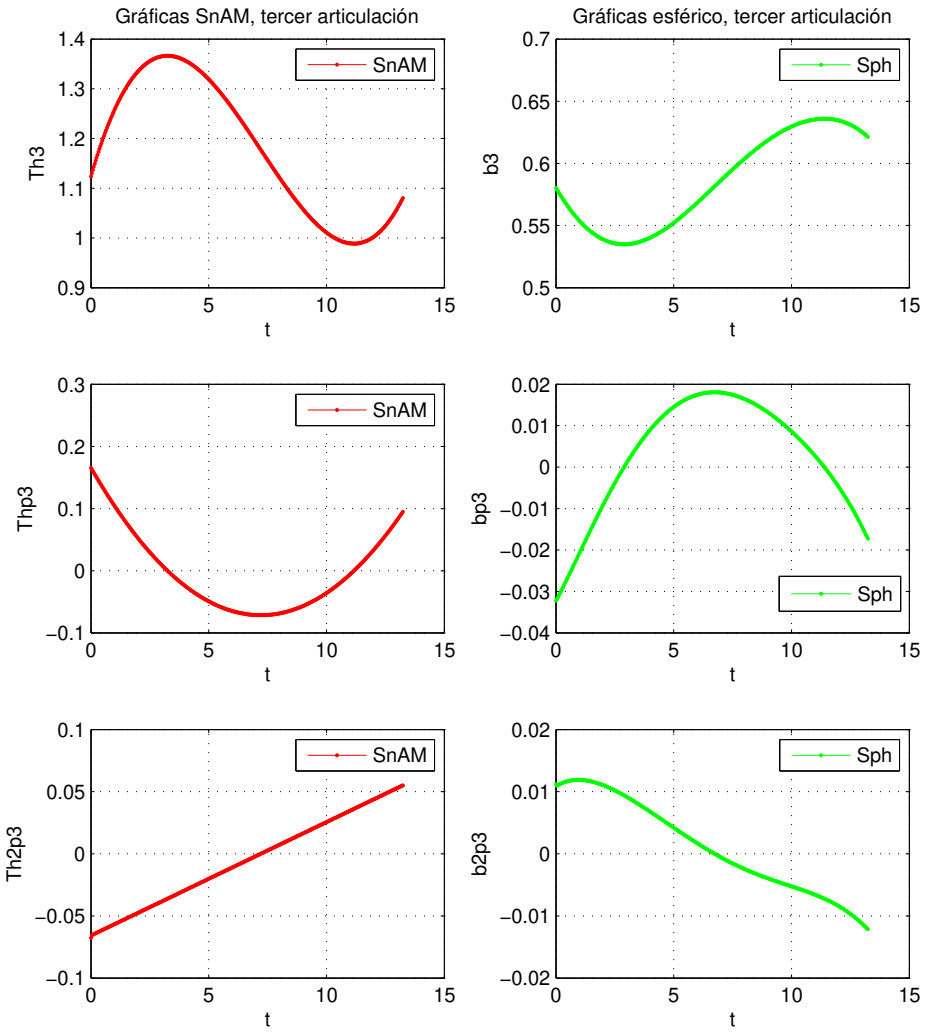

Figura D.42: Graficación de valores de la tercer articulación.

Bibliografía

[1] Spong Mark W., Seth Hutchinson, and Vidsagaray M., *Robot Modeling and Control*, John Wiley and Sons Inc. New York, first edition, 2008.

[2] 3-Axis EZ Module RP-HMSz, Imagen tomada de: `http://robots.epson.com/product-photos`. Consultado el: 26 de Mayo de 2015.

[3] IRB 2400, Imagen tomada de: `http://new.abb.com/products/robotics/es/robots-industriales/irb-2400`. Consultado el: 26 de Mayo de 2015.

[4] Adept eCobra 800, Imagen tomada de: `http://www.adept.com/products/robots/scara/ecobra-800/downloads`. Consultado el: 26 de Mayo de 2015.

[5] González Palacios M. A., *Advanced engineering platform for industrial development*, Journal of Applied Research and Technology, 2012, 10(3), pp. 309-326.

[6] Cuevas Ledesma, Stephanie Melissa, *Análisis y simulación de vibraciones en medios continuos mediante la plataforma ADEFID*, Tesis de Licenciatura, Universidad de Guanajuato, 2015.

[7] González Barbosa, Erick Alejandro, *Síntesis, simulación y control de posición de manipuladores seriales no redundantes*, Tesis de Maestría, Universidad de Guanajuato, 2008.

[8] González Palacios M.A., González Barbosa E.A. y Aguilera Cortés L.A., *An interactive software package for the simulation of serial manipulators*, , ECTC Proceedings, 2009.

[9] Peña Gallo, Rogelio, *Diseño y construcción de una mesa de dos grados de libertad para pruebas de procesos industriales*, Tesis de Maestría, Universidad de Guanajuato, 2011.

[10] *SNAP PAC SYSTEM SPECIFICATION GUIDE* (2015), (Disponible en `http://www.opto22.com/documents/1696_SNAP_PAC_System_Specification_Guide.pdf`). Consultado el: 15 de junio de 2015.

[11] *C Time Library*, Disponible en: `http://www.cplusplus.com/reference/ctime/?kw=time.h`. Consultado el: 15 de junio de 2015.

[12] Chapra Steven C., Canale Raymond P., *Métodos numéricos para ingenieros*, McGraw-Hill, Quinta edición, 2006.

[13] MathWorks Inc., *MATLAB, The language of technical computing*, Version 6, 1984-2001.

[14] González-Palacios M. A., *The unified orthogonal architecture of industrial serial manipulators*, Robotics and Computer-Integrated Manufacturing, Volume 29, Issue 1, February 2013.