DISEÑO DEL PROTOTIPO DE CONTROL COMPUTARIZADO DE UN BRAZO ROBÓTICO PARA MANIPULACION DE MATERIAL Y RESIDUOS BIOLÓGICOS

JOHNNY ZULUAGA ALVIS (10033648)

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE TECNOLOGÍA

INGENIERÍA MECATRÓNICA

PEREIRA

2016

DISEÑO DEL PROTOTIPO DE CONTROL COMPUTARIZADO DE UN BRAZO ROBÓTICO PARA MANIPULACION DE MATERIAL Y RESIDUOS BIOLÓGICOS

JOHNNY ZULUAGA ALVIS (10033648)

Trabajo de grado presentado para optar por el título de Ingeniero en Mecatrónica

Director de proyecto

EDUARDO GIRALDO SUÁREZ (Ph.D en sistemas de Control)

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE TECNOLOGÍA

INGENIERÍA MECATRÓNICA

PEREIRA

2016

AGRADECIMIENTOS

Agradezco infinitamente a Dios por permitirme culminar mis estudios universitarios después de tantos años anhelando este resultado y en vista de tantos tropiezos a lo largo de mi vida.

A mi esposa quien ha sabido tener la paciencia necesaria para soportar los momentos de tensión generados por el estudio y el sacrificio de espacios personales para permitir que mis sueños hagan parte de los suyos.

A mi madre por inculcarme desde niño el interés por la superación a pesar de no contar con recursos suficientes para tal fin y por estar siempre ahí con su apoyo.

A mis profesores quienes compartieron sus conocimientos aún en horarios extra clase importando más el aprendizaje que sus propios intereses personales, con paciencia comprendiendo que no todos los estudiantes tienen los mismos espacios de aprendizaje ni el tiempo para dedicar a tal fin.

CONTENIDO

Tabla de Figuras

INTRODUCCIÓN

El presente trabajo de grado es realizado con el fin de aplicar los conocimientos adquiridos durante toda la carrera para tratar de dar solución a un problema que se presenta en los laboratorios clínicos y en las empresas donde se manipulan materiales de riesgo biológico y/o corrosivo pues por lo general no se dispone de una extensión del cuerpo como un brazo robótico para tal fin y los operarios deben hacerlo por sus propios medios exponiéndose así a los peligros que ello implica. Se pretende realizar el diseño del prototipo de control computarizado de un brazo robótico que se mueva de la mejor manera reduciendo los derrames para manipular tales materiales minimizando la exposición por parte del personal y así reforzar el tema de la salud ocupacional y reducir las enfermedades derivadas de las labores cotidianas.

CAPITULO 1

Este capítulo contiene: Planteamiento, contextualización y delimitación del problema de investigación que corresponde a la definición de problema. Contiene además preguntas o interrogantes de la investigación. Objetivos de la investigación: General y específicos y la Justificación e importancia de la investigación.

1. TÍTULO

Diseño del prototipo de control computarizado de un brazo robótico para manipulación de material y residuos biológicos.

1.1 DEFINICIÓN DEL PROBLEMA

1.1.1 Planteamiento

En muchas áreas de la producción se interactúa con materiales corrosivos y/o contaminantes, más aún en el análisis clínico se generan residuos biológicos que es bien sabido, el nivel de contaminación es elevado teniendo en cuenta la inmensa cantidad de virus, bacterias y enfermedades que existen en la actualidad. Surge, entonces, la necesidad de limitar el contacto directo de las personas con dicho tipo de materiales y conseguir evitar al máximo el riesgo de infección sin que se vea afectado el resultado de la labor realizada por el personal encargado. La idea es que la mecatrónica nos permita dar solución a esta problemática.

1.1.2 Formulación

¿Es la mecatrónica capaz de dar solución con eficiencia a un tema de riesgo biológico tanto en laboratorios como en cualquier otra área de la producción para minimizar el contacto directo y la manipulación de materiales contaminantes o corrosivos?

1.1.3 Sistematización

- ¿Un sistema de control computarizado es tan fácil de manejar por un usuario promedio de forma tal que no se convierta en un problema mayor que el que se desea solucionar?
- ¿Un sistema de control puede llegar a ser confiable desde el punto de vista operativo?
- ¿Puede un sistema de este tipo acoplarse fácilmente a cualquier brazo robótico sin verse afectado el resultado final?

1.2 JUSTIFICACIÓN

En los laboratorios de análisis clínico se estudian muestras biológicas de pacientes que por alguna razón y bajo criterio médico necesitan la realización de determinadas pruebas cuyos resultados servirán a los médicos para empezar un tratamiento. Las muestras tomadas corresponden, en su mayoría a sangre, materia fecal, orinas entre otros. EL nivel de contaminación es elevado en cualquiera de las fases de los procesos de análisis. Los resultados son arrojados por equipos electrónicos y electromecánicos diseñados para análisis específicos pero deben tener unas preparaciones previas tales como centrifugado, mezclas con soluciones reactivas y demás; al finalizar los procesos de análisis, quedan unos desechos de igual o mayor nivel de contaminación; en estas etapas del proceso es donde el riesgo de contaminación e infección aumenta para las personas encargadas de la manipulación directa de dichas muestras y residuos biológicos. Es allí donde nace la idea de minimizar los riesgos de contaminación e infección al diseñar e implementar el prototipo de un sistema de control computarizado para un brazo robótico que permita a los usuarios cambiar la ubicación de las muestras entre equipos de análisis y la manipulación de los residuos finales sin tener contacto directo con los mismos y sin derrames durante la reubicación de los recipientes que contienen dicho material contaminado.

Este sistema de control no estaría limitado a laboratorios clínicos sino que tendría la posibilidad de ser adaptado por el usuario a un espacio determinado y empleando el brazo robótico que más se acomode a sus necesidades.

1.3 OBJETIVOS

1.3.1 Objetivo General

Diseñar un control óptimo operado desde computador para un sistema multivariable de brazo robótico que pueda desplazarse por unas trayectorias óptimas y aplicado a la manipulación de material biológico y/o contaminado para que los usuarios no los manipulen directamente.

1.3.2 Objetivos Específicos

- Dar al prototipo la posibilidad de generar las trayectorias óptimas por donde se desplazará para un mínimo tiempo de transporte y sin derrames.
- Diseñar el controlador tal que se pueda acoplar fácilmente a cualquier espacio dentro del rango de acción del brazo robótico.
- Sintonizar este control multivariable con cinco entradas (cinco servomotores) y tres salidas (ejes X, Y, Z) para que responda de la manera más suave a las perturbaciones y evitar así el derrame de material.
- Diseñar un sistema de control con una interfaz de usuario fácil de interpretar y operar.

CAPITULO 2

Este capítulo contiene las bases teóricas de los temas involucrados en el desarrollo del trabajo así como la definición de algunos términos empleados y la sustentación matemática de las expresiones utilizadas en los cálculos de cinemática inversa y de los controladores.

2 BASES TEORICAS

El usuario, desde la interfaz diseñada en MATLAB para tal motivo, ingresa los datos de la posición en los ejes X, Y y Z a la cual desea que el brazo se dirija, con estos datos se encuentran los ángulos requeridos en cada uno de los servo motores para que la punta llegue a ese lugar. Para tal fin se utiliza la Cinemática Inversa.

2.1 Cinemática Inversa

La cinemática se ocupa de la descripción del movimiento sin tener en cuenta sus causas. El objetivo de la cinemática inversa consiste en encontrar el gesto que deben adoptar las diferentes articulaciones para que el final del sistema articulado llegue a una posición concreta.

En cinemática inversa, la posición deseada y, posiblemente la orientación del efector están dadas por el usuario y los ángulos de la articulación requerida para alcanzar esa configuración se pueden calcular. El problema puede tener cero, una, o más soluciones.

2.1.1 Solución geométrica

En un método geométrico para encontrar la solución de un manipulador, tratamos de descomponer la geometría espacial del brazo en varios problemas de geometría plana. Para muchos manipuladores (especialmente cuando los ángulos son iguales a cero), esto puede hacerse con mucha facilidad. Los ángulos de articulación pueden así resolverse utilizando las herramientas de geometría plana. Inicialmente se empezará con un brazo de tres grados de libertad como se muestra en la figura 1. Como dicho brazo es planar podremos aplicar directamente la geometría plana para encontrar una solución. [1]

En la figura se muestra el triángulo formado por las longitudes L1 y L2y la línea que une el origen con el punto (x , y), considerando el triángulo sólido podremos aplicar la teoría de la ley de Cosenos que es usada para encontrar las partes faltantes de un triángulo oblicuo (no rectángulo) cuando las medidas de dos lados y la medida del ángulo incluido son conocidas o las longitudes de los tres lados son conocidas. En cualquiera de estos casos, es imposible usar la ley de los senos porque no podemos establecer una proporción que pueda resolverse. La ley de cosenos establece que:

$$c^2 = a^2 + b^2 - 2*a*b*\cos C \qquad \textit{ecuación 2.1.1.1}$$

Donde las minúsculas son los catetos y las mayúsculas son los ángulos. Esto se parece al teorema de Pitágoras excepto que para el tercer término y si *C* es un ángulo recto el tercer término es igual 0 porque el coseno de 90° es 0 y se obtiene el teorema de Pitágoras. Así, el teorema de Pitágoras es un caso especial de la ley de los cosenos.

La ley de los cosenos también puede establecerse como

$$b^2 = a^2 + c^2 - 2*a*c*\cos B \qquad \textit{ecuación 2.1.1.2}$$

$$a^2 = b^2 + c^2 - 2*b*c*\cos A \qquad \textit{ecuación 2.1.1.3}$$

Para que el triángulo sólido exista la distancia al punto de destino $\sqrt{x^2 + y^2}$ debe ser menor o igual a la suma de los vínculos L1 y L2. El ángulo *Alfa* puede estar en cualquier cuadrante dependiendo de los signos de **x** y de **y** por ello se debe usar una función Arco tangente de dos argumentos (Atan2).

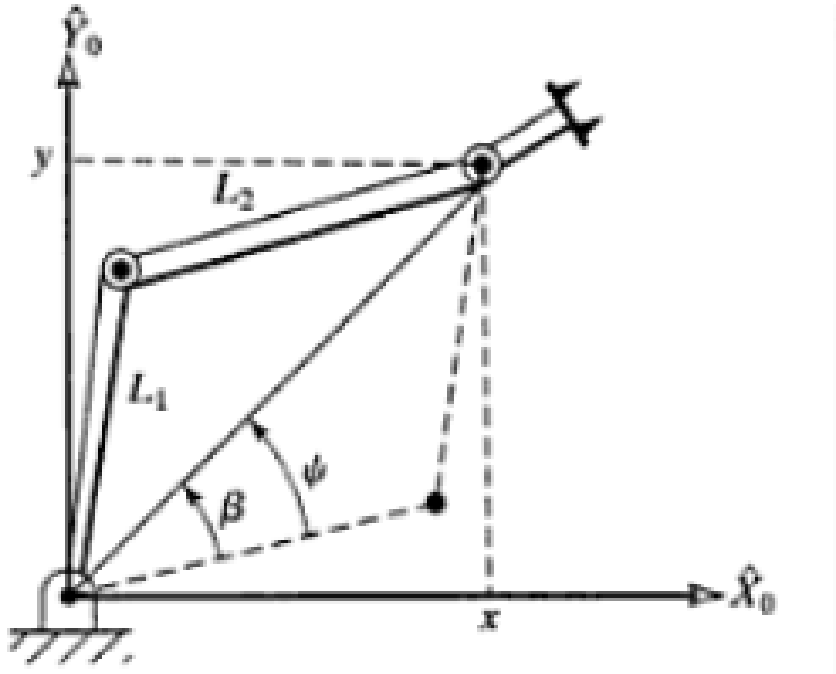

Figura 1, brazo con tres grados de libertad.

Los ángulos a calcular se muestran en la figura 2, adicionalmente se expresan las ecuaciones halladas luego de aplicar el método geométrico con ayuda de la ley de Cosenos.

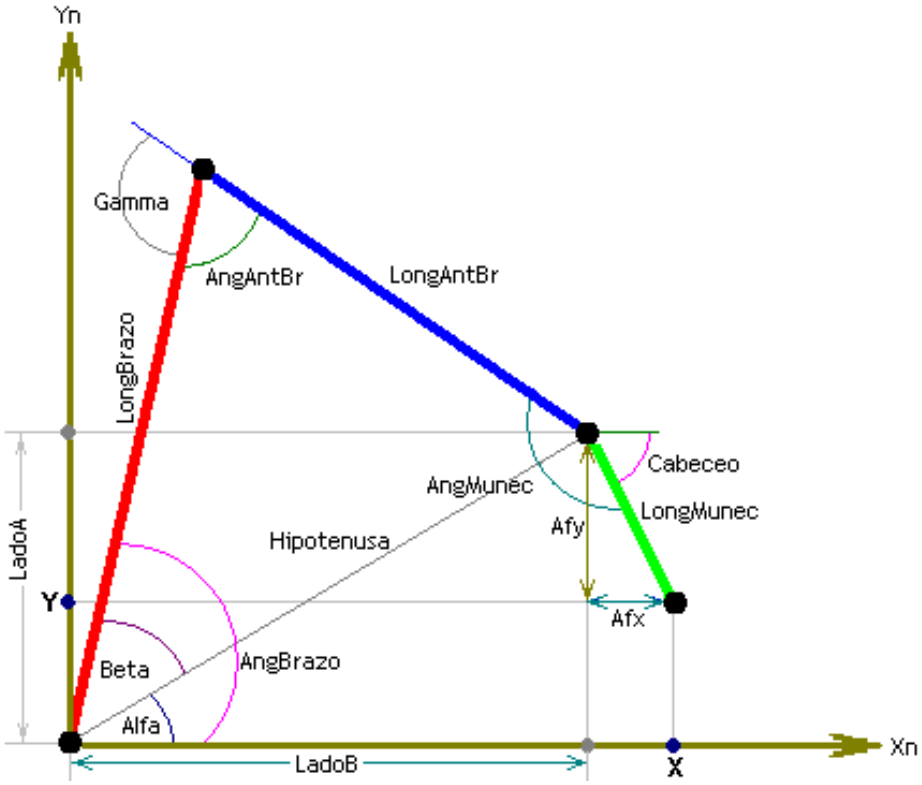

Figura 2, Ubicación de cada ángulo necesario

Afx = Cos (cabeceo)*LongMunec *ecuación 2.1.1.4*

LadoB = X – Afx *ecuación 2.1.1.5*

Afy = Sin(Cabeceo)*LongMunec *ecuación 2.1.1.6*

LadoA = Y – Afy *ecuación 2.1.1.7*

Hipotenusa = sqrt (LadoA + LadoB) *ecuación 2.1.1.8*

Alfa = Atan2(LadoA,LadoB) *ecuación 2.1.1.9*

Beta = Acos((LongBrazo^2 – LongAntBr^2 + Hipotenusa^2) / (2*LongBrazo*Hipotenusa)) *ecuación 2.1.1.10*

AngBrazo = Alfa + Beta *ecuación 2.1.1.11*

Gamma = Acos((LongBrazo^2 +LongAntBr^2 - Hipotenusa^2) / (2* LongBrazo* *LongAntBr)) *ecuación 2.1.1.12*

AngAntBr = - (180 - Gamma) *ecuación 2.1.1.13*

AngMunec = Cabeceo – AngBrazo – AngAnrBr *ecuación 2.1.1.14*

El ángulo de Cabeceo es un valor que se dá para que la punta permanezca constante desde el punto de vista del observador aunque se mueva el brazo a otra posición. El ángulo de muñeca es diferente pero dependiente uno del otro.

El resto de las expresiones se encuentra aplicando el método geométrico con la ayuda del teorema de Cosenos.

Ahora con estos datos es mucho más fácil encontrar las ecuaciones de cinemática inversa para un brazo antropomorfo de 5 grados de libertad (5GDL), en realidad son las mismas pero con algunos ajustes para la nueva dimensión de movimiento. El eje X se mantendrá en la misma posición sobre el plano, el eje Y pasará a convertirse en la profundidad y el eje Z pasa a ser la altura como se muestra en la figura 3.

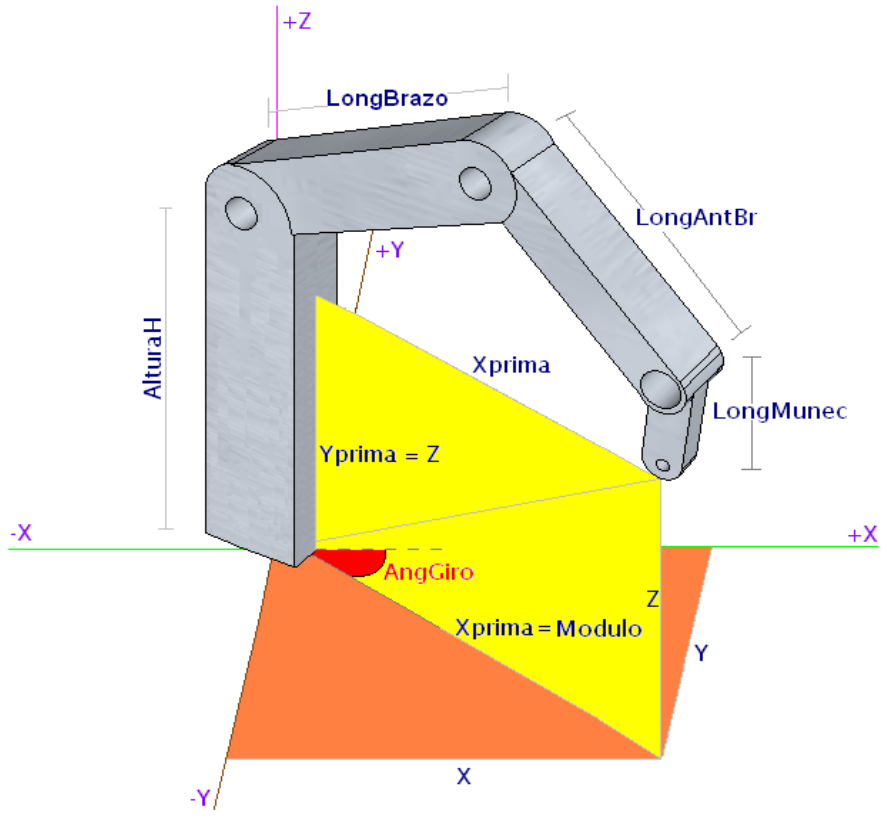

Figura 3, tercer plano de desplazamiento para el brazo robótico de 5GDL

El ángulo de giro se haya con la siguiente expresión:

AngGiro = Atan2(Y,X) *ecuación 2.1.1.15*

Se crean dos variables de ejes ficticios para facilitar el cálculo **Xprima** e **Yprima** que en realidad son los ejes X e Y visto en dos dimensiones para el brazo de 3GDL.

Se halla una variable que se llamará **Módulo**

Módulo = sqrt(X^2 + Y^2) *ecuación 2.1.1.16*

Xprima = Modulo *ecuación 2.1.1.17*
Yprima = Z *ecuación 2.1.1.18*

Después se hace una reconversión de variables, las cuales harán de puente para usar el mismo método de resolución de ángulos expuesto para el brazo de 3GDL. Se añade la variable **AlturaH** que es la distancia entre la base (suelo) y el hombro del brazo. En la figura 4 se muestran las variantes para el brazo robótico de 5GDL.

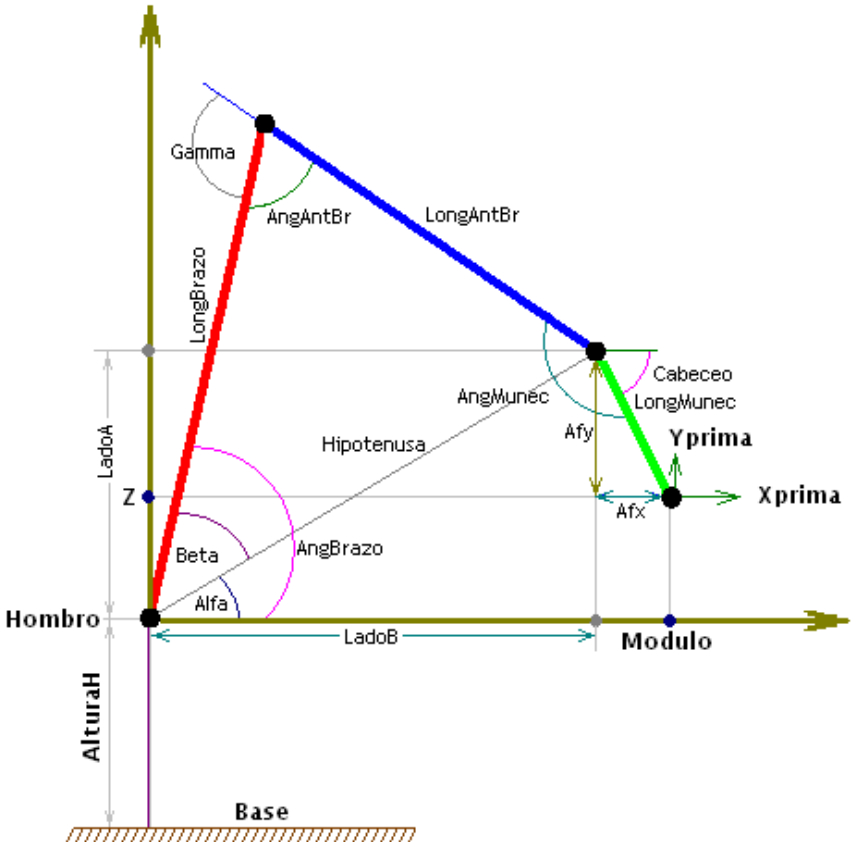

Figura 4, nuevas variantes para el brazo de 5GDL

Las ecuaciones quedan de la siguiente forma:

AngGiro = Atan2(Y,X)

Módulo = sqrt(X^2 + Y^2)

Xprima = Modulo

Yprima = Z

Afx = Cos (cabeceo)*LongMunec *ecuación 2.1.1.19*

LadoB = Xprima – Afx *ecuación 2.1.1.20*

Afy = Sin(Cabeceo)*LongMunec *ecuación 2.1.1.21*

LadoA = Yprima – Afy – AturaH *ecuación 2.1.1.22*

Hipotenusa = sqrt (LadoA^2 + LadoB^2) *ecuación 2.1.1.23*

Alfa = Atan2(LadoA,LadoB) *ecuación 2.1.1.24*

Beta = Acos((LongBrazo^2 – LongAntBr^2 + Hipotenusa^2) / (2*LongBrazo*Hipotenusa)) *ecuación 2.1.1.25*

AngBrazo = Alfa + Beta *ecuación 2.1.1.26*

Gamma = Acos((LongBrazo^2 +LongAntBr^2 - Hipotenusa^2) / (2* LongBrazo* LongAntBr)) *ecuación 2.1.1.27*

AngAntBr = - (180 - Gamma) *ecuación 2.1.1.28*

AngMunec = Cabeceo – AngBrazo – AngAnrBr *ecuación 2.1.1.29*

Tenemos las soluciones para calcular los 4 grados de libertad siendo estos: el ángulo de giro de todo el brazo, ángulo del brazo, ángulo del antebrazo y ángulo de la muñeca (pitch). Si añadimos el giro de la muñeca (éste ángulo no afecta al resto de los ángulos del brazo ó a la posición final de la punta por tanto no afecta a la cinemática inversa) pasamos a tener 5 grados de libertad. La pinza (o terminal) no se cuenta como grado de libertad pero sí se tiene en cuenta como parte de la

longitud de la muñeca a efectos de cálculos; es decir, la longitud total de la muñeca es la suma de la longitud de la muñeca real más la longitud del terminal o pinza. [2]

2.2 CONTROLADORES

Se implementan los controles PID y Modelo de referencia que serán explicados a continuación.

2.2.1 Controladores PID

Los controladores PID son compensadores que permiten incorporar acciones Proporcionales, Integrales y Derivativas sobre la señal de error del sistema. Estos controladores son los más usados actualmente por su versatilidad y facilidad de implementación, se empieza hablando del controlador P (proporcional).

2.2.1.1 Control P (proporcional)

Este tipo de controlador es directamente proporcional a la señal de error, cuando este aumenta, la constante de proporcionalidad Kp aumenta, si la señal de error disminuye la constante Kp también disminuye. Esto es bueno mientras la salida se acerca a cero en su gráfica pero es muy malo cuando esta señal cruza el cero y se hace negativa, con lo cual la oscilación se mantiene y la planta, según sea su naturaleza podría no estabilizarse generando una onda senoidal en la gráfica de la salida tal como lo muestra la figura 5.

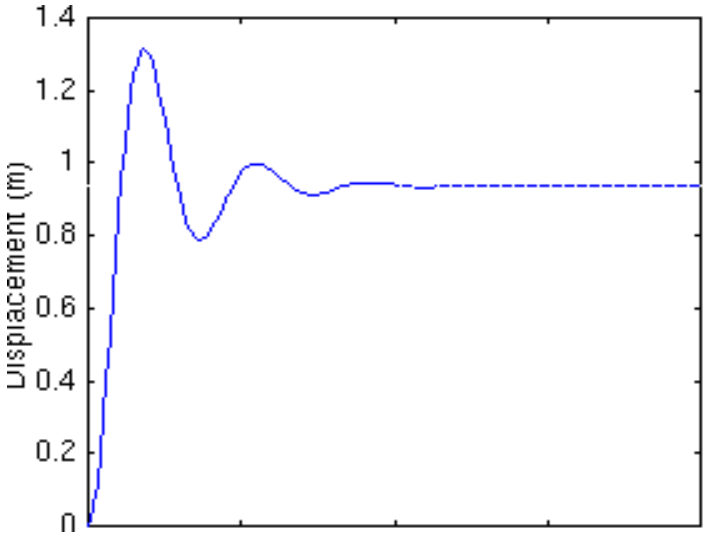

Figura 5, respuesta de una planta con control Proporcional

2.2.1.2 Control Proporcional – Integral

Esta acción elimina el problema del error en estado estacionario frente a perturbaciones de carga constante. Por eso se utiliza para determinar de forma automática el valor correcto de la señal de error. Además se usa para corregir el error en régimen permanente. Este controlador integra las áreas bajo las curvas de la señal de salida minimizándolas y haciendo un poco mas estable al sistema. La respuesta a este controlador se muestra en la figura 6.

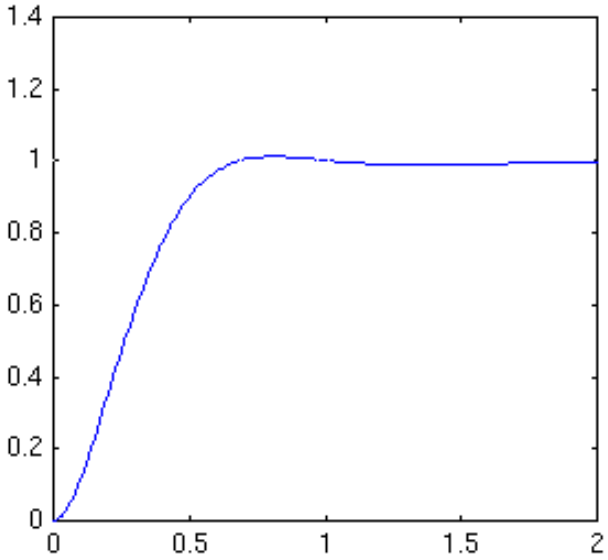

Figura 6, respuesta de la planta a un controlador Proporcional - Integral

2.2.1.3 Controlador Proporcional - Derivativo

Uno de los problemas del controlador PI y que limita su comportamiento es que solo considera los valores del error que han ocurrido en el pasado, es decir, no intenta predecir lo que pasará con la señal en un futuro inmediato. Cuando la oscilación empieza a ser negativa, el Derivativo la detecta de inmediato. La acción derivativa realiza ese tipo de compensación, que se basa en introducir una acción de predicción sobre la señal de error. Una forma sencilla de predecir es extrapolar la curva de error a lo largo de su tangente. La figura 7 muestra la respuesta.

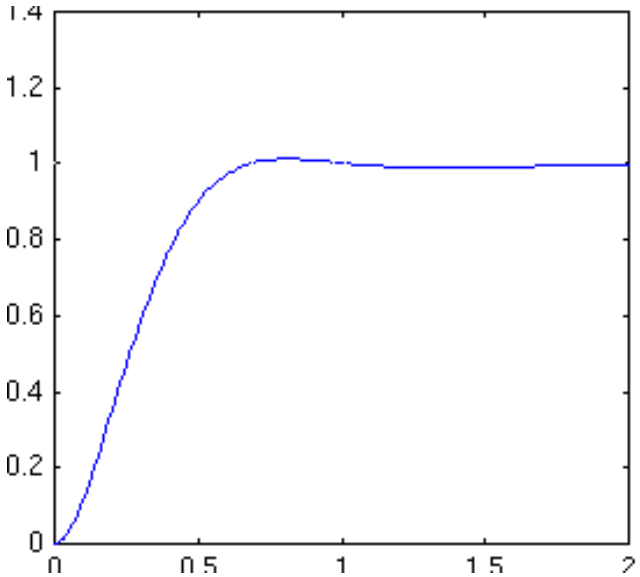

Figura 7, respuesta de la planta al controlador Proporcional – Derivativo

2.2.1.4 Controlador Proporcional - Integral - Derivativo

El controlador PID combina en un único controlador la mejor característica de estabilidad del controlador PD con la ausencia de error en estado estacionario del controlador PI.

La adición de la acción integral a un controlador PD es esencialmente lo mismo que añadir dicha acción a un controlador P.

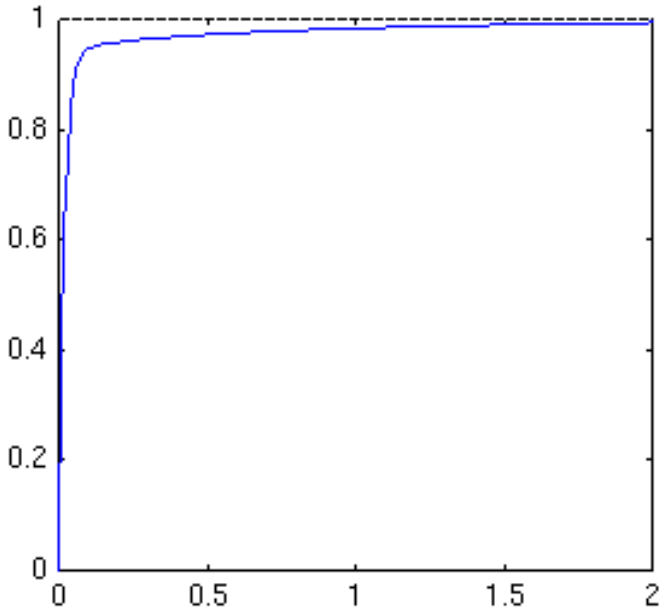

Figura 8, respuesta a un controlador Proporcional – Integral - Derivativo

Se puede concluir que los controladores anteriores tienen las siguientes funciones:

- un control proporcional para mejorar el tiempo de elevación.
- un control derivativo para mejorar el sobrepico.
- un control integral elimina el error de estado estacionario. [3]

La figura 9 muestra el diagrama de bloques del control PID y la planta

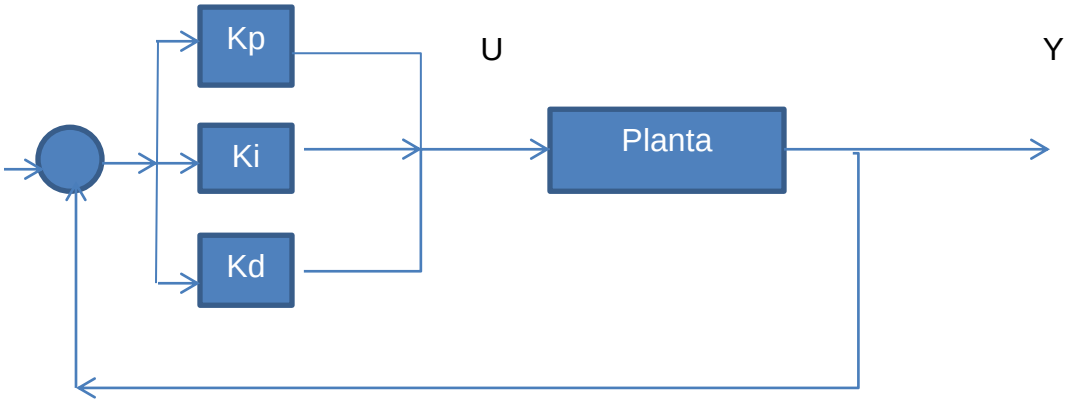

Figura 9, diagrama de flujo del controlador PID.

Donde el controlador completo está dado por la siguiente expresión, siendo ***e*** el error:

$$U(t) = Kp*e(t) + Ki*\int e(t)dt + Kd*\frac{de}{dt} \qquad \textit{ecuación 2.2.1.4.1}$$

2.2.2 Control por Modelo de Referencia

Los MROC (Controladores Óptimos por Modelo de Referencia), intentan alcanzar un comportamiento en bucle cerrado deseado que viene especificado por un modelo de referencia. Las ventajas del MROC pasan por una rápida adaptación y la posibilidad de utilizar formulaciones que garanticen estabilidad. Sin embargo, la capacidad de adaptación de estas estrategias dependen en gran medida de la riqueza dinámica de la señal de control. Se debe partir de un conjunto de especificaciones deseadas de bucle cerrado que se expresan mediante el modelo de referencia. El controlador ajustable deberá adaptar sus parámetros para que el modelo de bucle cerrado del conjunto coincida o se acerque lo más posible al modelo de referencia. No es realista escoger un modelo de referencia con una dinámica muy rápida en comparación con la de la planta en bucle abierto.

Para diseñar un MROC se ha de definir el modelo de referencia, el controlador y la ley de adaptación. Para el controlador primario se puede pensar en casi cualquier estructura de control lineal, incluyendo los populares PI, PID, etc. Se deben cumplir sin embargo varios requisitos, entre ellos que la señal de control debe ser una función lineal de los parámetros. También (suponiéndose fijado el modelo) se debe escoger un controlador ajustable que permita reproducir el modelo. [4]

Las características mas relevantes para el diseño de un Modelo de referencia son que tenga ganancia unitaria y que sea de orden inferior al de la planta.

La figura 10 representa una configuración para los Modelos de Referencia aunque no es la única que existe.

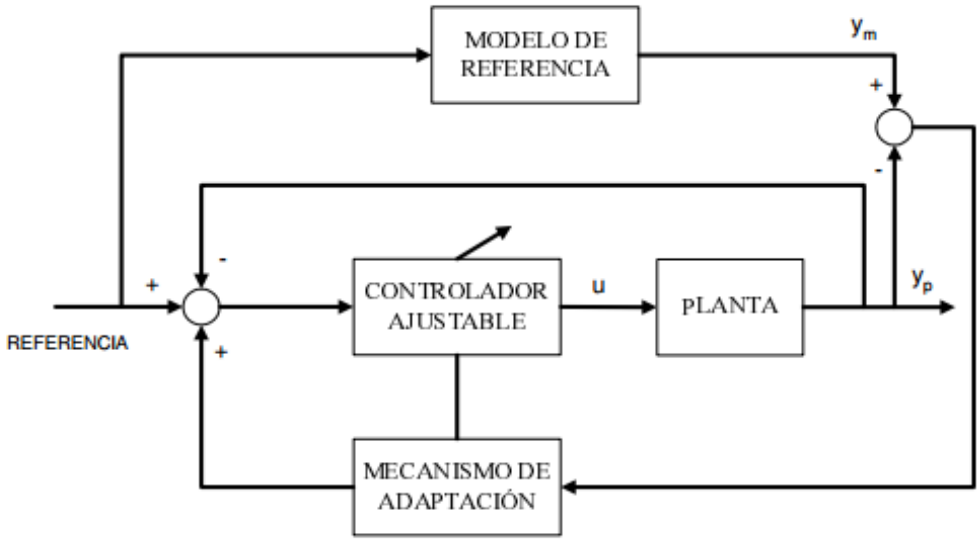

Figura 10, posible configuración para Modelos de Referencia.

El controlador primario o ajustable puede tener en principio cualquiera de las configuraciones conocidas para el diseño de controladores lineales. Sin embargo, debe cumplir la condición de que sea posible que el conjunto del proceso y el controlador puedan reproducir al modelo de referencia. Este requisito, supone restricciones sobre el orden y la estructura del controlador. Por otro lado para que pueda aplicarse una adaptación directa, la señal de control debe ser una función lineal de los parámetros. [5]

El modelo de referencia en el plano Z está dado de la siguiente forma:

Modelo de referencia = $\dfrac{b_0}{z+ a_1}$ *ecuación 2.2.2.1*

Aplicando el teorema del valor final obtenemos:

$$\lim_{z\to 1}(1 - z^{-1})\left(\frac{b_0}{z + a_1}\right) \quad * \quad \frac{1}{(1 - z^{-1})}$$

Con lo cual se obtiene:

Modelo de referencia = $\dfrac{b_0}{1+ a_1}$ *ecuación 2.2.2.2*

Para garantizar la ganancia unitaria tenemos que $b_0 = 1 + a_1$. Donde a_1 es el polo discreto.

En tiempo continuo, el modelo de referencia está dado por la expresión:

Modelo de referencia = $\frac{1}{\tau s+1}$ *ecuación 2.2.2.3*

Se factoriza τ y se obtiene:

Modelo de referencia = $\frac{1/\tau}{s+ 1/\tau}$ *ecuación 2.2.2.4*

Donde:

$1/\tau$ = *a*

a = es el Polo deseado

τ = un cuarto del tiempo de establecimiento, es decir el tiempo que se demora la salida en estabilizarse ($4/ts$); valor que será modificado a gusto para esta aplicación.

Se tiene que *a* = $1/\tau$; reemplazando el valor de τ ($4/ts$) se tiene que *a* = $4/ts$; siendo:

Ts = el tiempo de establecimiento especificado por el usuario.

Por lo tanto el Polo deseado es igual a $4/ts$.

Retomando la ecuación 2.2.2.1 y aplicando operador de desplazamiento (que se explica en el apartado 2.2.3):

Y[k] = $\frac{b_0 * q^{-1}}{1 + a_{1*} q^{-1}}$ *ecuación 2.2.2.5*

Pasándolo a ecuaciones en diferencias queda de la siguiente forma:

Y[k] = $a_1 * Y[k-1] + b_0 * r[k-1]$ *ecuación 2.2.2.6*

El polo deseado en el plano Z esta dado por la expresión:

Zd = $e^{-a*h} = a_1$

Reemplazando el valor de a se tiene que:

Zd = $e^{\frac{-1}{\tau}*h}$ = a_1

Se reemplazan los parámetros de los subíndices por los encontrados anteriormente, el equivalente del operador de desplazamiento y el valor de τ:

Y[k] = $e^{\frac{4}{ts}*h}$*Y[k-1] + (1 - $e^{\frac{4}{ts}*h}$)*r[k-1] *ecuación 2.2.2.7*

Esta es la ecuación que se implementa en el software para el control por Modelo de Referencia.

2.2.3 Operador Desplazamiento

E1 uso del operador diferencial p = d/dx es adecuado para el trabajo con ecuaciones lineales diferenciales a coeficientes constantes. Un operador análogo se definiría para sistemas expresados en ecuaciones en diferencias lineales a coeficientes constantes. En la definición de este operador los sistemas son vistos como operadores que relacionan señales de entrada con señales de salida.
Es necesario especificar el rango de validez del operador es decir definir la clase de señal de entrada y qué tipo de actuación tiene el operador sobre las señales. Para los cálculos con este operador todas las señales son consideradas como secuencias doblemente infinitas, debe ir desde −∞ a + ∞. Por conveniencia el per´ıodo de muestreo es elegido como la unidad de tiempo T = 1.[6]

2.2.3.1 Operador Retardo

La inversa del operador desplazamiento hacia adelante es llamado operador desplazamiento hacia atrás y su notación es q^{-1}. Resulta:

$$q^{-1} f_k = f_{k-1}$$

Notemos la importancia de definir el rango del operador para secuencias doblemente infinitas dado que de otra manera el operador desplazamiento hacia atrás no podría existir. En problemas relacionados con la ecuación característica

del sistema, como estabilidad y orden del sistema es mejor utilizar el operador desplazamiento hacia adelante. En cambio en problemas relacionados con la causalidad es conveniente usar el operador desplazamiento hacia atrás.
El cálculo con estos operadores da una compacta descripción de los sistemas y hace muy fácil relacionar variables ya que el manejo de ecuaciones en diferencias se reduce a un problema puramente algebraico. En muchos textos se usa z como el operador desplazamiento al igual que la variable compleja de la transformada en Z. Esta diferenciación es la misma que se hace entre la variable compleja s de la transformada de Laplace y el operador diferencial p = d/dt.

Este operador desplazamiento es usado para simplificar el manejo de las ecuaciones en diferencias de un alto orden. Sea la ecuación expresada en función del operador de retardo: [7]

$$y_{k+n_a} + a_1 y_{k+n_a-1} + \cdots + a_{n_a} y_k = b_0 u_{k+n_b} + \cdots + b_{n_b} u_k$$

$$n_a > n_b$$

$$(q^{n_a} + a_1 q^{n_a-1} + \cdots + a_{n_a}) y_k = (b_0 q^{n_b} + b_1 q^{n_b-1} + \cdots + b_{n_b}) u_k$$

$$A(q) = q^{n_a} + a_1 q^{n_a-1} + \cdots + a_{n_a}$$

$$B(q) = b_0 q^{n_b} + b_1 q^{n_b-1} + \cdots + b_{n_b}$$

$$A(q) y_k = B(q) u_k$$

2.2.4 Retenedor de orden Cero

En un muestreador ideal, un interruptor se cierra cada período de muestreo T para admitir una señal de entrada. Un muestreador convierte una señal de tiempo continuo en un tren de pulsos que se presenta en los instantes de muestreo t=0,T,2T... como se muestra en la figura 11.

La retención de datos es un proceso de generación de una señal de tiempo continuo h(t) a partir de una secuencia en tiempo discreto x(kT). Un circuito de retención convierte la señal muestreada en una señal de tiempo continuo, que reproduce aproximadamente la señal aplicada al muestreador. El circuito de retención más simple es el Retenedor de Orden Cero (ROC), este circuito retiene la amplitud de la muestra en un instante de muestreo al siguiente. La función de transferencia del ROC (ZOH) es: [8]

$$ZOH = \frac{1 - e^{-Ts}}{s} \qquad \textit{ecuación 2.2.4.1}$$

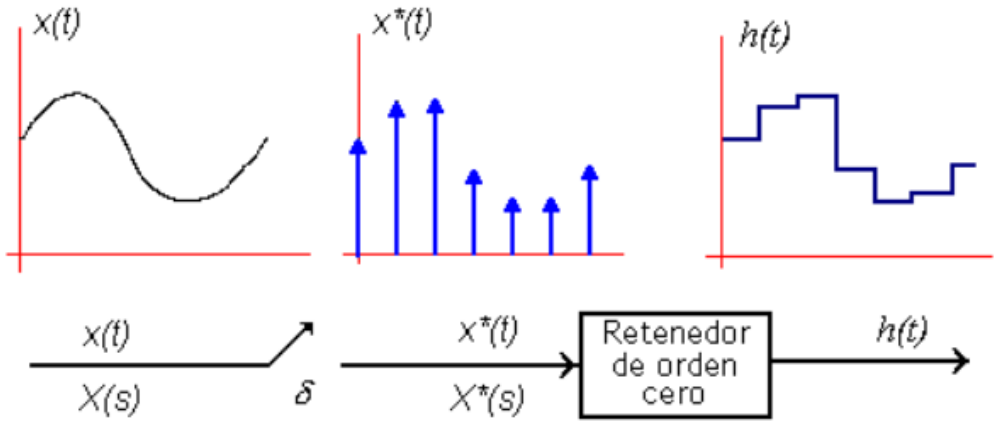

Figura 11, Señales muestreadas

Para discretizar la matriz de estados se tienen las expresiones:

$X[k+1] = \phi X[k] + \Gamma u[k]$ *ecuación 2.2.4.2*

$Y[k] = CX[k] + Du[k]$ *ecuación 2.2.4.3*

Donde:

$$\phi = I + h*A + \frac{h^2*A^2}{2_¡} + \frac{h^3*A^3}{3_¡} \ldots \qquad \textit{ecuación 2.2.4.4}$$

$$\Gamma = \left(I*h + \frac{h^2*A^1}{2_¡} + \frac{h^3*A^2}{3_¡} \ldots\right) *B \qquad \textit{ecuación 2.2.4.5}$$

siendo:

A = matriz de estados del sistema

B = matriz de entradas

h = tiempo de muestreo

I = matriz identidad

Para hallar al función de transferencia se obtienen las ecuaciones para X1[k+1], X2[k+2], X3[k+3], X4[k+4] y Y[k] a partir de las matrices que se acaban de obtener, se despejan las “X[k+n]” luego de aplicar el operador de desplazamiento “q” que nos deja cada variable de la siguiente forma: X1[k+1] = qX1[k] , X2[k+2] = qX2[k], X3[k+3] = qX3[k] y X4[k+4] = qX4[k]. Se reemplazan en las demás ecuaciones hasta lograr obtener la expresión en la ecuación de salida Y[k] y por consiguiente la función de transferencia en términos de “q”.

2.2.5 Reducción del orden de un sistema. Teoría de los polos dominantes

Dado que el valor de las raíces reales o el de las partes reales de las raíces imaginarias dan la estabilidad de los sistemas y como el valor tiene que ser negativo para que los sistemas sean estables, se tiene que cuando mayor sea su valor absoluto, el sistema es tanto mas estable dado que los exponenciales decrecen mas rápidamente. Los polos causantes de la mayor estabilidad de los sistemas, conocidos como polos dominantes, son los que dan lugar a exponenciales que decrecen más lentamente y son los polos mas cercanos el eje imaginario. La existencia de estos, da la posibilidad de reducir el orden de un sistema mediante el estudio del mapa de polos y ceros sin mas que tener en cuenta la posición de estos, además el polo más alejado debe estar a una distancia superior a 5 veces la presente entre los polos dominantes. La mayoría de veces los sistemas de orden superior admiten dos polos dominantes por lo que su estudio queda reducido al de los sistemas de segundo orden. [9]

Para reducir dicho polo basta con dividir el numerador y el denominador de la función de transferencia entre el polo más pequeño del sistema.

2.2.6 Control óptimo

Cuando se habla de la solución óptima de un problema, intuitivamente se piensa en que esta es “la mejor solución”, es decir “insuperable”. Sin embargo, como muchos otros adjetivos, la palabra optimo tiene un alto grado de subjetividad. Efectivamente, un pésimo control desde el punto de vista del comportamiento dinámico podría ser óptimo desde el punto de vista económico y viceversa. Luego, para calificar la bondad de un control (en particular para poder decir que es óptimo) es necesario asociarlo a un “índice” de performance. En términos de control diremos que un control es óptimo si minimiza un funcional de costo en el

que claramente se manifiesta un compromiso entre distintas especificaciones y restricciones. [9]

2.2.7 Teoría de los Polos Dominantes

Los modelos de las plantas, en la práctica, suelen superar a los sistemas de segundo orden. Sin embargo, la influencia de los polos de la cadena cerrada no son todos de igual importancia. Aquellos que están más cerca del semiplano positivo son más lentos en su evolución temporal que otros orientados hacia el -∞ del semiplano negativo.

A los polos más próximos al semiplano positivo se les llama dominantes y a los otros polos insignificantes.
La regla práctica de clasificación de unos sobre otros depende de si el polo dominante es complejo conjugado o de primer orden. Si es complejo conjugado, debe de haber una distancia sobre el eje real de 5 a 10 veces el valor de la constante de amortiguamiento, entre el polo dominante y el resto de los polos. Para los polos dominantes de primer orden, el valor de la constante del polo dominante debe de ser al menos 5 a 10 veces mayor que el de los polos insignificantes.

La reducción del orden del sistema simplifica tanto la fase de análisis como de diseño. En la práctica, se emplean las características dinámicas de los sistemas de primer o de segundo orden para definir los requisitos de diseño, aunque el sistema sea de mayor orden. Desde luego no tiene sentido hablar del coeficiente de amortiguamiento o de la frecuencia natural de un sistema si es de tercer, cuarto o de orden superior. El comportamiento de los sistemas de orden elevado puede aproximarse por otro equivalente de segundo o primer orden. La respuesta del equivalente no es idéntica, no tiene tantos matices, pero se aproximan y se hace factible aplicar reglas sencillas tanto para la predicción de su comportamiento como para el diseño. [10]

Hay dos formas de reducir el orden de un sistema:

1. Por aplicación de la teoría de polos dominantes. Los polos ubicados en la región de insignificantes pueden ser eliminados.
2. Mediante la cancelación entre el efecto de un polo y un cero próximo entre sí.

En este proyecto se aplicará el método 1 ya que es la opción que más se ajusta según la función de transferencia encontrada. Para eliminar el polo más insignificante basta con dividir el numerador y el denominador de la función de transferencia entre dicho polo con lo cual el orden ya queda reducido.

CAPÍTULO 3

En el presente capítulo se expresan los resultados de la implementación y utilización de la teoría del capítulo anterior para el desarrollo final del presente proyecto.

3 DESARROLLO DEL PROYECTO

El desarrollo del actual proyecto de grado trae consigo una serie de cálculos necesarios para sintonizar correctamente los controladores, tanto el PID como el Modelo de referencia. Inicialmente se encuentra la expresión para cada controlador de forma independiente que al final se traducirá en el controlador P+I+D (PID).

3.1 Expresiones del controlador PID en cada una de sus partes

3.1.1 Controlador D (Derivativo)

Se utiliza el método de aproximaciones hacia atrás que se basa en tomar el valor del error en el momento actual y restar el valor del error en el momento anterior dividiéndolo entre el tiempo de muestreo, así:

$$\frac{de}{dx} = \frac{e[k]-e[k-1]}{h} \qquad \textit{ecuación 3.1.1.1}$$

Ahora se utiliza el operador de desplazamiento (q^{-1}), que va relacionado al instante anterior:

$$\frac{e[k]-e[k]*q^{-1}}{h} = \frac{\left(1-q^{-1}\right)*e[k]}{h} \qquad \textit{ecuación 3.1.1.2}$$

La señal de control derivativa (Ud[k]) queda de la siguiente forma:

$$\text{Ud[k]} = \text{Kd} * \frac{\left(1-q^{-1}\right)*e[k]}{h} \qquad \textit{ecuación 3.1.1.3}$$

Donde:

Kd = constante de derivación

k = instante de muestreo

$\boldsymbol{q^{-1}}$ = operador de desplazamiento

h = tiempo de muestreo

3.1.2 Control I (Integral)

Para hallar el controlador Integral, se hace de forma numérica, es decir hallando el área bajo la curva del error anterior y del actual, se sabe que el área de un rectángulo es el producto de la base por la altura del mismo, así:

$$ei[k] = h * e[k] \qquad \textit{ecuación 3.1.2.1}$$

dónde:

ei[k] = error integral en un lapso de tiempo ó suma de los errores anteriores

e[k] = es el error en el instante k

ahora, si se busca integrar varios rectángulos (figura 12) lo que se hace es la suma de todas las áreas y este resultado será el valor del error Integral, así:

$$ei[k] = h * e[k] + ei[k-1] \qquad \textit{ecuación 3.1.2.2}$$

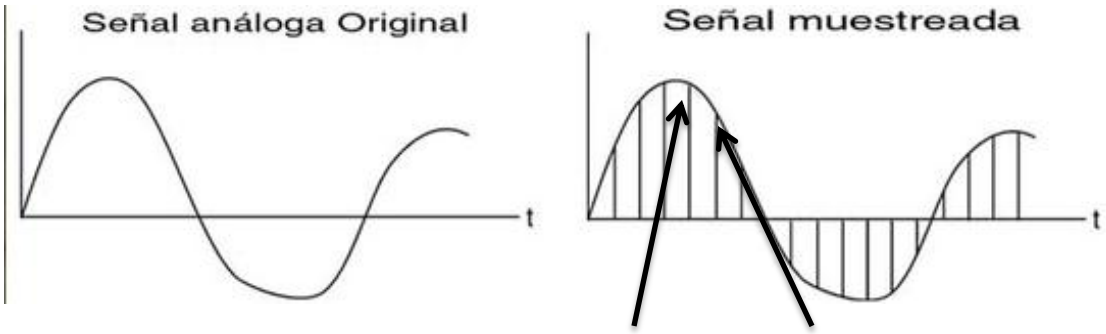

Figura 12, señal muestreada, instante e[k-1] , instante e[k]

Aplicando el operador de desplazamiento y despejando ei[k], la expresión queda de la siguiente forma:

$$ei[k] = \frac{h}{1-q^{-1}} * e[k] \qquad \textit{ecuación 3.1.2.3}$$

la señal de control integral (Ui[k]) queda de la siguiente forma:

$$Ui[k] = Ki * \frac{h}{1-q^{-1}} * e[k]$$ *ecuación 3.1.2.4*

3.1.3 Controlador P (Proporcional)

Para el controlador derivativo solo basta con encontrar el valor de la constante de proporcionalidad KP y aplicarlo a la siguiente expresión:

$$Up[k] = Kp * e[k]$$ *ecuación 3.1.3.1*

Con todo lo anteriormente explicado se encuentra que la señal de control PID queda:

$$U[k] = Up[k] + Ui[k] + Ud[k]$$ *ecuación 3.1.3.2*

Realizando la correspondiente suma y factorizando

$$U[k] = \left(Kp + \frac{Ki*h}{1-q^{-1}} + Kd * \frac{(1-q^{-1})}{h}\right) * e[k]$$ *ecuación 3.1.3.3*

3.2 Función de transferencia de un controlador PID

Se realiza la correspondiente suma de fracciones y se reorganiza como numerador y denominador. La Función de transferencia para un controlador PID queda así:

$$C(q^{-1}) = \frac{\left(Kp+Ki*h+\frac{Kd}{h}\right)+\left(-Kp-\frac{2*Kd}{h}\right)*q^{-1}+\frac{Kd}{h}*q^{-2}}{1-q^{-1}}$$ *ecuación 3.2.1*

La señal de control que sale del PID viene con la señal de error por lo que la expresión de dicha señal queda:

$$U[k] = C(q^{-1}) * e[k]$$ *ecuación 3.2.2*

Para implementarla en un sistema computacional o en un microcontrolador se debe pasar a ecuación en diferencias, pero para que sea más fácil trabajar con ella se separa por bloques y queda como se muestra a continuación:

$$C1 = \left(Kp + Ki * h + \frac{Kd}{h}\right)$$

$$C2 = \left(-Kp - \frac{2*Kd}{h}\right)$$

$$C3 = \frac{Kd}{h}$$

$$U[k] = U[k-1] + C1 *e[k] + C2*e[k-1]+ C3*e[k-2] \quad \textit{ecuación 3.2.3}$$

3.3 Sintonización de los parámetros del controlador

Para encontrar los valores correctos de los parámetros con los cuales operar el controlador se aplica la fórmula para ***sistemas en lazo cerrado*.**

$$Y[z] = \frac{C(z)*H(z)}{1+C(z)*H(z)} \quad \textit{ecuación 3.3.1}$$

Pero antes de esto se debe encontrar la función de transferencia de la planta para lo cual se encuentran las ecuaciones de espacio de estados. Los sistemas complejos pueden tener entradas y salidas múltiples y pueden variar en el tiempo. El análisis en espacio de estados, se concentra en tres tipos de variables involucradas en el modelado de sistemas dinámicos: variables de entrada, variables de salida y variables de estado. [9] Como se muestra en las siguientes ecuaciones:

$$\dot{X}(s)=A*X(s)+B*U(s) \text{ [ecuación de estados]} \quad \textit{ecuación 3.3.2}$$

$$Y(s)=C*X(s)+D*U(s) \text{ [ecuación de salida]} \quad \textit{ecuación 3.3.3}$$

Siendo A la matriz de estados, B la de entradas, C la de salidas, D la de perturbaciones y U las entradas.

Para determinar el orden de la planta utilizada, se emplea el software Matlab y su comando “Linmode” dado que al importar el modelo desde el software de diseño “Inventor”, este trae consigo todos los parámetros necesarios para su modelamiento y este simula los movimientos que generaría el sistema real. En vista de la complejidad de los cálculos, todas las operaciones se realizaron en el

programa nombrado y tomando cada grado de libertad como un bloque independiente ya que para cada motor se genera una señal de control independiente de las demás entregadas por la cinemática inversa. Tomando cada bloque se encuentra que son de orden 4. Los valores encontrados por medio de Matlab fueron:

A =

	x1	x2	x3	x4
x1	-1.455e+06	-9964	0	0
x2	8487	-1.087	0	0
x3	0	1	0	0
x4	0	0	0	0

B =

	u1
x1	3.636e+05
x2	8.487e-06
x3	0
x4	0

C =

	x1	x2	x3	x4
y1	0	0	1	0

D =

	u1
y1	0

De la ecuación de estados 3.3.2 se despeja la X(s), se agrupa y se reemplaza en la ecuación de salida 3.3.3, obteniendo la siguiente expresión:

G(s) = C[((S*I)-A)^-1 * B] *ecuación 3.3.4*

Con la anterior ecuación y las matrices encontradas anteriormente se puede encontrar la función de transferencia en tiempo continuo:

$$\frac{3.086e9\ s}{s\text{^}4 + 1.455e06\ s\text{^}3 + 8.614e07\ s\text{^}2}$$ *ecuación 3.3.5*

De la anterior expresión se puede factorizar una “s” quedando el sistema de tercer orden así:

$$\frac{3.086e9}{s\text{^}3 + 1.455e06\ s\text{^}2 + 8.614e07\ s}$$ *ecuación 3.3.6*

Dicho sistema se puede aproximar a uno de segundo orden aplicando la teoría de polos dominantes para lo cual se debe encontrar el polo más alejado que tiene el sistema.

Aplicando Ruffini se encuentra la solución al sistema de tercer orden pero en vista de los valores tan altos de los coeficientes se hace demasiado extenso y demorado el cálculo por lo cual se aplica comando “pzmap” de matlab para la localización de los polos y los ceros del sistema ó simplemente una calculadora que solucione sistemas de este tipo encontrando los siguientes resultados:

p =

1.0e+06 *

0

-1.454487309202461

-0.000059226038488

z =

Empty matrix: 0-by-1

Donde:

p = polos

z = ceros

Se puede observar que el polo más alejado es de valor -1.454 e6, este valor se utiliza para aplicar la reducción por polos dominantes que se explicó en la sección 2.2.5. La aproximación a sistema de segundo orden queda:

$$\frac{2120}{s\text{^}2 + 59.22\ s}$$ *ecuación 3.3.7*

Al graficar ambas funciones de transferencia se encuentra que la respuesta a una entrada de tipo escalón es casi idéntica utilizando la función de tercer orden ó la de segundo orden, lo cual significa que es correcto usar el método de aproximación por polos dominantes para trabajar con un sistema reducido solo para facilitar las operaciones sin afectar su comportamiento. La figura 13 muestra lo indicado.

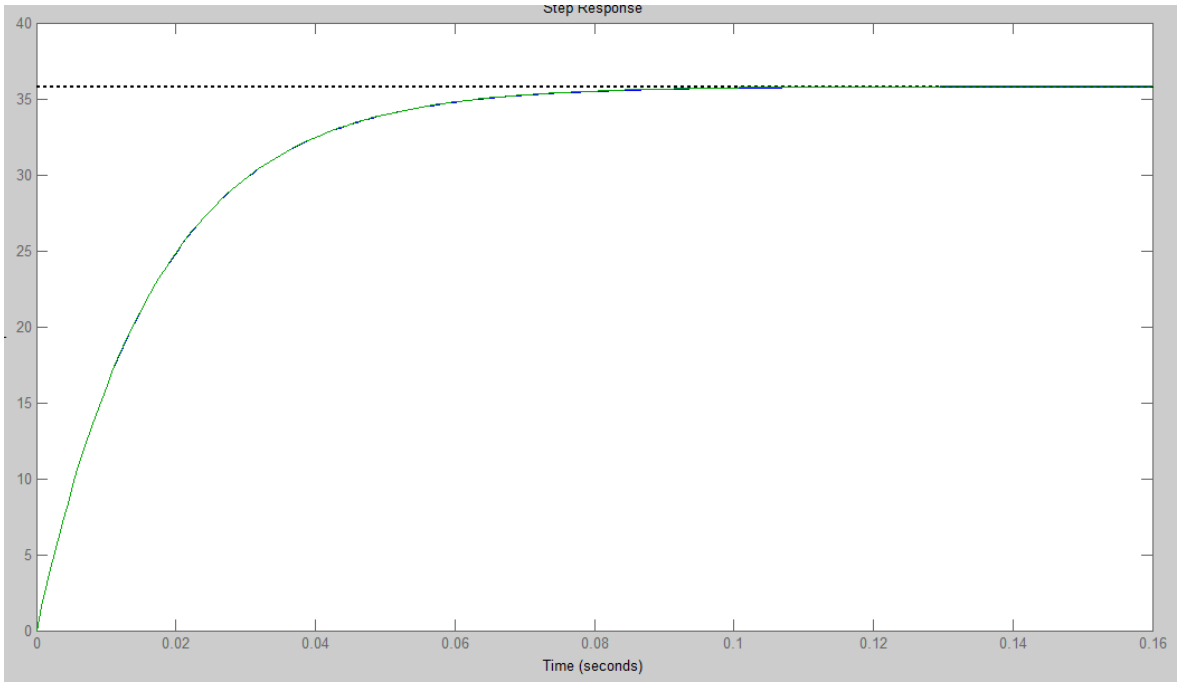

Figura 13, respuesta de las funciones de transferencia de segundo y tercer orden

Un acercamiento a un punto aleatorio en cualquier posición de la gráfica ilustra mucho mejor la existencia de las dos curvas, figura 14.

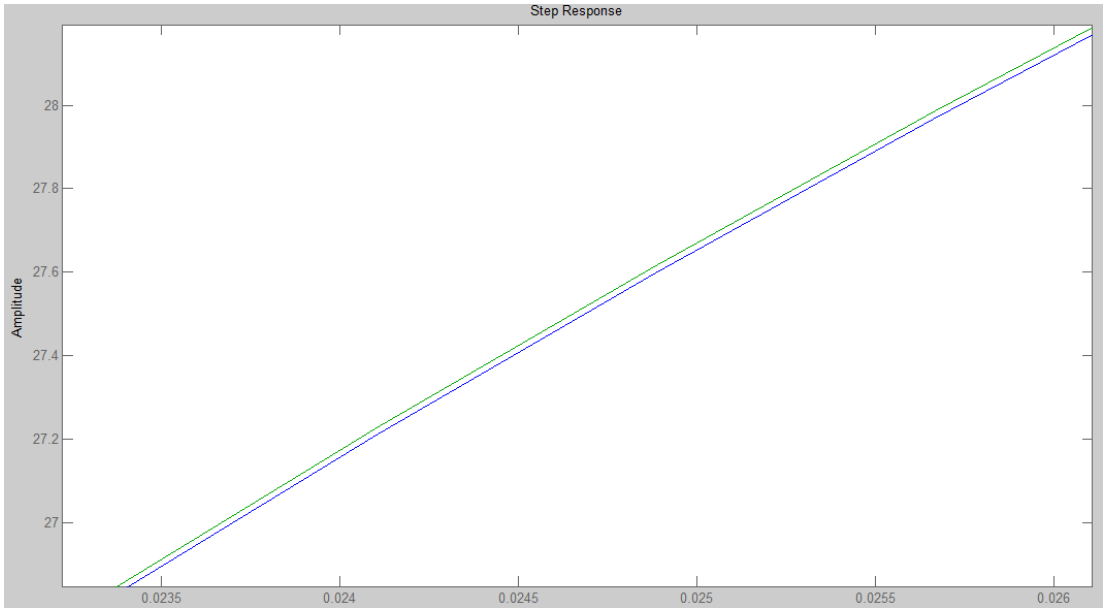

Figura 14, respuesta de las funciones de transferencia de segundo y tercer orden

Ahora se encuentra la función de transferencia a partir de las matrices de estado, de entrada y de salida aplicando *retenedor de orden cero* como se explicó en la sección 2.2.4 y para un tiempo de muestreo "h" de 0.1 obteniendo como resultado:

$$\frac{2.977\ z + 0.5933}{z\text{^}2 - 1.003\ z + 0.00268} \qquad \textit{ecuación 3.3.8}$$

Para sistemas de orden 2, necesitamos un controlador PID de tipo 2 que está dado de la forma:

$$C(q^{-1}) = \frac{C1 + C2 * q^{-1} + C3 * q^{-2}}{(1 + C4 * q^{-1}) * (1 - q^{-1})} \qquad \textit{ecuación 3.3.9}$$

Se vuelve a cambiar el factor de desplazamiento a su equivalente en Z, recordando que el factor de desplazamiento es una variable algebraica y Z es una variable compleja.

$$C(z) = \frac{C1z^2 + C2*z + C3}{(z+C4)*(z-1)}$$ *ecuación 3.3.10*

Se aplica la fórmula para sistemas en lazo cerrado (ecuación 3.3.1) obteniendo el siguiente denominador que también es el Polinomio Característico:

Pc = z^4 **+** z^3*(2.977*C1+C4 + 0.003) **+** z^2*(2.997*C2 + 0.5933*C1 – C4 +1.003*C4 – 1.00032) **+** z *(2.977*C3 + 05933*C2 + 1.003*C4 + 0.00268*C4 – 0.00268) **+** (0.5933*C3 – 0.00268*C4)

ecuación 3.3.11

Ahora se indican los polos deseados, para este caso se ubicarán en un valor de 0.2 para los cuatro polos así:

$(z - 0.2)^4$ **= Pd =** z^4 - 0.8*z^3 + 0.24*z^2 – 0.032*z^1 + 0.0016

ecuación 3.3.12

Se compara término a término entre el polinomio característico y el polinomio deseado (Pc y Pd) igualando los coeficientes del Pd con los de Pc en cada potencia igual de Z para encontrar la matriz de Sylvester así:

Para los coeficientes de z^3 de cada polinomio

(2.977*C1 + C4 + 0.003) = -0.8

Para los coeficientes de z^2 de cada polinomio

(0.5933*C1 + 2.977*C2 + 0.003*C4) = 1.20032

Para los coeficientes de z^1 de cada polinomio

(0.5933*C2 + 2.977*C3 + 1.00568*C4) = -0.02932

Para los términos independientes de cada polinomio

(.59933*C3 – 0.00368*C4) = 0.0016

Las matrices quedan de la forma:

M *X = B

Donde:

M = matriz de Sylvester

X = las incógnitas C1, C2, C3 y C4

B = son los valores conocidos de las ecuaciones

M =

2.9770	0	0	1.0000
0.5933	2.9770	0	0.0030
0	0.5933	2.9770	1.0057
0	0	0.5993	-0.0027

X =

C1

C2

C3

C4

B =

-0.8030

1.2003

-0.0029

16.0000

Al solucionar el sistema de 4 ecuaciones por 4 incógnitas, los resultados son:

C1 = 25.0475

C2 = -4.5127

C3 = 26.3595

C4 = -753695

Estos son los valores que se utilizan para la implementación de cada controlador PID de tipo 2 utilizados en el presente proyecto dado que la cinemática inversa entrega valores independientes para cada grado de libertad y estos se calculan de la misma forma.

3.4 Control por modelo de referencia

La ecuación en diferencias aplicada al proyecto en cada uno de sus grados de libertad tiene la forma de la ecuación 2.2.2.7, un ejemplo de la implementación de la misma es la siguiente:

cintura = ((exp(poten)) * cintura) + ((1-exp(poten)) * AngGiro)

ecuación 3.4.1

El ángulo de giro es el valor que la cinemática inversa calculó y al que el sistema debe llegar, se calcula la potencia de la exponencial aparte pues toma el valor del tiempo de establecimiento dado por el usuario según sus necesidades, el valor de la variable cintura se asocia al motor que se moverá en cada instante de tiempo según se va calculando.

3.5 Simulación del brazo robótico

Para este fin se diseñó el brazo robótico de una forma relativamente sencilla por medio del software “Inventor” sin entrar en muchos detalles pues solo se requiere que la simulación traiga consigo los datos de dinámica del sistema y a modo de simulación solo se mostrará el movimiento sin carga. Dicho proyecto de Inventor se exportó a Simulink por medio de Simmechanics que es un toolbox de matlab

especial para simulación de sistemas que constan de multicuerpos o varias partes unidas para lograr un desarrollo final; el archivo importado de inventor se muestra en la figura 15, al oprimir el botón de Run del simulador aparece el brazo que se muestra en la figura 16.

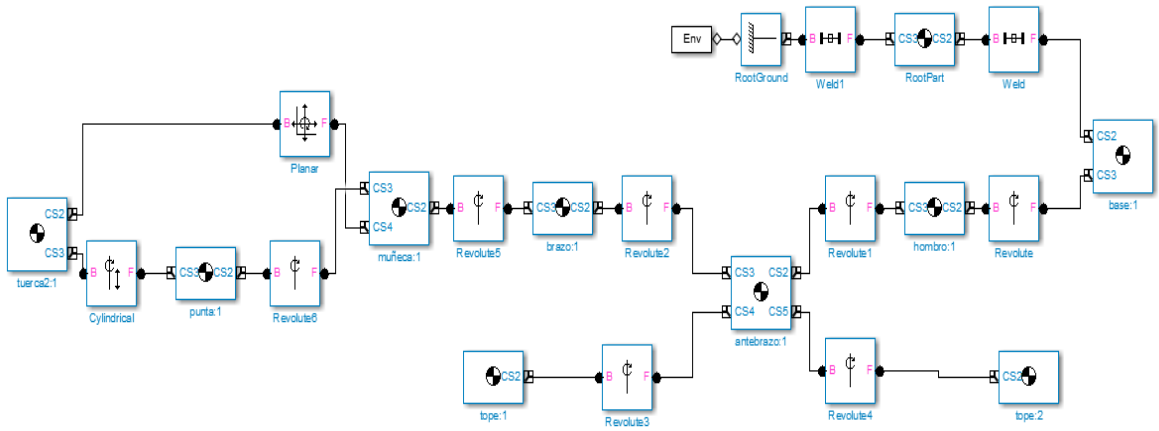

Figura 15, archivo importado desde inventor.

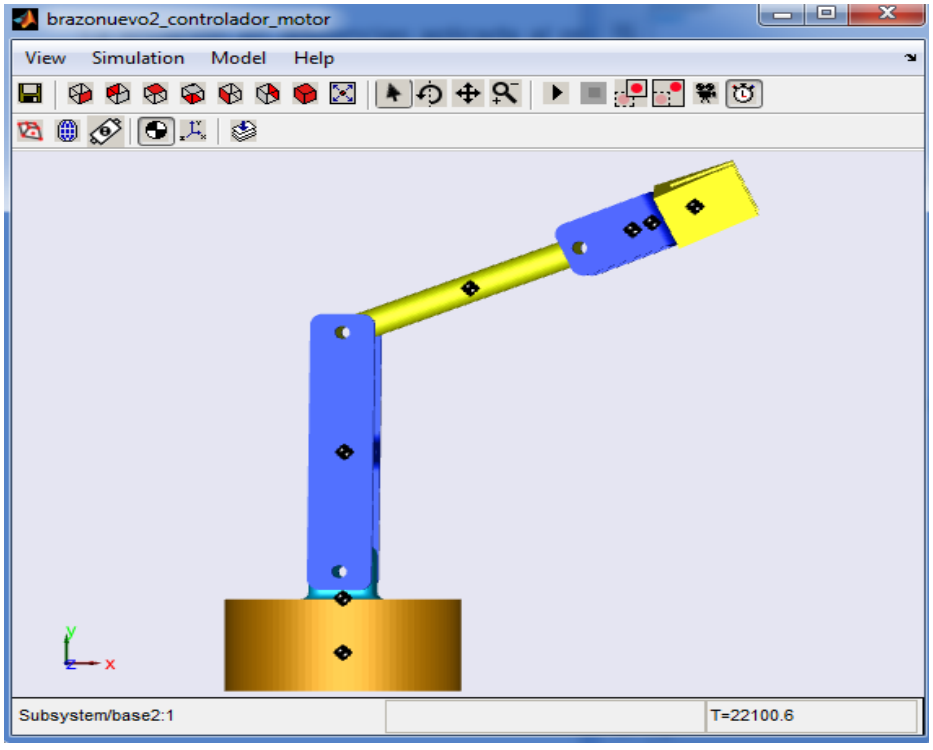

Figura 16, brazo diseñado en Inventor e importado a Simulink.

El brazo por si solo simula la fuerza de gravedad ejercida en todos los eslabones a excepción de la base dado que no hay aún ningún tipo de controlador que corrija este efecto por lo cual se empieza por incluir en cada bloque de “revoluta ”de los mostrados en la figura 15, un motor DC simulado que a su vez traerá la dinámica del dispositivo para acercar la simulación aún más a la realidad, dicho motor se muestra en la figura 17. El control de la figura 18 se asocia a cada motor y será del tipo PID.

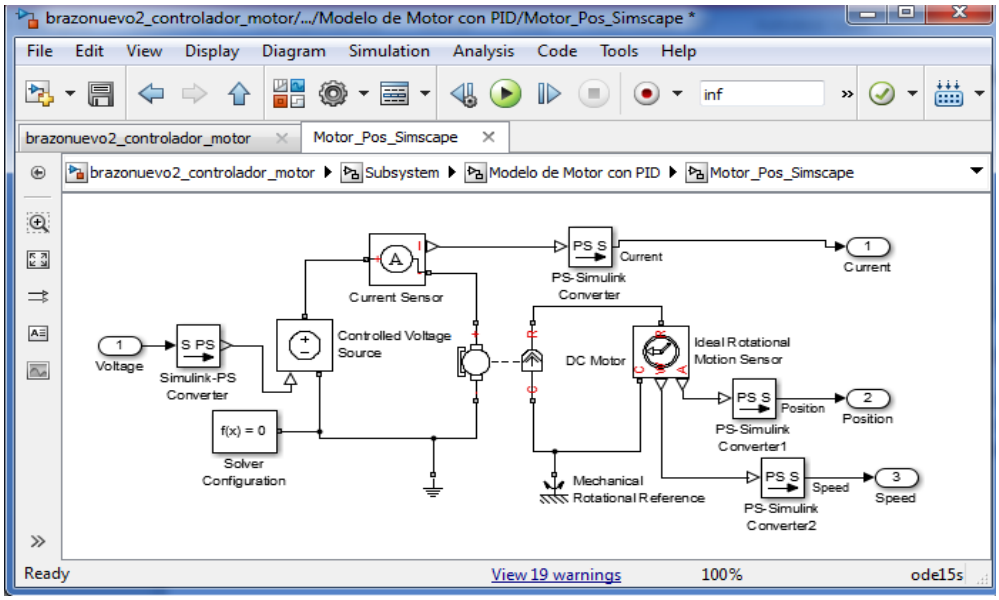

Figura 17, diseño del motor DC en simulink.

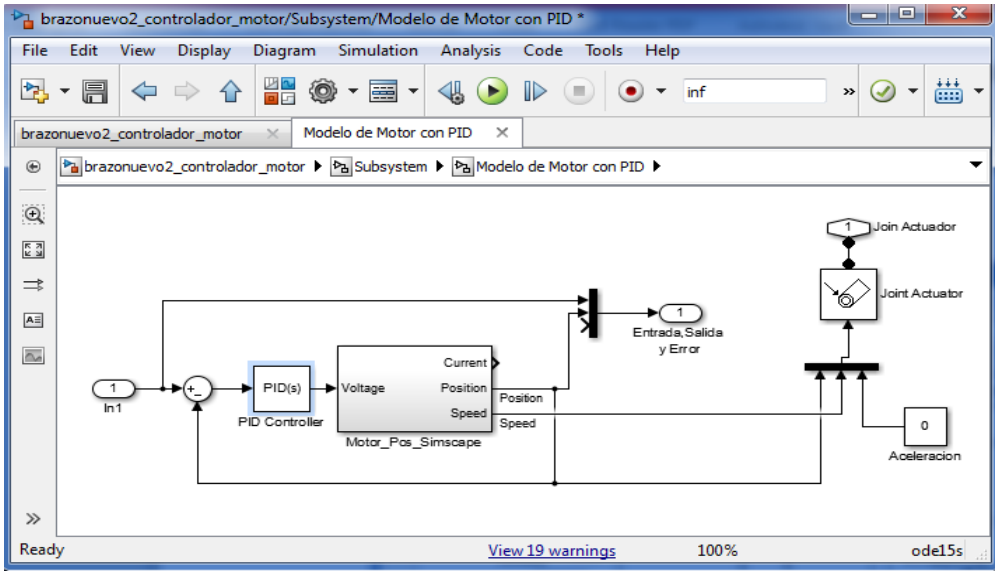

Figura 18, motor DC con PID adicionado

Cada motor con su respectivo PID se unifican en un “subsistema” como se le llama en simulink a la reducción de un sistema relativamente grande a un solo bloque, al final a todo el sistema generado por los mostrados en las figuras 15, 17 y 18 se les nombra de nuevo como subsistema quedando de la forma mostrada en la figura 19.

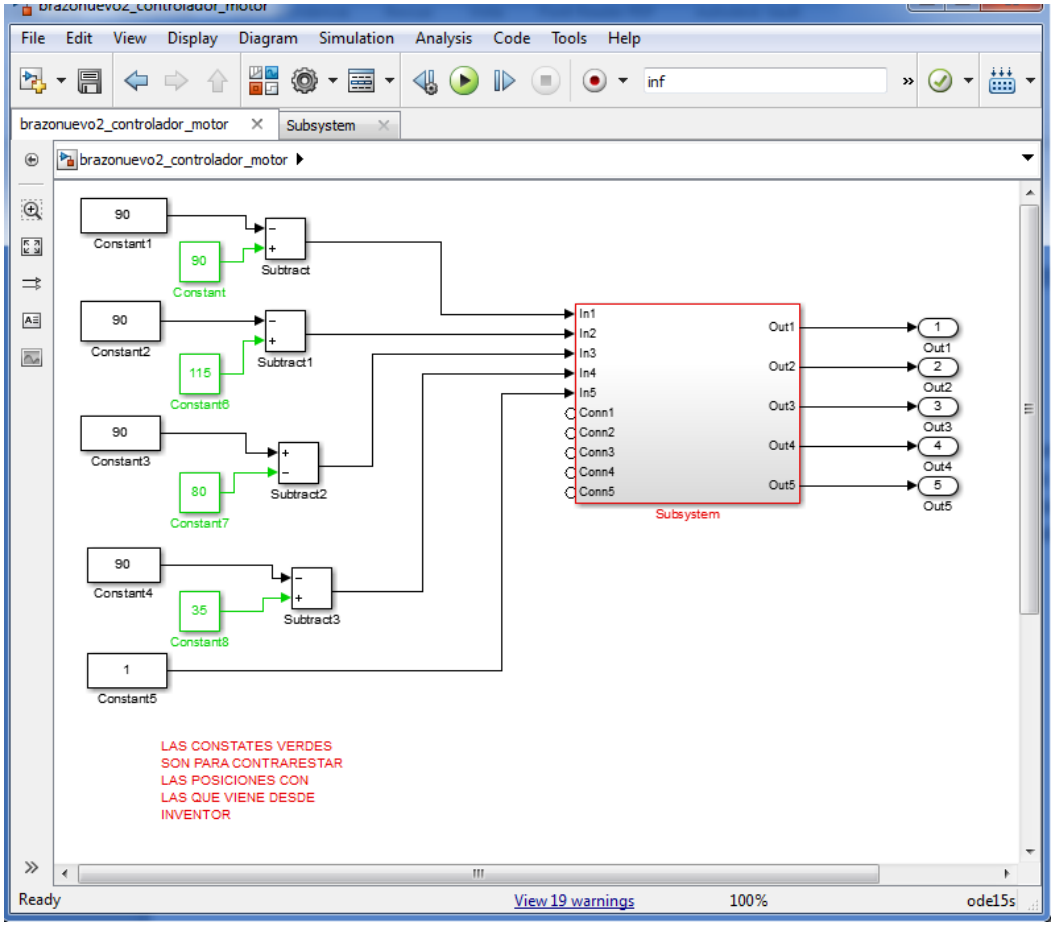

Figura 19, subsistema de subsistemas.

Las constantes de color verde se adicionan para compensar la posición original y que el sistema empiece en la posición de 90° para cada servomotor toda vez que el sistema importado desde Autodesk Inventor tiene puntos de referencia diferentes, de no colocarse estas constantes, el sistema simulado empezaría en una posición totalmente diferente a la del brazo físico.

4 RESULTADOS

A continuación se presentan los resultados finales del proyecto y sus variantes ante factores como cálculo y limitaciones de implementación según los elementos y métodos utilizados.

Interfaz de usuario: se debe ingresar en las casillas correspondientes a los tres ejes los valores a los cuales se quiere que llegue la punta del brazo, adicionalmente se debe ingresar el tiempo de establecimiento que es el tiempo que se demoraría el sistema en estabilizarse y ese dato es con el cual se evalúa el Modelo de Referencia en cada instante de tiempo. La interfaz de usuario muestra los valores objetivos de los ángulos de cada grado de libertad y la variación de cada ángulo en el momento que se está evaluando. La interfaz es muy sencilla y fácil de comprender y se muestra en la figura 15.

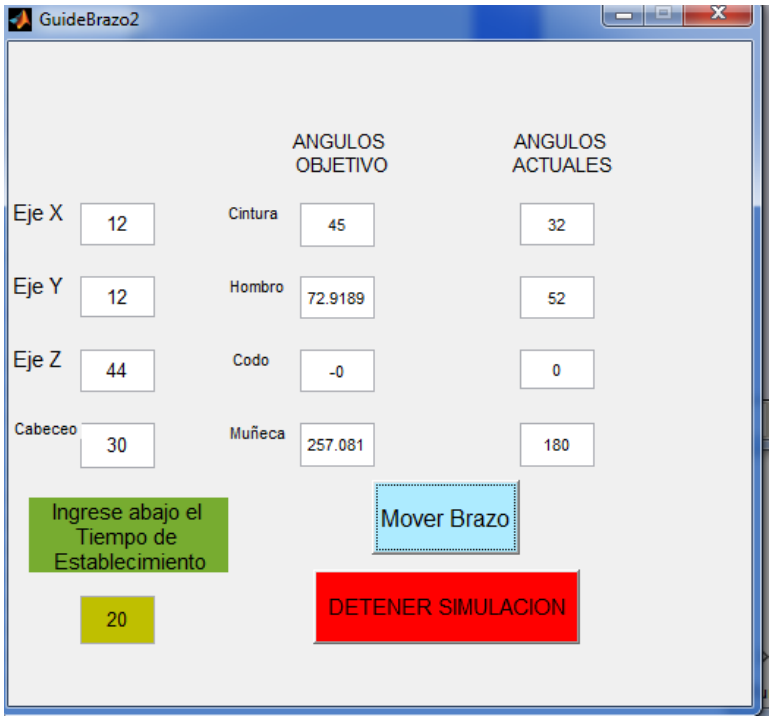

Figura 20, interfaz de usuario

Se deben ingresar las coordenadas X,Y y Z a las cuales el usuario quiere enviar la punta del brazo. El ángulo de Cabeceo es la posición que tendrá la punta todo el tiempo.Los Ángulos Objetivo son los que calcula la cinemática inversa y los Ángulos Actuales van cambiando en la interfaz de usuario a medida que son recalculados con el tiempo.

4.1 Software de la interfaz

El software de la interfaz realizado en matlab tiene indexada la cinemática inversa, necesaria para el cálculo de los ángulo requeridos para cada uno de los 5 grados de libertad del sistema; así también tiene el cálculo del modelo de referencia necesario para que los movimientos sean suaves, cálculo realizado con la ecuación 3.4.1.

```
function varargout = GuideBrazo2(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GuideBrazo2_OpeningFcn, ...
                   'gui_OutputFcn',  @GuideBrazo2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before GuideBrazo2 is made visible.
function GuideBrazo2_OpeningFcn(hObject, eventdata, handles, varargin)
global a cintura hombro codo muneca;

find_system('Name','brazonuevo2_controlador_motor');
open_system('brazonuevo2_controlador_motor');
set_param(gcs,'SimulationCommand','Start');

a = arduino('com3');%se asigna el arduino y su puerto a la variable a
servoAttach(a,3);   %servo para la cintura
servoWrite(a,3,90);
servoAttach(a,4);   %servo para el hombro
servoWrite(a,4,90);
servoAttach(a,6);   %servo para el codo
servoWrite(a,6,90);
servoAttach(a,9);   %servo para la muñeca
servoWrite(a,9,90);

cintura = 0;    %estas veriables se inicializan en esta parte del
                %código para que el brazo empieza el movimiento desde
                     %la última parte donde quedó y no arranque en la
      posición %de inicio a cada vez que se le indique nuevas coordenadas
hombro = 0;
codo = 0;
```

```
muneca = 0;
% Choose default command line output for GuideBrazo2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% --- Outputs from this function are returned to the command line.
function varargout = GuideBrazo2_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function editEjeX_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editEjeX_CreateFcn(hObject, eventdata, handles)
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editEjeY_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editEjeY_CreateFcn(hObject, eventdata, handles)
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editEjeZ_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editEjeZ_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function editCabeceo_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editCabeceo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushMoverBrazo.
function pushMoverBrazo_Callback(hObject, eventdata, handles)
global a b cintura hombro codo muneca;

LongMunec = 7;
LongBrazo = 12;
LongAntBr = 12;
AlturaH = 5;

X = str2num(get(handles.editEjeX,'string'));
Y = str2num(get(handles.editEjeY,'string'));
```

```
Z = str2num(get(handles.editEjeZ,'string'));
Cabeceo = str2num(get(handles.editCabeceo,'string'));

grados = (180/pi);
radianes = (pi/180);

Cabeceorad = real (Cabeceo*radianes);
AngGiro = atan2 (Y,X) * grados

%¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨´CINEMATICA INVERSA ¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨
Modulo = sqrt(X^2 + Y^2);
Xprima = Modulo;
Yprima = Z;

Afx = cos (Cabeceorad)*LongMunec;
LadoB = Xprima - Afx;
Afy = sin(Cabeceorad)*LongMunec;
LadoA = Yprima - Afy - AlturaH;
Hipotenusa = sqrt (LadoA^2 + LadoB^2)
Alfa = real( atan2(LadoA,LadoB) );
Beta  = real( acos( (LongBrazo^2 - LongAntBr^2 + Hipotenusa^2) /
(2*LongBrazo*Hipotenusa)  ))
AngBrazo = (Alfa + Beta)*grados;
Gamma = real(acos( (LongBrazo^2 +LongAntBr^2 - Hipotenusa^2) / (2*
LongBrazo*LongAntBr)  ))
AngAntBr = - (180 - (Gamma* grados))
AngMunec = 360 - (Cabeceo + AngBrazo + AngAntBr);
%´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´´
set(handles.editCintura,'String',AngGiro) %coloco los Angulos objetivo en
set(handles.editHombro,'String',AngBrazo) %el guide para visualizar hasta
set(handles.editCodo,'String',AngAntBr)   %donde debería llegar el brazo
set(handles.editMuneca,'String',AngMunec)
i = 0;
h = 0.1; %tiempo de muestreo
estable =str2num (get(handles.editEstablecimiento,'String')); %tiempo de
establecimiento
a1 = (-4/estable);   % Polo discreto
poten = (a1*h);  %potencia de la exponencial
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
while round(cintura) ~= round(AngGiro) || round(hombro) ~=
round(AngBrazo) || round(codo) ~= round(AngAntBr) || round(muneca) ~=
round(AngMunec)
cintura = ((exp(poten)) * cintura) + ((1-exp(poten)) * AngGiro );
%i = i+1 ;    %variable para saber cuantas iteraciones se hacen
if cintura < 0
    cintura = 0;
end
if cintura > 180
   cintura = 180;
end
servoWrite(a,3,round(cintura));
set(handles.editCintuAc,'String',round(cintura));   %coloco los Angulos
objetivo en
%pause (0.1);
```

```
AngGiroSimulink = num2str(cintura); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant1','Value',
AngGiroSimulink);
hombro= ((exp(poten)) * hombro) + ((1-exp(poten)) * AngBrazo );
if hombro < 0
    hombro = 0;
end
if hombro > 180
   hombro = 180;
end
servoWrite(a,4,round(hombro));
set(handles.editHombroAc,'String',round(hombro));%el guide para
visualizar hasta el valor objetivo
AngBrazoSimulink = num2str(hombro); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant2','Value',
AngBrazoSimulink);
codo = -1;  %lo inicializo en este valor para que siempres entre a este
bucle
codo = ((exp(poten)) * codo) + ((1-exp(poten)) * AngAntBr );
i = i+1     %variable para saber cuantas iteraciones se hacen
if codo < 0
   codo = 0;
   AngAntBr = codo;
end
   if codo > 180
    codo = 180;
    AngAntBr = codo;
   end
servoWrite(a,6,round(180-codo)); %se resta a 180° por que el servo esta
invertido
set(handles.editCodoAc,'String',round(codo));%donde debería llegar el
brazo
AngAntBrSimulink = num2str(codo); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant3','Value',AngAntBrSimul
ink);
muneca = ((exp(poten)) * muneca) + ((1-exp(poten)) * AngMunec );
%i = i+1 ;    %variable para saber cuantas iteraciones se hacen
if muneca < 0
    muneca = 0;
    AngMunec = muneca;
end
if muneca > 180
   muneca = 180 ;
   AngMunec = muneca ;
end
servoWrite(a,9,round(muneca));
set(handles.editMunecaAc,'String',round(muneca));
pause (0.1);
AngMunecSimulink = num2str(muneca); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant4','Value',AngMunecSimul
ink);
end
% --- Executes on button press in pushDetener.
function pushDetener_Callback(hObject, eventdata, handles)
```

```
global a;

clear all
delete(GuideBrazo2)%para cerrar la interfaz
cintura= 90;      %para que el brazo termine en el valor inicial
AngGiroSimulink = num2str(cintura); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant1','Value',AngGiroSimuli
nk);
hombro = 90;
AngBrazoSimulink = num2str(hombro); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant2','Value',AngBrazoSimul
ink);%se inicializa el brazo
codo = 90;
AngAntBrSimulink = num2str(codo); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant3','Value',AngAntBrSimul
ink);%para la siguiente simulación
muneca = 90;
AngMunecSimulink = num2str(muneca); %se envia el dato a simulink
set_param('brazonuevo2_controlador_motor/Constant4','Value',AngMunecSimul
ink);
clear all

function editCintura_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editCintura_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editHombro_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editHombro_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editCodo_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editCodo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editMuneca_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editMuneca_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editCintuAc_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editCintuAc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editHombroAc_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editHombroAc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editCodoAc_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editCodoAc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editMunecaAc_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editMunecaAc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editEstablecimiento_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function editEstablecimiento_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editEstablecimiento_ButtonDownFcn(hObject, eventdata, handles)
```

5 Conclusiones

- Dado que la cinemática inversa calcula el valor del ángulo para cada uno de los grados de libertad, dicho valor puede ser tanto positivo como negativo y en la mayoría de los casos con valores que superan los 180°.

- Se encuentra que los servomotores que hacen parte del brazo robótico de la marca “lynxmotion” y como la mayoría de servos tienen un rango de operación que va de los 0° Hasta los 180° por lo cual es imposible que se dé el resultado final y esperado. Para evitar que el programa desarrollado en matlab arroje errores a causa de valores fuera de los rangos especificados, se saturan o limitan los valores objetivos con ciclos condicionales del tipo If, Else, End, donde para valores negativos la interfaz se encarga de entregar un valor objetivo de 0° y para valores superiores a 180° el sistema se ubica en este valor. Se explica mejor en el siguiente trozo de código.

```
if muneca < 0
   muneca = 0;
   AngMunec = muneca;
end
if muneca > 180
   muneca = 180 ;
   AngMunec = muneca ;
end
```

  La condición anterior se repite en cada uno de los grados de libertad.

- La cinemática inversa es una muy buena opción para encontrar los ángulos necesarios para cada servomotor sin tener que experimentar cuales serían los valores que nos llevarían a la mejor respuesta sino que esta lo hace por nosotros.

- Al aplicar los conocimientos adquiridos durante la carrera se hace interesante ver los resultados y como cada tema estudiado aporta su parte para dar solución a casi cualquier problema físico.

- Este método de cinemática inversa es aplicable y con excelentes resultados para sistemas simulados ó sistemas físicos que cuenten con servomotores que tengan una movilidad de 360°. En vista de los valores entregados que pueden ser negativos ó superiores a 180°, este método no es aplicable a cualquier brazo robótico como se esperaba que fuera al iniciar este proyecto por lo cual los resultados obtenidos con el brazo fabricado por

Lynxmotion y empleado para este prototipo no llega a todas las coordenadas deseadas limitando la movilidad del brazo robótico a distancias muy reducidas.

- Lo anterior no impide que la teoría de control opere como se espera toda vez que se pudo evidenciar que al aplicar el Modelo de Referencia los movimientos del brazo son mucho más sutiles a diferencia del comportamiento que tiene sin aplicar dicho modelo de control y en este caso cumple con lo esperado al principio del proyecto que es evitar el derrame de líquidos durante el desplazamiento del brazo robótico.

6 SUGERENCIAS PARA OPTIMIZAR ESTE TIPO DE PROYECTOS

En primer lugar, se observó que la cinemática inversa entrega valores tanto positivos como negativos y estos pueden superar los 180° radianes. Para que el método empleado en este proyecto para hallar los valores de los ángulos de cada grado de libertad (cinemática inversa) a partir de las coordenadas deseadas funcione adecuadamente y el resultado final sea el esperado se deben usar servomotores que tengan movimiento de 360° que aunque no son muy populares ya empiezan a aparecer en algunos mercados.

Otro método interesante para la implementación en este tipo de proyectos es el de Jacques Denavit y Richard Hartenberg que son cuatro parámetros asociados con un convenio en particular para la fijación de marcos de referenica a los eslabones de una cadena cinemática especial o de un manipulador robótico. En este se utiliza una matriz para representar la posición de un cuerpo respecto a otro.

REFERENCIAS

[1] Craig John J. Robotica tercera edición.

[2] http://www.lynxmotion.com/t-contact.aspx. Lynxmotion, a RobotShop Inc. company 555 VT Route 78 Suite 367 Swanton, Vermont, USA, 05488

[3] http://www.ib.cnea.gov.ar/~instyctl/Tutorial_Matlab_esp/index.html Carnegie Mellon, Copyright (C) 1996 by the Regents of University of Michigan.

[4] Rodríguez Ramirez, Daniel. Bordóns Alba, Carlos. APUNTES DE INGENIERÍA DE CONTROL. Rev. 5/05/2005

[5] Rodríguez Rubio Francisco , López Sánchez Manuel Jesús. 2006. Control Adaptativo y Robusto, Universidad de Sevilla.

[6] Palmieri Diego. Capítulo 3. Modelo de Sistemas Discretos. http://iaci.unq.edu.ar/Materias/Cont.Digital/Apuntes/ApuntePagina/CODIES2010_Cap3_Modelos%20de%20Sistemas%20Discretos.pdf

[7] DR Vasquez López Virgilio, departamento de Mecatrónica y Automatización ITESM - CEM http://homepage.cem.itesm.mx/vlopez/notas2p.pdf

[8] Fernández Sarásola Armando. Control de los Sistemas continuos.

[9] Mantz, Ricardo Julián. Introducción al control óptimo.

[10] Platero Dueñas, Carlos. 2008, Apuntes de Regulación Automática. Departamento de Electrónica, Automática e Informática Industrial. http://www.elai.upm.es/webantigua/spain/Asignaturas/Servos/Apuntes/7_OrdenSup.pdf

[11] Ogata, Katsuhiko. Modern Control Engineering. Fourth edition. New Jersey: Prentice hall inc, 2002. 976 p.