GENERACIÓN DE TRAYECTORIAS PARA UNA MESA CARTESIANA DE TRES GRADOS DE LIBERTAD

SANTIAGO MONTOYA SABAS

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
FACULTAD DE INGENIERÍA MECÁNICA
MEDELLÍN

2015

GENERACIÓN DE TRAYECTORIAS PARA UNA MESA CARTESIANA DE TRES GRADOS DE LIBERTAD

SANTIAGO MONTOYA SABAS

Trabajo de grado para optar por título de Ingeniero Mecánico

Director
Julio César Correa Rodriguez
Doctor of Philosophy in Mechanical Engineering

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
FACULTAD DE INGENIERÍA MECÁNICA
MEDELLÍN

2015

Nota de aceptación

_____

_____
_____
_____

_____

Firma

Nombre:

Presidente del jurado

_____

Firma

Nombre:

Jurado

_____

Firma

Nombre:

Jurado

Medellín, Mayo de 2015

DEDICATORIA

Le dedico este trabajo a mi abuelita que siempre me ha apoyado y se ha preocupado por mi bienestar.

AGRADECIMIENTOS

Agradezco a todos los docentes que durante toda la carrera me brindaron conocimiento por medio de sus clases, en especial a mi director de tesis Julio Correa del cual aprendí enormemente, gracias a ellos es que me encuentro donde estoy en el momento.

CONTENIDO

LISTA DE FIGURAS

1 INTRODUCCIÓN

Actualmente, la robótica es parte fundamental en las industrias automatizadas. Entre los robots manipuladores más importantes se encuentran: robots seriales, los paralelos y las mesas cartesianas o mesas XYZ. Su debida utilización hace a las instituciones y empresas más competitivas en su entorno. Los sistemas cartesianos son importantes por su simplicidad, sencilla manipulación, programación, amplia variedad de aplicaciones y bajo costo comparándolas con otros sistemas robóticos.

Usualmente, industrias fabricantes de mesas cartesianas no suministran el ¨*know how*¨ de sus equipos, ni proporcionan información detallada sobre el *software* o el *hardware.* El conocimiento del funcionamiento de la máquina no es compartido con el público.

En el mercado existe diversidad de software para la manipulación de mesas XY, tales programas son proporcionados por el mismo proveedor de la mesa. Éstos varían dependiendo de la necesidad del cliente, pero todos concuerdan en que son software para el seguimiento de una trayectoria.

Desarrollar una mesa cartesiana en la Universidad permitiría adquirir conocimiento relacionado con el hardware y el software de este tipo de sistemas robóticos, donde todos los desarrollos serían propios y se daría una solución efectiva a los propósitos académicos de la institución.

En los Laboratorios de Mecánica de la Universidad Pontificia Bolivariana el grupo A+D tiene una mesa cartesiana diseñada y construida por Santiago Flórez Toro[1]. La mesa con sus accesorios está casi lista físicamente, pero no cuenta con el sistema eléctrico ni con un software controlador, por lo que la mesa aún no está operando. En este trabajo se desarrollará un programa que permita generar trayectorias a la mesa cartesiana de la UPB.

2 MARCO TEÓRICO

En el siguiente capítulo se dará información de las mesas cartesianas encontradas en la industria y se hablará de la mesa con la que dispone la UPB. Se hablará sobre el tipo de movimientos que deben ejecutan las mesas Se considerarán todos los elementos para la gestión de los elementos faltantes físicos de la mesa y se contextualizará los elementos teóricos para la elaboración del programa de seguimiento de trayectoria deseado.

2.1 MESAS CARTESIANAS

Una mesa cartesiana, es una estructura rígida que normalmente tiene tres ejes móviles X,Y,Z. Usualmente el eje X soporta el eje Y, el eje Y a su vez soporta el eje Z. Este último hace de efector final y es el que soporta la herramienta.

El eje X se conoce como riel de desplazamiento, el eje Y es conocido como viga móvil, el eje Z se conoce como portaherramientas o simplemente eje Z y por lo tanto es de especial interés. La combinación de movimientos sobre los ejes X y Y, le permiten a la mesa seguir todo tipo de trayectorias en el plano y el eje Z le permite una altura definida o variable a esta trayectoria.

2.2 COMPONENTES

Las mesas cartesianas incluyen distintos componentes: bases que soportan las cargas generadas, actuadores y elementos móviles. A continuación se hace una breve descripción de algunos de ellos.

2.2.1 Eje X

Éste normalmente es el eje apoyado a la tierra y es el más robusto de los tres. Se encuentra doblemente apoyado para poder soportar las cargas que generan los otros ejes. Permite el desplazamiento en el eje X, usualmente es el más largo.

2.2.2 Eje Y

El eje Y está en la mayoría de los casos apoyado en dos rieles del eje X, esto debido a que este eje también tiene que soportar el peso y las cargas generadas por el eje Z. En casos de mesas pequeñas éste solo está apoyado en un riel de X. El movimiento del eje Y proporciona movimiento en dirección a lo ancho del eje X.

2.2.3 Eje Z

El eje Z es donde se encuentra el efector final, éste se encuentra apoyado sobre el eje Y y permite un cambio de altura del sistema. Comúnmente se encuentra apoyado en un punto ya que solo debe soportar el peso de la herramienta y las cargas generadas por la operación.

2.2.4 Base

La base es una estructura rígida donde van apoyados los rieles del eje X, ésta va directamente en el punto donde se desea poner la mesa. No tiene movimiento alguno, es el punto de referencia (tierra) respecto a los ejes.

2.2.5 Motores

El movimiento de los ejes es generado por servomotores o motores paso a paso que entregan una fuente de movimiento rotacional que al ser transformada en lineal permite el movimiento de los ejes.

2.2.6 Sistema de Transmisión

El movimiento rotacional de los motores es transformado comúnmente por medio de bandas dentadas que se encuentran dentadas que se encuentran dentro de los perfiles de la base. La utilización de éstas proporciona un sistema de transmisión un sistema de transmisión simple y confiable. Permite que el movimiento en ambos rieles del eje X sea uniforme. En la eje X sea uniforme. En la

Figura 1se muestra uno de los sistemas más comunes usados en las mesas cartesianas. [2].

Figura 1. Sistema de transmisión.

Fuente: Bahr –Modultechnik [en línea]. Fuente [11]

2.3 MESA CARTESIANA DE LA UPB

En la Universidad Pontificia Bolivariana, se cuenta con una mesa cartesiana de tres grados de libertad, se hará una descripción de la mesa, los componentes y cambios que hacen falta para poder funcionar en la parte física. Se debe de tener en cuenta que la mesa no cuenta con un programa que genere trayectorias por medio de la intervención del usuario, éste hará parte de la entrega de este trabajo. En la Figura 2 se muestra un esquema de la mesa cartesiana de la Universidad con algunas partes remarcadas.

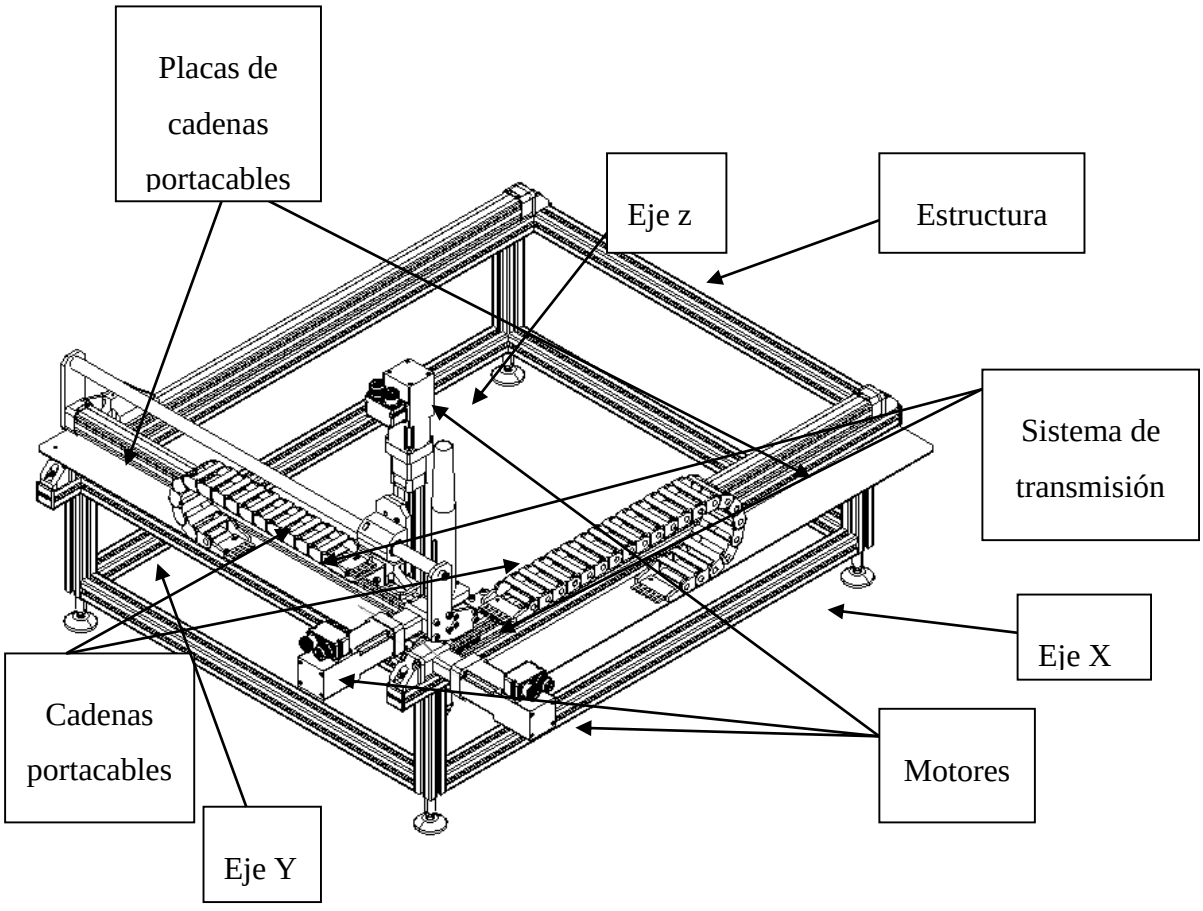

Figura 2.Mesa cartesiana de la UPB.

Fuente: [1]

2.3.1 Estructura

La mesa tiene una estructura rectangular en perfiles de aluminio de 35 mm, permite a la mesa apoyarse en cualquier superficie plana.

2.3.2 Motores

Los motores utilizados por la mesa son tres servomotores, para el eje X se utilizó un Baldor BSM50N-233AF con torque pico 3.65 [N/m] y torque stall (torque de frenado) de 0.9 [N/m] y para los ejes Y y Z se utiliza para cada uno un motor Baldor BSM50N-133AF con torque pico 1.8 [N/m] y torque stall (troque de frenado) de0 .45 [N/m]. Cada uno de los motores permite el desplazamiento de la mesa en cada uno de los ejes. El servomotor del eje Z tiene una abolladura en el conector a los cables de poder, la cual es necesaria corregir para poder acoplar el cable, necesario para suministrar corriente y poder funcionar.

2.3.3 Cadenas Portacables

Los motores van conectados a dos cables cada uno, uno para el control y otro para el suministro de corriente, estos cables pasan por dentro de las cadenas portacables y estas les permite moverse de manera que no se enreden mientras se están moviendo. Estas cadenas portacables fue necesario adquirirlas ya que la mesa no contaba con ellas, los parámetros para la compra fueron el diámetro de los cables y la cantidad de cables que pasaban por cada una de estas, se adquieren las cadenas a la empresa COLDECON (controles de Colombia).

2.3.4 Placas de Cadenas Portacables

Las placas donde van apoyadas las cadenas portacables se encuentran en los laboratorios, no están acopladas y debido a un cambio en el diseño original de la estructura es necesario hacerles unas modificaciones que constan en maquinar las placas y hacer las perforaciones necesarias de manera que se puedan acoplar correctamente con las tuercas y tornillos y con las vigas que hace arte de la base de la mesa.

2.3.5 Porta Herramientas

La mesa cuenta con dos tipos de porta herramientas: una para la herramienta de corte por plasma (no adquirida aún) y otra para adaptar un gripper o pinza neumática.

2.3.6 Partes Estructurales

Para el buen funcionamiento, la mesa cuenta con barras, acoples, soportes y partes, asegurando la estabilidad y seguridad de la mesa para asegurar el movimiento correcto en los dos rieles de X, Y y Z.

2.3.7 Actividades mecanicas llevadas a cabo

Para el buen funcionamiento de la mesa y la terminación física de la misma, se lleva a cabo las siguientes actividades

2.3.7.1 Corte de las placas que soportan las cadenas portacables de manera que puedan ser acopladas correctamente a la base de la mesa.

2.3.7.2 Tramite de piezas de ajuste como tornillos y tuercas, faltantes para el correcto ensamble de la misma.

2.3.7.3 Cotización y compra de las cadenas portacables.

2.3.7.4 Acoplamiento y ensamble de todos los componentes sueltos, barras estabilizadoras de los ejes X,Y, placas que soportan las cadenas portacables, cadenas portacables, piezas de ajuste, paso de cables por las cadenas portacables hacia los motores.

2.4 SOFTWARE DE CONTROL

El movimiento de la mesa va a estar dado por una combinación de acciones de los motores correspondientes a cada uno de los ejes, el control de las acciones de los motores es indispensable para el buen funcionamiento de la mesa. Hasta el momento no se ha desarrollado ningún software controlador para la mesa cartesiana de la UPB. La entrega de unas trayectorias controladas por el software hace parte de la realización de este trabajo.

2.4.1 Mint

El lenguaje de programación MINT (Motion Intelligence Multitasking) es el utilizado para el desarrollo de actividades en los motores Baldor, cuenta con un lenguaje muy sencillo y fácil de programar cuando se trata control de movimiento. Las cualidades más relevantes del lenguaje de programación MINT para el proyecto son:

- Compatibilidad con los servomotores y motores pasa a paso, permitiendo que los ejes se combinen.
- Palabras claves en inglés.
- Capacidad de hacer varias tareas a la vez.
- Control de torque, velocidad y posición.
- Interpolaciones.
- Ciclos controlados.

2.4.2 Matlab®

El lenguaje de programación Matlab® es uno de los lenguajes más conocidos para efectuar cálculos de ingeniería. Por este motivo se desea utilizarlo para desarrollar un código que calcule los datos necesarios para los servomotores, esta información va a ser transportada al lenguaje MINT, los servos van a leer la información y ejecutar el movimiento deseado. En la Figura 3 se encuentra el diagrama de proceso desde la obtención de los datos hasta la ejecución de movimiento de los servomotores.

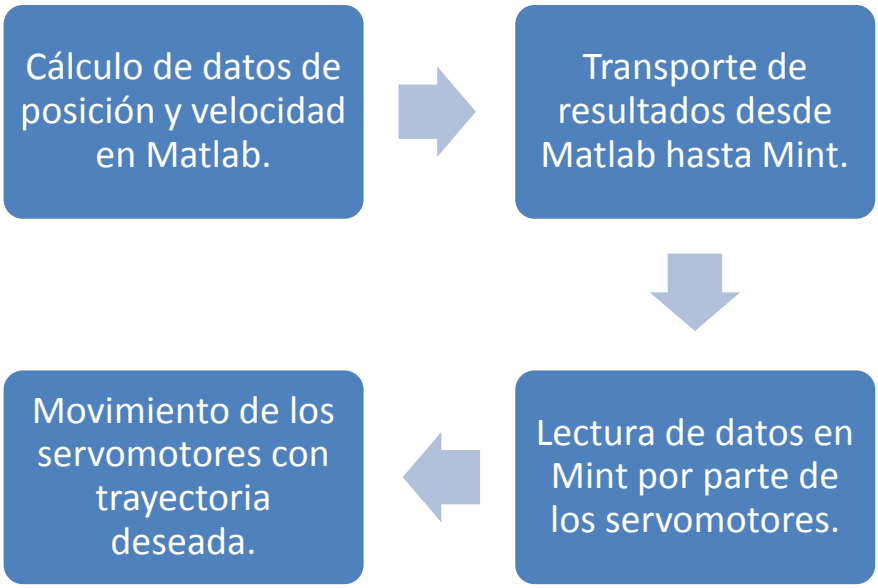

Figura 3. Diagrama de proceso de los datos.
Fuente: Elaboración propia.

2.4.3 Seguimiento de Trayectoria

Una trayectoria se define como el recorrido que sigue un cuerpo al desplazarse de un punto a otro, teniendo en cuenta que puede haber más de una posible línea o recorrido deseado. En las mesas cartesianas el seguimiento de una trayectoria está determinado por la combinación controlada de los movimientos de los tres motores.

Se debe tener en cuenta que se está trabajando con servomotores que tienen una aceleración una velocidad y una posición configurables, se debe tener cuidado cuando se programen, ya que cambios o paros repentinos en el proceso pueden afectar el buen funcionamiento de los servomotores y las partes estructurales de la mesa.

Los movimientos que deben hacer los motores de la mesa deben ser suaves, continuos y con pocos picos. Los movimientos típicos de un servomotor para un intervalo en el cual inicia y termina con velocidad cero se describen en la Figura 4. La imagen superior muestra la posición de un motor dada en grados vs tiempo, en la imagen del medio se ve la gráfica de velocidad en

grados por segundo vs tiempo, y por último se observa la aceleración en grados por segundo cuadrado vs tiempo.

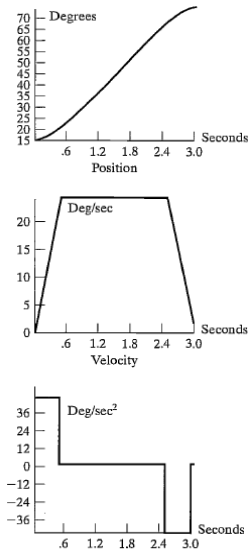

Figura 4.Curvas de posición, velocidad y aceleración.

Fuente:[3]

La curva de posición para el caso de la Figura 4, aumenta durante el intervalo de movimiento. Inicialmente está detenido y por lo tanto la velocidad es cero. Cuando el movimiento va a dar comienzo la aceleración toma un valor constante positivo, generando una velocidad lineal positiva. Cuando el motor llega a la velocidad pedida, está se vuelve constante y por ende el valor de la aceleración es cero. Cuando se aproxima al punto final (deseado) los motores deben de llegar suave tal como arrancaron, por lo anterior el movimiento inicial se repite con valores de velocidad y aceleración negativos, asegurando la llegada suave al punto final.

En vez de utilizar solo dos puntos, las trayectorias pueden ser curvas definidas por una serie de puntos. Para poder lograr movimientos suaves y continuos, las trayectorias no pasan exactamente por los puntos deseados y deben calcularse cuidadosamente los tiempos para cada uno de los segmentos de aceleración y desaceleración entre los puntos de la curva. Esta teoría se conoce como *Via Points* (puntos intermedios entre el punto inicial y final de la trayectoria). Los detalles sobre cómo hacerlo, se presentan en el capítulo 3 siguiente y está basado en el material presentado por Craig [3].

2.5 GABINETE DE CONTROL

La mesa cartesiana requiere un gabinete de control para poder ejercer cualquier tipo de movimiento, el gabinete de control con controlador con plataforma Mint workbench permite controlar los motores de la mesa cartesiana. En el controlador es donde se ingresa el código por medio de comunicación USB y este a su vez al ser ejecutado manda las órdenes a los drivers para que estos realicen las acciones ordenadas en los motores. A continuación se hará listado de los algunos de los componentes del gabinete de control existente y una breve descripción e ilustración de los más influyentes para el propósito del trabajo.

2.5.1 Gabinete

En Figura 5 se observa como se ve físicamente el gabinete de control, el gabinete debe estar conectado a 110 V.

Figura 5. Gabinete abierto

Fuente propia

2.5.2 Drivers de Motores

Cada uno de los motores cuenta con un driver por lo que en el gabinete se tiene tres drivers que sirven para controlar y alimentar cada uno de los tres motores. Los driver son alimentados en el gabinete de control, monofásicamente a 110V. La señal del encoder proveniente del

motor información de posición y velocidad se conecta al driver y luego es replicada al controlador, ver Figura 6.

Figura 6. Drivers de los motores.

Fuente propia

2.5.3 Controlador Principal

El ciclo de control de la mesa cartesiana está compuesto por los drivers y el controlador, éste es un controlador multiejes que tiene la capacidad de controlar el movimiento de los tres motores sincronizadamente, ver Figura 7.

Figura 7. Controlador Baldor Nexmode e100.

Fuente propia

3 DESARROLLO DEL PROBLEMA

En este capítulo se mostrará la manera en la que se llega a la solución del problema y se logra desarrollar un programa donde la mesa cartesiana pueda realizar la trayectoria que se desee dentro de sus capacidades.

En el manual de programación de Mint Workbench[4] se encuentran muchas ayudas y aplicaciones para mesas cartesianas más ninguna que cumpla con las necesidades del proyecto, ninguna función de las encontradas permite realizar la entrada de datos directamente en el programa y que este ejecute las trayectorias en los tiempos deseados. Debido a la falta de una función que cumpla con el objetivo, se llega a la necesidad de dos códigos en distintas plataformas, el primero en Matlab® donde por medio de la teoría de *Via Points*[3] y teniendo los ángulos deseados y los tiempos entre ellos, se obtienen los datos necesarios para poder ejecutar el movimiento. El segundo código se desarrolla en Mint Workbench, donde partir de los datos encontrados en el primer código, se puede generar la trayectoria deseada en los tiempos deseados cumpliendo así con las condiciones requeridas en el proyecto.

3.1 SOFTWARE EN MATLAB®

En Matlab® se desarrolla un código que permite al usuario ingresar cuantos ángulos se requieran y el tiempos entre los ángulos deseados. El código al leer los datos y por medio de la teoría de *Via Points*[3] arroja todos los datos con los que se resuelve la teoría. En la Figura 8 se muestra una gráfica donde se pueden observar los datos necesarios y los deseados para objetivos de este trabajo. El código se puede dividir en cuatro partes fundamentales entrada de datos, inicio de movimiento, tramos medios, último tramo, brevemente explicados más adelante. El código mencionado se puede observar en el ANEXO 2.

Los datos de entrada (ver Figura 8) son los siguientes:

- Ángulos objetivos (θ_i, θ_j, θ_k).
- Tiempo entre cada par de ángulos (td_{12},td_{23}).
- Aceleración ($\ddot{\theta}$, no se visualiza en la gráfica, es constante en cada intervalo de movimiento).

Los valores que se quieren hallar (ver Figura 8) son:

- Velocidad lineal para cada trayecto entre dos ángulos ($\dot{\theta}_{ij}$, $\dot{\theta}_{jk}$, $\dot{\theta}_{kl}$).
- Tiempo para el tramo lineal (t_{ij},t_{jk},t_{kl}).
- Tiempo para el tramo con aceleración constante (t_j,t_k).

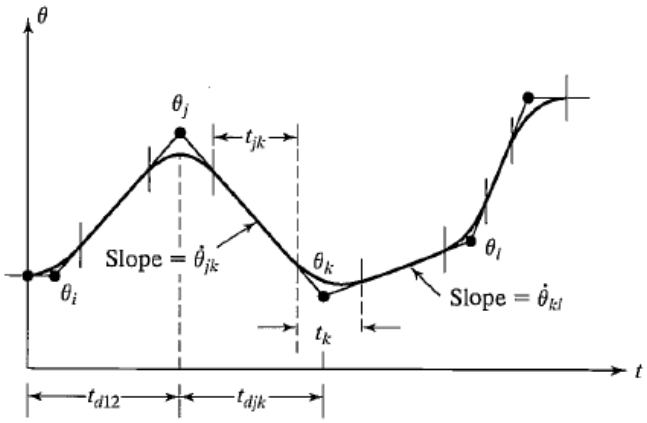

Figura 8. Datos necesarios *Via Points*.

Fuente [3]

3.1.1 Entrada de Datos

En la primera parte del código se requiere que el usuario entre la aceleración con la que desea el movimiento (teniendo en cuenta las capacidades de los motores), los ángulos por los que se desea que pase la trayectoria y el tiempo entre cada uno de los ángulos deseados.

3.1.2 Inicio del Movimiento

El comienzo del movimiento se requiere hacerlo de una manera suave. Se utiliza la aceleración estipulada en la entrada de datos y los dos primeros valores de los ángulos, para calcular posiciones y velocidades que serán los primeros datos obtenidos y con estos se ayuda a la solución de los puntos siguientes. En Figura 8 se puede identificar este tramo inicial acotado desde θi hasta θj. Las ecuaciones que se utilizan para desarrollar este tramo inicial son:

$$\ddot{\theta}1 = \mathrm{SGN}(\theta_2 - \theta_1) * \mathrm{abs}(\ddot{\theta}) \qquad (1)$$

$$t1 = td_{12}t - \sqrt{(td_{12})\text{^}2 - \frac{2*(\theta_2-\theta_1)}{\ddot{\theta}_2}} \quad (2)$$

$$\dot{\theta}12 = \frac{\theta_2-\theta_1}{td_{12}-\frac{t1}{2}} \quad (3)$$

$$t_{12} = td_{12} - t_1 - \frac{t2}{2} \quad (4)$$

3.1.3 Tramos Internos

Los tramos internos son todos los ángulos por los que se desea pasar pero que no corresponden al primero y al último. En la Figura 8 se puede observar este tramo desde θ_j hasta θ_k**.** Para ejecutar estos movimientos internos se considera que el movimiento no debe de parar y la transición de cada ángulo debe ser suave y continua. Para resolver las incógnitas en esta parte del código se requieren los datos obtenidos del ángulo anterior para resolver el siguiente (deseado) y así igualmente para todos los ángulos siguientes se va a necesitar los datos obtenidos del ángulo anterior. En este tramo se calculan los datos de los ángulos hasta el penúltimo. Las ecuaciones que describen estos tramos son:

$$\dot{\theta}jk = \frac{\theta k - \theta j}{td_{jk}} \quad (5)$$

$$\ddot{\theta} = SGN\left(\ddot{\theta}_{kl} - \ddot{\theta}_{jk}\right) * abs(\ddot{\theta}_k) \quad (6)$$

$$t_k = \frac{\dot{\theta}_{kl}-\dot{\theta}_{jk}}{\ddot{\theta}k} \quad (7)$$

$$t_{jk} = td_{jk} - \frac{t_j - t_k}{2} \quad (8)$$

3.1.4 Último Tramo

Éste corresponde al último ángulo, donde termina el movimiento. La terminación del movimiento debe ser suave como el inicio, por lo tanto se calcula tomando el penúltimo movimiento y se calculan los datos necesarios que en este caso en la Figura 8 se puede observar como θj.las ecuaciones que describen este tramo son;

$$\ddot{\theta}_n = SGN(\theta_{(n-1)} - \theta_{(n)}) * abs(\ddot{\theta}_{(n)}) \quad (9)$$

$$t_{(n)} = td_{(n-1)} - \sqrt{\left(td_{(n-1)n}\right)^2 - \frac{2*(\theta_{(n)}-\theta_{(n-1)})}{\ddot{\theta}_n}} \quad (10)$$

$$\dot{\theta}(n-1)n = \frac{\theta_{(n)}-\theta_{(n-1)}}{td_{(n-1)}-\frac{t_{(n)}}{2}} \quad (11)$$

3.2 DATOS GENERADOS POR EL CÓDIGO DE MATLAB®

Al ejecutar el código de Matlab®, éste calcula internamente los datos de velocidad lineal entre dos puntos sucesivos. Esta información junto con la aceleración y los ángulos deseados se presentan en un formato adecuado para que pueda ser ingresado a la plataforma de Mint Workbench, el formato de los datos se muestra a continuación.

Aceleración (ACCEL=$\ddot{\theta}$)

Desaceleración (DECEL=$\ddot{\theta}$)

Ángulos deseados (x_pos= θ_i, θ_j, θ_k....)

Velocidad deseada (velocidad_x=$\dot{\theta}_{ij}$, $\dot{\theta}_{jk}$, $\dot{\theta}_{kl}$...)

Se entiende que los valores simbólicos serán remplazados por valores numéricos, una vez hayan efectuado los cálculos correspondientes.

3.3 PROGRAMACIÓN EN MINT

En la plataforma de Mint Workbench se programa siguiendo las guías de programación *MINT WORKBENCH PROGRAMING GUIDE*[4]. En el código siguiente se busca que lo motores pasen por los ángulos deseados y cumplan con los tiempos requeridos, estos requerimientos los debe cumplir con la característica de suavidad y continuidad en los movimientos.

Para dar cumplimiento a los requerimientos del proyecto es necesario generar un código el cual se compone de:

- Cancelación de ordenes anteriores
- Dimensionamiento de los datos (ya que la plataforma lo exige)
- Entrada de datos
- Llevar los motores a la primera posición
- Ciclo donde se hace la lectura de los datos y ejecución de los mismo

A continuación se explican cada uno de estos pasos con los tramos del código para ayudar a ilustrar y el código completo se puede encontrar en el ANEXO 2.

3.3.1 Cancelar Órdenes Anteriores

En el primer tramo del código se utilizan tres comandos los cuales se encargan de borrar órdenes anteriores y configuraciones que tengan anteriormente los motores, es necesario que estas tres operaciones estén al principio del código para que pueda ser ejecutado.

```
TIMEREVENT=0
CANCELALL
RESETALL
```

3.3.2 Dimensionamiento de Variables

Es este tramo del código se dimensionan y se crean las variables para que el código pueda utilizar posteriormente, con la función "Dim" se la da el nombre a la variables y el número de datos que contiene dicha variable, así el programa va a crear la variable y estará lista para cuando se haga el ingreso de los datos de dichas variables, en el caso de 'a' no es necesario establecer cuantos datos va a contener ya que hace parte de un contador de un ciclo, explicado adelante. En este caso se trabaja con dos motores por lo que se deben dimensionar y dar valores a las variables de ambos.

```
Dim a
Dim velocidad_x((número de datos))
Dim velocidad_y((número de datos))
Dim x_pos((número de datos))
Dim y_pos((número de datos))
```

3.3.3 Ingreso de Datos

En esta sección del código se define todas las variables que requieren los motores para realizar la trayectoria deseada, la aceleración y la desaceleración (ACCEL,DECEL) se ingresan directamente como un único valor en el código ya que estas son constantes en toda la trayectoria y al no especificar si son para un motor o otro el programa asume que se utiliza para todos los movimientos independiente del motor que se mueva. La velocidad y la posición son variantes para cada par de puntos por lo que se hace necesario crear vectores para almacenar los datos (velocidad_x, x_pos, velocidad_y,y_pos) que previamente fueron obtenidos por Matlab®, es importante recalcar que estos datos se deben de ingresar con el mismo nombre que se les dio en el dimensionamiento de variables.

ACCEL=(valor de aceleración)
DECEL=(valor de desaceleración)
y_pos=(posición_1,posición_2,....)
x_pos= (posición_1,posición_2,....)
velocidad_y=(velocidad_1,velocidad_2,....)
velocidad_x=(velocidad_1,velocidad_2,....)

3.3.4 Llevar los Motores a Primera Posición

En esta parte del código se desea llevar los motores desde cualquier posición en la que se encuentren antes de empezar el movimiento, hasta la primera posición deseada, esto con el fin de no forzar un movimiento brusco al principio del movimiento deseado. Las instrucciones utilizadas para este tramo son: "MOVEA,SPEED,Pause IDLE". Para definir la velocidad a la cual se desea que se realice el movimiento se utiliza "SPEED" seguido por el valor de velocidad deseado, en este caso determinado por una posición de un vector. Para determinar a qué posición se desea mover el motor se utiliza "MOVEA" la cual mueve en posición absoluta los motores a la posición requerida, igualmente determinada por un vector. La instrucción Pause IDLE espera a que los motores que están en movimiento terminen el mismo y cuando lo hagan, seguir adelante con el código. Se logra ver que las instrucciones "SPEED " y "MOVEA" tiene un punto seguido por un número, éste indica que motor se desea mover (debe corresponder al motor

deseado para que la trayectoria sea la deseada), en el caso de "Pause IDLE" se utilizan corchetes separados por comas para determinar cuáles motores van a depender de esa instrucción.

SPEED[0]=velocidad_x(1)
SPEED[1]=velocidad_y(1)
MOVEA.0=x_pos(1) : GO
MOVEA.1=y_pos(1) : GO
Pause IDLE[0,1]

3.3.5 Ciclo de Ejecución

El objetivo de este tramo del código es leer los datos anteriormente declarados y ejecutar el movimiento hasta el final, para lo cual se utiliza un ciclo, al principio del ciclo se define desde y hasta que posición y a que paso se desea que corra el ciclo en este caso de uno en uno (for a=1 to (número de datos) step 1), después define el valor de velocidad para cada ángulo (SPEED) tomándolo del vector de velocidad anteriormente definido.

La instrucción "INCA" permite llevar a una posición indicada con una velocidad definida a una tasa de aceleración o desaceleración fija o variable, en este caso la aceleración y desaceleración son fijas, en el caso de la velocidad varía para cada posición. La variación de la velocidad y las distintas posiciones se hace por medio del ciclo con la variable "a" la cual incrementa cada vez que realice un movimiento completo permitiendo que la velocidad y la posición a la que se desea llevar sea la siguiente deseada. Se utiliza la instrucción "Pause IDLE" que espera a que los motores terminen un movimiento antes de seguir y "NEXT" para llevar en ciclo a la posición siguiente.

For a=1 To (número de valores) 300 Step (paso)
SPEED[0]=velocidad_x(a)
SPEED[1]=velocidad_y(a)
INCA.0=x_pos(a+1) : GO
INCA.1=y_pos(a+1) : GO
Pause IDLE[0,1]
Next

4 PRUEBAS EXPERIMENTALES EN LOS MOTORES

Para verificar la validez de los desarrollos efectuados, es necesario realizar pruebas experimentales, se deben comparar los valores deseados de posición con los arrojados experimentalmente por los motores. A continuación se mostrarán varios ejemplos de datos reales tomados de los motores. Debido dificultades con la mesa cartesiana que estaban más allá del alcance de este proyecto, las pruebas ejecutadas fueron hechas en los motores del robot paralelo de UPB. Como los motores son de la misma marca y el controlador es el mismo los experimentos realizados de esta forma cumplen con los objetivos de este proyecto. Los datos arrojados por los motores al realizar las distintas trayectorias fueron guardados y posteriormente interpretados en Matlab® para ser mostrados de una manera más cómoda y poder compararlos con las trayectorias deseadas.

El código de Mint Workbench requiere de la aceleración, los ángulos y velocidades entre cada par de ángulos de la trayectoria deseada, éstos son encontrados por medio de programa en Matlab®, donde el usuario tiene la opción de ingresar la aceleración, ángulos y el tiempo entre cada uno de ellos manualmente como se muestra en el ejemplo 1. Otra opción es introducir la ecuación de la trayectoria deseada, ingresar la aceleración deseada del movimiento, decidir qué tan fina (en cuantos puntos se divide toda la trayectoria) quiere que sea y luego ingresar el tiempo que desea entre cada punto (teniendo en cuenta que entre más fina el tiempo entre cada punto será menor). Las trayectorias de los ejemplos 2 y 3 fueron obtenidas de esta forma. El código de Matlab® arroja las gráficas teóricas y los valores necesarios para introducir al código de Mint Workbench y de este modo poder ejecutar la trayectoria deseada.

En los ejemplos son entregados datos con unidades dados en grados, debido a que los datos que recibe el programa de Mint Work Bench son ángulos, la relación entre los motores y el movimiento en ejes de la mesa es igual a 360° de giro de un motor equivalen a 6 centímetros de movimiento axial del eje,, la relación se cumple entre los motores 1 y 2 y los ejes X y Y respectivamente, esta relación se puede encontrar en los ejemplos 2 y 3, la ecuación que describe la relación mencionada es:

$$\theta[Grados] = X[cm] * \frac{360[Grados]}{6[cm]} \qquad (12)$$

X= Movimiento en el eje real de la mesa cartesiana [cm]

ɵ= Ángulos del motor [Grados]

4.1 EJEMPLO 1

En este primer ejemplo se desea que un motor desacoplado de la mesa, ejecute una serie de ángulos que dan lugar a una trayectoria arbitraria, dando valores de tiempo distintos cada ángulo. En la Tabla 1, se pueden encontrar los valores de los ángulos, tiempo entre ángulos y velocidad para cada par de ángulos. Los valores de los ángulos y tiempos entre cada par de ángulos fueron dados al azar por el usuario mientras que los valores de velocidad fueron encontrados por el código de Matlab®. La aceleración con la que se calculó el movimiento fue de 300 grados/s^2.

Ángulo [Grados]	Tiempo [s]	Velocidad [Grados/s]
0		
200	3.5	58.78
-100	3	-100
70	2.5	85
0	1	-80.91

Tabla 1.Datos requeridos ejemplo 1.

Fuente propia

En la Figura 9 se logra ver los datos experimentales de posición y tiempo arrojados por el motor. Es claro que se cumplen los tiempos y los ángulos especificados en la

.

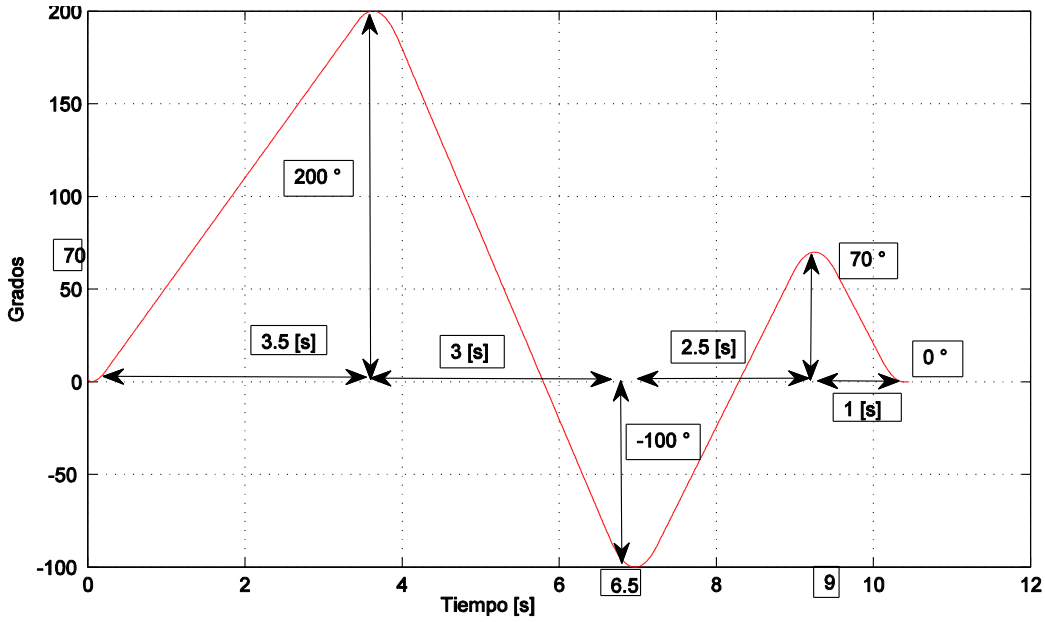

Figura 9. Posición motor trayectoria aleatoria.

Fuente propia

En la Figura 10 se aprecian los datos experimentales de velocidad arrojados por el motor, esta permite comparar los teóricos de la Tabla 1 con los experimentales y como se observa se cumple con las velocidades deseadas en los tiempos requeridos.

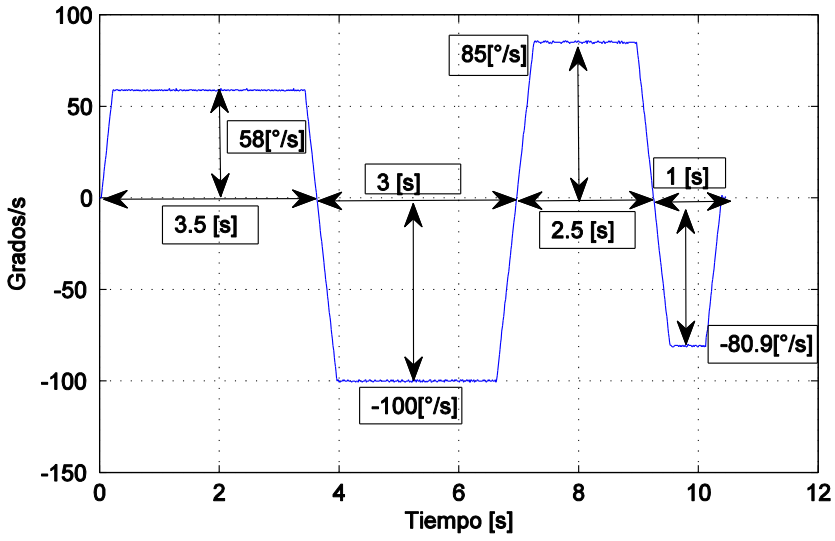

Figura 10.Velocidad Motor 1 trayectoria aleatoria.

Fuente propia

Al igual que los datos de posición y velocidad en la Figura 11 se muestran los datos de aceleración del motor y se observa que tanto la aceleración como la desaceleración cumplen el tiempo y toman ambos valores de 300°/s^2, que fueron los valores deseados.

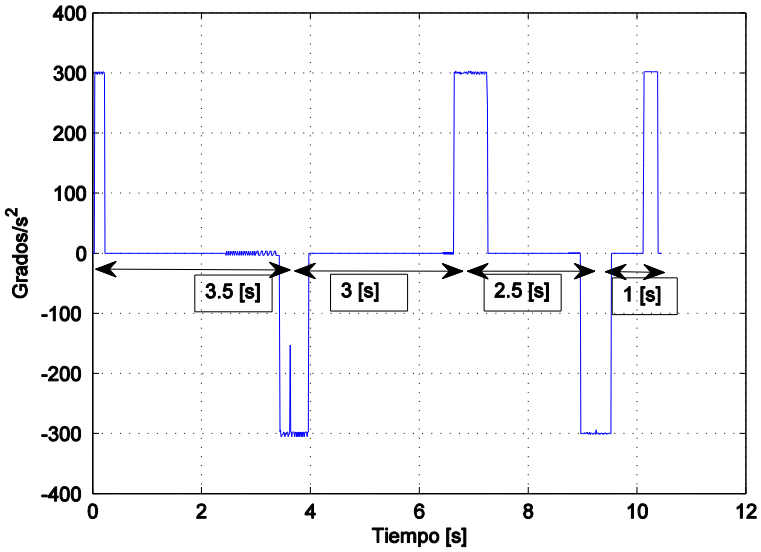

Figura 11.Aceleración Motor 1 trayectoria aleatoria.

Fuente propia

De este ejemplo resulta claro que el movimiento cumple con todas las características deseadas, los movimientos son suaves y sin picos, los tiempos deseados entre cada movimiento se cumplen, las velocidades que entrega el código de Matlab® son las que alcanza los motores y la aceleración es la descrita. Se concluye que el movimiento deseado fue el realizado efectivamente por el motor.

4.2 EJEMPLO 2

En este ejemplo se ejecutó una trayectoria senoidal, se quiere mostrar que con el programa se puede obtener cualquier trayectoria deseada. En este caso la trayectoria que se hizo fue Y=10*sin(4*X) donde la combinación de los movimientos de dos motores deben generan la trayectoria deseada. Para que la combinación de los dos motores puedan llevar a cabo la trayectoria senoidal es necesario que el motor del eje X ejecute un movimiento oscilatorio, simultáneamente el motor del eje Y debe variar su posición en forma creciente . La trayectoria esta descrita por 200 puntos y el tiempo deseado entre par de puntos es 0.2 segundos. Según lo

anterior el movimiento debe demorar 40 segundos en ejecutarse. En las graficas es necesario recordar que por cada 360° el eje de la mesa cartesiana avanzaría 6 centímetros, para ambos ejes.

Las velocidades fueron calculadas por el código de Matlab® e ingresadas posteriormente a Mint Workbench En la Figura 12 se pueden ver los movimientos experimentales del motor del eje X, contra los deseados, en la Figura 13 se observan los movimientos deseado y experimental del motor del eje Y y en la Figura 14 se puede observar la combinación de los datos de posición de los dos motores mostrando la trayectoria deseada y la realizada.

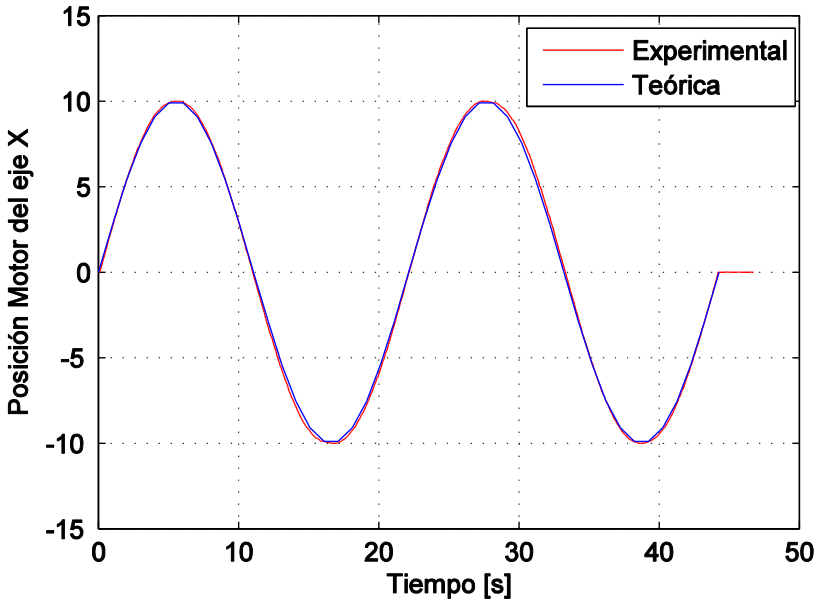

Figura 12. Posición motor eje X, trayectoria senoidal.

Fuente propia

Como se logra observar en la Figura 12 la línea experimental es muy aproximada a la línea teórica, cumpliendo lo deseado por parte del primer motor, se aprecia que el movimiento fue ejecutado en aproximadamente 40 segundos, como fue calculado.

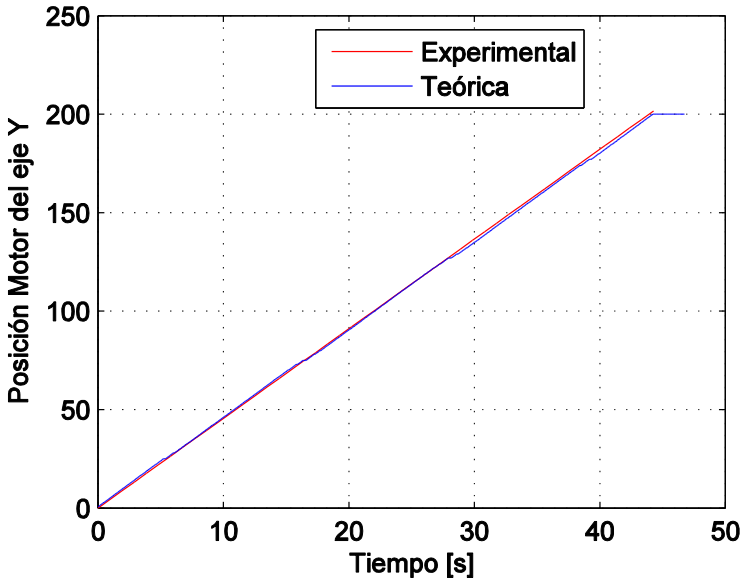

Figura 13. Posición motor eje Y, trayectoria senoidal.

Fuente propia

En la Figura 13 se observa los datos teóricos contra los experimentales del motor 2, se logra observar que la posición deseada es trazada por el motor e igualmente cumple con el tiempo estipulado.

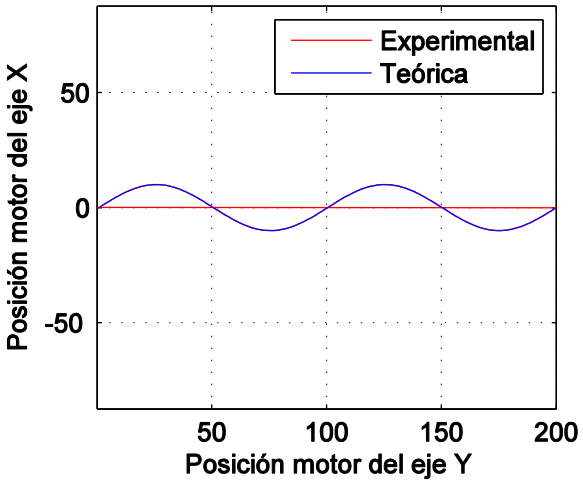

Figura 14.Posición Motor eje X vs Motor eje Y, trayectoria senoidal.

Fuente propia

En la Figura 14 se logra ver la combinación del movimiento simultáneo generado por los motores del eje X y Y. Se puede observar que la curva realizada pasa con buena exactitud por la curva deseada mostrando que la trayectoria deseada se realizó con éxito. Es de notar que al principio de la gráfica se ve un tramo de línea experimental, este se puede explicar ya que el programa graba todos los movimientos desde que el motor empieza a correr y lo primero que hacen los motores es llegar a la posición inicial. Se puede concluir que ese tramo de línea es el trayecto que se mueven los motores para llegar al primer punto de la trayectoria deseada.

Para efectos de ilustrar los datos de velocidad de los motores en la Figura 15 y Figura 16 se muestran la graficas de velocidad de un par de puntos específicos de la trayectoria en cada motor, no se muestra completa que ya en su conjunto no resulta explicativa.

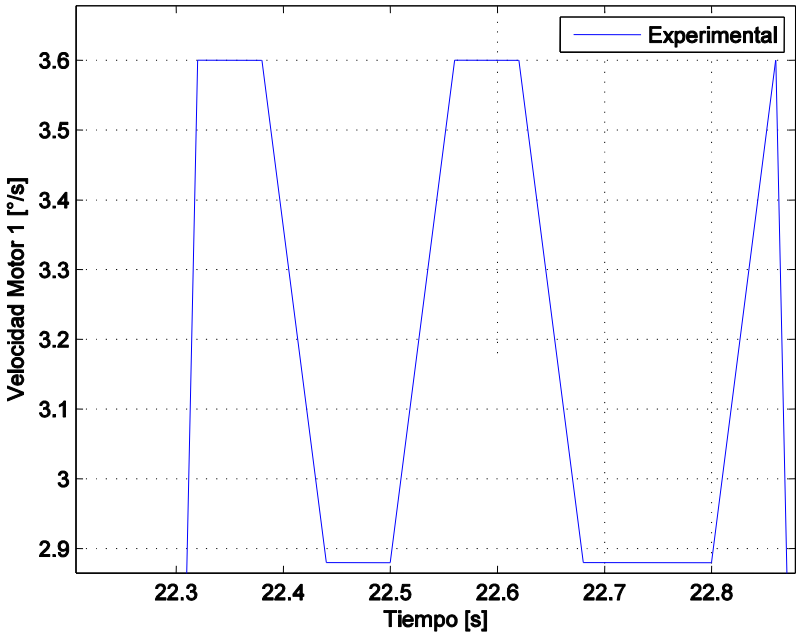

Figura 15. Velocidad motor del eje X, en el intervalo de 22.3 a 22.7 segundos.

Fuente propia

Como se logra observar en la Figura 15, la velocidad logra ejecutar la rampa ideal pasando por un pequeño intervalo de velocidad constante.

En la Figura 16, se observa que en ciertos tramos la los valores de velocidad llegan un valor pero no alcanzan a tomar un valor de velocidad constante ya que el tiempo para ejecutar ese punto pudo haber sido muy corto, de igual manera se aprecian tramos donde si es alcanzada el tramo de velocidad constante.

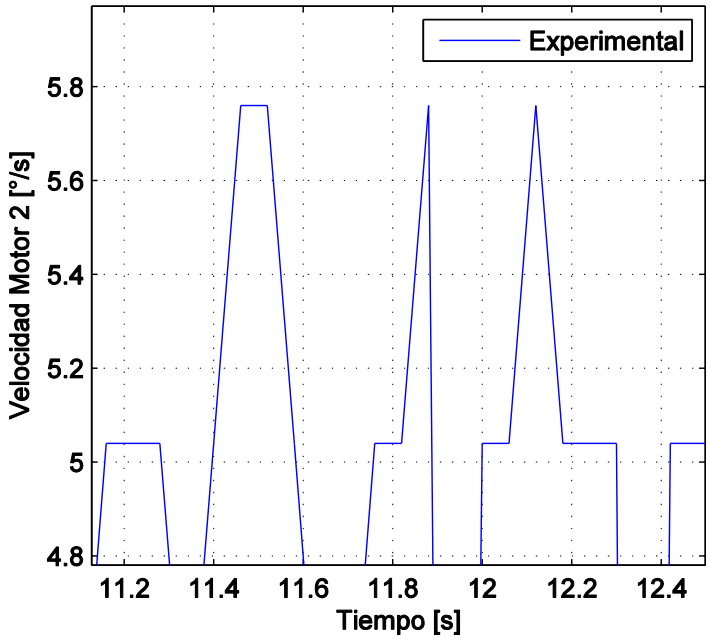

Figura 16. Velocidad motor del eje Y, en el intervalos de 11.2 a 12.6 segundos.

Fuente propia

4.3 EJEMPLO 3

En este ejemplo se presenta una trayectoria donde la combinación de los movimientos de los dos motores debe ejecutar la trayectoria de un círculo de radio de 0.83 centímetros que con la ayuda de la ecuación (12) equivalen a 50 grados, en un tiempo de 40 segundos. Para ejecutar la tarea deseada, cada uno de los motores deberá pasar por una serie de ángulos positivos y negativos durante su ciclo de trabajo. La trayectoria teórica del círculo fue obtenida por medio de la ecuación que lo describe en coordenadas paramétricas, para el eje X, X=radio*cos(th) y para el eje Y, Y=radio*sen(th), en este caso el radio seria 50 y "th" tendría un valor de 200 que equivale el número de puntos en los que se divide la trayectoria.

Los movimientos teórico y experimentales del motor del eje X se muestran en la Figura 17 y los del motor del eje Y en la Figura 18. La trayectoria deseada puede apreciarse en la Figura 19.

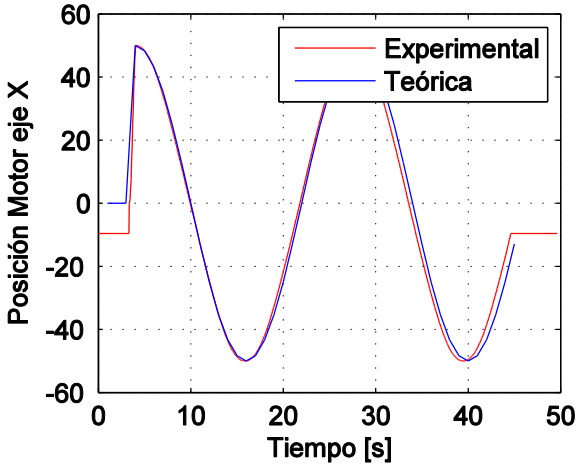

Figura 17. Posición motor del eje X, trayectoria círculo.

Fuente propia

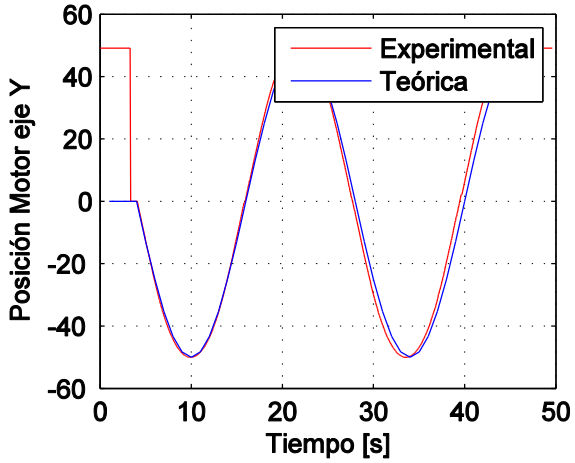

Figura 18. Posición motor del eje Y, trayectoria círculo.

Fuente propia

Tanto en la Figura 18 como en la Figura 19 se observa que la curva experimental sigue la teórica satisfactoriamente. Al principio de la gráfica se observan unas líneas que no corresponden a la

trayectoria del círculo, estas se deben a los trazos que debe de dar el motor para llegar a la primera posición deseada. La duración total del movimiento es de aproximadamente de 40 segundos.

En la Figura 19 se observa la combinación de los movimientos de los motores de los ejes X y Y, obteniendo la trayectoria del círculo de radio 50 unidades deseada. Se logra ver que los motores cumplen con la trayectoria. La línea que se ve llegando al centro del círculo es el movimiento que deben hacer los motores para llegar al primer punto de la trayectoria deseada.

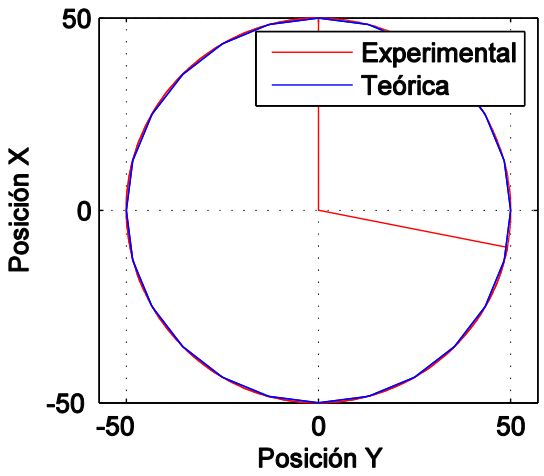

Figura 19. Posición Motor del eje X vs Motor del eje Y, trayectoria círculo.

Fuente propia

5 SUGERENIAS

Para el funcionamiento de la mesa cartesiana, es necesario ejecutar un código en Matlab® y luego pasar los resultados a la plataforma MintWorkbench, de esta manera la mesa cartesiana funciona como se especifica en este trabajo.

Para hacer el manejo de la mesa más ágil y amigable con el usuario, se recomienda implementar una interfaz gráfica en Matlab, de manera que se pueda manejar la mesa desde este programa sin tener que transportar los datos manualmente a Mintworkbench. Se debe tener en cuenta al momento de implementar la interfaz, que la mesa debe cumplir requerimientos básicos de una mesa cartesiana, además pensar que la mesa está en las instalaciones de la Universidad y por ende ésta va a ser objeto de estudio, tanto sus mecanismos como los movimientos ejecutados por esta.

Teniendo en cuenta las características de la mesa y el uso que se le va a dar, se recomienda que la interfaz gráfica programada en Matllab tenga las siguientes características:

- Controles de velocidad, para aumentar y disminuir la velocidad de los movimientos.
- Control de aceleración, para manejar la aceleración deseada.
- Un control manual de la mesa, donde la mesa se pueda mover en distintas direcciones manualmente, tanto de un solo eje como en combinación de los mismos.
- Un control de paso de la mesa que permita aumentar o disminuir la distancia que se mueve al presionar el comando de movimiento manual.
- Una casilla donde se puedan ingresar las trayectorias deseadas.
- Avisos que arroje la misma interfaz cuando las trayectorias que se introduzcan no puedan ser ejecutadas por los parámetros físicos de la mesa.
- Diagramas visibles de variables como: movimiento, velocidad y aceleración de los tres ejes.
- Diagramas donde se pueda visualizar el movimiento de la mesa antes de ser realizado.

Además de las siguientes características, se pueden sumar las que el usuario recomiende o requiera al momento de realizar la interfaz. Siguiendo las especificaciones recomendadas la mesa cartesiana sería más amigable con el usuario y a la vez facilitaría el estudio en sus componentes y movimientos. En la Figura 20. Sugerencia de interfaz gráfica.Figura 20 se puede visualizar como se sugiere la interfaz gráfica.

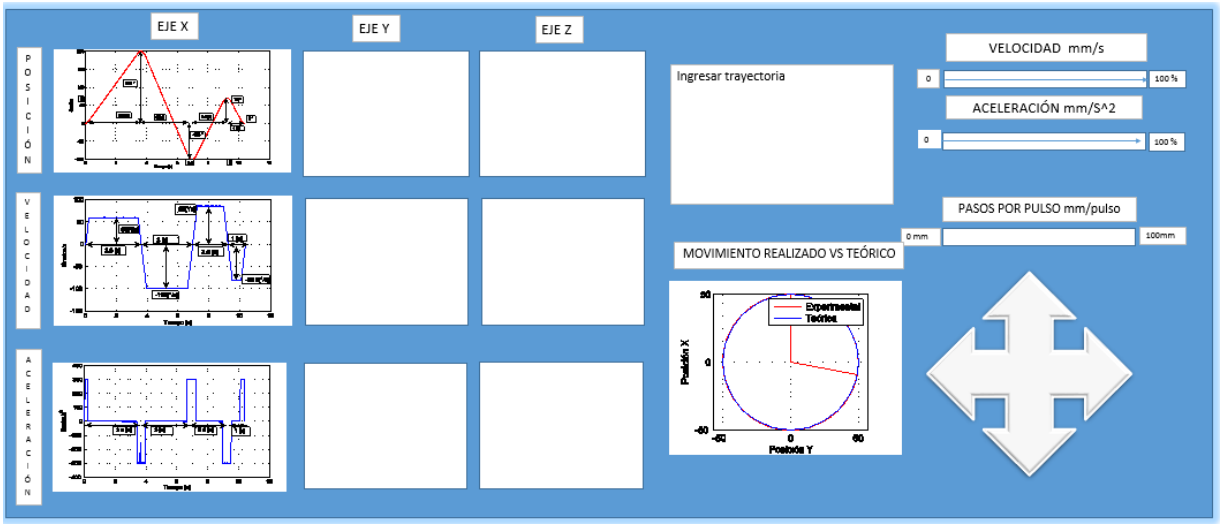

Figura 20. Sugerencia de interfaz gráfica.

(Fuente propia)

6 CONCLUSIONES

- La combinación de los programas entregados en este trabajo, permite lograr que el robot llegue a realizar cualquier trayectoria que este dentro de los límites del robot. Adicionalmente se puede conocer el tiempo que tarda toda la trayectoria en realizarse.
- Los movimientos realizados por medio del programa siguen curvas suaves y continuas.
- Estos programas funcionan para cualquiera de los motores del robot paralelo y el robot cartesiano, que se encuentran en el Laboratorio de Robótica de la UPB.
- Es necesario tener en cuenta los parámetros de los motores con los que se estén trabajando ya que los motores del cartesiano y del paralelo no son los mismos y por lo tanto las capacidades de velocidad y aceleración van a variar de uno al otro.

7 BIBLIOGRAFÍA

[1] S. Florez -Toro.,J. A. Ramírez -Macías, “DISEÑO MECÁNICO DE UNA MESA CARTESIANA XYZ” Universidad Pontificia Bolivariana., trabajo de grado,2010.

[2] S. Function, “Function : Fitting position : Carriage mounting : Unit mounting : Belt type : Formula : MLZ,” vol. 70000, no. mm, pp. 34–35.

[3] J.Criag,"trayectory generation". in *introduction to robotics*,A. Dworkin,USA, 2005, pp 212-216.

[4] S. Smartstep and E. Euroservo, “MINT ™ Programming Guide.”

[5] Ping-Lang Yen, Chi-Chung Lai, *Dynamic modeling and control of a 3.DOF cartesian parallel manipulator,Vol 1*, taiwan, 2009.

[6] *Mint workbench User Guide*,2.1.Baldor,UK,2000.

[7] ABB,"Host comms protocool 2″, aplication note,, ABB.UK, 2012.

[8] ABB,"Corner rounding", aplication note",aplication note, ABB, UK, 2012.

[9] P Smid, *CNC programing handbook*. second edition, I Press. New York, USA 2003.

[10] R. P. Paul, "Robot Manipulators: Mathematics, Programming, and Control", The MIT Press. Cambridge Massachussets, 1981.

[11] Bahr Modultechnik "Products" [online] 2014 Disponible en: http://www.bahr-modultechnik.de/index.php/en/products/product-galery

8 ANEXOS

En este capítulo se encuentran los anexos nombrados en el documento.

8.1 ANEXO 1

Código Mint Workbench

```
 TIMEREVENT=0
  CANCELALL
  RESETALL
 Dim a  ' Contador para ciclo
  Dim velocidad_x(300)
 Dim velocidad_y(300)
 Dim x_pos(300)
 Dim y_pos(300)
'ACCEL=300.0
'DECEL=300.0
'x_pos=0.0,200.0,-100.0,70.0,0.0
'velocidad_x=58.789,100.000,85.000,80.911
' Primer punto.
SPEED[0]=velocidad_x(1)
SPEED[1]=velocidad_y(1)
MOVEA.0=x_pos(1) : GO
MOVEA.1=y_pos(1) : GO
 Pause IDLE[0,1] '
 For a=1 To 300 Step 1
SPEED[0]=velocidad_x(a)
SPEED[1]=velocidad_y(a)
 INCA.0=x_pos(a+1) : GO
```

INCA.1=y_pos(a+1) : GO

Pause IDLE[0,1] '

Next

8.2 ANEXO 2

```
%Via Points
clear all
close all
clc

%% Ingreso de datos

n=input('ingrese # de ángulos ' );
ac(1)=input('ingrese aceleración ');

for i=1:n
    num=num2str(i);
    text='ingrese ángulos_';
    textFull=strcat(text,num,'=');
    th(i)=input(textFull);
end
for i=1:n-1
    ang=num2str(th(i));
    ang_s=num2str(th(i+1));
    text_2='ingrese el tiempo entre los ángulos ';
    texcom=strcat(text_2 ,'',ang,' y ',ang_s,'=');
     td(i)=input(texcom);
end
%% Ejemplo Ingresando ttrayectoria

% trayectoria sin
% th=linspace(0,4*pi,200);
% y=10*sin(th);
% figura_sin=plot(y);
% x=get(figura_sin,'XData');
% y=get(figura_sin,'YData');
% % % para x
% %  th=x;
% %  td(1:length(x))=0.0005;
% %  ac(1)=150;
```

```
% % % %para y
% th=y;
% td(1:length(x))=0.001;
% ac(1)=150;
%% círculo
% r=50;
% th=linspace(4*pi,0,200);
% x=r*cos(th);
% y=r*sin(th);
% plot(x,y)
% axis equal
% grid on
% % %para x
%  th=x;
%  td(1:length(x))=0.001;
%  ac(1)=150;
% % %para y
% % th=y;
% % td(1:length(x))=0.001;
% % ac(1)=150;

%%
%incial
n=size(th,2);
ac(1)=sign(th(2)-th(1))*abs(ac);
ta(1)=td(1)-sqrt(td(1)^2-(2*(th(2)-th(1))/ac(1)));
v(1)=(th(2)-th(1))/(td(1)-0.5*ta(1));
%
%puntos medios
%velocidades medias
for i=2:n-1
 v(i)=(th(i+1)-th(i))/td(i);
end

%aceleración medias
for i=2:n-1
    ac(i)=sign(v(i)-v(i-1))*abs(ac(1));
end
%tiempos de acleleración
for i=2:n-1
    ta(i)=(v(i)-v(i-1))/ac(i);
end
%tiempo del 1 tramo lineal
t(1)=td(1)-ta(1)-0.5*ta(2);
```

```
%tiempo de n-2  tramos lineales
for i=2:n-2
    t(i)=td(i)-0.5*ta(i)-ta(i+1);
end

%último tramo
ac(n)=sign(th(n-1)-th(n))*abs(ac(1));
ta(n)=td(n-1)-sqrt(td(n-1)^2+(2*(th(n)-th(n-1))/ac(n)));
v(n-1)=(th(n)-th(n-1))/(td(n-1)-0.5*ta(n));
t(n-1)=td(n-1)-ta(n)-0.5*ta(n-1);
% resultado
mat.theta=th;
mat.vel=v;
mat.tiempo=td;
mat.acel=ac;
mat.t_acel=ta;
mat.t_lineal=t;
mat;
vel=abs(v);
fprintf('ACCEL=%3.1f,',ac(1))
fprintf('\n')
fprintf('DECEL=%3.1f,',ac(1))
fprintf('\n')
fprintf('x_pos=')
fprintf('%3.1f,',th)
fprintf('\n')
fprintf('velocidad=')
fprintf('%3.3f,',vel)
fprintf('\n')
```