Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

Puesta en marcha del robot *Omni Phantom* de *Sensable*

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO Y ELECTRÓNICO

P R E S E N T A:

MAURO GILBERTO LÓPEZ RODRÍGUEZ

Director de tesis:

DR. MARCO ANTONIO ARTEAGA PÉREZ

MÉXICO, D. F. **2014**

Jurado:

PRESIDENTE: ING. Gabriel Alejandro Jaramillo Morales

VOCAL: DR. Marco Antonio Arteaga Pérez

SECRETARIO: M.I. Ricardo Garibay Jiménez

1ER. SUPLENTE: DR. Gerardo René Espinosa Pérez

2DO. SUPLENTE: DR. Paul Rolando Maya Ortiz

Esta tesis fue elaborada en el Laboratorio de Robótica del Edificio de Estudios de Posgrado de la Facultad de Ingeniería; a cargo del Dr. Marco Antonio Arteaga Pérez.

Tutor de tesis:

Dr. Marco Antonio Arteaga Pérez

A mi madre, Vicenta

A mi padre, Hilario

A mi hermano, César

Agradecimientos

A la Universidad Nacional Autónoma de México por ofrecerme tanto en mi paso como estudiante, de la cual me siento orgulloso en pertenecer.

Al Dr. Marco Arteaga Pérez quien me aceptó para desempeñar el servicio social y posteriormente como tesista, estoy en deuda por la oportunidad que me brindó en trabajar con usted.

Al futuro doctor, maestro y amigo Alejandro Giles por ser la persona que me llevó por buena dirección durante el desarrollo de la tesis.

A mis compañeros de laboratorio Javier Pliego, Daniel Castro, Karla García, Juan Carlos Trujillo y Juan Carlos Albarrán por toda la ayuda y consejos que me brindaron durante el desarrollo de la tesis.

A mis queridos colegas de control Michel Rojas, Isaac Ortega, Oscar Vásquez, Daniel Mendoza y Raúl Canseco, con quienes disfruté pasar cada minuto durante la realización de la tesis.

A mis queridos amigos de la Facultad Efraín Flores, David Torres, Diana Montaño, Servando Rafael, Diego Hernández y Felipe Ascencio.

Agradezco a PAPIIT IN109611, por poner la confianza en mí y mis compañeros para desarrollar y experimentar con el control de robots con restricciones no holonómicas y modelo dinámico desconocido.

Finalmente, a todas aquellas personas que intervinieron directa o indirectamente en mi formación como ingeniero. A todos ellos ¡Gracias!

Índice de figuras

Índice de cuadros

Índice general

Capítulo 1

Introducción

1.1. Antecedentes y estado del arte

A medida que crece la población mundial, la demanda de bienes se va incrementando rápidamente, la necesidad de producir más en menor tiempo es un objetivo en común de muchas empresas. Éstas se ayudan de las mejoras tecnológicas, del control realimentado y del desarrollo de herramientas especializadas y así dividir el trabajo en tareas más pequeñas y específicas. En los comienzos de la automatización, las primeras máquinas especializadas se empleaban sólo para tareas simples como poner tapones a las botellas o verter caucho líquido en moldes para neumáticos, lo que fue un avance importante para la época. Sin embargo, ninguna de estas máquinas podía alcanzar objetos alejados y colocarlos en la posición y orientación adecuada. Debido a algunas limitaciones como éstas, se dudaba si se podía sustituir la versatilidad de un brazo humano.

Las primeras máquinas a las que podemos llamar robots fueron los teleoperadores, que consistían en simples servo-mecanismos, que repetían lo que un operador a distancia hacía al mismo tiempo. A partir de estos robots comenzaron a surgir nuevos manipuladores para manejar elementos radiactivos. Estos eran del tipo maestro-esclavo, desarrollados en los laboratorios de *Oak Ridge y Argonne National Laboratories.*

A inicios de la década de 1960 se pone a prueba el primer manipulador industrial en una planta de *General Motors*, (el *Unimate* resultaría ser un éxito rotundo en la industria automotriz). Este robot era un manipulador controlado por computadora y fue programado para manejar y ensamblar piezas. Desde entonces surgen ideas para obtener realimentación sensorial, como los sensores de posición, de presión e incluso ya se pensaba en realimentación por visión. Poco después se desarrolló un sistema que incluía manipuladores, cámaras y micrófonos.

En los años siguientes surgen los robots articulados. Luego, se comenzaron a desarrollar nuevas configuraciones de robots, los que son la base de muchos robot actuales: *el PUMA (Programable Universal Universal Arm) de IBM en 1975 y el SCARA de Digital Electric Automation en 1979*, por mencionar algunos ejemplos.

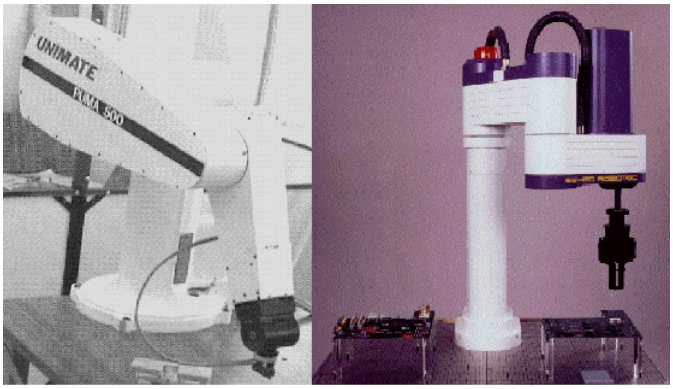

Figura 1.1: Manipulador PUMA (izquierda) y robot SCARA (derecha).

En la actualidad, con el desarrollo de ordenadores cada vez más poderosos, poner en práctica las teorías y los algoritmos de control desarrollados por los investigadores no resulta tan difícil en cuanto a la implementación se refiere. Incluso se puede predecir el comportamiento del sistema con la ayuda de una simulación, siempre y cuando se tenga un modelo suficientemente aproximado que describa su comportamiento.

Aplicación háptica

El robot *Omni Phantom* está diseñado para realizar tareas hápticas. Se pueden encontrar muchas aplicaciones en áreas tan distintas como diseño y medicina, siendo esta última un área con muchos avances en beneficio de la humanidad. Actualmente se encuentran disponibles en el mercado una variedad de simuladores que pueden ser usados de manera gratuita gracias a su código abierto, como lo es APSiDeTH, un simulador para entrenamiento de futuros dentistas que se basa en técnicas de visión 3D estereoscópica, interacción háptica y animación facial. Otra aplicación un poco más sofisticada es el sistema pre operativo desarrollado por el departamento de Mecatrónica de la Universidad Politécnica Timisoara de Rumania, para la reconstrucción de la rodilla humana. Este entorno es controlado por un sistema háptico que incluye al robot *Omni Phantom* el cual sitúa, orienta y calcula la superficie virtual de la tibia donde la reconstrucción del ligamento se ubicará. En el Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET) de la Universidad Nacional Autónoma de México se ha logrado realizar un simulador con sensación táctil en una cirugía de próstata.

El robot *Omni Phantom* Figura 1.2 cuenta con seis articulaciones, de las cuales sólo las primeras tres están actuadas, mediante motores de corriente directa. Para las seis articulaciones, se cuenta con encoders que permiten medir la posición con buena precisión.

Figura 1.2: Robot *Omni Phantom* de *Sensable*

Características relevantes

Para determinar las ecuaciones cinemáticas y dinámicas del robot *Omni*, es necesario conocer algunas características importantes, como son: intervalos de valores que pueden tomar las variables articulares, la resolución que ofrecen los encoders, el espacio de trabajo, etc. Estos datos (ofrecidos por el fabricante) se muestran en la Tabla 1.1.

Especificación	Descripción
Peso	3 Libras (1360.78 g)
Grados	6 GDL
Rigidez	>Eje X 1.26 N / mm >Eje Y 2.31 N / mm >Eje Z 1.02 N / mm
Espacio de trabajo	160 w x 120 h x 70 d [mm]
Comunicación	IEEE-1394 Fire Wire 6 a 6 pines
Fuerza máxima nominal	0.75 lbf (3.3 N)
Resolución nominal de la posición	0.055 [mm]

Tabla 1.1 Especificaciones técnicas del *Omni Phantom*

Con ayuda del paquete de librerías *Open Haptics* proporcionado por *Sensable*, es posible obtener el número de pulsos de los encoders en las seis articu-

laciones del robot, así como aplicar voltaje a los motores que activan las tres primeras articulaciones. Cabe mencionar que se utiliza un convertidor digital analógico de 2 bytes para enviar los voltajes a los motores.

0	θ_1	θ_2	θ_3
Cuentas	-2424 *a* 2356	-80 *a* -4440	*No es constante*
Grados	-50 *a* 55	0 *a* 105	*No es constante*

Tabla 1.2 Relación entre giro y cuentas de encoders (posición para el robot)

El valor de los pulsos para la tercera articulación no es constante ya que debido al diseño mecánico del robot, el movimiento de la tercera articulación es limitada por la posición de la segunda, importante mencionar que cuando θ_2 tiende a su máximo valor (105 °), el rango de valores de θ_3 disminuye, si θ_2 llega a su mínimo valor 0°, el rango de θ_3 es el máximo, incluso llegando a colisionar con la base del robot.

$\theta_2[^\circ]$	$\theta_3[^\circ]$
0	-119
0	-13
24	4
45	25
73	25
105	-32
105	23

Tabla 1.3 Valores de θ_3 dado θ_2

1.2. Motivación

El control automático siempre ha estado ligado a la robótica. Desde sistemas sencillos en lazo abierto (*e.g.*, con motores a pasos) hasta sistemas complejos con controladores no lineales. Las aplicaciones de los robots son amplias y van desde los sistemas microelectromecánicos hasta los sistemas biorrobóticos.

El robot *Omni Phantom* abordado en esta tesis puede utilizarse en una gran variedad de tareas que involucran las interfaces hápticas y la teleoperación. En háptica se pueden encontrar aplicaciones en medicina. En la actualidad, los simuladores quirúrgicos son una realidad. Aplicando control se logra hacer que los actuadores permitan sentir las texturas de un tejido vivo sin poner vidas en peligro. En la misma medicina ya se utilizan robots para ayudar a los cirujanos, donde la precisión de una brazo humano no basta para garantizar que no haya riesgo, en procedimientos como puede ser una cirugía delicada en los ojos o el colocar caderas artificiales. En un futuro muy cercano será común no estar en la misma sala que el cirujano. La implementación de robots esclavos y maestros en telecirugía podría facilitarle a cualquier persona en el mundo acudir con el mejor cirujano, aunque ambos estén en lugares muy distantes.

Para el diseño e implementación de algoritmos de control no lineales o de observadores de estados que eliminen la necesidad de sensores físicos, es de gran utilidad contar con un modelo matemático del robot. Este modelo es útil no sólo para el diseño de algoritmos, sino para la predicción de escenarios reales mediante la simulación numérica de los mismos, lo cual tiene muchas ventajas como pueden ser: ahorrar tiempo de experimentación, validar un diseño, evitar daños al equipo real, etc. Por lo tanto, la obtención de un modelo matemático que aproxime suficientemente el comportamiento real del robot será de gran utilidad para el desarrollo de la investigación en teleoperación y en dispositivos hápticos que se basarán en dicho manipulador.

1.3. Objetivos de la tesis

- Realizar una interfaz gráfica sencilla con *Visual Studio* para realizar el control del robot, lectura de encoders, posición del efector final, con la finalidad de ser utilizada como plataforma para implementar algoritmos de control y de observación.
- Realizar pruebas experimentales con controladores PD, PID en seguimiento de trayectorias.
- Comparar el desempeño de un control PD y un control adaptable.
- Obtener el modelo dinámico del robot y una parametrización lineal del mismo.

1.4. Planteamiento del problema

A pesar de que el robot *Omni phantom* de *Sensable*, es un robot diseñado para realizar háptica, también puede ser muy útil en actividades de teleoperación. Debido a que este robot es relativamente nuevo, no se encuentra reportado un modelo dinámico del mismo, además de que el *software* dado por el fabricante no permite la implementación directa de algoritmos de control o técnicas de estimación. Por lo tanto, se plantea:

- Realizar la interfaz de programación para implementar algoritmos de control utilizando como base la lectura de posición articular y la manipulación de los voltajes de los motores que actúan las tres primeras articulaciones.
- Identificar los parámetros del modelo dinámico del robot mediante una parametrización lineal del mismo y técnicas de identificación reportadas en la literatura.

1.5. Logros y aportaciones

En la realización de este trabajo se utilizaron conceptos vistos a lo largo de la licenciatura, de modelado de sistemas físicos, control, programación estructurada, etc.

Se diseñó un programa con una interfaz gráfica que servirá como base para la implementación de algoritmos de control y de observación.

Se obtuvieron las ecuaciones que describen el comportamiento cinemático del efector final. La importancia de la cinemática directa e inversa radica en el hecho de poder mapear las variables articulares a coordenadas cartesianas ó viceversa.

Con la obtención del Jacobiano geométrico se logró obtener el modelo dinámico que se aproxima muy bien al comportamiento del robot *Omni Phantom*.

Se logró obtener una parametrización lineal del mismo, las aportaciones anteriores servirán para desarrollos posteriores donde se necesita conocer la dinámica, como lo son en las áreas de teleoperación e interfaces hápticas.

1.6. Organización del trabajo

Para seguir el desarrollo de la tesis se comenzará explicando la debida asignación de los sistemas coordenados siguiendo el algoritmo de Denavit-Hartenberg, para la cinemática directa e inversa. El capítulo dedicado al modelo cinemático se encarga de la obtención de estas ecuaciones y culmina con la obtención del jacobiano geométrico del efector final.

El Capítulo 3 esta dedicado a la obtención de cada una de las variables involucradas en el modelo dinámico del robot *Omni Phantom*. Esta sección comienza con el Jacobiano geométrico, pero a diferencia del Jacobiano obtenido en el capítulo anterior, esta vez se obtendrá respecto a cada uno de los centroides de cada eslabón, así la relación necesaria entre las velocidades de eslabón y articulación, para determinar la energía cinética del robot estará hecha.

Continua con la obtención de la energía potencial de cada eslabón. Para este fin es necesario conocer los vectores de posición de cada centroide, que se obtendrán del Jacobiano geométrico de cada centroide. Las fuerzas de Coriolis se obtendrán con los símbolos de Cristoffel, la única variable que no necesita estar bien detallada es la de disipación de energía, debido a que dificultad en cuantificar los coeficientes de fricción requieren un análisis más profundo. El contenido de este capítulo resulta ser fundamental para la representación paramétrica.

En el Capítulo 4 se diseñan trayectorias que no pasen por singularidades o que causen colisiones entre los eslabones y la base, con el objetivo de ser las señales de referencia para poner a prueba los controladores PD, PID y adaptable.

Para concluir este trabajo el Capítulo 5 está dedicado a la obtención de una representación paramétrica del modelo dinámico y de un regresor en función de los errores como lo plantean Slotine y Li [5]. Este regresor modificado será utilizado para el seguimiento de trayectorias y comparar las señales de error de un PD y del adaptable.

Capítulo 2

Modelo Cinemático

Para analizar el comportamiento de un manipulador, el primer paso suele ser obtener los modelos cinemáticos, los cuales proporcionan información sobre el movimiento del manipulador considerando únicamente sus cracterísitcas geométricas e ignorando las fuerzas y pares que lo generan.

Para determinar las ecuaciones cinemáticas, se utilizará el método matricial propuesto por Denavit y Hartenberg en 1955 [2]. El objetivo que tiene la cinemática directa es poder determinar la posición del efector final respecto a un sistema base, con ecuaciones en función únicamente de las variables articulares del robot, mientras que la cinemática inversa es útil para determinar las variables articulares cuando se conoce la posición del efector final respecto a un sistema base.

La utilidad de estas ecuaciones radica en poder realizar un mapeo del espacio articular al cartesiano y viceversa. A continuación se muestra la asignación del sistema base y los ángulos de giro (variables articulares) en la siguiente imagen isométrica del robot:

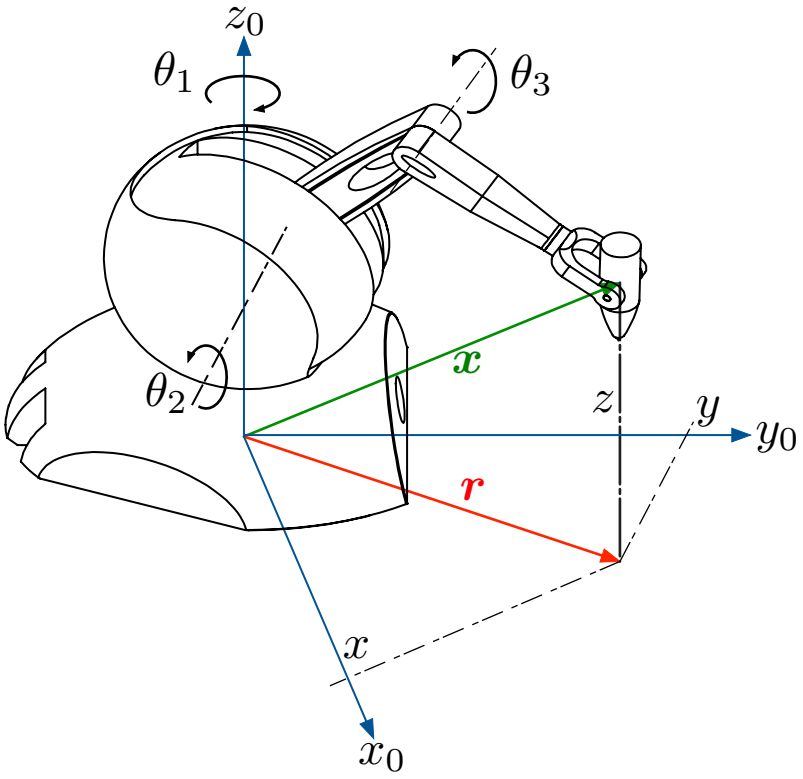

Figura 2.1: Vista isométrica del robot y asignación del sistema base

2.1. Convención y Parámetros de Denavit Hartenberg

Para poder describir el movimiento de cualquier eslabón rígido respecto al sistema base, se utiliza el algoritmo de Denavit-Hartenberg (1955), este algoritmo nos permite hacer una adecuada asignación de sistemas coordenados inerciales a cada eslabón, teniendo en cuenta dos restricciones.

Restricciones para Sistemas Coordenados DH
(**DH1**) $\mathbf{x}_i$ debe ser perpendicular a $\mathbf{z}_{i-1}$
(**DH2**) $\mathbf{x}_i$ debe intersectar a $\mathbf{z}_{i-1}$

Para los robots del tipo serial los sistemas se asignan respetando las dos restricciones D-H. Sin embargo este robot tiene una característica peculiar: la tercer variable articular no es afectada por el movimiento de la segunda articulación como suele pasar en los robots seriales. Por lo tanto el tercer sistema

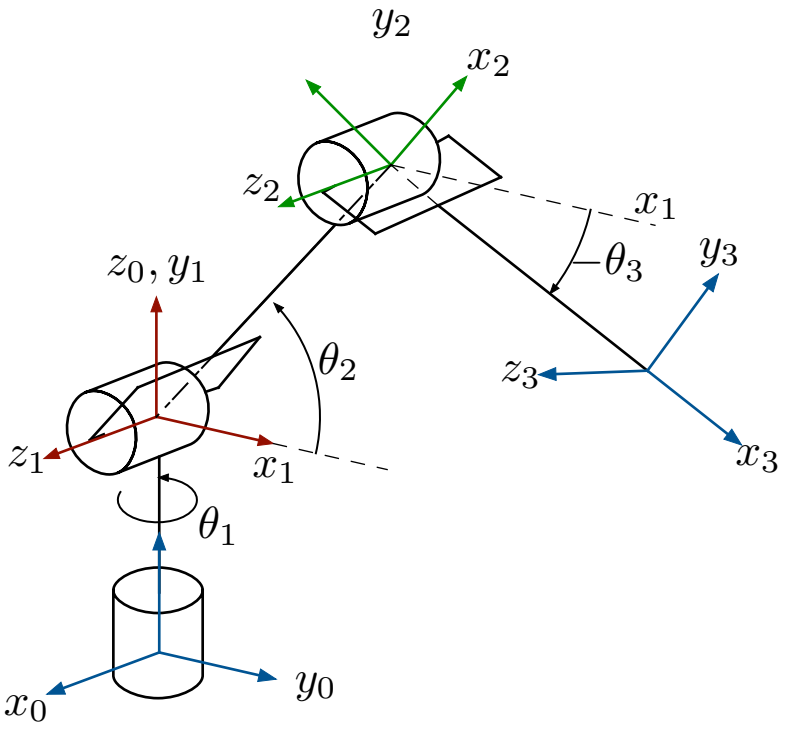

Figura 2.2: Asignación de sistemas coordenados y ángulos de giro

coordenado afectará un poco la obtención de los parámetros D-H. A pesar de esta singularidad, se puede representar cualquier desplazamiento o rotación con una transformación homogénea $\mathbf{A}_i$ de 4×4, es decir como el producto de cuatro transformaciones básicas. De acuerdo al convención D-H los parámetros D-H para el robot *Omni Phantom* son los siguientes:

Eslabón	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	a_3	0	0	θ_3^*

Tabla 2.1 Parámetros D-H robot *Omni Phantom* (*=Variable)

Es importante mencionar que la tercer articulación está medida respecto a un eje horizontal, y no de $\mathbf{x}_2$ a $\mathbf{x}_1$ como lo indica el algoritmo D-H. Con los parámetros D-H del robot *Omni Phantom* obtenidos, se pueden utilizar para hacer una representación matricial mediante la siguiente transformacón homogénea:

$$\mathbf{A}_i = \mathbf{Rot}_{z,\theta_i}\mathbf{Trans}_{z,di}\mathbf{Trans}_{x,ai}\mathbf{Rot}_{x,\alpha_i} \tag{2.1}$$

Desarrollando (2.1) se obtiene:

$$\mathbf{A}_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

donde θ_i: Es el ángulo medido del eje $\mathbf{x}_{i-1}$ al eje $\mathbf{x}_i$ teniendo como eje de giro a $\mathbf{x}_{i-1}$.

$\mathbf{d}_i$ es la distancia del sistema coordenado i-ésimo$_{-1}$ hasta la intersección de los ejes $\mathbf{z}_{i-1}$ y $\mathbf{x}_i$ medida sobre $\mathbf{z}_{i-1}$.

a_i es la distancia del sistema coordenado i-ésimo hasta la intersección de $\mathbf{z}_{i-1}$ y $\mathbf{x}_i$ medida sobre $\mathbf{x}_i$.

α_i es el ángulo medido del eje Z_{i-1} al eje $\mathbf{z}_i$, tomando como eje de giro al eje $\mathbf{x}_i$.

La transformción homogénea $\mathbf{A}_i$ de 4×4, contiene al mismo tiempo una rotación (matriz de 3×3) y una traslación (vector columna de 3×1), por lo tanto se puede representarla de la siguiente manera:

$$\mathbf{A}_i = \begin{bmatrix} {}^0\mathbf{R}_i & {}^0\mathbf{o}_i \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

donde ${}^0\mathbf{R}_i$ es una matriz de rotación, que representa la rotación del sistema coordenado i-ésimo, respecto al sistema base y ${}^0\mathbf{o}_i$ representa la traslación del sistema coordenado i-ésimo, respecto al sistema base. Estas matrices serán muy útiles para obtener la matriz de inercia del robot.

Por lo tanto las matrices homogéneas estarán dadas por:

$$\mathbf{A}_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$\mathbf{A}_2 = \begin{bmatrix} c_2c_1 & -s_2c_1 & s_1 & a_2c_2s_1 \\ c_2s_1 & -s_2s_1 & -c_1 & a_2c_2c_1 \\ s_2 & c_2 & 0 & a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$\mathbf{A}_3 = \begin{bmatrix} c_3c_1 & -s_3c_1 & s_1 & (a_2c_2 + a_3c_3)c_1 \\ c_3s_1 & -s_3s_1 & -c_1 & (a_2c_2 + a_3c_3)s_1 \\ s_3 & c_3 & 0 & a_2s_2 + d_1 + a_3s_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Para comprobar la ecuación (2.6), la cual representa la posición y orientación del efector final, considérese la posición del robot mostrada en la Figura 2.3 donde: $\theta_1 = 90°$, $\theta_2 = 90°$ y $\theta_3 = 0°$. Este caso particular se escogió por ser demostrativo, ya que de acuerdo al espacio de trabajo esta posición no podría ser alcanzada.

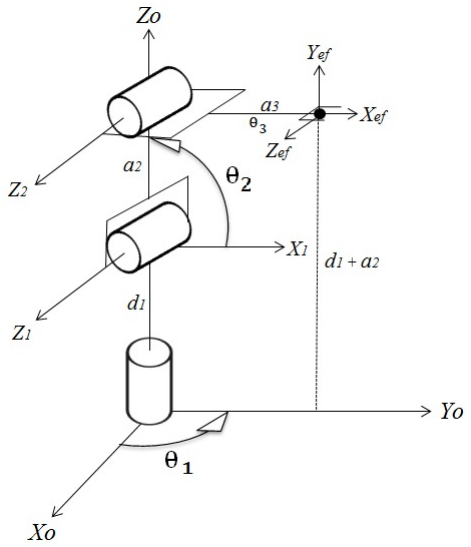

Figura 2.3: Caso particular para la matriz homogénea

$$\mathbf{A}_3 = \begin{bmatrix} c_0c_{90} & -s_0c_{90} & s_{90} & (a_2c_{90} + a_3c_0)c_{90} \\ c_0s_{90} & -s_0s_{90} & -c_{90} & (a_2c_{90} + a_3c_0)s_{90} \\ s_0 & c_0 & 0 & a_2s_{90} + d_1 + a_3s_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\mathbf{A}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Saber interpretar los valores de la matriz de transformación homogenea proporciona la información suficiente para conocer la posición y orientación del efector final. Por parte de la matriz de rotación se obtienen la orientación debido a que se sabe que $\mathbf{z}_3$ está proyectada sobre $\mathbf{x}_0$, $\mathbf{x}_3$ sobre $\mathbf{y}_0$ y $\mathbf{y}_3$ sobre $\mathbf{z}_0$. Nótese que la magnitud de cada renglón o columna es unitario. Cómo se esperaba, la posición es determinada con el último vector columna, sobre el eje coordenado $\mathbf{x_0}$ no hay proyección alguna, a_3 es la distancia del efector final proyectada sobre $\mathbf{y_0}$, mientras que en $\mathbf{z_0}$ la proyección es $a_2 + d_1$.

2.2. Cinemática Directa

Para obtener las ecuaciones que relacionan la posición del efector final respecto al sistema base, hay que explicar algunas carcaterísticas propias del robot *Omni Phantom* es decir, hay que tener en cuenta que tipo de articulaciones, cuantos grados de libertad y sus dimensiones físicas. El robot *Omni Phantom* es del tipo antropomórfico con muñeca esférica. Sin embargo, es importante mencionar que para el desarrollo de la tesis solamente se utilizarán las tres primeras articulaciones, es decir solo interesa la parte antropomórfica del robot.

Un robot antropomórfico tiene tres articulaciones de rotación (RRR), también suele llamarsele configuración articulada. Por convención para nombrar a las variables articulares se han adoptado las letras q_i y d_i. Para este caso sólo se utilizará q_i y que representa el ángulo de rotación e i es el número de articulación y para cualquier tipo de configuración con n articulaciones se obtendran $n+1$ eslabones.

A partir de la Figura 2.2 se determinan las ecuaciones para la cinemática directa, utilizando algunas relaciones geométricas simples.

Ecuaciones para la Cinemática Directa:

$$x = (a_2c_2 + a_3c_3)c_1 \tag{2.8}$$

$$y = (a_2c_2 + a_3c_3)s_1 \tag{2.9}$$

$$z = (a_2s_2 + a_3s_3) + d_1 \tag{2.10}$$

donde x, y, z es la posición de la proyección del efector final sobre el sistema coordenado base $(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$, d_1 es la altura de la base del robot, decir la distancia del sistema coordenado base al 1 y mide 127 [mm], a_2 y a_3 son las distancias de los eslabones, miden 129 [mm] y 133 [mm] respectivamente, q_1, q_2, q_3 son las variables articulares, q_1 está medido de $\mathbf{x}_0$ a $\mathbf{x}_1$, q_2 está medido en sentido antihorario, desde $\mathbf{x}_1$ a $\mathbf{x}_2$ y q_3 es la tercer variable articular independiente de q_2, es decir el ángulo de giro se mide a partir de una horizontal o una paralela siempre a $\mathbf{x}_1$.

2.3. Cinemática Inversa

El objetivo de la cinemática inversa es determinar las variables articulares en función únicamente de la posición del efector final. En general, es más complicado que en la cinemática directa, ya que no siempre puede obtenerse una solución analítica cerrada a este problema. Sin embargo, se puede obtener una solución para el problema cinemático inverso mediante un enfoque geométrico.

Para obtener las variables articulares es conveniente hacer proyecciones sobre planos para comprender su comportamiento. En este sentido, la solución para q_1 resulta sencilla de apreciar, así que la primer variable articular que se obtendrá es q_1. Para esto basta con hacer una proyección de x, y, z, (Posición del efector final) sobre el plano $\mathbf{x}_0$ y $\mathbf{y}_0$, y nótese que no hay relación del efector en Z_0 con q_1.

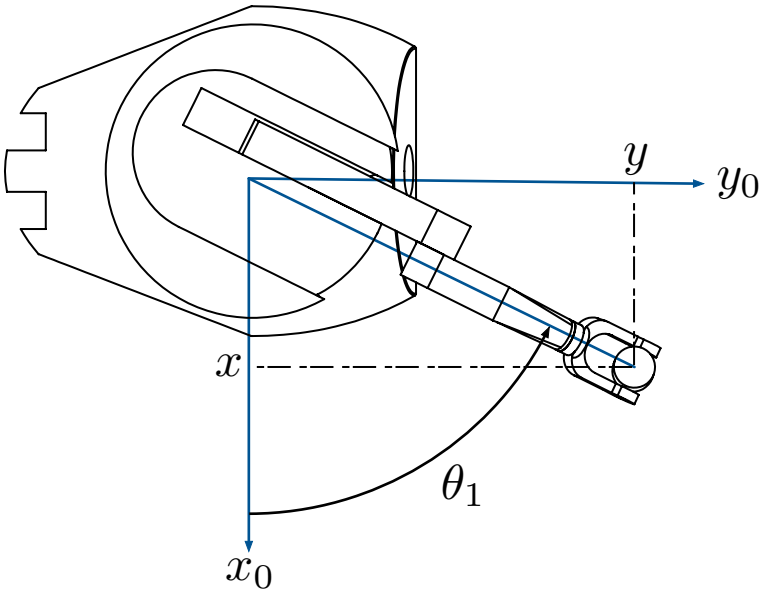

Figura 2.4: Proyección sobre el plano $\mathbf{x_0}$ y $\mathbf{y_0}$, para mostrar la obtención de q_1

De la Figura 2.4 q_1 se obtiene directamente que:

$$q_1 \;=\; \text{atan2}(y, x) \tag{2.11}$$

Para poder determinar q_2 y q_3 se hace una proyección sobre el plano $\mathbf{z}_1$, $\mathbf{y}_1$, tal que sea más evidente la trayectoria que pueden describir los eslabones del manipulador. Es importante mencionar que se obtendrá primero q_2, ya que q_3 está en función de q_2.

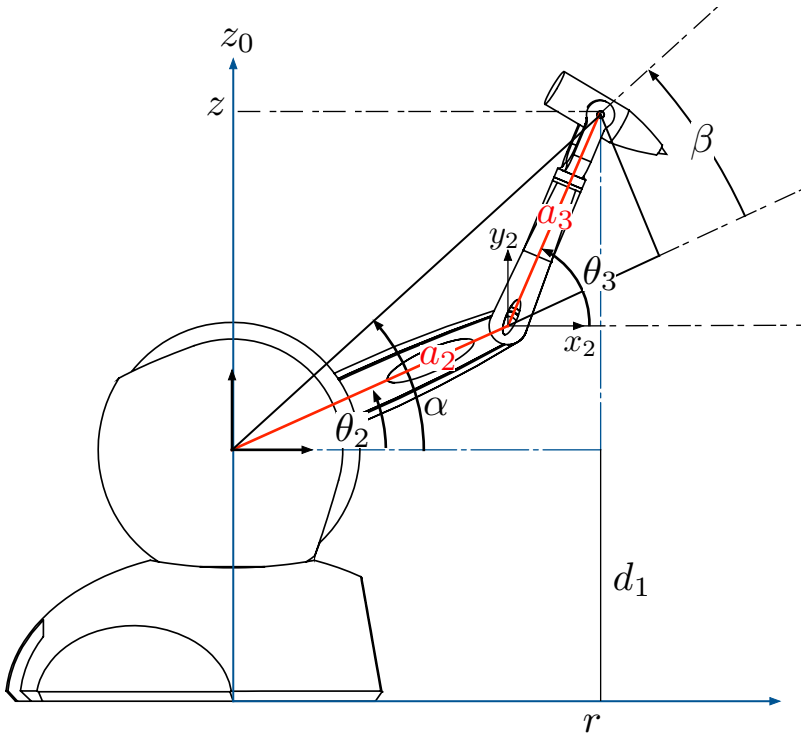

Figura 2.5: Proyección sobre el plano z_0 y y_0, para mostrar la obtención de q_2 y q_3

Para visualizar más fácilmente la obtención de q_2 y q_3 apóyese de la Figura 2.6 donde se muestran las variables auxiliares necesarias:

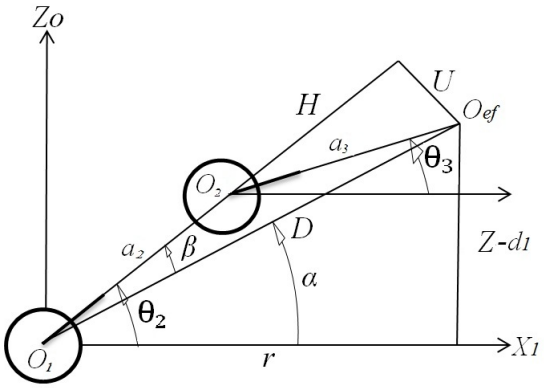

Figura 2.6: Asignación de variables auxiliares

De la Figura 2.6 se observa que q_2 es la suma de $\alpha + \beta$ donde α es el ángulo formado por los segmentos D, r, y β es el ángulo formado por los segmentos D, a_2, para determinar β se utilizaran relaciones trigonométricas obtenidas a partir del triángulo formado por los segmentos $a_2 + H$, $Uy\ D$:

Ecuaciones útiles para determinar β:

$$D^2 = X^2 + Y^2 + (Z + d_1)^2 \tag{2.12}$$

$$a_3^2 = H^2 + U^2 \tag{2.13}$$

$$\cos\beta = \frac{a_2 + H}{D}; \tag{2.14}$$

$$D^2 = (a_2 + H)^2 + U^2 \tag{2.15}$$

Desarrollando (2.15) se tiene:

$$D^2 = a_2^2 + 2a_2H + H^2 + U^2 \tag{2.16}$$

Sustituyendo (2.13) en (2.16) tenemos:

$$D^2 = a_2^2 + 2a_2H + a_3^2 \tag{2.17}$$

Tomando H de (2.14) y sustituyendo en (2.17) se tiene:

$$D^2 = 2a_2D\cos\beta + a_3^2 \tag{2.18}$$

De (2.18) se despeja $\cos\beta$ y se sustituye por B:

$$\cos\beta = \frac{D^2 - a_3^2 + a_2^2}{2a_2D} = B \tag{2.19}$$

$$\beta = atan2(\sqrt{1 - B^2}, B) \tag{2.20}$$

$$\alpha = atan2(Z - d_1, r) \tag{2.21}$$

$$\theta_2 = \alpha - \beta = atan2(-\sqrt{1 - B^2}, B) - atan2(Z - d_1, r) \tag{2.22}$$

$$\theta_3 = atan2(Z - d_1 - a_2s_2, r - a_2c_2) \tag{2.23}$$

La ecuación (2.22) permite obtener el tipo de solución eligiendo el signo en la raíz cuadrada, para este caso se elige el signo negativo de la raíz, ya que representa la solución de codo arriba del robot *Omni*, físicamente no existe la solución codo abajo.

2.4. Velocidad Cinemática

Para complementar las ecuaciones de la cinemática directa e inversa de la poscición del efector final, se expondrá la relación que existe entre la velocidad lineal y angular del efector final respecto a cada una de las articulaciones. Recordemos que ya se ha obtenido la posición del efector con (2.2). Por lo tanto, se pueden extraer los elementos necesarios para encontrar esta relación a partir de (2.4), (2.5) y (2.6). Para determinar la velocidad lineal y angular del efector final, dado que únicamente hay articulaciones rotacionales, aplicando directamente:

$$\begin{bmatrix}\mathbf{J}_{vi}\\ \mathbf{J}_{\omega i}\end{bmatrix} = \begin{bmatrix}\mathbf{z}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1})\\ \mathbf{z}_{i-1}\end{bmatrix}$$

Determinando $\mathbf{J}_{v1}$ de la siguiente forma:

$$\mathbf{J}_{v1} = \begin{bmatrix}\mathbf{z}_0 \times ({}^0\mathbf{o}_1 - {}^0\,\mathbf{o}_0) & 0 & 0\end{bmatrix}$$

$$\mathbf{J}_{v1} = \begin{bmatrix}\mathbf{z}_0 \times ({}^0\mathbf{o}_1 - {}^0\,\mathbf{o}_0) & 0 & 0\end{bmatrix} = \begin{bmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \times \begin{pmatrix}0\\0\\d_1\end{pmatrix} - \begin{pmatrix}0\\0\\0\end{pmatrix} & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \tag{2.24}$$

$$\mathbf{J}_{\omega 1} = \begin{bmatrix}\mathbf{z}_0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\end{bmatrix} \tag{2.25}$$

Para comprender la utilidad del Jacobiano, comencemos con las velocidades lineales y angulares (2.24) y (2.25), recuérdese que $\mathbf{J}_{v1}$ representa la velocidad lineal del sistema coordenado o_1 respecto al sistema coordenado base o_0. Pero el sistema o_1 está fijo al extremo de d_1 por lo que la velocidad lineal de d_1 es la misma que o_1 es decir cero, esto resulta evidente al ver la Figura 2.2. Esto significa que por más que se mueva el manipulador no existe un movimiento lineal de o_1. Sin embargo existe movimiento rotacional $\mathbf{J}_{\omega 1}$ entorno al eje $\mathbf{z}$. Nótese que únicamente existe elemento en $\mathbf{J}_{\omega 1}$ en su primer columna. La interpretación de este elemento es que solamente el sistemas coordenado O_1 es afectado por si mismo y no se afecta por el movimiento de los sistemas coordenados o_2 y o_3. Los Jacobianos $\mathbf{J}_{v2}$ y $\mathbf{J}_{\omega 2}$ contienen elementos en sus dos primeras columnas lo que significa que el sistema O_2 se ve afectado por su propio movimiento y por movimiento de o_1 sin que el movimiento de o_3 lo afecte. Para finalizar los Jacobianos $\mathbf{J}_{v3}$ y $\mathbf{J}_{\omega 3}$ se ven afectados por o_1, o_2 y por su propio movimiento, su objetivo es relacionar las velocidades de cada uno de los eslabones con las articulaciones.

Para $\mathbf{J}_{v2}$ se desarrollará :

$$\mathbf{J}_{v2} = \begin{bmatrix}\mathbf{z}_0 \times ({}^0\mathbf{o}_2 - {}^0\,\mathbf{o}_0) & \mathbf{z}_1 \times ({}^0\mathbf{o}_2 - {}^0\,\mathbf{o}_1) & 0\end{bmatrix}$$

donde :

$$\mathbf{z}_0 \times ({}^0\mathbf{o}_2 - {}^0\,\mathbf{o}_0) = \begin{pmatrix}0\\0\\1\end{pmatrix} \times \begin{pmatrix} o_2c_2c_1 \\ o_2c_2s_1 \\ o_2s_2 + d_1 \end{pmatrix} = \begin{pmatrix} -o_2c_2s_1 \\ o_2c_2c_1 \\ 0 \end{pmatrix}$$

$$\mathbf{z}_1 \times ({}^0\mathbf{o}_2 - {}^0\,\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} o_2c_2c_1 & - & 0 \\ o_2c_2s_1 & - & 0 \\ o_2s_2 + d_1 & & d_1 \end{pmatrix} = \begin{pmatrix} -c_1s_2o_2 \\ -s_1s_2o_2 \\ c_2o_2 \end{pmatrix}$$

obteniendo finalmente $\mathbf{J}_{v2}$:

$$\mathbf{J}_{v2} = \begin{bmatrix} -o_2c_2s_1 & -c_1s_2o_2 & 0 \\ o_2c_2c_1 & -s_1s_2o_2 & 0 \\ 0 & c_2o_{c2} & 0 \end{bmatrix} \tag{2.26}$$

el Jacobiano de velocidad angular $\mathbf{J}_{\omega\mathbf{2}}$ será:

$$\mathbf{J}_{\omega 2} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & s_1 & 0 \\ 0 & -c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.27}$$

Para determinar J_{v3} se desarrolla:

$$\mathbf{J}_{v3} = \begin{bmatrix} \mathbf{z}_0 \times ({}^0\mathbf{o}_3 - {}^0\,\mathbf{o}_0) & \mathbf{z}_1 \times ({}^0\mathbf{o}_3 - {}^0\,\mathbf{o}_1) & \mathbf{z}_2 \times ({}^0\mathbf{o}_3 - {}^0\,\mathbf{o}_2) \end{bmatrix}$$

donde :

$$\mathbf{z}_0 \times ({}^0\mathbf{o}_3 - {}^0\,\mathbf{o}_0) = \begin{pmatrix}0\\0\\1\end{pmatrix} \times \begin{pmatrix} (o_3c_3 + a_2c_2)c_1 \\ (o_3c_3 + a_2c_2)s_1 \\ o_3s_3 + a_2s_2 + d_1 \end{pmatrix} = \begin{pmatrix} -(o_3c_3 + a_2c_2)s_1 \\ (o_3c_3 + a_2c_2)c_1 \\ 0 \end{pmatrix}$$

$$\mathbf{z}_1 \times ({}^0\mathbf{o}_3 - {}^0\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} (o_3c_3 + a_2c_2)c_1 & - & 0 \\ (o_3c_3 + a_2c_2)s_1 & - & 0 \\ o_3s_3 + a_2s_2 + d_1 & & d_1 \end{pmatrix} = \begin{pmatrix} -c_1(o_3s_3 + a_2s2_) \\ -s_1(o_3s_3 + a_2s2_) \\ o_3c_3 + a_2c_2 \end{pmatrix}$$

$$\mathbf{z}_2 \times ({}^0\mathbf{o}_3 - {}^0\,\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} c_1(o_3c_3 + a_2c_2) & & a_2c_2c_1 \\ s_1(o_3c_3 + a_2c_2) & - & a_2c_2s_1 \\ o_3s_3 + a_2s_2 + d_1 & & a_2s_2d_1 \end{pmatrix} = \begin{pmatrix} -c_1s_3o_3 \\ -s_1s_3o_3 \\ c_3o_3 \end{pmatrix}$$

obteniendo finalmente el Jacobiano de velocidad lineal de $\mathbf{J}_{\mathbf{v3}}$:

$$\mathbf{J}_{v3} = \begin{bmatrix} -s_1(o_3c_3 + a_2c_2) & -c_1(o_3s_3 + a_2s_2) & -c_1s_3o_3 \\ c_1(o_3c_3 + a_2c_2) & -s_1(o_3s_3 + a_2s_2) & -s_1s_3o_3 \\ 0 & o_3c_3 + a_2c_2 & c_3o_3 \end{bmatrix} \tag{2.28}$$

el Jacobiano de velocidad angular $\mathbf{J}_{\omega 3}$:

$$\mathbf{J}_{\omega 3} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.29}$$

Capítulo 3

Modelo Dinámico

En el Capítulo 2 se obtuvieron las ecuaciones cinemáticas, directa e inversa. Estas ecuaciones sólo dependen de la geometría del manipulador y permiten relacionar la posición del efector final y las variables articulares. A diferencia de la cinemática, la dinámica estudia la relación del movimiento del robot y las fuerzas que lo causan. Para implementar los algoritmos de control es muy importante tener en cuenta todas las fuerzas que interactúan con el robot.

En esta sección se obtendrá un modelo que toma en cuenta las fuerzas que interactuan con el manipulador. La necesidad de conocer esto resulta muy útil ya que se puede conocer los pares que se necesitan administrar para lograr el movimiento deseado. Sin embargo, modelar un sistema como lo es un manipulador o cualquier sistema mecánico no es tarea sencilla; las ecuaciones que modelan su comportamiento tomarán en cuenta ciertas variables como lo son las inercias, fuerzas de reacción (Coriolis y centrífugas) y el efecto de la gravedad sobre cada uno de los eslabones. Para modelar el robot se utilizarán las ecuaciones de Euler-Lagrange, que pueden ser usadas para sistemas mecánicos sujetos a restricciones holonómicas. Considérese el Lagrangiano del sistema.

$$\mathcal{L} = \mathcal{K} - \mathcal{P}$$

donde $\mathbf{K}$ es la energá cinética y $\mathbf{P}$ es la energía potencial, por lo tanto la ecuación Euler-Lagrange para cada grado de libertad del manipulador es:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \quad i = 1, ..., n$$

Estas ecuaciones pueden escribirse en forma matricial como:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{3.1}$$

donde la matriz $\mathbf{H}$ es la matriz de inercia del manipulador, $\mathbf{C}$ es la matriz de Coriolis, $\mathbf{D}$ es una matriz diagonal con los coeficientes de fricción (matriz de disipación de energía) y $\mathbf{g}$ es el vector de torques debidos a la gravedad. El objetivo es modelar al sistema de la forma dada por (3.1).

Antes de obtener el modelo dinámico se deben determinar los Jacobianos de los centros de masa de cada eslabón, ya que proporciona las relaciones entre las velocidades angulares y lineales del efector final respecto a las velocidades de cada articulación, relaciones que serán muy útiles para obtener el modelo dinámico.

3.1. Jacobiano Geométrico

La relación de las velocidades en el espacio articular con las velocidades en el espacio cartesiano puede establecerse mediante el Jacobiano analítico o el Jacobiano geométrico. En este trabajo caso se utilizará el geométrico, que es una matriz de $6 \times n$, donde n es el número de eslabones, y donde los tres primeros renglones son debidos al Jacobiano de velocidad lineal, y los otros renglones (inferiores) son debidos a la parte del Jacobiano de velocidad angular, es decir:

$$\mathbf{J}^{6\times n} = \begin{bmatrix} \mathbf{J}_v^{3xn} \\ \mathbf{J}_\omega^{3xn} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{v1} \ldots \mathbf{J}_{vn} \\ \mathbf{J}_{\omega 1} \ldots \mathbf{J}_{\omega n} \end{bmatrix}$$

Debido a que el robot *Omni Phantom* sólo tiene articulaciones rotacionales para la posición, para determinar el Jacobiano basta con seguir las siguientes relaciones:

$$\begin{bmatrix} \mathbf{J}_{\mathrm{vi}} \\ \mathbf{J}_{\omega \mathrm{i}} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} \tag{3.2}$$

Para obtener el Jacobiano se necesita conocer algunas proyecciones y distancias, estos valores se pueden obtenerse de las matrices de transformación homogéneas.

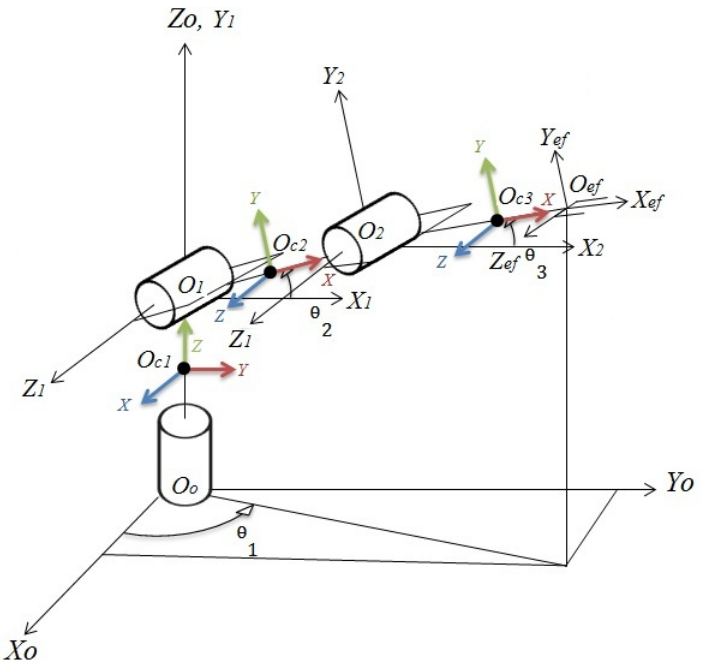

Figura 3.1: Asignación de sistemas coordenados a los centroides de cada eslabón.

De acuerdo a la Figura 3.1, en donde se han resaltado nuevos sistemas coordenados en los centroides de cada eslabón y con las matrices de transformación homogénea se determinó el Jacobiano y se obtuvo lo siguiente:

Las matrices homogéneas para los centroides son:

$$\mathbf{A}_{c1} = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & o_{c1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_{c2} = \begin{bmatrix} c_2c_1 & -s_2c_1 & s_1 & o_{c2}c_2s_1 \\ c_2s_1 & -s_2s_1 & -c_1 & o_{c2}c_2c_1 \\ s_2 & c_2 & 0 & o_{c2}s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_{c3} = \begin{bmatrix} c_3c_1 & -s_3c_1 & s_1 & (a_2c_2 + o_{c3}c_3)s_1 \\ c_3s_1 & -s_3s_1 & -c_1 & (a_2c_2 + o_{c3}c_3)c_1 \\ s_3 & c_3 & 0 & a_2s_2 + d_1 + o_{c3}s_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Por lo que puede determinarse $\mathbf{J}_{vc1}$ de la siguiente forma:

$$\mathbf{J}_{vc1} = \begin{bmatrix} \mathbf{z}_0 \times ({}^0\mathbf{o}_{c1} - {}^0\,\mathbf{o}_0) & 0 & 0 \end{bmatrix}$$

donde $\mathbf{z}_0$ es la proyección del eje $\mathbf{z}_0$ respecto a $\mathbf{z}_0$, ${}^0\mathbf{o}_{c1}^0$ este elemento es extraído de $\mathbf{A}_{c1}$ y son los tres primeros elementos de la cuarta columna y representa la distancia del sistema coordenado $\mathbf{o}_{c1}$ respecto al origen y donde ${}^0\mathbf{o}_0$ es la distancia de $\mathbf{o}_0$ respecto a $\mathbf{o}_0$.

$$\mathbf{J}_{\mathrm{vc1}} = \begin{bmatrix} {}^0\mathbf{z}_0 \times ({}^0\mathbf{o}_{c1} - {}^0\,\mathbf{o}_0) & 0 & 0 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0_{c1} \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

Para determinar el Jacobiano de la velocidad angular basta con proyectar $\mathbf{z}_0$ sobre $\mathbf{z}_0$ es decir:

$$\mathbf{J}_{\omega \mathrm{c1}} = \begin{bmatrix} \mathbf{z}_0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.4}$$

Para los Jacobianos de velocidades lineales y angulares de los centroides del eslabón a_2 y a_3, se realiza algo similar, para $\mathbf{J}_{vc2}$ y $\mathbf{J}_{wc2}$ se tiene:

$$\mathbf{J}_{\mathrm{vc2}} = \begin{bmatrix} \mathbf{z}_0 \times ({}^0\mathbf{o}_{c2} - {}^0\,\mathbf{o}_0) & \mathbf{z}_1 \times ({}^0\mathbf{o}_{c2} - {}^0\,\mathbf{o}_1) & 0 \end{bmatrix}$$

Donde :

$$\mathbf{z}_0 \times ({}^0\mathbf{o}_{c2} - {}^0\,\mathbf{o}_0) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} o_{c2}c_2c_1 \\ o_{c2}c_2s_1 \\ o_{c2}s_2 + d_1 \end{pmatrix} = \begin{pmatrix} -o_{c2}c_2s_1 \\ o_{c2}c_2c_1 \\ 0 \end{pmatrix}$$

$$\mathbf{z}_1 \times ({}^0\mathbf{o}_{c2} - {}^0\,\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} o_{c2}c_2c_1 \\ o_{c2}c_2s_1 \\ o_{c2}s_2 + d_1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ d_1 \end{pmatrix} = \begin{pmatrix} -c_1s_2o_{c2} \\ -s_1s_2o_{c2} \\ c_2o_{c2} \end{pmatrix}$$

Obteniendo:

$$\mathbf{J}_{\mathrm{vc2}} = \begin{bmatrix} -o_{c2}c_2s_1 & -c_1s_2o_{c2} & 0 \\ o_{c2}c_2c_1 & -s_1s_2o_{c2} & 0 \\ 0 & c_2o_{c2} & 0 \end{bmatrix}$$

$$\mathbf{J}_{\omega \mathrm{c2}} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & s_1 & 0 \\ 0 & -c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Para determinar $\mathbf{J}_{vc3}$ y $\mathbf{J}_{\omega c3}$ desarrollaremos:

$$\mathbf{J}_{\text{vc3}} = \begin{bmatrix} \mathbf{z}_0 \times ({}^0\mathbf{o}_{c3} - {}^0\,\mathbf{o}_0) & \mathbf{z}_1 \times ({}^0\mathbf{o}_{c3} - {}^0\,\mathbf{o}_1) & {}^0\mathbf{z}_2 \times ({}^0\mathbf{o}_{c3} - {}^0\,\mathbf{o}_2) \end{bmatrix}$$

Donde :

$$\mathbf{z}_0 \times ({}^0\mathbf{o}_{c3} - {}^0\,\mathbf{o}_0) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} (o_{c3}c_3 + a_2c_2)c_1 \\ (o_{c3}c_3 + a_2c_2)s_1 \\ o_{c3}s_3 + a_2s_2 + d_1 \end{pmatrix} = \begin{pmatrix} -(o_{c3}c_3 + a_2c_2)s_1 \\ (o_{c3}c_3 + a_2c_2)c_1 \\ 0 \end{pmatrix}$$

$$\mathbf{z}_1 \times ({}^0\mathbf{o}_{c3} - {}^0\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} (o_{c3}c_3 + a_2c_2)c_1 & - & 0 \\ (o_{c3}c_3 + a_2c_2)s_1 & - & 0 \\ o_{c3}s_3 + a_2s_2 + d_1 & & d_1 \end{pmatrix} = \begin{pmatrix} -c_1(o_{c3}s_3 + a_2s2_) \\ -s_1(o_{c3}s_3 + a_2s2_) \\ o_{c3}c_3 + a_2c_2 \end{pmatrix}$$

$$\mathbf{z}_2 \times ({}^0\mathbf{o}_{c3} - {}^0\mathbf{o}_1) = \begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} c_1(o_{c3}c_3 + a_2c_2) & - & a_2c_2c_1 \\ s_1(o_{c3}c_3 + a_2c_2) & - & a_2c_2s_1 \\ o_{c3}s_3 + a_2s_2 + d_1 & & a_2s_2d_1 \end{pmatrix} = \begin{pmatrix} -c_1s_3o_{c3} \\ -s_1s_3o_{c3} \\ c_3o_{c3} \end{pmatrix}$$

Obteniendo finalmente:

$$\mathbf{J}_{\text{vc3}} = \begin{bmatrix} -s_1(o_{c3}c_3 + a_2c_2) & -c_1(o_{c3}s_3 + a_2s2_) & -c_1s_3o_{c3} \\ c_1(o_{c3}c_3 + a_2c_2) & -s_1(o_{c3}s_3 + a_2s2_) & -s_1s_3o_{c3} \\ 0 & o_{c3}c_3 + a_2c_2 & c_3o_{c3} \end{bmatrix}$$

$$\mathbf{J}_{\omega c3} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix}$$

El conocimiento de estas velocidades en términos de las coordenadas generalizadas, perimite encontrar la energía cinética de cada eslabón.

3.2. Energía cinética del robot *Omni*

Se le llama energía cinética a la energía que posee un cuerpo por el hecho de moverse. La energía cinética de un cuerpo depende únicamente de la masa y de su velocidad. La energía es una función cuadrática de la velocidad articular y matricialmente es una función cuadrática la podemos escribir como:

$$\mathbf{K} = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{H}\dot{\mathbf{q}} = \frac{1}{2}\sum_{i,j} d_{i,j}(\mathbf{q})\dot{\mathbf{q}}_i\dot{\mathbf{q}}_j \tag{3.5}$$

La forma general para la energía cinética para cualquier cuerpo es:

$$\mathbf{K} = \sum_{i=1} \frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i \tag{3.6}$$

Sin embargo, para un cuerpo rígido en movimiento se debe calcular la energía cinética debido al movimiento lineal y rotacional. Para calcular la energía cinética debido al movimiento lineal se puede suponer que la masa total del cuerpo está contenida en el centro de masa del mismo, mientras que para la debida al movimiento rotacional se debe de considerar la geometría y la distribución de masa de cada eslabón.

La energía cinética total tiene componentes debido a un movimiento lineal y rotacional:

$$\mathbf{K} = \frac{1}{2} m \mathbf{v}^T \mathbf{v} + \frac{1}{2} \mathbf{w}^T \mathbf{I} \mathbf{w} \tag{3.7}$$

Para n grados de libertad de un manipulador tenemos:

$$\mathbf{K} = \sum_{i=1}^{n} \frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i + \frac{1}{2} \mathbf{w}_i^T \mathbf{I}_i \mathbf{w}_i \tag{3.8}$$

donde $\mathtt{I}_i$ es el tensor de inercía referenciado al sistema base, pero mediante una transformación de similitud se puede quitar la dependencia del tensor de inercia con respecto a la posición. Por lo tanto se logra hacer $\mathtt{I}_i$ constante, y bajo la suposición de que la densidad de cada eslabón es uniforme en cualquier parte, I_i es el momento principal de inercia, por lo tanto los productos cruzados de inercia son cero.

Transformación de similitud:

$$\mathtt{I}_i = {}^0\mathbf{R}_i \mathbf{I}_i^0 \mathbf{R}_i^T$$

Por lo que la energía cinética será:

$$\mathbf{K} = \sum_{i=1}^{n} \frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i + \frac{1}{2} \mathbf{w}_i^T \, {}^0\mathbf{R}_i^0 \mathbf{I}_i^0 \mathbf{R}_i^T \, \mathbf{w}_i \tag{3.9}$$

Para realizar nuestro análisis necesitamos agregar las coordenadas generalizadas, esto se puede lograr relacionando los Jacobicanos como:

$$\mathbf{v}_i = \mathbf{J}_{vi}(q) \dot{\mathbf{q}} \tag{3.10}$$

$$\mathbf{w}_i = \mathbf{J}_{wi}(q) \dot{\mathbf{q}} \tag{3.11}$$

Sustituyendo (3.10)y (3.11) en (3.9) se logra obtener finalmente una ecuación de la energía cinética $\mathbf{K}$ en función de nuestras coordenadas generalizadas.

$$\mathbf{K} = \sum_{i=1}^{n} \frac{1}{2} m_i \mathbf{J}_{vi}(q) \dot{\mathbf{q}}^T \mathbf{J}_{vi}(q) \dot{\mathbf{q}} + \frac{1}{2} \mathbf{w}_i^T \mathbf{R}_i^0 \mathbf{I}_i \mathbf{R}_i^{0T} \mathbf{w}_i \tag{3.12}$$

La matriz $\mathbf{H}(\mathbf{q})$ la podemos extraer de (3.12), y queda determinada como:

$$\mathbf{H}(\mathbf{q})(\mathbf{q}) = \sum_{i=1}^{n} m_i \mathbf{J} v_{ci}^T \mathbf{J} v_{ci} + \mathbf{J} w_{ci} \mathbf{R}_i^0 \mathbf{I}_i \mathbf{R}_i^{0T} \mathbf{J} \omega_{ci} \tag{3.13}$$

Donde:
m_i es la masa de cada eslabón
$\mathbf{Jv_{ci}}$ es el Jacobiano de velocidad lineal del eslabón i
$\mathbf{J}\omega_{\mathbf{ci}}$ es el Jacobiano de velocidad angular del eslabón i
${}^0\mathbf{R_i}$ Matrices de rotación del sistema coordenado i respecto al sistema base
$\mathbf{I}_i$ Momento principal de inercia

A partir de este momento se hará un agrupamiento de términos repetidos en parámetros (de θ_1 a θ_8). Para simplificar un poco las matrices, más adelante se utilizarán estos mismos parámetros, por el momento se definen:

$$\theta_1 = m_3 o_{c3} + I_3 \tag{3.14}$$

$$\theta_2 = a_2 m_3 o_{c3} \tag{3.15}$$

$$\theta_3 = m_2 o_{c2} + a_2^2 m_3 + I_2 \tag{3.16}$$

A partir del desarrollo de (3.13) para cada variable articular, se obtiene la matriz de inercia como:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.17}$$

Cada elemento de H es:
$h_{11} = \theta_1 c_3^2 + 2\theta_2 c_2 c_3 + \theta_3 c_2^2$
$h_{12} = 0$
$h_{13} = 0$
$h_{21} = 0$
$h_{22} = 2\theta_2 s_2 s_3 + 2\theta_2 c_2 c_3 + \theta_1 + \theta_3$
$h_{23} = \theta_2 s_2 s_3 + \theta_2 c_2 c_3 + \theta_1$
$h_{31} = 0$
$h_{32} = \theta_2 s_2 s_3 + \theta_2 c_2 c_3 + \theta_1$
$h_{33} = \theta_1$

Una característica muy importante es que la matriz $\mathbf{H}(\mathbf{q})$ es simétrica y positiva definida.

3.3. Energía potencial del robot *Omni*

La energía potencial es la energía almacenada por un cuerpo dada su posición en un instantes. Para conocer el vector $\mathbf{g(q)}$, se obtiene primero la energía

potencial de cada eslabón del manipulador.

Energía Potencial para n eslabones

$$\mathbf{P} = \sum_{i=1}^{n} m_1 \mathbf{g}^T \mathbf{r}_{ci} \tag{3.18}$$

donde m_i es la masa de cada eslabón, $\mathbf{g}$ es un vector con la constantes gravitacional ($9.79\frac{m}{s^2}$) en su componente en z_0, referido al sistema base y r_{ci} es el vector posición para cada instante de cada centro de masa respecto al sistema base. Los vectores de cada centro de masa son:

$$\boldsymbol{r}_{c1} = \begin{bmatrix} 0 \\ 0 \\ o_{c1} \end{bmatrix} r_{c2} = \begin{bmatrix} o_{c2}c_2s_1 \\ o_{c2}c_2c_1 \\ o_{c2}s_2 + d_1 \end{bmatrix} \boldsymbol{r}_{c3} = \begin{bmatrix} (a_2c_2 + O_{c3}c3)c_1 \\ (a_2c_2 + O_{c3}c3)s_1 \\ a_2s_2 + d_1 + O_{c3}s_3 \end{bmatrix} g^T = \begin{bmatrix} 0 \\ 0 \\ 9.79 \end{bmatrix}$$

Hay que mencionar que el vector $\mathbf{g}$ contiene un signo negativo en z, este signo representa el sentido de crecimiento de la energía potencial, es decir dado el sistema coordenado base, la energía potencial crece en sentido opuesto a la fuerza de atracción gravitacional. Obteniendo la derivada parcial de la energía potencial respecto a cada una de las variables articulares:

$$\mathbf{g(q)} = \begin{bmatrix} \frac{\partial \mathbf{P}}{\partial \mathbf{q_1}} \\ \frac{\partial \mathbf{P}}{\partial \mathbf{q_2}} \\ \frac{\partial \mathbf{P}}{\partial \mathbf{q_3}} \end{bmatrix} = \begin{bmatrix} 0 \\ \theta_7 \mathbf{gc_2} \\ \theta_8 \mathbf{gc_3} \end{bmatrix} \tag{3.19}$$

Donde los parámetros θ_7 y θ_8 son:

$$\theta_7 = m_2 O_{c2} + a_2 m_3 \tag{3.20}$$

$$\theta_8 = m_3 O_{c3} \tag{3.21}$$

3.4. Disipación de energía

Para obtener un modelo más completo del manipulador deben tomarse en cuenta los elementos disipativos. En los sistemas mecánicos la disipación de energía se debe a la fricción. En el caso de los manipuladores es generada entre los eslabones. En la representación matricial para el modelo, los coeficientes serán los elementos de la diagonal principal de la matriz D

$$\mathbf{D} = \begin{bmatrix} \theta_4 & 0 & 0 \\ 0 & \theta_5 & 0 \\ 0 & 0 & \theta_6 \end{bmatrix} \tag{3.22}$$

donde los parámetros θ_4, θ_5 y θ_6 son:

$$\theta_4 = \mathbf{C_{f1}} \tag{3.23}$$

$$\theta_5 = \mathbf{C_{f2}} \tag{3.24}$$

$$\theta_6 = \mathbf{C_{f3}} \tag{3.25}$$

3.5. Coriolis y fuerzas centrifugas

El último elemento de la ecuación (3.1) que nos falta modelar es la fuerza de Coriolis. Esta fuerza recibe su nombre en honor al ingeniero y matemático francés Gaspard Gustav Coriolis (1835), quien propuso la existencia de una fuerza presente en todo sistema en rotación, la cual ejerce sobre cualquier objeto que se desplace sobre él una fuerza perpendicular a la dirección de su movimiento, torciendo su trayectoria. Para obtener las fuerza de Coriolis se utilizarán los símbolos de Christoffel del primer tipo:

$$\mathbf{c_{kij}} := \frac{\mathbf{1}}{\mathbf{2}} \left(\frac{\partial \mathbf{h_{ij}}}{\partial_{\mathbf{qk}}} + \frac{\partial \mathbf{h_{ik}}}{\partial_{\mathbf{qj}}} - \frac{\partial \mathbf{h_{kj}}}{\partial_{\mathbf{qi}}} \right) \tag{3.26}$$

La matriz $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ puede ser fácilmente calculada a partir de la matriz de inercia. La matriz de Coriolis está dada por:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{3.27}$$

donde cada elemento de $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ es:

$c_{11} = -s_2\dot{q}_2(\theta_3 c_2 + \theta_2 c_3) - \theta_1 s_2 c_3 \dot{q}_3 - \theta_2 s_3 c_2 \dot{q}_3$
$c_{12} = \theta_3 s_2 c_2 \dot{q}_1 - \theta_2 s_2 c_3 \dot{q}_1$
$c_{13} = -\theta_1 s_3 c_3 \dot{q}_1 - \theta_2 c_2 s_3 \dot{q}_1$
$c_{21} = \theta_3 s_2 c_2 \dot{q}_1 + \theta_2 s_2 c_3 \dot{q}_1$
$c_{22} = \theta_2(c_2 s_3 - s_2 c_3)(\dot{q}_2 - \dot{q}_3)$
$c_{23} = -\theta_2(c_2 s_3 - s_2 c_3)(\dot{q}_2 + \dot{q}_3)$
$c_{31} = \theta_1 s_3 c_3 \dot{q}_1 + \theta_2 c_2 s_3 \dot{q}_1$
$c_{32} = 2\theta_2(c_2 s_3 - s_2 c_3)\dot{q}_2$
$c_{33} = 0$

Hasta este momento se han determinado todas las variables involucradas en el modelo dinámico (3.1) del manipulador *Omni Phantom*, en forma matricial. De esta forma se puede aprovechar algunas propiedades importantes de las matrices. En el siguiente capítulo se utilizará la ecuación de Euler-Lagrange en su forma matricial para el desarrollo e implementación de un algoritmo adaptable propuesto por Slotine y Li en 1987 [5].

Capítulo 4

Control PD y PID

Desde 1784 cuando James Watt logró controlar la velocidad de una máquina de vapor con su gobernador centrífugo, a pesar de que no lograba explicar matemáticamente lo que estaba logrando y lo trascendental que resultaría a lo largo de la historia, el control automático se hizo presente. Sin embargo no fue hasta la década de 1930, que surge un esquema de control PID desarrollado para controlar la posición del timón de lo buques.

Los métodos de control han variado muchísimo hasta el día de hoy. Sin embargo, en la industria los controladores PD y PID son los que suelen ocuparse más, dado su gran desempeño y su facilidad de implementación. La elección del controlador a utilizar depende de la aplicación. En la robótica se han desarrollado esquemas de control para realizar tareas con mayor velocidad y alta precisión. A continuación se le aplicará al manipulador teorías de control que ignoren su dinámica.

4.1. Diseño de trayectorias

Para implementar esquemas de control en el manipulador *Phantom Omni*, es necesario determinar las trayectorias de referencia cuyo seguimiento será el objetivo de control. Idealmente, el seguimiento exacto se logra cuando el error entre las trayectorias de referencia y las reales es cero. En la práctica esto no suele pasar y lo que se busca es hacer este error tan cercano a cero como sea posible.

4.1.1. Segmento de recta

Como primera trayectoria a diseñar se utilizará un segmento de recta, para lo que se empleará un polinomio de quinto orden para realizar hacer una trayectoria suave. El grado se elige tomando en cuenta el número de restricciones que se deseen, en este caso se proponen 6 restricciones: la posición inicial, la

posición final, la velocidad inicial, la velocidad final, la aceleración inicial y la acelaración final del efector.

Las restricciones resultantes son:

$$\mathbf{p}_\mathrm{f} = \begin{bmatrix} q_1(t_f) \\ q_2(t_f) \\ q_3(t_f) \end{bmatrix} \quad \mathbf{P}_0 = \begin{bmatrix} q_1(t_0) \\ q_2(t_0) \\ q_3(t_0) \end{bmatrix}$$

$$\dot{\mathbf{p}}_\mathrm{f} = \begin{bmatrix} \dot{q}_1(t_f) \\ \dot{q}_2(t_f) \\ \dot{q}_3(t_f) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \dot{\mathbf{p}}_0 = \begin{bmatrix} \dot{q}_1(t_0) \\ \dot{q}_2(t_0) \\ \dot{q}_3(t_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\ddot{\mathbf{p}}_\mathrm{f} = \begin{bmatrix} \ddot{q}_1(t_f) \\ \ddot{q}_2(t_f) \\ \ddot{q}_3(t_f) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \ddot{\mathbf{p}}_0 = \begin{bmatrix} \ddot{q}_1(t_0) \\ \ddot{q}_2(t_0) \\ \ddot{q}_3(t_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Por lo que se tiene el siguiente polinomio:

$$P(t) = a_0 t + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{4.1}$$

Obteniendo primera y segunda derivada de (4.1) para velocidad y aceleración

$$\dot{P}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \tag{4.2}$$

$$\ddot{P}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \tag{4.3}$$

Nótese que cuando el tiempo es cero en (4.2) y (4.3), se pueden obtener los coeficientes a_1 y a_2;

$$a_1 = a_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

El problema se reduce a es encontrar los coeficientes del polinomio que nos permitan llegar desde P_i, hasta P_f, punto a punto, es decir:

$$qt_f - qt_0 = a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5$$

Por lo que se puede escribir la ecuación anterior y sus derivadas como:

$$\begin{gathered} q_f - q_0 = a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ 0 = 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ 0 = 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{gathered}$$

Cabe mencionar que se necesitan a_3, a_4, a_5, pero en forma matricial se tiene

$$\mathbf{Ax} = \mathbf{b};$$

$$\begin{bmatrix} t_f^3 + t_f^4 + t_f^5 \\ 3t_f^2 + 4t_f^3 + 5t_f^4 \\ 6t_f + 12t_f^2 + 20t_f^3 \end{bmatrix} \begin{bmatrix} a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_f - q_0 \\ 0 \\ 0 \end{bmatrix}$$

El primer paso es obtener el determinante de A, *i.e.* :
Para encontrar los coeficientes, se utilizá el método de Cramer:

$$det(\mathbf{A}) = \mathbf{2t_f^4}$$

Ahora se sustituye el vector columna $\mathbf{b}$ en cualquier vector columna de A, se hace para la primer columna de A, posteriormente se determina el determinante de $\mathbf{A}$, y es dividido entre $det(\mathbf{A})$, de esta manera se obtiene a_3.

$$a_3 = \frac{Det\begin{pmatrix} q_f - q_0 & t_f^4 & t_f^5 \\ 0 & 4t_f^3 & 5t_f^4 \\ 0 & 12t_f^2 & 20t_f^3 \end{pmatrix}}{Det(A)} = \frac{10(P_f - P_0)}{t_f^3}$$

$$a_4 = \frac{Det\begin{pmatrix} t_f^3 & q_f - q_0 & t_f^5 \\ 3t_f^2 & 0 & 5t_f^4 \\ 6t_f & 0 & 20t_f^3 \end{pmatrix}}{Det(A)} = \frac{-15(P_f - P_0)}{t_f^4}$$

$$a_5 = \frac{Det\begin{pmatrix} t_f^3 & t_f^4 & q_f - q_0 \\ 3t_f^2 & 4t_f^3 & 0 \\ 6t_f & 12t_f^2 & 0 \end{pmatrix}}{Det(A)} = \frac{6(P_f - P_0)}{t_f^5}$$

Sustituyendo los coeficientes en la ecuación del polinomio, se obtiene:

$$P(t) = P_0 + \dot{P}_0 t + \ddot{p}_0 t^2 + \frac{10(P_f - P_0)}{t_f^3} t^3 - \frac{15(P_f - P_0)}{t_f^4} t^4 + \frac{6(P_f - P_0)}{t_f^5} t^5 \quad (4.4)$$

4.1.2. Circunferencia

Otra trayectoria que se utilizará para evaluar los controladores es una trayectoria circular. Este tipo de trayectorias utiliza funciones acotadas y cíclicas lo que evita que haya colisiones entre articulaciones; ademas esta trayectoria logra hacer que los actuadores sean un poco más exigidos y es un poco más fácil de programar. Para diseñar una trayectoria circular sobre un plano basta con programar las ecuaciones paramétricas de una circunferencia, la trayectoria será

sobre el plano X-Y, por lo que Z es constante:

Ecuaciones paramétricas de la circunferencia:

$$x = 65 + r\cos(\omega t)[\text{mm}] \tag{4.5}$$

$$Y = r\cos(\omega t)[\text{mm}] \tag{4.6}$$

$$Z = 50[\text{mm}] \tag{4.7}$$

donde r, es el radio de la circunferencia, y z=50 [mm] es la distancia desde el plano que forman los vectores x_0 y y_0 al efector final. Estas ecuaciones son llamadas parámetricas ya que ambas dependen de un solo parámetro t, ω es la frecuencia y es libre de fijar esta frecuencia como una constante, un poco más adelante se utilizará una frecuencia de 0.2 [rad/seg] para los experimentos. También se sabe que $wt = 2\pi$, por lo que sólo falta determinar en cuanto tiempo se realizará un ciclo o una circunferencia a esa frecuencia. A diferencia del segmento de recta si el tiempo se excede del tiempo fijo, en este tipo de trayectoria se repetirá la circunferencia.

4.2. Control PD

El control Proporcional Derivativo, o simplemente PD, tiene como objetivo incrementar la estabilidad del sistemas mejorando su respuesta. Sus principales ventajas son reducir el sobre impulso y el tiempo de estabilización. La ley de control esta dada por:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}$$

donde K_p es la ganancia proporcional. Un controlador proporcional por sí mismo puede controlar cualquier planta estable. Sintonizando esta ganancia se puede reducir el tiempo de subida, pero tiene la desventaja de que no puede eliminar el error en estado estacionario.

T_d es una constante de tiempo derivativo y tiene como objetivo anticiparse al crecimiento del error. Ajustando la ganancia derivativa tendrá el efecto de aumentar la estabilidad del sistema, reduciendo el sobre impulso, la desventaja es que amplifica señales de ruido, por lo que podría dañar a los actuadores. $e(t)$ *y* $\dot{e}(t)$ es el error y su derivada respectivamente.

Es importante mencionar que el error articular lo definimos como el valor deseado menos el real, es decir:

$$\widetilde{q} = q_\text{d} - q$$

En las siguientes gráficas se muestra el comportamiento de este control, siguiendo la trayectoria circular descrita por las ecuaciones (4.5), (4.6) y (6.7).

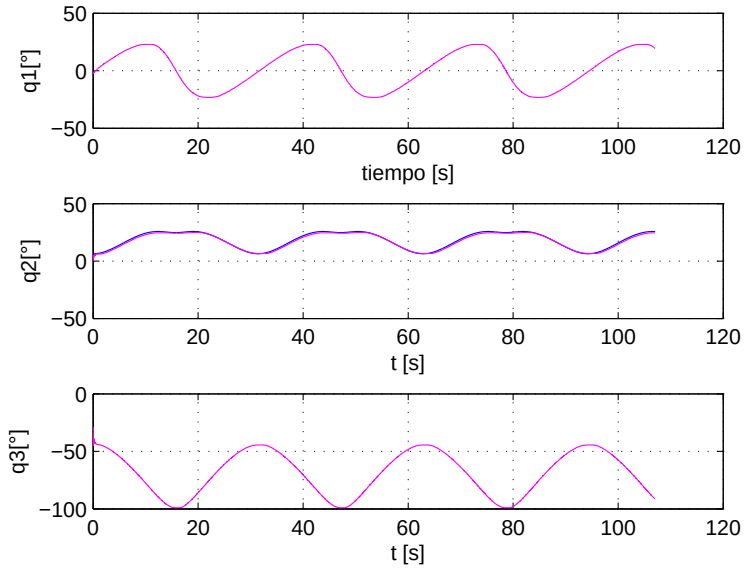

Figura 4.1: Coordenadas articulares reales y deseadas

En la Figura 4.1 se muestra el comportamiento real de cada articulación y la señal deseada correspondiente; nótese que ambas señales son parecidas, es decir, en todo momento con un control PD se logró hacer que las articulaciones siguieran la señal de referencia.

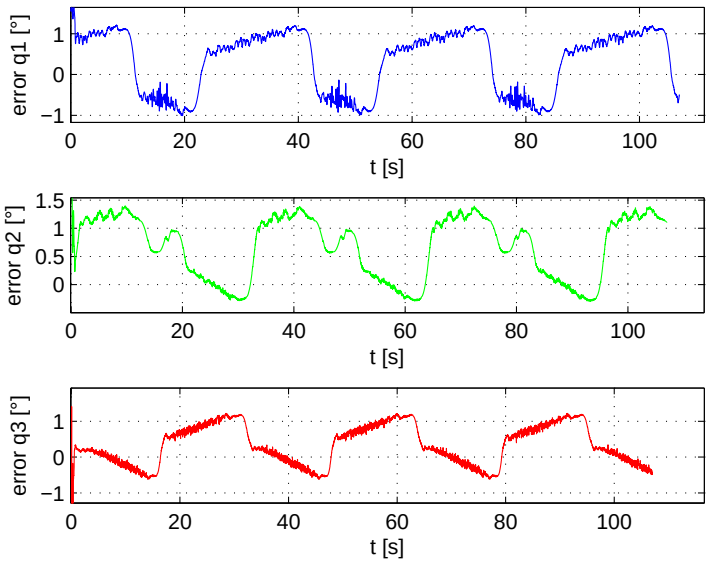

Figura 4.2: Error articular

En la Figura 4.2 se comprueba el seguimiento de las articulaciones con un error pequeño en cada articulación.

4.3. Control PID

Otro controlador muy utilizado es el PID. A diferencia del PD, el PID cuenta con una componente proporcional a la integral del error, cuya ganancia es K_i. Esta ganancia tendrá el efecto de eliminar el error en estado estacionario, aunque la desventaja es que podría afectar la respuesta transitoria. Hay que tener en cuenta que estas correlaciones pueden no ser exactamente precisas, ya que K_p, K_i y K_d tienen cierta dependencia, es decir, el cambio de una de estas variables puede cambiar el efecto de las otras dos. La estructura de un control PID es la siguiente:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} + \frac{K_p}{T_i} \int e(t)\, dt$$

Al igual que con el control PD, se realizaron experimentos similares, se implementó este controlador en el manipulador y se graficaron las posiciones articulares las posiciones articulares reales y deseadas. El tiempo del experimento

fue un poco mayor que para el PD. La trayectoria usada fue la misma que el PD, es decir, se usaron las ecuaciones (4.5), (4.6) y (4.7).

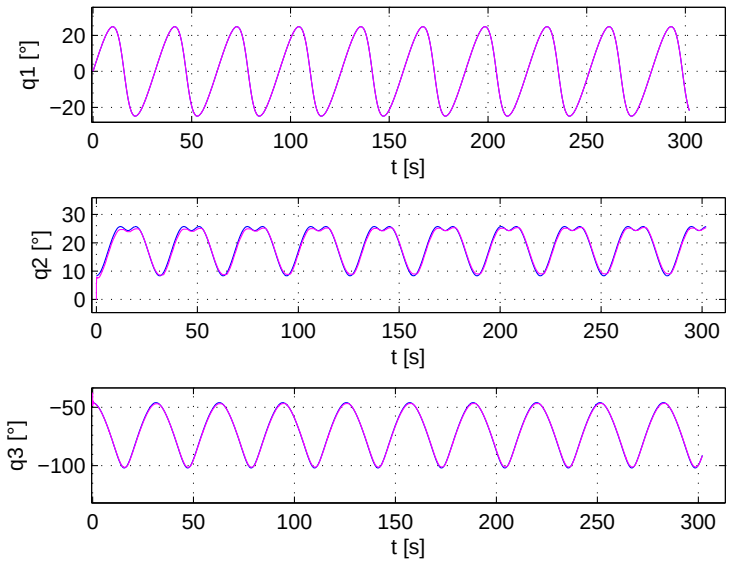

Figura 4.3: Coordenadas articulares reales y deseadas con controlador PID

En la Figura 4.3 se muestra el comportamiento de cada coordenada articular real y deseada. Nótese que ambas señales son parecidas, es decir, en todo momento con un control PID se logró que las articulaciones siguieran la señal de referencia.

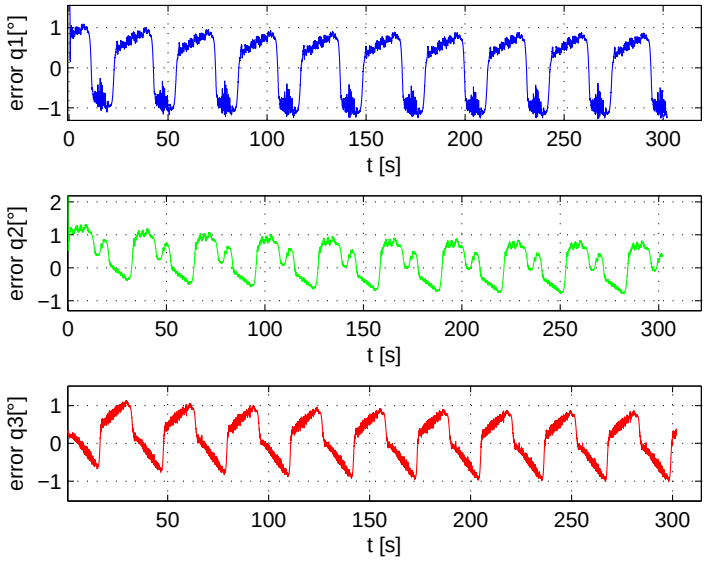

Figura 4.4: Error articular con el controlador PID

En la Figura 4.4 se comprueba el seguimiento de las articulaciones con un error pequeño en cada articulación. A diferencia del PD la duración del experimento fue más larga con el objetivo de resaltar la acción de la ganancia integral. Se observa que el error al final del experimento se redujo un poco más.

Otra trayectoria que ya se había mencionado es el segmento de recta que fue implementada, a partir de la (4.4). Se diseñó la trayectoria con las velocidades iniciales y finales iguales a cero, de igual forma para las aceleraciones, pero para las posiciones se tomaron las siguientes consideraciones:

Restriccin	*Punto inicial*	*Punto final*
X	84[mm]	160[mm]
Y	-111[mm]	0[mm]
Z	8.44[mm]	100[mm]

Tabla 4.1 Puntos iniciales y finales en el espacio cartesiano

Para poder mostrar la trayectoria que siguió el efector final en el espacio cartesiano se muestra en la siguiente gráfica 3D

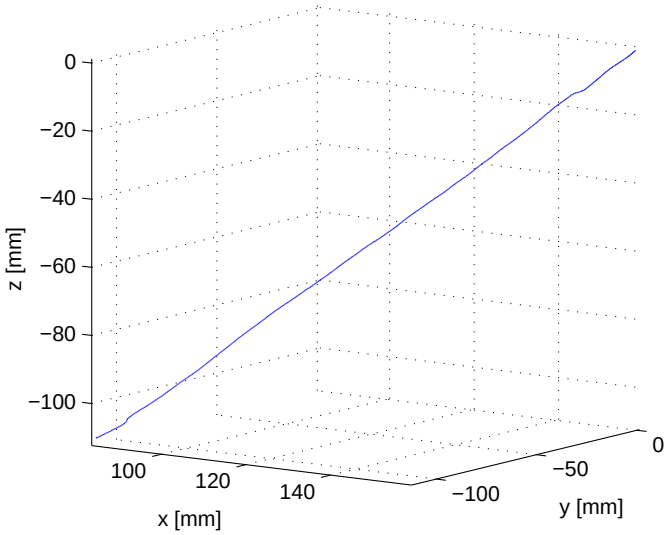

Figura 4.5: Recta en el espacio cartesiano generada con un polinomio

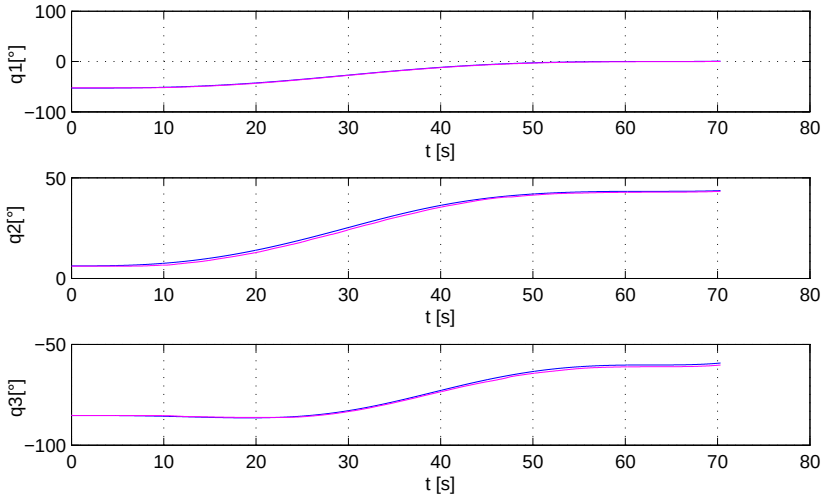

Figura 4.6: Variables articulares deseadas y reales

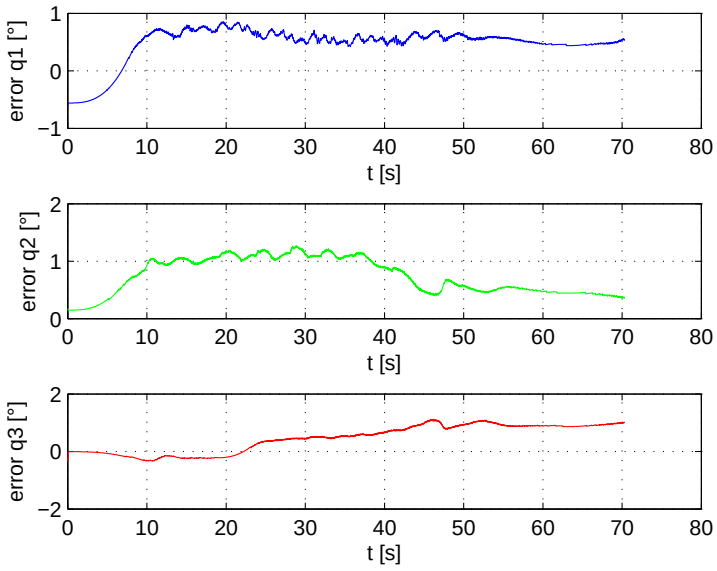

Figura 4.7: Error articular

En la Figura 4.5 se muestra la trayectoria generada con el polinomio de quinto orden de la ecuación (4.4). El tiempo de duración de la trayectoria fue diseñado para realizarse en 70 segundos. En la Figura 4.6 la trayectoria generada con el polinomio. Las variables articulares siguen trayectorias suaves, es decir, para seguir la referencia, las variables articulares reales sufren cambios muy pequeños. En la Figura 4.7 se muestra el error y siempre es pequeño, menor a un grado.

Se realizó otro experimento utilizando el controlador este PID con una trayectoria un poco más compleja. La señal consistió en el producto de señales cosenoidales, para obtener una trayectoria formada por varias lemniscatas. Las ecuaciones paramétricas que describen este comportamiento son las siguientes:

$$x = 65 + r\cos(\omega t)\cos(\omega_2 t)[\text{mm}] \tag{4.8}$$

$$Y = r\cos(\omega t)\cos(\omega_2 t)[\text{mm}] \tag{4.9}$$

$$Z = 50[\text{mm}] \tag{4.10}$$

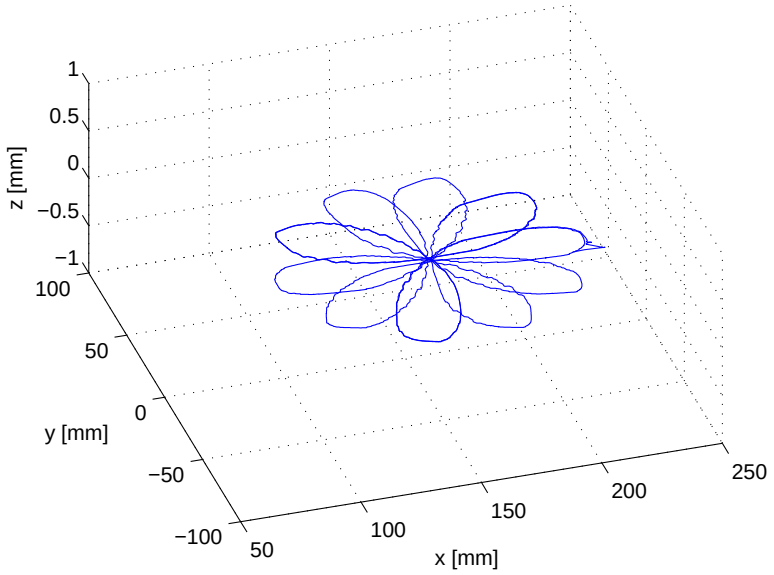

Figura 4.8: Flor en el espacio cartesiano

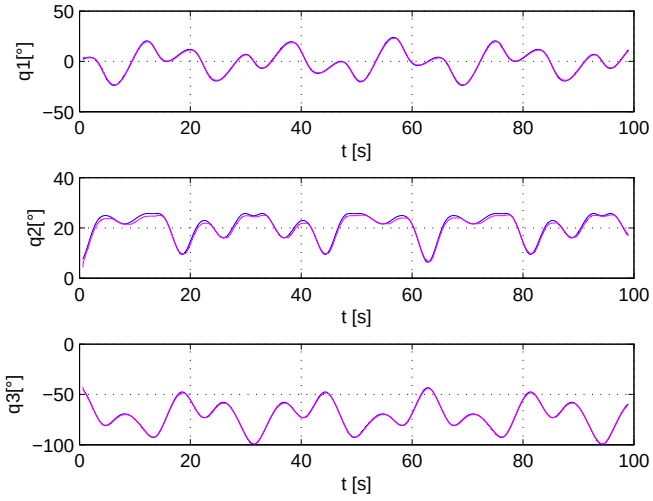

Figura 4.9: Variables articulares deseadas y reales

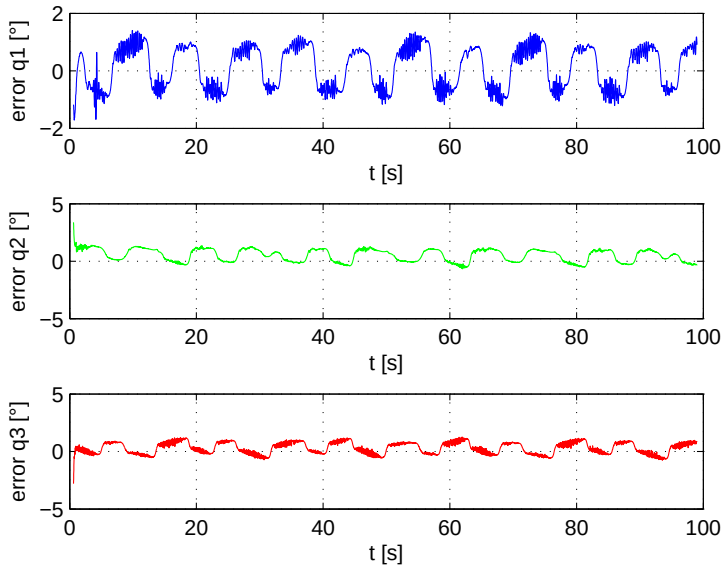

Figura 4.10: Error articular

Como se observa en las figuras anteriores el seguimiento de trayectorias un poco más complejas o con mayor velocidad con un control PID sigue siendo relativamente bueno. Las articulaciones siguieron en todo momento la señal de referencia como se muestra en la Figura 4.9. Sin embargo, es importante mencionar que el error mostrado en la Figura 4.10 creció un poco. A primera vista se puede concluir que la precisión del control PID decae con trayectorias más complejas o con mayor velocidad. El controlador PID no toma en cuenta los efectos dinámicos no lineales del robot. Por lo tanto, es de esperarse que el desempeño no sea el mejor posible para todas las trayectorias deseadas. Introducir el conocimiento de un modelo dinámico del manipulador en la ley de control podría complicar el diseño y la implementación. Sin embargo, al considerar estos efectos se debería obtener un mejor seguimiento.

Capítulo 5

Estimación de parámetros

El modelo (3.1) toma en cuenta todas las fuerzas que interactúan con el manipulador. Sin embargo, los parámetros que fueron introducidos, son difíciles de conocer con exactitud. Además, los datos mecánicos del robot no están disponibles, lo que dificulta más la obtención de dichos parámetros. Una solución a este problema es utilizar técnicas de estimación de parámetros basadas en datos de entrada y salida, como pueden ser voltajes de entrada a los motores y lectura de posición articular.

5.1. Linealidad de los parámetros

En el Capítulo 3 se escribió el modelo del robot en forma matricial. Pero se hizó una simplificación en los elementos de cada matriz, estos nuevas términos quedaron en función de las masas de los eslabones, momentos de inercía, coeficientes de fricción y de la gravedad. Estos parámetros también pueden expresarse como el producto de una función Y y un vector columna Θ.

La función Y es conocida como *regresor* y está en función únicamente de la posición, velocidad y las aceleraciones. Es importante mencionar que queda en función de variables que pueden ser determinadas por sensores de posición, encoders en nuestro caso. El vector columna $\mathbf{\Theta}$ contiene cada uno de los parámetros obtenidos anteriormente, tres parámetros son debidos a la matriz de inercia y fuerzas centrífugas, tres más son de los coeficientes de fricción, y dos más del vector de gravedad.

La ecuación de Euler-Lagranse (3.1) se puede representar como:

$$\mathbf{H(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = Y(q,\dot{q},\ddot{q})\Theta} = \tau \tag{5.1}$$

Dado que se tienen 8 parámetros y 3 grados de libertad, $\mathbf{Y} \in \mathbb{R}^{\mathbf{3\times 8}}$ y $\mathbf{\Theta} \in \mathbb{R}^{\mathbf{8}}$. El regresor puede ser expresado como:

$$\mathbf{Y^{3\times 8}} = \begin{bmatrix} \mathbf{Y_{HC}^{3\times 3}} & Y_D^{3\times 3} & Y_G^{3\times 2} \end{bmatrix} \tag{5.2}$$

donde la parte del regresor debida a las matrices $\mathbf{H}$ y $\mathbf{C}$ es la suma de $\mathbf{Y_H}$ y $\mathbf{Y_C}$ y se denominará $\mathbf{Y}_{HC}$, $\mathbf{Y}_D$ es la parte del regresor de los coeficientes de fricción, y la debida a la gravedad es $\mathbf{Y_G}$. Para los términos relacionados con la matriz de inercia se tiene:

$$\mathbf{Y_H} = \begin{bmatrix} \ddot{q}_1 cq_3^2 & 2\ddot{q}_1 cq_2 cq_3 & \ddot{q}_1 cq_2^2 \\ \ddot{q}_2 + \ddot{q}_3 & \ddot{q}_2(2sq_2 sq_3 + 2cq_2 cq_3) + \ddot{q}_3(sq_2 sq_3 + cq_2 cq_3) & \ddot{q}_2 \\ \ddot{q}_2 + \ddot{q}_3 & \ddot{q}_2(sq_2 sq_3 + cq_2 cq_3) & 0 \end{bmatrix} \tag{5.3}$$

Para los términos de fuerzas centrífugas y de Coriolis se tiene:

$$\mathbf{Y_C} = \begin{bmatrix} -2s_3 c_3 \dot{q}_1 \dot{q}_3 & -2(s_2 c_3 \dot{q}_1 \dot{q}_2 + s_3 c_2 \dot{q}_1 \dot{q}_3) & -2s_2 c_2 \dot{q}_1 \dot{q}_2 \\ 0 & s_2 c_3 \dot{q}_1^2 + (c_2 s_3 - s_2 c_3)(-2\dot{q}_2 \dot{q}_3 + \dot{q}_2^2 - \dot{q}_3^2) & s_2 c_2 \dot{q}_1^2 \\ s_3 c_3 \dot{q}_1^2 & c_2 s_3 \dot{q}_1^2 + 2(c_2 s_3 - s_2 c_3)\dot{q}_2^2 & 0 \end{bmatrix} \tag{5.4}$$

Por último, los términos de fricción y de gravedad $\mathbf{Y_D}$ y $\mathbf{Y_g}$, son:

$$\mathbf{Y_D} = \begin{bmatrix} \dot{q}_1 & 0 & \\ 0 & \dot{q}_2 & 0 \\ 0 & 0 & \dot{q}_3 \end{bmatrix} \tag{5.5}$$

$$\mathbf{Y_g} = \begin{bmatrix} 0 & 0 \\ gC_2 & 0 \\ 0 & gC_3 \end{bmatrix} \tag{5.6}$$

El vector de parámetros Θ es:

$$\mathbf{\Theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \end{bmatrix} = \begin{bmatrix} m_3 O_{c3} + I_3 \\ a_2 m_3 O_{c3} \\ m_2 O_{c2} + a_2^2 m_3 + I_2 \\ C_{f1} \\ C_{f2} \\ C_{f3} \\ m_2 O_{c2} + a_2 m_3 \\ m_3 O_{c3} \end{bmatrix} \tag{5.7}$$

Sustituyendo los elementos obtenidos en (5.3), (5.4), (5.5) y (5.6) en (5.1) se obtiene finalmente $\mathbf{Y\Theta}$.

5.2. Control adaptable

Para lograr el control del robot *Omni Phantom* las leyes de control PD y PID logran el seguimiento de trayectorias. Sin embargo, estas leyes de control no toman en cuenta los efectos dinámicos no lineales presentes en el movimiento del manipulador. Más aún, si el robot cambia alguno de sus parámetros (por ejemplo el cambio en la masa al levantar un objeto) el desempeño de los controladores mencionados se vería afectado. Una buena solución en este caso es utilizar un controlador adaptable. El control adaptable para el movimiento de actuadores tiene como principal objetivo mantener el desempeño en términos de la estabilidad y del seguimiento del error, compensando incertidumbres paramétricas.

A diferencia del control robusto, el control adaptable hace que el seguimiento de error tienda a decrementarse conforme los parámetros estimados se acercan a los reales, resulta mucho más factible usar control adaptable si la tarea a realizar por el manipulador es repetitiva.

Como se ha mencionado, el robot *Omni Phantom*, tiene ciertos parámetros en la dinámica que no pueden ser medidos fácilmente, en el enfoque de control adaptable. El vector de parámetros estimados $\widehat{\Theta}$ es una estimación aproximada del vector de parámetros reales Θ. Esta aproximación puede mejorarse utilizando señales de entrada y salida del sistema.

Para realizar el control de trayectorias y la estimación paramétrica , se utilizará un algoritmo adaptable propuesto por Slotine y Li (1987). La ley de control de este algoritmo es:

$$\tau = \widehat{\mathbf{H}}(\mathbf{q})\dot{\mathbf{r}} + \widehat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{r} + \widehat{\mathbf{D}}\mathbf{r} + \widehat{\mathbf{g}}(\mathbf{q}) - \mathbf{K_v}\mathbf{s} \tag{5.8}$$

Se observa que la dinámica esta en función de los valores estimados (Notación utilizada$\widehat{()}$), y de nuevas variables s y r. Dado que (5.8) es lineal con respecto a los parámetros estimados, puede ser expresado como el producto de un regresor Y por un vector de parámetros estimados, por lo que la ley de control se convierte en:

$$\tau = \mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_\mathbf{r},\ddot{\mathbf{q}}_\mathbf{r})\widehat{\mathbf{\Theta}} - \mathbf{K_v}\mathbf{s} \tag{5.9}$$

donde se define $\dot{\mathbf{q}}_r$, $\ddot{\mathbf{q}}_r$ y $\dot{\mathbf{s}}$ en función de los errores como sigue:

$$\dot{\mathbf{q}}_r = \dot{\mathbf{q}} - \mathbf{\Delta}\tilde{\mathbf{q}} \tag{5.10}$$

$$\ddot{\mathbf{q}}_r = \ddot{\mathbf{q}} - \mathbf{\Delta}\dot{\tilde{\mathbf{q}}} \tag{5.11}$$

$$\mathbf{s} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{r} = \dot{\tilde{\mathbf{q}}} + \mathbf{\Delta}\tilde{\mathbf{q}} \tag{5.12}$$

En la ley de control (5.9) se observa que tiene una parte PD y otra parte que será la compensación de parámetros en línea, tal vez la parte PD no se aprecia a simple vista, pero si se expresa de la siguiente manera, es más evidente ;

$$\tau = \mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_\mathbf{r},\ddot{\mathbf{q}}_\mathbf{r})\widehat{\mathbf{\Theta}} - \mathbf{K_v}\mathbf{s} = \mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_\mathbf{r},\ddot{\mathbf{q}}_\mathbf{r})\widehat{\mathbf{\Theta}} - \mathbf{K_v}\dot{\tilde{\mathbf{q}}} + \mathbf{K_v}\mathbf{\Delta}\tilde{\mathbf{q}} \tag{5.13}$$

De la ecuación (5.13) solo falta obtener $\widehat{\mathbf{\Theta}}$. Esto se puede lograr proponiendo una ley de adaptación, pero antes se explica como se involucra esta ley, el objetivo se convierte a encontrar una función que con el tiempo tenga una convergencia global a cero, para ello se propone una función que dependa de la energía, y del error de nuestros parámetros. Por lo tanto proponiendo la siguiente función candidata de Lyapunov.

$$\mathbf{V} = \frac{1}{2}\mathbf{s}\mathbf{H}\mathbf{s}^T + \frac{1}{2}\widetilde{\theta}^T\mathbf{\Gamma}^{-1}\widetilde{\theta} \tag{5.14}$$

Tomando en cuenta que $\widetilde{\theta} = \theta - \widehat{\theta}$ y que al derivar se obtiene que $\dot{\widetilde{\theta}} = -\dot{\widehat{\theta}}$, ya que el valor real de los parámetros es constante

Derivando V respecto al tiempo:

$$\dot{\mathbf{V}} = \mathbf{s^T}\mathbf{H}\dot{\mathbf{s}} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{s^T}\dot{\mathbf{H}}\mathbf{s} - \widetilde{\theta}^{\mathbf{T}}\mathbf{\Gamma}^{-\mathbf{T}}\dot{\widehat{\theta}} \tag{5.15}$$

Se sabe que del modelo dinámico

$$\mathbf{H}\dot{\mathbf{s}} = \mathbf{Y}\widetilde{\mathbf{\Theta}} - \mathbf{C}\mathbf{s} - \mathbf{D}\mathbf{s} \tag{5.16}$$

Sustituyendo en la función candidata de Lyapunov (5.16):

$$\mathbf{s}^T(\mathbf{Y}\widetilde{\theta} - \mathbf{C}\mathbf{s} - \mathbf{D}\mathbf{s} - \mathbf{K}_V\mathbf{s}) + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{H}}\mathbf{s} - \widetilde{\theta}^T\mathbf{\Gamma}^{-\mathbf{T}}\dot{\widehat{\mathbf{\Theta}}} \tag{5.17}$$

desarrollando se obtiene:

$$\dot{\mathbf{V}} = \mathbf{s}^T\mathbf{Y}\widetilde{\theta} - \mathbf{s}^T\mathbf{C}\mathbf{s} - \mathbf{s}^T\mathbf{D}\mathbf{s} - \mathbf{s}^T\mathbf{K}_V\mathbf{s} + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{H}}\mathbf{s} - \widetilde{\theta}^T\mathbf{\Gamma}^{-\mathbf{T}}\dot{\widehat{\theta}} \tag{5.18}$$

Nótese que la siguiente ecuación se simplifica haciendo cero un término, debido a una propiedad de las matrices antisimétricas en:

$$\dot{\mathbf{V}} = \mathbf{s}^T\mathbf{Y}\widetilde{\mathbf{\Theta}} + \mathbf{s}^T(\dot{\mathbf{H}} - 2\mathbf{C})\mathbf{s} - \mathbf{s}^T\mathbf{D}\mathbf{s} - \mathbf{s}^T\mathbf{K}_v\mathbf{s} - \widetilde{\mathbf{\Theta}}^T\mathbf{\Gamma}^{-\mathbf{T}}\dot{\widehat{\mathbf{\Theta}}} \tag{5.19}$$

obteniendo finalmente que:

$$\dot{\mathbf{V}} = -\mathbf{s}^T(\mathbf{D} + \mathbf{K}_v)\mathbf{s} + \mathbf{s}^T\mathbf{Y}\widetilde{bf\theta} - \widetilde{\theta}^T\mathbf{\Gamma}^{-\mathbf{1}}\dot{\widehat{\theta}} \tag{5.20}$$

El objetivo era obtener una función que fuese negativa semidefinida. El primer término siempre será negativo ya que $\mathbf{D}$ y $\mathbf{K_v}$ son matrices con elementos positivos (ganancias)en sus diagonales principales. No se puede hacer que los demás términos sean negativos en $\tilde{\theta}$ mediante la elección de la ley de adaptación. Sin embargo, se puede lograr que se haga cero, mediante la llamada ley de adaptación del tipo *gradiente*

$$\dot{\widehat{\Theta}} = -\mathbf{\Gamma}\mathbf{Y}^{\mathbf{T}}\mathbf{s} \tag{5.21}$$

Para llevar a cabo un experimento con el algoritmo adaptable es necesario definir el regresor en función de la nueva variable $\dot{\mathbf{q}}_r$ y $\ddot{\mathbf{q}}_r$, que a su vez está en

función de los errores, de (5.8) se puede parametrizar el término

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}_r + \mathbf{D}\dot{\mathbf{q}}_r + \mathbf{g}(\mathbf{q}) = \mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_r,\ddot{\mathbf{q}}_r)\boldsymbol{\Theta} = \tau \tag{5.22}$$

El cambio de variable no es tan sencillo de hacer, ya que la matriz de fuerzas centrifugas puede tener productos de $\dot{\mathbf{q}}$ y $\dot{\mathbf{q}}_r$.

El regresor modificado por parte de $\mathbf{H}$ es:

$$\mathbf{Y}_H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{5.23}$$

Cada elemento de $\mathbf{Y}_H$ se encuentra definido como:
$h_{11} = \ddot{q}_{r1}c_2^2$
$h_{12} = 2\ddot{q}_{r1}c_2c_3$
$h_{12} = \ddot{q}_{r1}c_2^2$
$h_{21} = \ddot{q}_{r2} + \ddot{q}_{r3}$
$h_{22} = \ddot{q}_{r2}(2sq_2s_3) + 2c_2c_3)) + \ddot{q}_{r3}(s_2s_3) + c_2c_3)$
$h_{23} = \ddot{q}_{r2}$
$h_{31} = \ddot{q}_{r2} + \ddot{q}_{r3}$
$h_{32} = \ddot{q}_{r2}(sq_2s_3 + c_2c_3)$
$h_{33} = 0$

El regresor modificado por parte de $\mathbf{Y}_C$ es:

$$\mathbf{Y}_C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{5.24}$$

$c_{11} = -s_3c_3(\dot{q}_3\dot{q}_{r1} + \dot{q}_1\dot{q}_{r3})$
$c_{12} = -c_2s_3(\dot{q}_3\dot{q}_{r1} + \dot{q}_1\dot{q}_{r3}) - s_2c_3(\dot{q}_2\dot{q}_{r1} + \dot{q}_1\dot{q}_{r2})$
$c_{13} = -s_2c_2\dot{q}_2\dot{q}_{r1}$
$c_{21} = 0$
$c_{22} = s_2c_3\dot{q}_1\dot{q}_{r1} + (c_2s_3 - s_2c_3)[(\dot{q}_2 - \dot{q}_3)\dot{q}_{r2} - (\dot{q}_2 + \dot{q}_3)\dot{q}_{r3}]$
$c_{23} = s_2c_2\dot{q}_1\dot{q}_{r1}$
$c_{31} = s_3c_3\dot{q}_1\dot{q}_{r1}$
$c_{32} = c_2s_3\dot{q}_1\dot{q}_{r1} + 2(c_2s_3 - s_2c_3)\dot{q}_2\dot{q}_{r2}$
$c_{33} = 0$

El regresor modificado por parte de $\mathbf{Y}_D$ y $\mathbf{Y}_g$ son respectivamente:

$$\mathbf{Y}_D = \begin{bmatrix} \dot{q}_{r1} & 0 & 0 \\ 0 & \dot{q}_{r2} & 0 \\ 0 & 0 & \dot{q}_{r3} \end{bmatrix} \tag{5.25}$$

$$\mathbf{Y}_G = \begin{bmatrix} 0 & 0 \\ gc_2 & 0 \\ 0 & gc_3 \end{bmatrix} \tag{5.26}$$

Sustutuyendo las ecuaciones (5.23), (5.24), (5.25) y (5.26) en (5.22), se obtiene:

$$\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_r \ddot{\mathbf{q}}_r)^{3\times 8} = \begin{bmatrix} \mathbf{Y}_{H+C}^{3\times 3} & \mathbf{Y}_{D}^{3\times 3} & \mathbf{Y}_{G}^{3\times 2} \end{bmatrix} \tag{5.27}$$

Con el el regresor modificado (en función del error) se puede poner a prueba el algoritmo de Slotine y Li [5] expuesto en (5.9). La idea es probar el algoritmo con la trayectoria circular descrita por las ecuaciones (4.5), (4.6) y (4.7) la cual fue puesta a prueba con un PD y comparar ambas señales de error, en teoría debe reducirla, al mismo tiempo que se obtiene una primera estimación paramétrica.

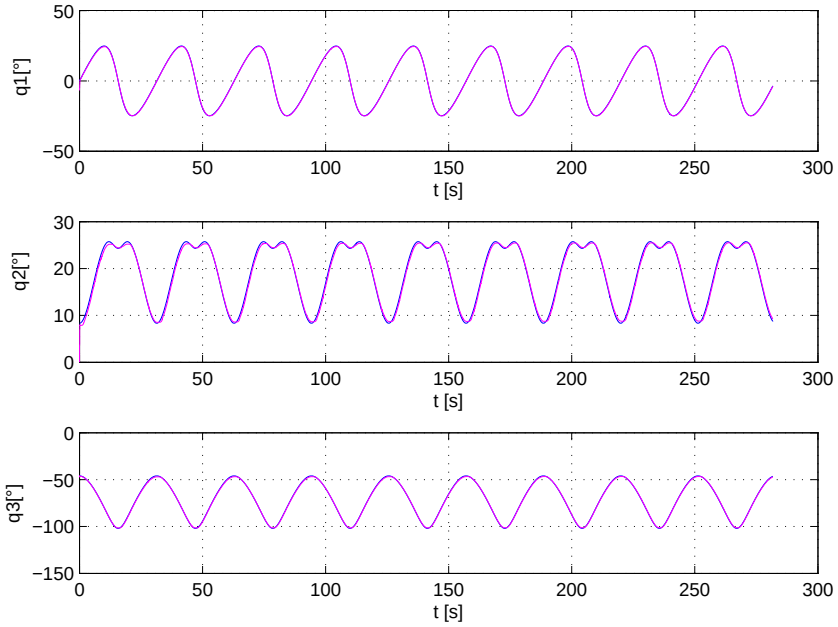

Figura 5.1: Variables articulares reales y deseadas control adaptable

En la gráfica anterior se observa que la trayectoria es seguida en todo momento, pero todavía falta analizar si el error disminuyó.

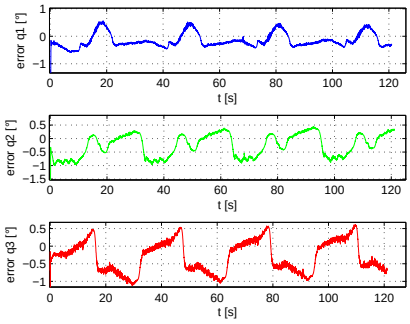

Figura 5.2: Error adaptable

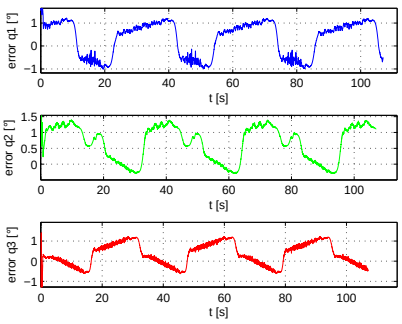

Figura 5.3: Error PD

En la Figura 5.2 se observa que la amplitud del error es menor que en la Figura 5.3; esto es, disminuyó el error cuando se agregó una parte del controlador en función de los parámetros, si la duración del experimento fuese más larga el error seguiría acercándose a cero como se muestra en la Figura 5.4.

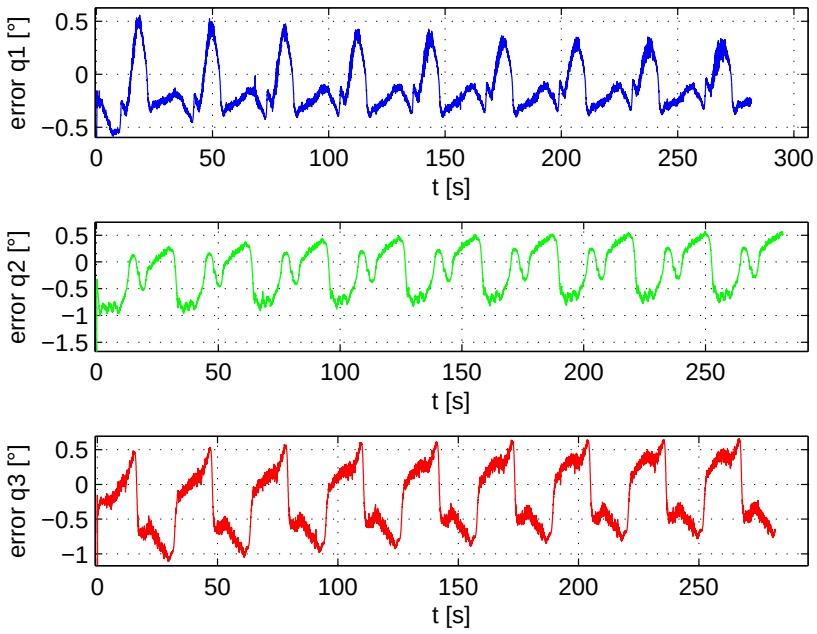

Figura 5.4: Error de control adaptable a 5 minutos

La estimación paramétrica obtenida se muestran en la tabla 5.2

Parámetro	Variables por parámetro	Experimento 1	Experimento 2
Θ_1	$m_3 o_{c3} + I_3$	9×10^{-6}	1.4×10^{-5}
Θ_2	$a_2 m_3 o_{c3}$	-6×10^{-6}	-5.5×10^{-6}
Θ_3	$m_2 o_{c2} + a_2^2 m_3 + I_2$	2×10^{-5}	4×10^{-5}
Θ_4	C_{f1}	2.1×10^{-5}	5.05×10^{-5}
Θ_5	C_{f2}	4.6×10^{-5}	9.3×10^{-5}
Θ_6	C_{f3}	2.6×10^{-5}	2.8×10^{-5}
Θ_7	$m_2 O_{c2} + a_2 m_3$	5×10^{-3}	5.2×10^{-3}
Θ_8	$m_3 O_{c3}$	5.27×10^{-3}	5.27×10^{-3}

Tabla 6.1 Estimación paramétrica

Capítulo 6

Conclusiones

Para el desarrollo de esta tesis se realizó un extenso análisis matemático del comportamiento del robot *Omni Phantom*, con el fin de proporcionar la parte medular de todo el trabajo y exponerlo en forma sencilla. Este análisis no es posible encontrarlo en la literatura, lo que aumenta el valor de su aportación. Se obtuvieron las ecuaciones que permiten mapear del espacio cartesiano al espacio cartesiano, sin duda el uso de la cinemática inversa y de la cinemática directa es de suma importancia en el diseño de trayectorias y aprovechar el espacio de trabajo del manipulador sin provocar colisiones entre eslabones.

La base de toda la experimentación, se logró mediante una interfaz gráfica que permite desde obtener la posición del efector final, hasta poner en marcha las tres teorías de control expuestas en la tesis. Este software es una excelente herramienta para poder extrapolar mis experimentos en distintas aplicaciones.

Se estimaron de los parámetros del robot *Omni Phantom*, como se muestra en la Tabla 6.1. Un factor importante que garantizaría que los parámetros estimados tiendan a los reales, es la excitación persistente. En otras palabras los torques de entrada deben ser capaces de seguir señales más ricas en frecuencia. Sin embargo, la señal de referencia utilizada y con la cual se obtuvieron los resultados de los experimento 1 y 2 de la Tabla 6.1, fue una senoidal en cada articulación, por lo que a pesar de que se obtuvieron valores para los parámetros no se garantiza que sean la mejor estimación.

En esencia un control adaptable compensa en línea parámetros e incertidumbres; sin embargo, a la excitación persistente se le suman otros factores que pueden hacer más difícil la obtención de parámetros más fiables. Un ejemplo puede ser el tiempo de muestreo. El temporizador que se utilizó dio el mínimo tiempo posible, 10 [ms]. es el tiempo mínimo que ofrece teóricamente pero en la práctica el tiempo de muestreo fue de 15 [ms].

Gracias a la implementación del algoritmo de Slotine y Li, se logró que el error fuera más pequeño que con un PD.

Para los controladores que no contemplan la dinámica de la planta, como lo son el PID y PD, se mostró que tienen limitantes al exigirle al robot seguir

trayectorias complejas a alta velocidad, lo que no significa que sean malos este tipo de controles. A pesar de que la dificultad sea un poco mayor un control adaptable demostró un mejor seguimiento de las trayectorias.

6.1. Trabajo futuro

La principales aplicaciones para el robot *Omni Phantom* se encuentran en la háptica, siendo esta área muy prometedora. Para diseñar aplicaciones hápticas es necesario conocer los parámetros del robot, así que el primer experimento es el validar los parámetros que se obtuvieron, de lo contrario habrá que realizar más experimentación. La obtención de una mejor aproximación de los parámetros estimados a los reales permitiría una compensación de la dinámica del robot, se reflejaría en la transparencia del sistema (compensación de gravedad, eliminar la percepción debida a la fricción etc.).

Debido a que las aplicaciones hápticas necesitan objetos virtuales, en cuanto a la programación, se puede desarrollar una plataforma de experimentación que presente un entorno gráfico, donde permita sentir texturas y formas, a la cual se le pueden agregar diferentes esquemas de control.

Capítulo 7

Apéndice A

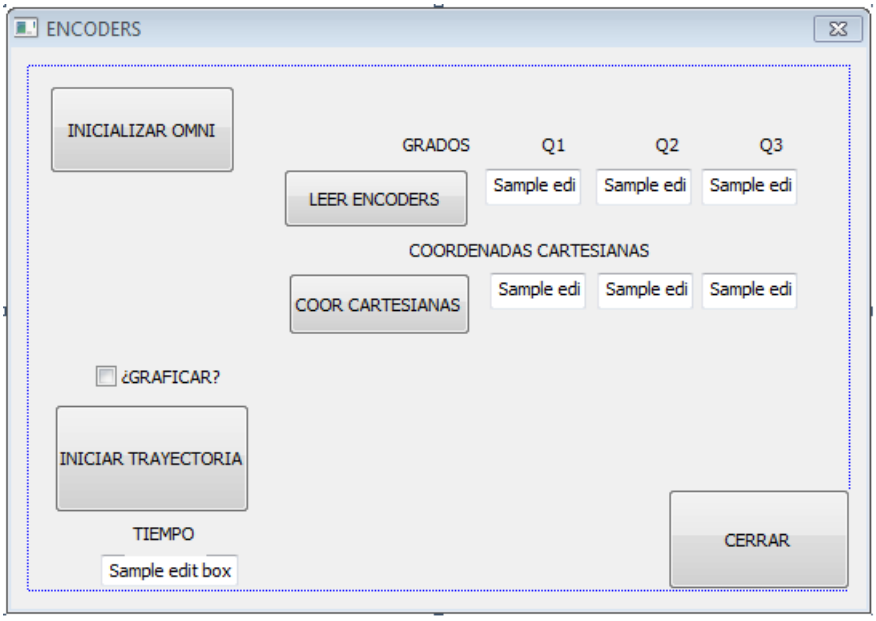

Figura 7.1: Interfaz gráfica para la implementación del control

La ventana principal de la interfaz gráfica cuenta con los siguientes botones:

- Botón INICIALIZAR OMNI, realiza la conexión del manipulador con Visual Studio.
- Botón LEER ENCODER nos permiten obtener las cuentas de los encoders de las tres primeras articulaciones.

- Botón COOR CARTESIANAS nos proporciona la posición del efector final en el espacio cartesiano para cualquier instante de tiempo.
- Botón más importantes es el que inicia el experimento, llamado INICIAR TRAYECTORIA al hacer click sobre él se ejecuta toda la programación para el seguimiento de trayectorias con PD, PID y adaptable, los códigos para el seguimiento de trayectorias se encuentran al final del apéndice A.

Para generar las gráficas se genera un archivo m desde *Visual Studio* donde se guarda toda la información necesaria para gráficar, el programa que realiza las gráficas se llama *Graficar.m*, disponible en Anexo A.

PROGRAMA DE LA INTERFAZ GRÁFICA DESARROLLADO EN MFC APLICATION

```
#include "stdafx.h"
#include "ENCODERS.h"
#include "ENCODERSDlg.h"
#include "Mmsystem.h" // LibrerÃya de sistema
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#include <windows.h>
#include <conio.h>
#include <afxwin.h>
#include <HD/hd.h>/*libreria PARA EL HAPTIC DEVICE*/
#include <HDU/hduError.h>
#include <HDU/hduVector.h>/*libreria para realizar operaciones de
    VECTORES*/
#include<HDU/hduMatrix.h>/*libreria para realizar operaciones de
    MATRICES*/
#include  "analysis.h"
#include <GL/glut.h>

//DECLARACION DE PARAMETROS DEL ROBOT
#define MAX_INPUT_DOF    3
#define MAX_OUTPUT_DOF   3

static int gNumMotors = 0;
static int gNumEncoders = 0;
static long alMotorDACValuesApp[MAX_OUTPUT_DOF];
static long alMotorDACValuesServo[MAX_OUTPUT_DOF];

#ifdef _DEBUG
#define new DEBUG_NEW
#endif
double T = .015; // Periodo de muestreo
UINT_PTR timer = 0;
#pragma comment(lib, "winmm.lib") // Se manda a llamar la libreria
    winmm para poder usar Mmsystem.h
#pragma comment(lib, "analysis.lib")
const double     pi=3.14159265358979323846264338327 95;
double qomni[6] = {pi};
```

```
//constantes de converiÃ§s de cuentas a grados rel1pos 55/2356
    rel1neg 50/2424 rel2=105/4360
const double rel1=0.021966527196652719665271966527196652719665272;
const double rel2=0.023648648648648648648648648648648648648648648648;
const double rel3=0.0234375;
const double g=9.79;
double qa[3] = {0.0};
double qalter[3] = {0.0};
const double DACR=1;//Factor de conversion
double d1=127, a2=129, a3=133;//UNIDAD MILIMETROS
double r, d, b1, b2, b, beta, alfa, q1, q2, q3, Q1, Q2, Q3;
double p0x=84, p0y=-111, p0z=8.44;
double X=0,Y=0,Z=0,x=0,y=0,z=0,radio=65;
float tfin=63;
double voltaje[3]={0, 0, 0};
HDSchedulerHandle gCallbackHandle;
HHD hHD;
//SE DECLARA LA POSICIÃŞN FINAL
double pfx=160.0;
double pfy=0.0;
double pfz=100.0;
double t=0;
int indice=0;
double datos[15][40000]={0.0};//ARREGLO PARA GRAFICAR
int gcount = 0;
DWORD tc, ts; // variables auxiliares para tiempos
FILE* archivo;

/*DECLARACIÃŞN DE LOS COEFICIENTES DEL POLINOMIO*/
double a1xyz=0;
double a2xyz=0;
double a3x=(10*(pfx-p0x))/(pow(tfin,3));
double a3y=(10*(pfy-p0y))/pow(tfin,3);
double a3z=(10*(pfz-p0z))/pow(tfin,3);
double a4x=(-15*(pfx-p0x))/pow(tfin,4);
double a4y=(-15*(pfy-p0y))/pow(tfin,4);
double a4z=(-15*(pfz-p0z))/pow(tfin,4);
double a5x=(6*(pfx-p0x))/pow(tfin,5);
double a5y=(6*(pfy-p0y))/pow(tfin,5);
double a5z=(6*(pfz-p0z))/pow(tfin,5);

//DEFINICIONES DE LAS GANANCIAS DEL PID

        double tau[3] = {0.0};
        double tauaux[3] = {0.0};
        const double kp[3] = {1.2,4.3,3.0};              //
            Sintonizacion proporcional
        const double kd[3] = {0.01,0.01,0.01};             //
            Sintonizacion derivativa
        const double ki[3] = {0.02,0.04,0.03};
                    //Sintonizacion integral
        const double PAR[3] = {100000,57142.8514,100000}; //
            Relacion de par a voltaje
        double ed[3] = {0.0};
        double edfp[3] = {0.0}, edf[3] = {0.0}, omegafilt = 80.0;
        static double ei[3] = {0.0};
```

```
//DEFINICION DE VARIABLES PARA EL ADAPTABLE
        double qa_1[3]={0.0};
        double dq[3]={0.0};
        double ddq[3]={0.0};
        double dq_1[3] = {0.0};
        double e[3] = {0.0};
        double de[3] = {0.0};
        double dde[3] = {0.0};
        double e_1[3] = {0.0};
        double de_1[3] = {0.0};
        double qf[3]={0.0};
        double dqf[3]={0.0};
        double ddqf[3]={0.0};
        double qf_1[3]={0.0};
        double dqf_1[3]={0.0};
        double ddqf_1[3]={0.0};
        //double ddqf_1[3]={0.0};
        double qr[3]={0.0};
        double dqr[3]={0.0};
        double ddqr[3]={0.0};

        const int ren = 3;//RENGLONES PARA EL REGRESOR
        const int col = 8;//COLUMNAS PARA EL REGRESOR
        double S[3]={0.0};
        //const double DELTA[3]={115,105,140};
        const double DELTA[3]={60.0,65.0,43.0};
        //const double KV[3]={.02040,.03675,.02280};
        const double KV[3]={.078,.075,.07};
        //const double KV[3]={.05010,.00918,.00520};
        double YY[ren][col]={0.0};
        double YYT[col][ren]={0.0};
        double AUX[col]={0.0};
        double AUX1[ren]={0.0};
        //const double GAMMA[col]={10.0};
        const double GAMMA[col
            ]={0.00000000175,0.0000000015,0.00000000175,
        0.000000025,0.000000025,0.000000025,0.000011,0.000011};
        double DTHES[col]={0.0};
        double THETA[col]={0.0};
        double W={8.0};
        double furier[col]={0.2};

        //PARA LA VALIDACION DE PARAMETROS
double H[ren][ren]={0.0};
double INVH[ren][ren]={0.0};
double C[ren][ren]={0.0};
double D[ren][ren]={0.0};
double G[ren]={0.0};
double CDG[ren]={0.0};
double CDQ[ren]={0.0};
double DDQ[ren]={0.0};
double ENTRADA[ren]={0.0};
double ACELERACION[ren]={0.0};
double ACELERACION_1[ren]={0.0};
double VELOCIDAD[ren]={0.0};
double VELOCIDAD_1[ren]={0.0};
double POSICION[ren]={0.0,0.0,-1.57};
```

```
double POSICION_1[ren]={0.0};

double T1=0.000014;
double T2=-0.0000055;
double T3=0.0004;
double T4=0.0000505;
double T5=0.000093;
double T6=0.000028;
double T7=0.0052;
double T8=0.00527;

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        enum { IDD = IDD_ABOUTBOX };
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);     // DDX/
            DDV support
// Implementation
protected:
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

CENCODERSDlg::CENCODERSDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CENCODERSDlg::IDD, pParent)
{
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CENCODERSDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        DDX_Control(pDX, IDC_GRADOSQ1, m_GRADOS1);
        DDX_Control(pDX, IDC_GRADOSQ2, m_GRADOS2);
        DDX_Control(pDX, IDC_GRADOSQ3, m_GRADOS3);
        DDX_Control(pDX, IDC_X, m_POSX);
        DDX_Control(pDX, IDC_Y, m_POSY);
        DDX_Control(pDX, IDC_Z, m_POSZ);
        DDX_Control(pDX, IDC_EDIT8, m_TIEMPO);
}

BEGIN_MESSAGE_MAP(CENCODERSDlg, CDialog)
```

```
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_INITOMNI, &CENCODERSDlg::
            OnBnClickedInitomni)
        ON_BN_CLICKED(IDC_CIERRE, &CENCODERSDlg::OnBnClickedCierre)
        ON_BN_CLICKED(IDC_LEER, &CENCODERSDlg::OnBnClickedLeer)
        ON_BN_CLICKED(IDC_HOME, &CENCODERSDlg::OnBnClickedHome)
        ON_BN_CLICKED(IDC_BUTTON1, &CENCODERSDlg::
            OnBnClickedButton1)

END_MESSAGE_MAP()

BOOL CENCODERSDlg::OnInitDialog()
{
        CDialog::OnInitDialog();
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
                CString strAboutMenu;
                strAboutMenu.LoadString(IDS_ABOUTBOX);
                if (!strAboutMenu.IsEmpty())
                {
                        pSysMenu->AppendMenu(MF_SEPARATOR);
                        pSysMenu->AppendMenu(MF_STRING,
                            IDM_ABOUTBOX, strAboutMenu);
                }
        }
        SetIcon(m_hIcon, TRUE);                 // Set big icon
        SetIcon(m_hIcon, FALSE);                // Set small icon
        return TRUE;  // return TRUE  unless you set the focus to a
             control
}

void CENCODERSDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}

void CENCODERSDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting
```

```
                SendMessage(WM_ICONERASEBKGND, reinterpret_cast<
                    WPARAM>(dc.GetSafeHdc()), 0);
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

HCURSOR CENCODERSDlg::OnQueryDragIcon()
{
        return static_cast<HCURSOR>(m_hIcon);
}

//Directly sets the DAC values.
HDCallbackCode HDCALLBACK ServoSchedulerCallback(void *pUserData)
{
    hdBeginFrame(hdGetCurrentDevice());
    assert(alMotorDACValuesServo);
        hdSetLongv(HD_CURRENT_MOTOR_DAC_VALUES, alMotorDACValuesApp
            );
    hdEndFrame(hdGetCurrentDevice());

    return HD_CALLBACK_CONTINUE;
}

typedef struct
{
     HDlong encoder_values[MAX_INPUT_DOF];
     HDlong motor_dac_values[MAX_OUTPUT_DOF];
     hduVector3Dd position;
} DeviceStateStruct;
/*******************************************************
 Callback that retrieves state.
 CON UN CALLBACK PODEMOS RECUPERAR EL ESTADO DEL ROBOT
*******************************************************/
DeviceStateStruct state;
HDCallbackCode HDCALLBACK GetDeviceStateCallback(void *pUserData)
{
        DeviceStateStruct *pState = static_cast<DeviceStateStruct
            *>(pUserData);

        hdGetLongv(HD_CURRENT_JOINT_TORQUE, pState->
            motor_dac_values);
    hdGetLongv(HD_CURRENT_ENCODER_VALUES, pState->encoder_values);
    hdGetDoublev(HD_CURRENT_POSITION, pState->position);

    return HD_CALLBACK_DONE;
```

```
}

bool CENCODERSDlg::lecturaOMNI(void)
{
                hdScheduleSynchronous(GetDeviceStateCallback, &
                    state,
        HD_MIN_SCHEDULER_PRIORITY);

        /*if (state.encoder_values[0]<=0)
                qa[0] = (state.encoder_values[0]*rel1neg)
                    *.01745329252;
        else{
                qa[0] = (state.encoder_values[0]*rel1pos)
                    *.01745329252;
        }*/
                qa[0] = (state.encoder_values[0]*rel1)
                    *.01745329252;
                qa[1] = (-state.encoder_values[1]*rel2)
                    *.01745329252;
                qa[2] = ((state.encoder_values[2]*rel3)-90)
                    *.01745329252;
        return true;
}

bool CENCODERSDlg::cartesianoOMNI(void)
{
        lecturaOMNI();
                 X=(a2*cos(qa[1])+a3*cos(qa[2]))*cos(qa[0]);
                 Y=(a2*cos(qa[1])+a3*cos(qa[2]))*sin(qa[0]);
            Z=d1+a2*sin(qa[1])+a3*sin(qa[2]);
return true;
}

void CENCODERSDlg::OnBnClickedLeer()
{
        bool LECTURAENCODERS;
                LECTURAENCODERS=lecturaOMNI();
                if(!LECTURAENCODERS)
                MessageBox(_T("No se pudieron leer los encoders del
                    OMNI"));
                else{
                CString GRADOS1;
                CString GRADOS2;
                CString GRADOS3;

                GRADOS1.Format(_T("%f"),qa[0]);
                GRADOS2.Format(_T("%f"),qa[1]);
                GRADOS3.Format(_T("%f"),qa[2]);

                CENCODERSDlg::m_GRADOS1.SetWindowTextW(GRADOS1);
                CENCODERSDlg::m_GRADOS2.SetWindowTextW(GRADOS2);
                CENCODERSDlg::m_GRADOS3.SetWindowTextW(GRADOS3);
                }
}

void CENCODERSDlg::OnBnClickedButton1()
{
```

```
        bool LECTURAENCODERS;
        LECTURAENCODERS=cartesianoOMNI();
        if(!LECTURAENCODERS)
                        MessageBox(_T("No se pudieron leer los
                            encoders del OMNI"));
        else{
                CString CARTESIANOX,CARTESIANOY,CARTESIANOZ;
                CARTESIANOX.Format(_T("%f"),X);
                CARTESIANOY.Format(_T("%f"),Y);
                CARTESIANOZ.Format(_T("%f"),Z);

                        CENCODERSDlg::m_POSX.SetWindowTextW(
                            CARTESIANOX);
                        CENCODERSDlg::m_POSY.SetWindowTextW(
                            CARTESIANOY);
                        CENCODERSDlg::m_POSZ.SetWindowTextW(
                            CARTESIANOZ);
        }
}

void CENCODERSDlg::OnBnClickedHome()
{               /*DECLARACION DE LOS PTS INICIALES Y FINALES*/
//AQUI CACHAMOS LA POSICIÃŞN INICIAL
//      cartesianoOMNI();
        /*p0x=92;
        p0y=-108;
        p0z=4.78;*/
        //timeKillOtherEvents();
        tc = timeGetTime();
        t = (timeGetTime()-tc)/1000;
        SetTimer( 0, T*1000, (TIMERPROC) &TimerOMNI );
        //timeBeginPeriod(1);
        //timer = timeSetEvent(static_cast<UINT>(1000*T), 0,
            TimerOMNI, 0, TIME_PERIODIC);
}

void CENCODERSDlg::MandarVoltajesOMNI(double art1, double art2,
    double art3)
{
        //SE DEBEN ELEGIR VALORES DE ENTRE -32768 A 32767
        voltaje[0]=-art1;
        voltaje[1]=art2;
        voltaje[2]=art3;
        for(int i=0; i<3; i++){
                if(voltaje[i]>=28000)
                        voltaje[i]=28000;
                if(voltaje[i]<=-28000)
                        voltaje[i]=-28000;
        alMotorDACValuesApp[i]= voltaje[i]*DACR;

        }
hdSetLongv(HD_CURRENT_MOTOR_DAC_VALUES, alMotorDACValuesApp);
}

void CALLBACK CENCODERSDlg::TimerOMNI(UINT uTimeID, UINT uMsg,
    DWORD dwUser, DWORD dw1, DWORD dw2)
{
```

```
        CENCODERSDlg *pMainWnd = (CENCODERSDlg* ) AfxGetApp()->
            m_pMainWnd;
        CString TIEMPO;
        bool LECTURAENCODERS;
        LECTURAENCODERS=lecturaOMNI();

SEÃŚAL CUADRADA CON SERIES FURIER
qf[0]=1.27324*(sin(W*t)+0.3333*sin(3*W*t)+0.2*sin(5*W*t)+0.14285*
    sin(5*W*t));
qf[1]=0;
qf[2]=-1.57;*/
//TRAYECTORIA PETALOS
                x=155+radio*cos(.2*t)*(cos(.5*t))+5*cos(.4*t);
                y=radio*sin(.2*t)*(cos(.5*t))+5*cos(.4*t);
                x=155+radio*cos(.2*t);
                y=radio*sin(.2*t);
                z=50;
                +2.34375*/
//TRAYECTORIA GENERADA CON UN POLINOMIO DE 5 ORDEN
//-------------------------------------------------------------
        x = p0x + a1xyz*t +a2xyz*pow(t,2) + a3x*pow(t,3) + a4x*pow(
            t,4) + a5x*pow(t,5);
        y = p0y + a1xyz*t +a2xyz*pow(t,2) + a3y*pow(t,3) + a4y*pow(
            t,4) + a5y*pow(t,5);
        z = p0z + a1xyz*t +a2xyz*pow(t,2) + a3z*pow(t,3) + a4z*pow(
            t,4) + a5z*pow(t,5);
//-------------------------------------------------------------
        //SE PASA POR LA CINEMÃĄTICA INVERSA
        r=sqrt(x*x+y*y);
        d=sqrt(pow(x,2)+pow(y,2)+pow((z-d1),2));
        b1=pow(d,2)-pow(a3,2)+pow(a2,2);
        b2=2*a2*d;
        b=b1/b2;
        /*SE ESCOGE SIGNO MENOS POR QUE LA SOLUCIÃŞN ES CODO ARRIBA
            */
        beta=atan2(-sqrt(1-pow(b,2)),b);
        alfa=atan2(z-d1,r);
        q1=atan2(y,x);
        q2=alfa-beta;
        q3=atan2(z-d1-a2*sin(q2),r-a2*cos(q2));
        qf[0]=q1;
        qf[1]=q2;
        qf[2]=q3;
        /*qf[0]=(180*q1)/pi;
        qf[1]=(180*q2)/pi;
        qf[2]=(180*q3)/pi;*/
//CONTROL PID
//-------------------------------------------------------------
        for(int i=0; i<3; i++)
        e[i] = qf[i] - qa[i]; // Error articular

        //CALCULO PARA LA DERIVADA
        for(int i=0; i<3; i++){
                ed[i] = (e[i] - e_1[i])/T;
                edfp[i] = omegafilt*(ed[i]-edf[i]);
        }
```

```
        //CALCULO PARA LA INTEGRAL
        for(int i=0; i<3; i++)
                ei[i] += e[i]*T;

        for(int i=0; i<3; i++)
                tauaux[i] = kp[i]*e[i] +kd[i]*edf[i] +ki[i]*ei[i];
        //Conversion de par avoltaje
for(int i=0; i<3; i++)
tau[i] = tauaux[i]*PAR[i];

        for(int i = 0; i<3; i++){
                e_1[i] = e[i];
                edf[i] += edfp[i]*T;
        }
//-------------------------------------------------
//ADAPTABLE
        //CALCULO PARA LA DERIVADA (velocidad y aceleraciÃšn)
        //qa=posiciÃ§n real, qf=posicion deseada
        /*for(int i=0; i<3; i++)
        e[i] = qa[i] - qf[i]; // Error articular

        for(int i=0; i<3; i++){
                if(t<2*T)
                        dq[i] = 0.0;
                else
                        dq[i] = (qa[i] - qa_1[i])/T;
                de[i] = (e[i] - e_1[i])/T;
                if(t<2*T)
                        dqf[i] = 0.0;
                else
                        dqf[i] = (qf[i] - qf_1[i])/T;}

        for(int i=0; i<3; i++){
                //ddq[i] = (dq[i] - dq_1[i])/T;
                //dde[i] = (de[i] - de_1[i])/T;
                ddqf[i] = (dqf[i] - dqf_1[i])/T;}

        //CONTROL ADAPTABLE tqd=qdeseada,
        for(int i=0; i<3; i++){
        dqr[i]=dqf[i]-(DELTA[i]*(e[i]));
        }

        for(int i=0; i<3; i++){
        ddqr[i]=ddqf[i]-(DELTA[i]*(de[i]));
        }

        for(int i=0; i<3; i++){
        S[i]=de[i]+DELTA[i]*(e[i]);
        }

//LA MATRIZ Y ES DE 3X8
//YY TIENE 8 PARÃĄMETROS
//EL REGRESOR MODIFICADO ES
//-------------------------------------------------
/*
YY[0][0]=(ddqr[1]*cos(qa[2])*cos(qa[2])-sin(qa[2])*cos(qa[2])*(dq
     [2]*dqr[0]+dq[0]*dqr[2]));
```

```
YY[0][1]=(ddqr[0]*2.0*cos(qa[1])*cos(qa[2])-cos(qa[1])*sin(qa[2])*(
    dq[2]*dqr[0]+dq[0]*dqr[2])-sin(qa[1])*cos(qa[2])*(dq[1]*dqr[0]+
    dq[0]*dqr[1]));
YY[0][2]=(ddqr[0]*cos(qa[1])*cos(qa[1])-sin(qa[1])*cos(qa[1])*dq
    [1]*dqr[0]);
YY[0][3]=dqr[0];
YY[0][4]=0.0;
YY[0][5]=0.0;
YY[0][6]=0.0;
YY[0][7]=0.0;
YY[1][0]=(ddqr[1]+ddqr[2]);
YY[1][1]=(ddqr[1]*(2.0*sin(qa[1])*sin(qa[2])+2.0*cos(qa[1])*cos(qa
    [2]))+ddqr[2]*(sin(qa[1])*sin(qa[2])+cos(qa[1])*cos(qa[2]))+sin
    (qa[1])*cos(qa[2])*dq[0]*dqr[0]+(cos(qa[1])*sin(qa[2])-sin(qa
    [1])*cos(qa[2]))*((dq[1]-dq[2])*dqr[1]-(dq[1]+dq[2])*dqr[2]));
YY[1][2]=(ddqr[1]+sin(qa[1])*cos(qa[1])*dq[0]*dqr[0]);
YY[1][3]=0.0;
YY[1][4]=dqr[1];
YY[1][5]=0.0;
YY[1][6]=g*cos(qa[1]);
YY[1][7]=0.0;
YY[2][0]=ddqr[1]+ddqr[2]+  sin(qa[2])*cos(qa[2])*dq[0]*dqr[0]; //YY
    [2][0]=ddqr[1]+ddqr[2]+  sin(qa[2])*cos(qa[2])*dq[0]*dqr[0];
YY[2][1]=(ddqr[1]*(sin(qa[1])*sin(qa[2])+cos(qa[1])*cos(qa[2]))+cos
    (qa[1])*sin(qa[2])*dq[0]*dqr[0]+2.0*(cos(qa[1])*sin(qa[2])-sin(
    qa[1])*cos(qa[2]))*dq[1]*dqr[1]);
YY[2][2]=0.0;
YY[2][3]=0.0;
YY[2][4]=0.0;
YY[2][5]=dqr[2];
YY[2][6]=0.0;
YY[2][7]=g*cos(qa[2]);

Transpose(YY,3,8,YYT);
MatrixMul(YYT,S,col,ren,1,AUX);//Y'*s

for(int i=0; i<col; i++)
        DTHES[i] = -GAMMA[i]*AUX[i];

MatrixMul(YY,THETA,3,8,1,AUX1);

        for(int i=0; i<ren; i++){
                tauaux[i]=AUX1[i]-KV[i]*S[i];
}
                //Conversion de par a voltaje
for(int i=0; i<3; i++)
tau[i]= tauaux[i]*PAR[i];

for(int i=0; i<col; i++){
        THETA[i] += DTHES[i]*T;
}

for(int i = 0; i<3; i++){
                dq_1[i] = dq[i];
                e_1[i] = e[i];
                de_1[i] = de[i];
                dqf_1[i] = dqf[i];
```

```
                    ddqf_1[i] = ddqf[i];
                    }
/*%EL VECTOR DE PARAMETROS ES: VALIDACION DE LOS PARAMETROS
T1=m3*Oc3^2+I3;
T2=a2*m3*Oc3;
T3=m2*Oc2^2+a2^2*m3+I2;
T4=cf1;
T5=cf2;
T6=cf3;
T7=m2*Oc2+a2*m3;
T8=m3*Oc3;

tau[0]=3000*cos(5*t);
tau[1]=0*cos(5*t);
tau[2]=3100*cos(5*t);

H[0][0]=T1*cos(qa[2])*cos(qa[2])+2*T2*cos(qa[1])*cos(qa[2])+T3*cos(
    qa[1])*cos(qa[1]);
H[0][1]=0;
H[0][2]=0;
H[1][0]=0;
H[1][1]=2*T2*sin(qa[1])*sin(qa[2])+2*T2*cos(qa[1])*cos(qa[2])+T1+T3
    ;
H[1][2]=T2*sin(qa[1])*sin(qa[2])+T2*cos(qa[1])*cos(qa[2])+T1;
H[2][0]=0;
H[2][1]=T2*sin(qa[1])*sin(qa[2])+T2*cos(qa[1])*cos(qa[2])+T1;
H[2][2]=T1;

C[0][0]=-T3*sin(qa[1])*cos(qa[1])*dq[1]-T2*sin(qa[1])*cos(qa[2])*dq
    [1]-T1*sin(qa[2])*cos(qa[2])*dq[2]-T2*sin(qa[2])*cos(qa[1])*dq
    [2];
C[0][1]=-T3*sin(qa[1])*cos(qa[1])*dq[0]-T2*sin(qa[1])*cos(qa[2])*dq
    [0];
C[0][2]=-T1*sin(qa[2])*cos(qa[2])*dq[0]-T2*cos(qa[1])*sin(qa[2])*dq
    [0];
C[1][0]=T3*sin(qa[1])*cos(qa[1])*dq[0]+T2*sin(qa[1])*cos(qa[2])*dq
    [0];
C[1][1]=T2*(cos(qa[1])*sin(qa[2])-sin(qa[1])*cos(qa[2]))*(dq[1]-dq
    [2]);
C[1][2]=-T2*(cos(qa[1])*sin(qa[2])-sin(qa[1])*cos(qa[2]))*(dq[1]+dq
    [2]);
C[2][0]=T1*sin(qa[2])*cos(qa[2])*dq[0]+T2*cos(qa[1])*sin(qa[2])*dq
    [0];
C[2][1]=2.0*T2*(cos(qa[1])*sin(qa[2])-sin(qa[1])*cos(qa[2]))*dq[1];
C[2][2]=0;

G[0]=T4;
G[1]=T5;
G[2]=T6;

D[0][0]=0;
D[1][1]=T7*g*cos(qa[1]);
D[2][2]=T8*g*cos(qa[2]);

for(int i=0; i<3; i++){
                    if(t<2*T)
                            dq[i] = 0.0;
```

```
                    else
                                dq[i] = (qa[i] - qa_1[i])/T;
                                }

InvMatrix(H,3,INVH);
MatrixMul(C,dq,ren,ren,1,CDQ);
MatrixMul(D,dq,ren,ren,1,DDQ);

for(int i=0; i<3; i++){
CDG[i]= tau[i]-CDQ[i]-DDQ[i]-G[i];
}

MatrixMul(INVH,CDG,ren,ren,1,ACELERACION);

for(int i=0; i<col; i++){
          VELOCIDAD[i] += ACELERACION[i]*T;
          POSICION[i] += VELOCIDAD[i]*T;
}

for(int i = 0; i<3; i++){
                    ACELERACION_1[i]=ACELERACION[i];
                    VELOCIDAD_1[i] = VELOCIDAD[i];
                    POSICION_1[i] = POSICION[i];
}
//-------------------------------------------------------------------------------

          X=(a2*cos(qa[1])+a3*cos(qa[2]))*cos(qa[0]);
          Y=(a2*cos(qa[1])+a3*cos(qa[2]))*sin(qa[0]);
          Z=d1+a2*sin(qa[1])+a3*sin(qa[2]);

          TIEMPO.Format(_T("%f"),t);
          pMainWnd->m_TIEMPO.SetWindowTextW(TIEMPO);
          MandarVoltajesOMNI(tau[0], tau[1], tau[2]);
          ts = timeGetTime()-t*1000-tc;// Periodo de muestreo real (
               ms).
          t = 1.0*(timeGetTime()-tc)/1000; //Tiempo transcurrido (seg
               ).

          if((pMainWnd->m_grafic)&&(gcount==0)){
          datos[0][indice] = t;
          datos[1][indice] = tau[0];
          datos[2][indice] = tau[1];
          datos[3][indice] = tau[2];
          datos[4][indice] = (e[0]*180)/pi;
          datos[5][indice] = (e[1]*180)/pi;
          datos[6][indice] = (e[2]*180)/pi;
//GRAFICAR PARAMETROS
          datos[7][indice] = THETA[0];
          datos[8][indice] = THETA[1];
          datos[9][indice] = THETA[2];
          datos[10][indice] =THETA[3];
          datos[11][indice] =THETA[4];
          datos[12][indice] =THETA[5];
          datos[13][indice] =THETA[6];
          datos[14][indice] =THETA[7];
//GRAFICAR A CARTESIANAS
          datos[7][indice] = (qf[0]*180)/pi;
```

```
            datos[8][indice] = (qf[1]*180)/pi;
            datos[9][indice] = (qf[2]*180)/pi;
            datos[7][indice] = (POSICION[0]*180)/pi;
            datos[8][indice] = (POSICION[1]*180)/pi;
            datos[9][indice] = (POSICION[2]*180)/pi;
            datos[10][indice] = (qa[0]*180)/pi;
            datos[11][indice] = (qa[1]*180)/pi;
            datos[12][indice] = (qa[2]*180)/pi;
            datos[13][indice] = X;
            datos[14][indice] = Y;
            datos[15][indice] = Y;
            datos[16][indice] = X;
            datos[17][indice] = Y;
            datos[18][indice] = Z;
            indice++;
            gcount==0;
            }
            else
            gcount--;
            }
return;
}

void CENCODERSDlg::OnBnClickedInitomni(){
            gCallbackHandle = HD_INVALID_HANDLE;
            HDErrorInfo error;
    HDboolean bDone = FALSE;
    hHD = hdInitDevice(HD_DEFAULT_DEVICE);
            memset(&state, 0, sizeof(DeviceStateStruct));
    if (HD_DEVICE_ERROR(error = hdGetError()))
    {
                    MessageBox(_T("ERROR NO SE INICIALIZO EL
                        DISPOSITIVO"));
            exit(-1);
    }

            MessageBox(_T("Â£INICIALIZAR ROBOT?"));

    hdGetIntegerv(HD_OUTPUT_DOF, &gNumMotors);
    hdGetIntegerv(HD_INPUT_DOF, &gNumEncoders);
    memset(alMotorDACValuesApp, 0, sizeof(long) * MAX_OUTPUT_DOF);
    memset(alMotorDACValuesServo, 0, sizeof(long) * MAX_OUTPUT_DOF)
            ;
    gCallbackHandle = hdScheduleAsynchronous(
            ServoSchedulerCallback, 0, HD_MAX_SCHEDULER_PRIORITY);
    if (HD_DEVICE_ERROR(error = hdGetError()))
    {
                    MessageBox(_T("Fallo el retorno de llamada"));
            hdDisableDevice(hHD);
            exit(-1);           //return -1;
    }

    hdEnable(HD_FORCE_OUTPUT);

    /* Start the haptic rendering loop */
    hdStartScheduler();
    if (HD_DEVICE_ERROR(error = hdGetError()))
```

```
    {
                MessageBox(_T("No se pudieron INICIAR EL SERVOLOOP"
                    ));

        hdDisableDevice(hHD);
         exit(-1);
    }
}

void CENCODERSDlg::OnBnClickedCierre()
{
        if (MessageBox(_T("Â£Desea salir?"),_T("Salir"),
            MB_ICONQUESTION+MB_YESNO)==IDYES){

                if(fopen_s(&archivo, "prueba.m","w")!=0){
                MessageBox(_T("No se pudo crear el archivo para
                    graficar"));
                        }
                else{
                for(int i=0; i<indice; i++){
                fprintf(archivo, "\t%f \t%f \t%f \t%f \t%f \t%f \t%
                    f \t%f \t%f \t%f \t%f \t%f \t%f \t%f \n",
                    datos[0][i], datos[1][i], datos[2][i], datos
                    [3][i], datos[4][i], datos[5][i], datos[6][i],
                    datos[7][i], datos[8][i], datos[9][i], datos
                    [10][i], datos[11][i], datos[12][i], datos[13][
                    i], datos[14][i]);
                }
                fclose(archivo);
                }
        CDialog::OnOK();
        KillTimer (0);
        hdStopScheduler();
    hdUnschedule(gCallbackHandle);
        hdDisableDevice(hHD);

        }
}
```

El cálculo para llegar al regresor podría involucrar errores, así que para demostrar que el regresor es correcto, se puede hacer una simulación en Matlab. La idea es darle una entrada a la dinámica τ y obtener las aceleraciones articulares a la salida, integrando una y dos veces, así se conocerán las velocidades y posiciones articulares, estas 3 señales, posiciones, velocidades y aceleraciones articulares se meten al regresor, recuérdese que el regresor esta en funciónde estas señales, así lo que se espera observar a la salida del regresor es la misma τ de la entrada de la dinámica, en la siguiente imagen se muestra el diagrama de bloques de *Simulink* de la simulación.

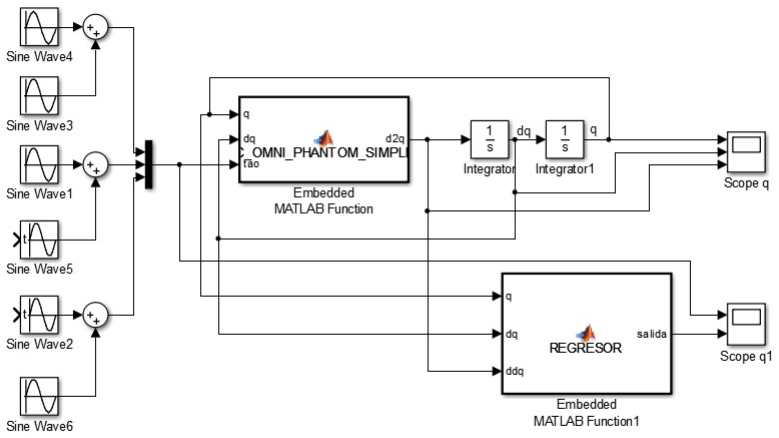

Figura 7.2: Diagrama de bloques para la comprobación del Regresor

La Figura (7.2) muestra el diagrama de bloques de *Simulink* de la simulación, se observa que en la entrada de cada articulación esta compuesta por la suma de dos señales senoidales, el código del modelo dinámico y del regresor se encuentran al final en el Apéndice A.

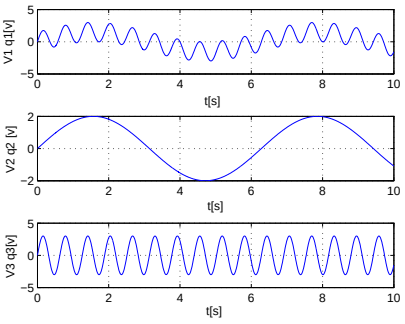

Figura 7.3: τ de entrada

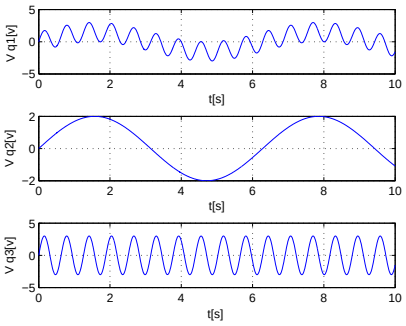

Figura 7.4: τ a la salida del regresor

En las figuras anteriores se observan las señales τ, a la entrada de la dinámica y la τ a la salida del regresor, las señales son indénticas, por lo que el regresor fue correctamente realizado.

PROGRAMA DEL REGRESOR REALIZADO EN MATLAB

```
function salida = REGRESOR(q,dq,tao)
//Bajo la suposicion de los siguientes parametros
m1=.1;
m2=.1;
m3=.1;
a2=.135;
a3=.13;
d1=.127;
Oc1=.06;
Oc2=.07;
Oc3=.07;
cf1=.01;
cf2=.01;
cf3=.01;
g=9.79;

%LA MATRIZ Y ES DE 3X8
%Y TIENE 8 PARÃĄMETROS
Y1=[ddq1*cos(q3)^2 ddq1*2*cos(q2)*cos(q3) ddq1*cos(q2)^2 0 0 0 0 0;
    ddq2+ddq3 ddq2*(2*sin(q2)*sin(q3)+2*cos(q2)*cos(q3))+ddq3*(sin(
        q2)*sin(q3)+cos(q2)*cos(q3)) ddq2 0 0 0 0 0;
    ddq2+ddq3 ddq2*(sin(q2)*sin(q3)+cos(q2)*cos(q3)) 0 0 0 0 0 0];

Y2=[dq3*(-cos(q3)*sin(q3))+dq1*(-cos(q3)*sin(q3)) ddq3*(-cos(q2)*
     sin(q3))+dq2*(-sin(q2)*cos(q3))+dq1*(-sin(q2)*cos(q3))+dq1*(-
     cos(q2)*sin(q3)) dq2*(-cos(q2)*sin(q2))+dq1*(-cos(q2)*sin(q2))
     0 0 0 0 0;
     0 dq1*(sin(q2)*cos(q3))+dq2*(2*cos(q2)*sin(q3)-2*sin(q2)*cos(q3
          ))+dq3*(.5*cos(q2)*sin(q3)-.5*sin(q2)*cos(q3))+.5*dq3*(sin(
          q2)*cos(q3)-cos(q2)*sin(q3))+dq3*(sin(q2)*cos(q3)-cos(q2)*
          sin(q3))+dq2*(sin(q2)*cos(q3)-cos(q2)*sin(q3))+.5*dq2*(cos(
          q2)*sin(q3)-sin(q2)*cos(q3))+dq3*(sin(q2)*cos(q3)-cos(q2)*
          sin(q3))+.5*dq2*(sin(q2)*cos(q3)-cos(q2)*sin(q3))+dq2*(sin(
          q2)*cos(q3)-cos(q2)*sin(q3)) dq1*(cos(q2)*sin(q2)) 0 0 0 0
          0;
     dq1*(cos(q3)*sin(q3)) dq1*(cos(q2)*sin(q3))+dq2*(cos(q2)*sin(q3
          )-sin(q2)*cos(q3))+.5*dq3*(cos(q2)*sin(q3)-sin(q2)*cos(q3))
          +dq2*(cos(q2)*sin(q3)-sin(q2)*cos(q3))+.5*dq3*(sin(q2)*cos(
          q3)-cos(q2)*sin(q3))+.5*dq2*(cos(q2)*sin(q3)-sin(q2)*cos(q3
          ))+.5*dq2*(sin(q2)*cos(q3)-cos(q2)*sin(q3)) 0 0 0 0 0 0];

Y3=[0 0 0 dq1 0 0 0 0;
     0 0 0 0 dq2 0 0 0;
     0 0 0 0 0 dq3 0 0];

Y4=[0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 g*cos(q2) 0;
     0 0 0 0 0 0 0 g*cos(q3)];

%EL REGRESOR ES:
Y=Y1+Y2+Y3+Y4;
%EL VECTOR DE PAR+AMETROS ES:

T1=m3*Oc3^2+I3;
T2=a2*m3*Oc3;
T3=m2*Oc2^2+a2^2*m3+I2;
```

```
T4=cf1;
T5=cf2;
T6=cf3;
T7=m2*Oc2+a2*m3;
T8=m3*Oc3;

THETA=[T1;
      T2;
      T3;
      T4;
      T5;
      T6;
      T7;
      T8];
salida=Y*THETA;

end
```

PROGRAMA DINÁMICA DE ROBOT OMNI REALIZADO EN MATLAB

```
function d2q = DINAMIC_OMNI_PHANTOM_SIMPLIFICADO(q,dq,tao)

%PARAMETROS DINAMICOS DEL ROBOT OMNI PHANTOM
m1=.1;
m2=.1;
m3=.1;
a2=.135;
a3=.13;
d1=.127;
Oc1=.06;
Oc2=.07;
Oc3=.07;
cf1=.1;
cf2=.1;
cf3=.1;
g=9.79;

%SIMPLIFINADO LA CAPTURA DE ENTRADAS
q1=q(1);
q2=q(2);
q3=q(3);
dq1=dq(1);
dq2=dq(2);
dq3=dq(3);

%CALCULO DE LOS MOMENTOS DE INERCIA
%ESLABÃŞN 1

Ix1=(1/12)*m1*d1^2;
IZ1=(1/12)*m1*d1^2;
%ESLABÃŞN 2
Iy2=(1/12)*m2*a2^2;
Iz2=(1/12)*m2*a2^2;
%ESLABÃŞN 3
```

```
Iy3=(1/12)*m3*a3^2;
Iz3=(1/12)*m3*a3^2;


%LA MATRIZ DE INERCIA H ES:
H=[(m3*Oc3^2+Iy3)*cos(q3)^2+2*a2*m3*Oc3*cos(q2)*cos(q3)+(m2*Oc2^2+
    a2^2*m3+Iy2)*cos(q2)^2  0  0
    0  2*a2*m3*Oc3*sin(q2)*sin(q3)+2*a2*m3*Oc3*cos(q2)*cos(q3)+m3*
        Oc3^2+m2*Oc2^2+a2^2*m3+Iz3+Iz2  a2*m3*Oc3*sin(q2)*sin(q3)+a2
        *m3*Oc3*cos(q2)*cos(q3)+m3*Oc3^2+Iz3;
    0  a2*m3*Oc3*sin(q2)*sin(q3)+a2*m3*Oc3*cos(q2)*cos(q3)+m3*Oc3^2+
        Iz3  m3*Oc3^2+Iz3];


%LA MATRIZ CON LOS SIMBÃŞLOS DE CRISTOFFEL C ES:
C11=dq3*(-(m3*Oc3^2+Iy3)*cos(q3)*sin(q3)-a2*m3*Oc3*cos(q2)*sin(q3))
    +dq2*(-a2*m3*Oc3*sin(q2)*cos(q3)-(m2*Oc2^2+a2^2*m3+Iy2)*cos(q2)
    *sin(q2));
C12=dq1*(-a2*m3*Oc3*sin(q2)*cos(q3)-(m2*Oc2^2+a2^2*m3+Iy2)*cos(q2)*
    sin(q2));
C13=dq1*(-(m3*Oc3^2+Iy3)*cos(q3)*sin(q3)-a2*m3*Oc3*cos(q2)*sin(q3))
    ;
C21=dq1*(a2*m3*Oc3*sin(q2)*cos(q3)+(m2*Oc2^2+a2^2*m3+Iy2)*cos(q2)*
    sin(q2));
C22=m3*a2*Oc3*(cos(q2)*sin(q3)-sin(q2)*cos(q3))*(dq2-dq3);
C23=-m3*a2*Oc3*(cos(q2)*sin(q3)-sin(q2)*cos(q3))*(dq2+dq3);
C31=dq1*((m3*Oc3^2+Iy3)*cos(q3)*sin(q3)+a2*m3*Oc3*cos(q2)*sin(q3));
C32=2*m3*a2*Oc3*(cos(q2)*sin(q3)-sin(q2)*cos(q3))*dq2;
C33=0;
C=[C11  C12  C13;
    C21  C22  C23;
    C31  C32  C33];

%LA MATRIZ G ES:
G=[0;g*m2*Oc2*cos(q2)+a2*g*m3*cos(q2);g*m3*Oc3*cos(q3)];

%LA MATRIZ CON LOS COEFICIENTES DE FRICCIÃŞN ES:
D=[cf1  0  0;0  cf2  0;  0  0  cf3];

d2q=inv(H)*(tao-(C*dq)-(D*dq)-G);
end
```

Bibliografía

[1] Norman S. Nise, Sistemas de control para ingeniería, 2004.

[2] Mark W. Spong and M. Vidyasagar, Robot Dynamics and Control, 1989.

[3] Petros Ioannou Baris Fidan, Adaptive control tutorial.

[4] Bruno Siciliano, Robotics Modelling, Planning and control, 2005.

[5] E. Slotine Jean-Jaques, Applied nonlineal control.

[6] Hassan K. Khalil, Nonlinear systems, 1996.

[7] ensable, Specifications for the PHANTOM Desktop and PHANTOM Omni. haptic devices, http//www.sensable.com/documents/STIJan2009DesktopOmniComparison print.pdf

[8] Eduardo Nájera Fernández y Antonio DÃŋaz Estrella Dept. de Tecnología Electrónica ETS Ingeniería Telecomunicación Univ. de Málaga 29071 MÃąlaga, Interacción pseudoháptica en el simulador de entrenamiento APSiDeTH. http://aipo.es/articulos/5/1374.pdf

[9] Implementación de una Plataforma Experimental para un Sistema de Teleoperación Robótica en Tiempo Real. *http* : *//www.iiis.org/CDs*2010/*CD*2010*CSC/CISCI*$_2$010/ *PapersPdf/CA*825*WV.pdf*

[10] Amanda Pauli Aldana Suárez y Susana del Pilar Yañez Mantilla, adaptación de pinzas de laparoscopia a dispostitivos hápticos phantom omni y desarrollo de software de evaluación. http://repository.unimilitar.edu.co/bitstream/10654/11365/1 /AldanaSuarezAmandaPaulin2013.pdf

[11] Geomagic, Sensable Phantom, Phantom Omni Overview, http://geomagic.com/en/products/Phantom-Omni/overview.

[12] Sensable new part de Geomatic. http://www.dentsable.com/haptic-phantom-omni.htmspecs