UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación de controladores difusos en ROS para la postura y caminata de un Robot Humanoide

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Edgar Muñoz Vivero

DIRECTOR DE TESIS

M. en A. Adalberto Joel Durán Ortega

Ciudad Universitaria, Cd. Mx., 2019

Dedicatorias

A mi papá Rogaciano Muñoz Bernal y a mi mamá Angela Vivero Mejía por su trabajo, paciencia y sacrificio en todos estos años, gracias por darnos la mejor herencia a mi hermano y a mi, la educación.

A mi Abuelita Guillermina Mejía Cruz, por sus consejos, apoyo incondicional y por tratar de que siempre fuera una mejor persona.

A mi Abuelita Juana Bernal, aunque ya no se encuetra a nuestro lado, gracias por la oraciones y por preocuparse por mi futuro.

A mi tío Eleazar Vivero Mejía y a mi tía Sara Tellez Velázquez gracias por preocuparse por mi formación y educación.

A mi hermano Francisco y mi primo David por su apoyo incondicional y por estar en todo momento conmigo gracias.

A toda mi familia porque con sus consejos y palabras de aliento hicieron de mi una mejor persona y gracias por acompañarme en todos mis sueños y metas.

Agradecimientos

A la Universidad Nacional Autónoma de México por darme la oportunidad de tener una educación, lo cual me ha hecho una mejor persona.

A la Facultad de Ingeniería por darme las herramientas científica, técnica y cultural para poder desempeñar la profesión en beneficio a la sociedad.

Al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas por darme la oportunidad de contribuir con un granito de arena en el conocimiento.

*Al el Laboratorio **UNAMoids** por darme la oportunidad de aplicar mis conocimientos adquiridos en la **UNAM**.*

A el M. en A. Adalberto Joel Durán Ortega por sus consejos, paciencia y tiempo que me dedicó durante este proceso.

Índice general

Capítulo 1

Introducción

Un robot humanoide es un robot que tiene la forma de un humano el cual esta destinado a realizar tareas peligrosas o repetitivas e incluso tareas dentro de la medicina, relaciones sociales, exploración espacial, etc. Para que algún robot humanoide desempeñe estas actividades debe de ser capaz de realizar movimientos similares a los de un humano por eso es interesante investigar en esta área de la robótica porque se abordan diferentes campos como la visión artificial, inteligencia artificial, caminata bípeda, etc.

La motivación de construir robots humanoides es debido a que la mayoría de las cosas que nos rodean están diseñadas con base a la geometría y movimientos de los humanos como por ejemplo las puertas, los escalones, las cabinas de control de alguna máquina así que se podrán introducir robots humanoides a la sociedad si tienen la forma de un humano [1].

Los humanos pueden realizar varios y complicados movimientos porque el cuerpo consiste de cartílagos, huesos, articulaciones, ligamentos, muscúlos y tendones [2] por eso es complicado realizar la caminata bípeda de un humano en un robot humanoide.

La caminata bípeda tiene que ser estable para el robot humanoide ya que esto permitirá realizar diferentes tareas o actividades, para lograr esto se utilizan algoritmos de control que permiten llevar al robot a regiones estables y dependiendo de la ley de control utilizada se requerirá modelar matemáticamente al robot si esta lo necesita. El modelo matemático se complica si se desea modelar la cinemática y dinámica de cada articulación por eso es que se buscan métodos para poder simular y modelar los movimientos del robot humanoide.

Al igual que en la mecánica celeste, los investigadores aproximan al sol y a los planetas como masas puntuales mientras que conservan sus propias estructuras internas y aún pueden calcular las órbitas del sistema solar con suficiente precisión [3, pág. 107], utilizando esta analogía con el robot humanoide se modelará su movimiento suponiendo un péndulo invertido en 2D como se muestra en la Figura 1.1.

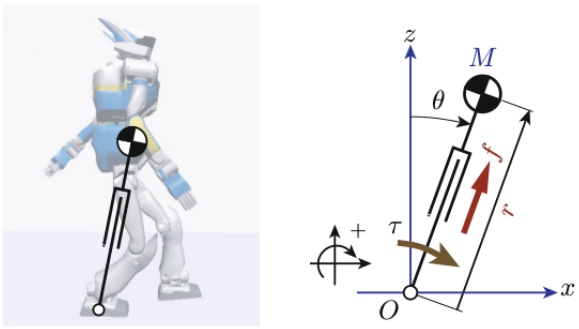


Figura 1.1: Péndulo invertido 2D: Es un modelo simple para la caminata de un robot. consiste en el centro de masa (CoM) y piernas telescópicas, θ es el ángulo de inclinación del péndulo[3, pág. 107]

El robot de la Figura 1.1 está simulado y modelado mediante un péndulo telescópico, en esté trabajo solo se tomará el péndulo invertido simple para controlar la pierna fija a la tierra cuando esté en marcha bípeda y también se utilizará esta suposición para estabilizar, controlar las dos piernas y la cadera del robot cuando no esté en marcha bípeda.

Existen diferentes algoritmos de control o leyes de control que se pueden utilizar para facilitar la manipulación del robot humanoide, estás tienen el digrama básico mostrado en la siguiente Figura 1.2. las técnicas para el diseño de las leyes de control tiene tres clasificaciones [4] las cuales se muestran en el diagrama de la Figura 1.3.

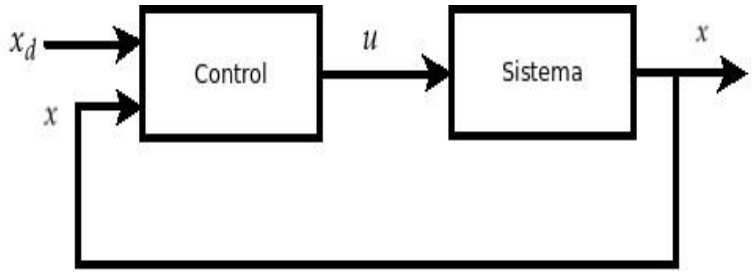


Figura 1.2: x_d punto de operación, x variables de estados,u salida del controlador

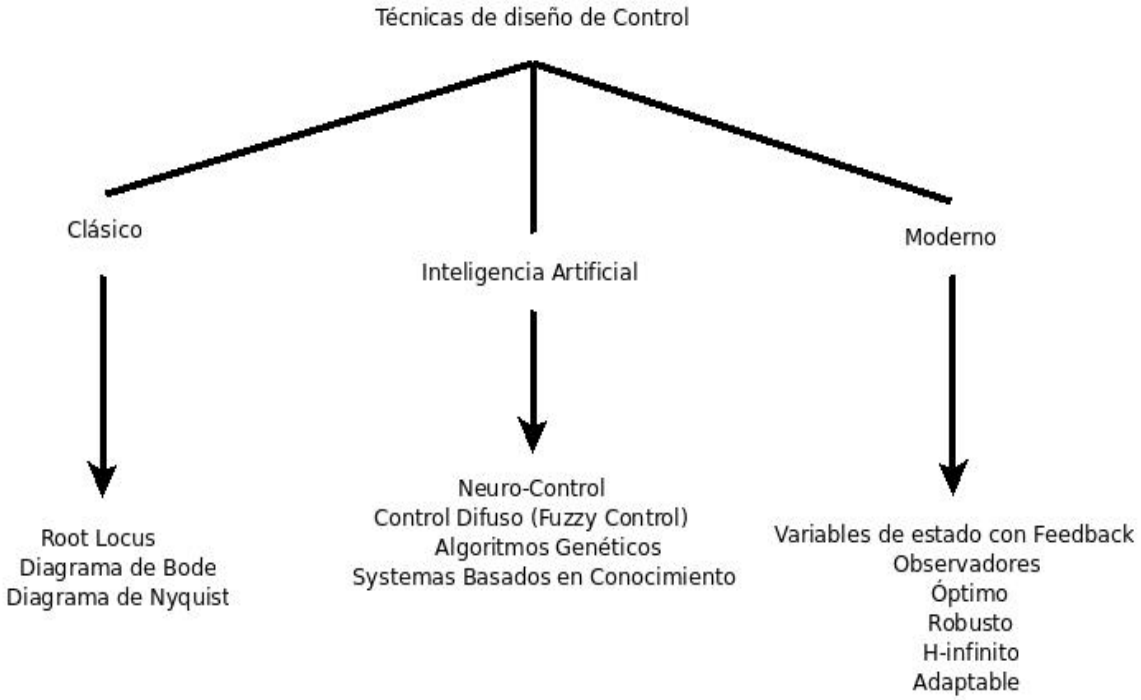


Figura 1.3: Clasificación de las técnicas de control

Entre las diferentes técnicas para diseñar leyes de control existen algunas que no requieren forzosamente un modelo matemático del sistema para poder calcular el controlador como por ejemplo el control difuso no se requiere forzosamente, en cambio las técnicas de control clásico o moderno si lo necesitan, el control de lógica difusa (**FLC**) se puede utilizar en procesos complejos donde pueden ser controlados por la experiencia de un humano sin ningún conocimiento de la dinámica del sistema o proceso[4, pag. 6]. Existen diferentes enfoques para diseñar controladores difusos como por ejemplo inferencia de Mamdani o Takagi-Sugeno, en la Figura 1.4 se muestra un diagrama de control más específico para un **FLC**.

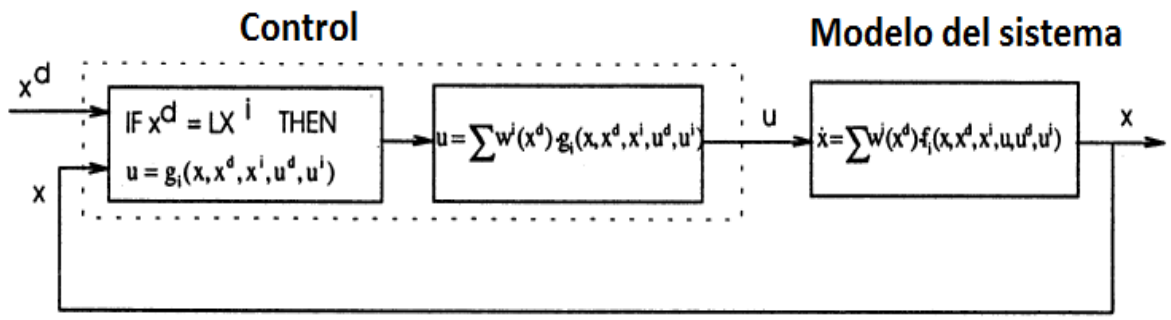
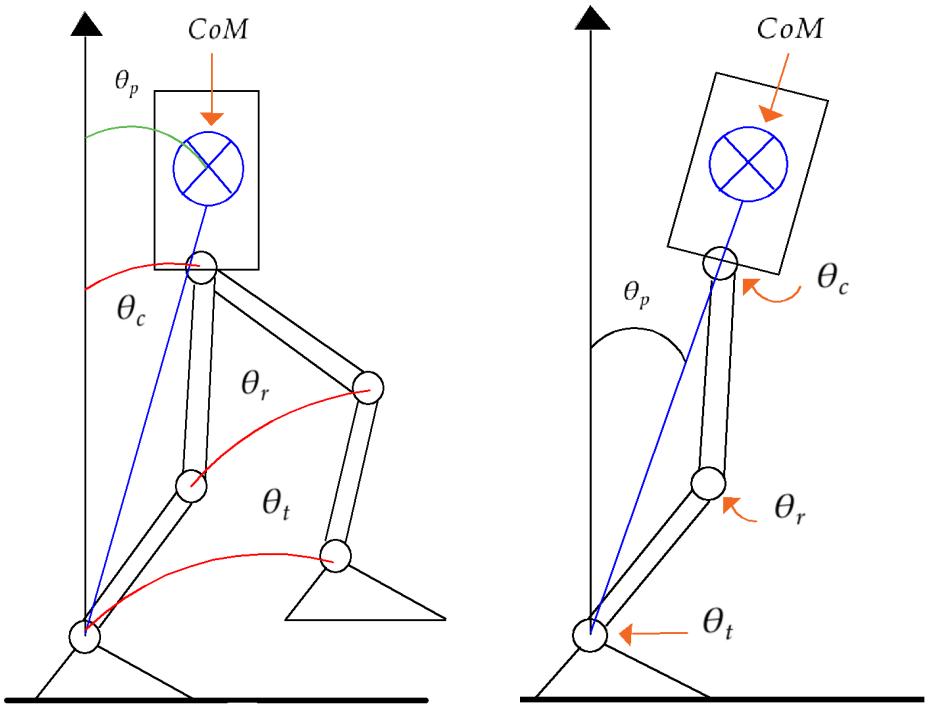


Figura 1.4: Diagrama de **FCL** basado en Takagi-Sugeno [5]

El control de la estabilidad de la postura y caminata bípeda involucra dos categorias de un sistema de control y estas son:

1. **Estabilización**: consiste en diseñar un sistema en lazo cerrado (Feedback) que permita estabilizar al robot humanoide alrededor de un punto, este punto también es llamado punto de operación o set-point, un ejemplo se muestra en la Figura 1.5 **b)** donde por medio del control se trata de llevar al robot a un punto estable mediante el modelo de un péndulo invertido.
2. ***Tracking* o Seguimiento**: consiste en diseñar un sitema en lazo cerrado que permita seguir una trayectoria variante con el tiempo para que el robot humanoide sea estable alrededor de la trayectoria. Las trayectorias pueden ser funciones conocidas como $sen(t)$, $cos(t)$, polinomios $P(t) = a_1 t^n + a_2 t^{n-1} + \cdots + a_{n-1} t + a_n$, etc. En la Figura 1.5 **a)** se muestra un ejemplo visual de las trayectorias posibles que el robot puede seguir durante el comienzo de la caminata.



(a) **Caminata bípeda** (b) **Estabilidad**

Figura 1.5: **a)** Ejemplo de trayectorias para caminata bípeda; péndulo($\theta_p = \theta_p(t)$), cadera($\theta_c = \theta_c(t)$), rodilla($\theta_r = \theta_r(t)$) y tobillo($\theta_t = \theta_t(t)$); Centro de Masa (CoM), **b)** Estabilidad de la postura mediante un péndulo invertido controlando los ángulos del péndulo(θ_p), cadera(θ_c), rodilla(θ_r) y tobillo(θ_t).

Con las leyes de control básicamente se quiere garantizar la estabilidad del robot para que sus movimientos sean suaves y lo más parecido a los de un humano, y que además ante perturbaciones externas el control tenga la capacidad de compensar y llevar al robot a regiones estables preprogramadas.

Capítulo 2

Justificación

El robot humanoide nombrado **BigBrother** (Figura 2.1) que está en el laboratorio de **UNAMoids** en el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (**IIMAS**) tiene varios propósitos uno de estos es participar en las competencias de Soccer RoboCup. El robot humanoide tendrá que realizar las actividades de portero o de alguna otra posición (defensa, medio o delantero) ver Figura 2.2 por eso es necesario tener una caminata y postura estables en todo momento durante la competencia para que el robot pueda tener un buen desempeño.

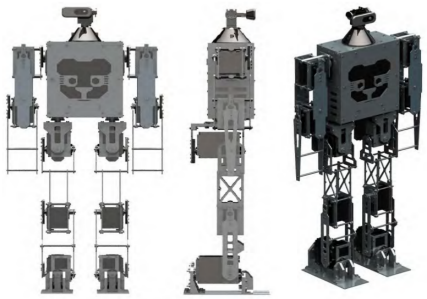


Figura 2.1: Robot Humanoide **BigBrother** [6]

Durante la competencia el robot está expuesto a choques con otros robots por lo que el control implementado para la postura debe tener la capacidad de compensar estas perturbaciones para que el robot tenga estabilidad en la competencia.
Cuando el robot detecta un balón este debe caminar hacia la dirección del balón por este motivo la caminata bípeda debe ser estable para que el robot no se tropiece y pueda llegar a la posición en la que se encuentra el balón dentro de la cancha.

La estabilidad de la caminata y postura del robot se garantizará mediante un algoritmo de control para un péndulo invertido basado en las características principales del robot como la concentración de la masa (6,77 $[Kg]$ [6]) y la altura del tobillo al centro de masa (0,525 $[m]$ [6]) del robot.
La solución del problema de diseño del algoritmo de control será mediante un controlador de lógica difusa, para cerrar el lazo de control se utiliza un sensor MPU6050 para la medición del ángulo del péndulo invertido.

Figura 2.2: RoboCup Humanoid League [7]

Capítulo 3

Objetivos, Hipótesis y Metas

3.1. Objetivos

1. **Objetivo General**
   Desarrollar los controladores difusos que permitan la postura erguida y caminata estable del robot humanoide.

2. **Objetivos Específicos**

   a) Calcular las trayectorias para la pierna fija al suelo y la pierna móvil durante la marcha bípeda.

   b) Análisis de la cinemática inversa para los movimientos del robot humanoide.

   c) Programar el controlador difuso para que pueda compensar y seguir las trayectorias solicitadas apoyándose del modelo de control del péndulo invertido.

3.2. Hipótesis

Con el uso del diseño de los controladores difusos basados en la inferencia de Takagi-Sugeno y la suposición de modelar matemáticamente al robot humanoide como un péndulo invertido, se podrá obtener la estabilidad para la postura y el seguimiento de trayectorias para la inclinación del robot humanoide y generar la caminata bípeda.

3.3. Metas

1. Calcular los polinomios con base en una regresión no lineal para relacionar los ángulos con los datos en microsegundos y de esa forma poder manipular los servomotores.

2. Diseñar filtros digitales para mejorar los datos obtenidos del sensor MPU 6050.

3. Emplear la cinématica inversa de un robot planar para generar los movimientos de la cadera y el tobillo del robot humanoide.

4. Calcular y aplicar las trayectorias para generar las inclinaciones con base en el modelo del péndulo invertido y los movimientos de la pierna móvil para que se pueda generar el patrón de caminata bípeda.

5. Calcular los controladores difusos de postura y seguimiento de trayectorias basados en la inferencia de Takagi-Sugeno con base en el modelo del péndulo invertido.

Capítulo 4

Matemáticas difusas utilizadas en control difuso

4.1. Conjuntos difusos

Zadeh [4] fue el que introdujó la teoria de conjuntos y la lógica difusa, esto ha tenido un desarrollo en varias direcciones, ahora la lógica difusa tiene aplicaciones en la ingeniería de control, procesamiento de señales, reconocimiento de patrones, toma de decisiones, etc.

Conjunto difuso

Definición **4.1** *Un conjunto es una colección de elementos o de objetos $x \in X$ esto puede ser finito o infinito. Cada elemento tiene una grado de pertenencia al conjunto difuso $\overline{A}$*

Función de membresía difusa

Definición **4.2** *X es una colección de objetos denotado genéricamente por x , entonces un conjunto difuso $\overline{A}$ en X es un conjunto de pares ordenados:*

$$\overline{A} = \{(x, \mu_{\overline{A}}(x)) \mid x \in X\} \tag{4.1}$$

donde $\mu_{\overline{A}}(x)$ es llamada la función de membresía asociada al conjunto difuso $\overline{A}$, $\mu_{\overline{A}}(x)$ es una función que mapea de los números reales al intervalo [0,1]:

$$\mu_{\overline{A}}(x) : \overline{A} \longrightarrow [0, 1] \tag{4.2}$$

Un conjunto difuso también puede ser representado usando las siguientes notaciones según [8]:

$$\overline{A} = \int_X \frac{\mu_{\overline{A}}(x)}{x} \qquad \overline{A} = \sum_{i=0}^{n} \frac{\mu_{\overline{A}}(x_i)}{x_i}, \ \forall \ i = 0, \ 1, \ 2, \ \cdots, \ n$$

La forma integral representa el caso continuo para el conjunto $\overline{A}$ donde su dominio es X. La sumatoria es la representación de un conjunto definido en forma discreta o un conjunto finito de n terminos en $\overline{A}$.

Ejemplo **4.1** *Una función de membresía triangular esta definida como una función a trozos:*

$$\mu_{\overline{A}}(x) = \begin{cases} \frac{x-a}{b-a} & si, \ a \leq x < b \\ \frac{-(x-c)}{c-b} & si, \ b \leq x \leq c \\ 0 & en \ otro \ caso \end{cases}$$

donde a, c son los extremos izquierdo o derecho respectivamente y b el centro de la función $\mu_{\overline{A}}(x)$ en la Figura 4.1 se muestra una función de forma triangular.

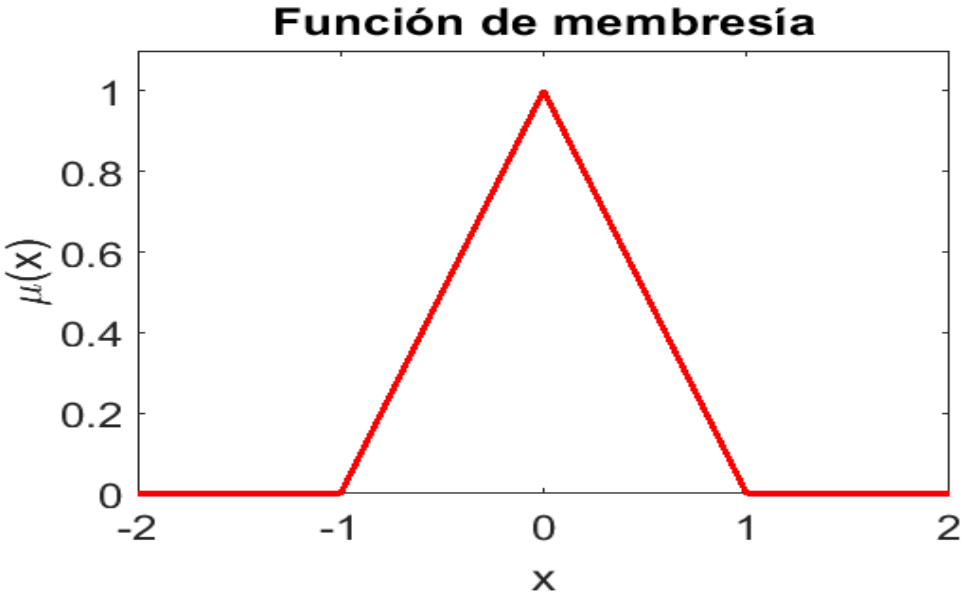


Figura 4.1: función de membresía con forma triangular a=-1, b=0, c=1.

Operaciones sobre conjuntos difusos

El complemento estándar puede ser extendido para un conjunto difuso $\overline{A}$ con respecto al conjunto universal X esta definido para toda $x \in X$ por la ecuación:

$$\mu_{\overline{A}}(x) = 1 - \mu_{\overline{A}}(x) \tag{4.3}$$

La unión y la intersección estándar también se pueden extender para los conjuntos difusos, dado dos conjuntos difusos $\overline{A}$ y $\overline{B}$ donde la intersección esta dada por $\overline{A} \cap \overline{B}$ y la unión por $\overline{A} \cup \overline{B}$, estas son definidas para toda $x \in X$ por las ecuaciones:

$$\mu_{\overline{A}}(x) \cap \mu_{\overline{B}}(x) = min\{\mu_{\overline{A}}(x), \mu_{\overline{B}}(x)\} \tag{4.4}$$

$$\mu_{\overline{A}}(x) \cup \mu_{\overline{B}}(x) = max\{\mu_{\overline{A}}(x), \mu_{\overline{B}}(x)\} \tag{4.5}$$

Definición **4.3** *[8] El soporte de un conjunto difuso $\overline{A}$, supp($\overline{A}$), es el conjunto clásico de todas las $x \in X$ tal que $\mu_{\overline{A}}(x) > 0$*

Definición **4.4** *[8] Un conjunto difuso $\overline{A}$ es convexo si:*

$$\mu_{\overline{A}}(\lambda x_1 + (1-\lambda)x_2) \geq min\{\mu_{\overline{A}}(x_1), \mu_{\overline{A}}(x_2)\},\ x_1,\ x_2 \in X,\ \lambda \in [0,1] \tag{4.6}$$

Considerando las siguientes conexiones básicas en la teoría de conjuntos difusos, donde las siguientes propiedades son verdaderas.

Involuntividad:

$$(\overline{A}^c)^c = \overline{A} \tag{4.7}$$

Conmutatividad:

$$\overline{A} \cup \overline{B} = \overline{B} \cup \overline{A} \tag{4.8}$$

$$\overline{A} \cap \overline{B} = \overline{B} \cap \overline{A} \tag{4.9}$$

Asociatividad:

$$(\overline{A} \cup \overline{B}) \cup \overline{C} = \overline{A} \cup (\overline{B} \cup \overline{C}) \tag{4.10}$$

$$(\overline{A} \cap \overline{B}) \cap \overline{C} = \overline{A} \cap (\overline{B} \cap \overline{C}) \tag{4.11}$$

Distributividad:

$$\overline{A} \cap (\overline{B} \cup \overline{C}) = (\overline{A} \cap \overline{B}) \cup (\overline{A} \cap \overline{C}) \tag{4.12}$$

$$\overline{A} \cup (\overline{B} \cap \overline{C}) = (\overline{A} \cup \overline{B}) \cap (\overline{A} \cup \overline{C}) \tag{4.13}$$

Idempotencia:

$$\overline{A} \cup \overline{A} = \overline{A} \tag{4.14}$$

$$\overline{A} \cap \overline{A} = \overline{A} \tag{4.15}$$

Ley de contradicción:

$$\overline{A} \cap \overline{A}^c = \emptyset \tag{4.16}$$

Ley del medio excluido:

$$\overline{A} \cup \overline{A}^c = X \tag{4.17}$$

De Morgan:

$$(\overline{A} \cap \overline{B})^c = \overline{A}^c \cup \overline{B}^c \tag{4.18}$$

$$(\overline{A} \cup \overline{B})^c = \overline{A}^c \cap \overline{B}^c \tag{4.19}$$

Identidad:

$$\overline{A} \cup \emptyset = \overline{A}, \quad \overline{A} \cap \emptyset = \emptyset \tag{4.20}$$

$$\overline{A} \cup X = X, \quad \overline{A} \cap X = \overline{A} \tag{4.21}$$

Absorción:

$$\overline{A} \cup (\overline{A} \cap \overline{B}) = \overline{A} \tag{4.22}$$

$$\overline{A} \cap (\overline{A} \cup \overline{B}) = \overline{A} \tag{4.23}$$

Observación: Las leyes de contradicción (4.16) y del medio excluido (4.17) fallan [9], si $\overline{A}$ es un conjunto difuso $\mu_{\overline{A}} : X \longrightarrow [0,1]$ (existen $x \in X$ con $\mu_{\overline{A}}(x) \notin [0,1]$) entonces:

$$\overline{A} \cap \overline{A}^c \neq \emptyset \tag{4.24}$$

$$\overline{A} \cup \overline{A}^c \neq X \tag{4.25}$$

4.2. Normas y Conormas T

Los operadores $max\{\}$ y $min\{\}$ utilizados en los conjuntos difusos son parte de las normas-T y conormas-T, ya que hay diferentes tipos de estas se estudiaran sus formas generales. Para esto primero comenzaremos con las normas-T.

Normas-T

Definición **4.5** *[9] Una norma-T es una función* $T = T_n(a,b)$, $0 \leq a,\ b,\ T \leq 1$

con las siguientes propiedades:

1. $T_n(a,1) = a$
2. $T_n(a,b) = T_n(b,a)$
3. si $b_1 \leq b_2$, entonces $T_n(a,b_1) \leq T_n(a,b_2)$
4. $T_n(a,T_n(b,c)) = T_n(T_n(a,b),c)$

Si $\overline{A}$ y $\overline{B}$ son subconjuntos de X y $\overline{C} = \overline{A} \cap \overline{B}$ entonces tenemos que:

$$\mu_{\overline{C}}(x) = T_n(\mu_{\overline{A}}(x), \mu_{\overline{B}}(x)) \tag{4.26}$$

La ecuación (4.26) es la forma general de una intersección en la teoría de los conjuntos y lógica difusa. A continuación se presentan algunas de las normas-T basicas:

Intersección estándar:

$$T_m(a,b) = min\{a,b\} \tag{4.27}$$

Suma limitada:

$$T_b(a,b) = max\{0, a+b-1\} \tag{4.28}$$

Producto algebraico:

$$T_p(a,b) = ab \tag{4.29}$$

Intersección drástica:

$$T^*(a,b) = \begin{cases} a & si,\ b=1 \\ b & si,\ a=1 \\ 0 & en\ otro\ caso \end{cases} \tag{4.30}$$

Conormas-T

Definición **4.6** *[9] Una conorma-T es una función* $C = C_n(a,b)$, $0 \leq a, b, C \leq 1$

con las siguientes propiedades:

1. $C_n(a,0) = a$
2. $C_n(a,b) = C_n(b,a)$
3. si $b_1 \leq b_2$, entonces $C_n(a,b_1) \leq C_n(a,b_2)$
4. $C_n(a,C_n(b,c)) = C_n(C_n(a,b),c)$

Si $\overline{A}$ y $\overline{B}$ son subconjuntos de X y $\overline{C} = \overline{A} \cup \overline{B}$ entonces tenemos que:

$$\mu_{\overline{C}}(x) = C_n(\mu_{\overline{A}}(x), \mu_{\overline{B}}(x)) \tag{4.31}$$

La ecuación (4.31) es la forma general de una unión en la teoria de los conjuntos y lógica difusa. A continuación se presentan algunas de las conormas-T basicas:

unión estándar:

$$C_m(a,b) = max\{a,b\} \tag{4.32}$$

Suma limitada:

$$C_b(a,b) = min\{1, a+b\} \tag{4.33}$$

Suma algebraica:

$$C_p(a,b) = a+b-ab \tag{4.34}$$

unión drástica:

$$C^*(a,b) = \begin{cases} a & si,\ b=0 \\ b & si,\ a=0 \\ 1 & en\ otro\ caso \end{cases} \tag{4.35}$$

En la aplicación de las normas-T y conormas-T se debe tener en cuenta una propiedad importante la cual es la dualidad. En otras palabras $T_n(a,b)$ y $C_n(a,b)$ deben de ser mutuamente duales, es decir:

$$T_n(a,b) = 1 - C_n(1-a, 1-b) \tag{4.36}$$

$$C_n(a,b) = 1 - T_n(1-a, 1-b) \tag{4.37}$$

Las siguientes $T_n(a,b)$ y $C_n(a,b)$ son duales:

- $T_m(a,b)$ y $C_m(a,b)$
- $T_b(a,b)$ y $C_b(a,b)$
- $T_p(a,b)$ y $C_p(a,b)$
- $T^*(a,b)$ y $C^*(a,b)$

Algunas normas y conormas ($T_n(a,b)$ y $C_n(a,b)$) son utilizadas en el control difuso ya que no todas la normas y conormas cumplen con las propiedades (4.7)-(4.23) y también tomando encuenta las propiedades (4.24) y (4.25), por ejemplo las normas (4.27) y (4.29) con sus conormas (4.32) y (4.34) cumplen con las propiedades antes mencionadas [9].

4.3. Relaciones difusas

Definición **4.7** *[10] X, Y son dos conjuntos clásicos. Un mapeo* $R : X \times Y \to [0,1]$ *es llamado una relación difusa. El número* $R(x,y) \in [0,1]$ *es interpretado como el grado de pertenencia entre x y y.*

El producto cartesiano puede ser definido como $X \times Y = \{(x,y) \mid x \in X \ \& \ y \in Y\}$ y utilizando la forma general de la intersección (4.26) se puede también representar a las relaciones difusas como:

$$R = \{(x,y) \mid x \in X,\ y \in Y :\ \mu_R(x,y) = T_n(x,y)\} \tag{4.38}$$

Donde $T_n(x,y)$ puede ser culquiera de las normas (4.27) o (4.29).
Una relación difusa de dos elementos dentro de dos conjuntos finitos $X = \{x_1,\ x_2,\ \cdots,\ x_m\}$, $Y = \{y_1,\ y_2,\ \cdots,\ y_n\}$ puede ser representada como una matriz.

$$R = \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \cdots & \mu_R(x_1,y_n) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & \cdots & \mu_R(x_2,y_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_m,y_1) & \mu_R(x_m,y_2) & \cdots & \mu_R(x_m,y_n) \end{bmatrix}$$

Operaciones sobre relaciones difusas
Sean R y S relaciones difusas con membresias $\mu_R(x,y)$ y $\mu_S(x,y)$

- Complemento: $\mu_R^c(x,y) = 1 - \mu_R(x,y)$
- Unión: $(\mu_R \cup \mu_S)(x,y) = \mu_R(x,y) \cup \mu_S(x,y)$
- Intersección: $(\mu_R \cap \mu_S)(x,y) = \mu_R(x,y) \cap \mu_S(x,y)$
- Inversa $\mu_R^{-1}(x,y) = \mu_R(y,x)$

Composición max-min

Definición **4.8** *[10] Conside dos relaciones difusas* $R \in X \times Y$ *y* $S \in Y \times Z$*, entonces* $R \circ S \in X \times Z$ *definida como:*

$$\mu_R \circ \mu_S(x,y) = \bigcup_{y \in Y} \mu_R(x,y) \cap \mu_S(y,z) \tag{4.39}$$

Para que esto quede más claro como se debe formar una composición de relaciones difusas lo haremos en notación indicial o en forma discreta para estos conjuntos finitos: $X = \{x_1,\ x_2,\ \cdots,\ x_n\}$, $Y = \{y_1,\ y_2,\ \cdots,\ y_m\}$, $Z = \{z_1,\ z_2,\ \cdots,\ z_p\}$. si $R = (r_{ij})_{i=1,\ldots,n, j=1,\ldots,m} \in X \times Y$ y $S = (s_{jk})_{j=1,\ldots,m, k=1,\ldots,p} \in Y \times Z$ entonces la composición $T = (t_{ik})i = 1,\ldots,n, k = 1,\ldots,p = R \circ S \in X \times Z$ esta dada por:

$$t_{ik} = \bigcup_{j=1}^{m} r_{ij} \cap s_{jk}$$

4.4. Reglas If-Then

Antes de empezar a definir las reglas if-then primero se tiene que estudiar las implicaciones difusas ya que también estas son fundamentales en la inferencia difusa.
Implicación difusa

Definición **4.9** *[10] Se define a la implicación difusa como una función* $I : [0,1] \times [0,1] \to [0,1]$ *que debe satisfacer:*

- I_1 :, si $x \leq y$, entonces $I(x,z) \geq I(y,z)$, es decir I es decreciente en su primera variable.
- I_2 :, si $y \leq z$, entonces $I(x,y) \leq I(x,z)$, es decir I es creciente en su segunda variable.
- $I_3 : I(1,0) = 0$, $I(0,0) = I(1,1) = 1$

Entonces I es llamada una implicacion difusa. La implicación tambien puede ser denotada como $x \to y$.
Estas son algunas operaciones que cumplen con de definición 4.9 [10]:

1. $I_1(x,y) = max\{1-x, y\}$
2. $I_2(x,y) = min\{1-x+y, 1\}$

Las reglas if-then necesitan de las variables lingüísticas para definir información antecedente y para la concecuencia, como no se ha introducido formalmente a las reglas if-then se puede visualizar como una implicación difusa de la forma antecedente (x) y concecuencia (y) $x \to y$. Entonces se define a las variables lingüísticas como:

Definición **4.10** *[10] Una variable lingüísticas es una quintupla*

$$(Z, T, U, G, M)$$

donde
Z es el nombre de la variable
T es el conjunto de terminos lingüisticos los cuales pueden ser valores de la variable
U es el universo de discurso
G es una colección de reglas que producen expresiones correctas en T
M es un conjunto de reglas semanticas que mapea T dentro de un conjunto difuso U

Ejemplo **4.2** *conciderando un ejemplo de una varible lingüistica* (Z, T, U, G, M) *[10]*
Z=Edad
$T = \{joven,\ muy\ joven, muy\ muy\ joven, \ldots\}$
$U = [0,1]$ *universo de discurso de la edad*
G : *las reglas peden ser expresadas como sigue: joven*$\in G$. *Si* $z \in G$ *entonces muy* $z \in G$.
$M : T \to \mathcal{F}(Z)$, $M(joven) = u$, *donde u=(0,0,18,40)*
$M(muy^n\ joven) = u^n(z)$

Las reglas if-then son para modelar una opinión de un experto o un cierto conocimiento a menudo se expresa en forma de términos lingüisticos. Las reglas if-then consisten de un antecesor $A \in X$ y una consecuencia $B \in Y$ estas son variables lingüisticas y se relacionan a través de una relación difusa $R \in X \times Y$, esta relación difusa tiene que ver con la Definición (4.9) ya que una implicación difusa es una relación difusa.
Con la ayuda de los conjuntos difusos las reglas difusas se pueden escribir de la siguiente forma:

$$\mathcal{R}^i = if\ x_1^i\ is\ A_1^i\ and\ x_2^i\ is\ A_2^i\ \ldots\ and\ x_n^i\ is\ A_n^i\ then\ y_m^i\ is\ B_m^i \tag{4.40}$$

Donde $\mathcal{R}$ es una regla difusa, i es la i-ésima regla difusa, x_1^i, x_2^i, …, x_n^i son las n-ésimas variables lingüisticas de la i-ésima regla difua asociadas a la parte if y y_m^i es la m-ésima variable lingüistica de la i-ésima regla difusa asociada a la parte then.
La implicación de una regla difusa puede ser representada como:

$$\mathcal{R}^i = (\ if\ x_1^i\ is\ A_1^i\ and\ x_2^i\ is\ A_2^i\ \ldots\ and\ x_n^i\ is\ A_n^i\ ) \to (\ y_m^i\ is\ B_m^i\ ) \tag{4.41}$$

$$\mu_A(x_1^i, x_2^i, \ldots, x_n^i) = \mu_{A_1^i}(x_1^i) \cap \mu_{A_2^i}(x_2^i) \cap \ldots \cap \mu_{A_n^i}(x_n^i) \tag{4.42}$$

$$R_{\mathcal{R}^i}(x_1^i, x_2^i, \ldots, x_n^i, y_m^i) = I\{\mu_A(x_1^i, x_2^i, \ldots, x_n^i), \mu_{B_m^i}(y_m^i)\}, \quad R_{\mathcal{R}^i} : [0,1]_1 \times \ldots \times [0,1]_n \times [0,1]_m \to [0,1] \tag{4.43}$$

4.5. Defusificación

La conclusión o la salida de un controlador difuso es una combinación de funciones de membresía de entrada, salida y reglas difusas, hasta este momento la salida es un valor difuso, para que tenga utilidad se debe convertir en una variable real a este proceso se le conoce como inferencia difusa, por eso es importante la defusificación. La defusificación es un proceso para transformar una variable difusa a una real [4].
Se mostrarán tres técnicas para defusificar:

Método del máximo

En este metodo se trata de calcular el promedio de las conclusiones o salidas difusas que tengan el mayor grado de pertenencia (Figura 4.2).

$$y_{\mathbb{R}} = \frac{\sum\limits_{x' \in T} x'}{T},\ y \in \mathbb{R},\ T = \{x' \mid \mu(x') = supp\{\mu(x)\}\} \tag{4.44}$$

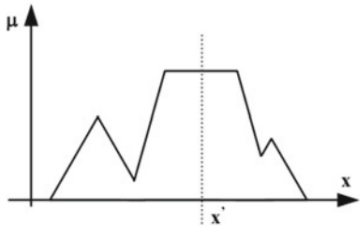


Figura 4.2: Defusificación por el método del máximo [4]

Método del centro de gravedad

Este método es uno de los más utilizados en diferentes aplicaciones. Es la misma fórmula para calcular el centro de gravedad en física, consiste en el promedio ponderado de funciones de membresía o el centro de gavedad para una región limitada (Figura 4.3).

$$y_{\mathbb{R}} = \frac{\sum_x \mu(x) \cdot x}{\sum_x \mu(x)} \tag{4.45}$$

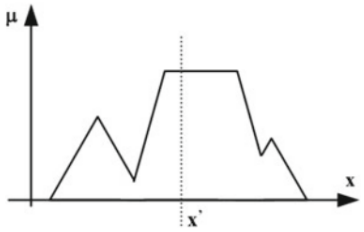


Figura 4.3: Defusificación por el método del centro de gravedad [4]

EL método HM (Height Method)

Esta técnica es solamente valida para cuando la concecuencia es una función que depende de las variables de entrada es decir $f_i(x_1, x_2, \ldots, x_n)$, además esta multiplicada y dividida por una variable de activación w_i. El i-ésimo termino esta asiciado a la i-ésima regla difusa.

$$y_{\mathbb{R}} = \frac{\sum\limits_{i=1}^{m} w_i f_i(x_1, x_2, \ldots, x_n)}{\sum\limits_{i=1}^{m} w_i} \tag{4.46}$$

4.6. Inferencia Difusa de Mamdani, Tsukamoto y Takagi-Sugeno

La inferencia difusa es un proceso donde se cálcula la toma de decisiones basadas en la lógica difusa donde se formulan mapeos no lineales del espacio de las entradas a el espacio de las salidas involucrando las funciones de membresía de estas, operadores de lógica difusa y las reglas if-then. En la aplicación para el control difuso se utilizan tres tipos de inferencia difusa las cuales son:

1. inferencia difusa de Mamdani
2. inferencia difusa Tsukamoto
3. inferencia difusa Takagi-Sugeno

Inferencia difusa de Mamdani
La inferencia de Mamdani (Figura 4.4) es el primer intento para un sitema de control, teniendo encuenta que los conjuntos difusos ($x \in A$, $y \in B$, $z \in C$) estan diseñados basados en la experiencia de un operador. La regla if-then básica para una inferencia de Mamdani es:

$$if\ x\ is\ A,\ y\ is\ B,\ then\ z\ is\ C$$

donde x y y son las variables de entrada y z la salida. A, B y C son los conjuntos asociados a las funciones de membresía ($\mu_A(x)$, $\mu_B(y)$, $\mu_C(z)$). La conclusión de una regla de Mamdani utiliza la composición max-min (ecuación (4.39)) o el max-product esto genera una realación difusa, esto consume mucho tiempo de prosesamiento ya que para cada valor que adquieran las variables x, y y z se tiene que calcular una relación difusa diferente.

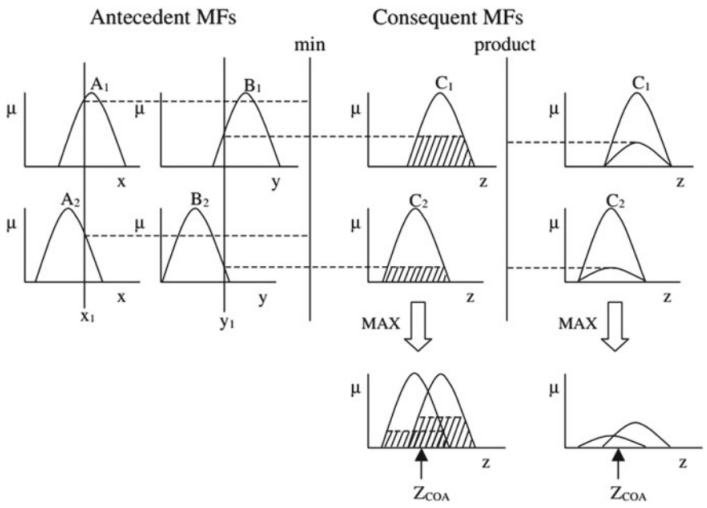


Figura 4.4: Inferencia difusa de Mamdani [4]

Inferencia difusa Tsukamoto
La regla de inferencia de Tsukamoto, las concecuencias de cada una de las reglas if-then están definidas por funciones de membresía monotonas, la inferencia de la salida esta definida en valores clásicos (z_i) para cada una de las reglas if-then. Donde la salida general es calculada mediante un promedio de las salidas de cada una de las reglas difusas (Figura 4.5) esto reduce el tiempo de procesamiento del cálculo, la ecuación del promedio para la salida esta dada como:

$$u = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i} \tag{4.47}$$

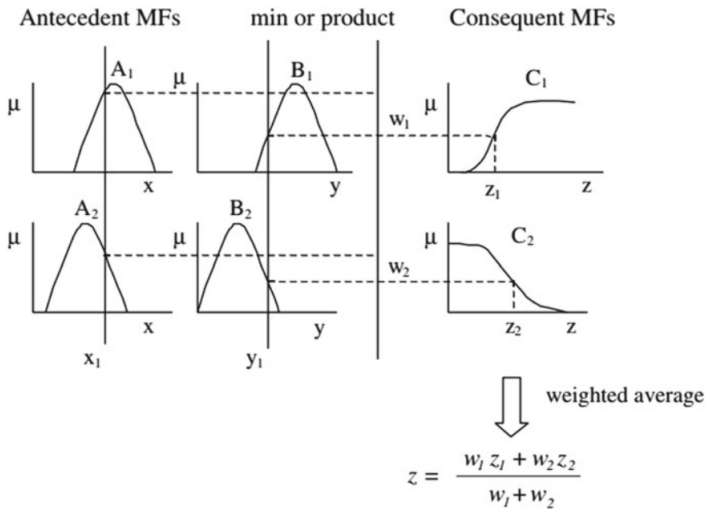


Figura 4.5: Inferencia difusa de Tsukamoto [4]

Inferencia difusa Takagi-Sugeno
La estructura de una inferencia de Takagi-Sugeno también esta dada por las reglas if-then de la siguiente forma:

$$if\ x\ is\ A\ and\ if\ y\ is\ B,\ then\ z = f(x,y)$$

por la forma de la conclusión se puede decir que es un caso particular de Tsukamoto [4], por lo regular $f(x,y)$ es una función lineal, polinomio o si se desea generar un control difuso basado en modelo matemático entonces $f(x,y)$ es una ley de control lineal o no lineal [5], al igual que en la inferencia de Tsukamoto la salida general esta dada por:

$$u = \frac{\sum_{i=1}^{m} w_i f_i(x_1, x_2, \ldots, x_n)}{\sum_{i=1}^{m} w_i} \tag{4.48}$$

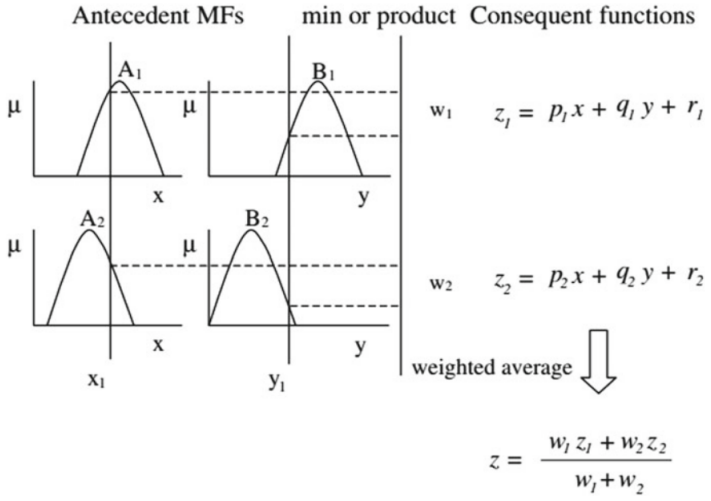


Figura 4.6: Inferencia difusa de Takagi-Sugeno [4]

Ejemplo **4.3** *A continuación se muestra un control difuso con inferencia difusa de Takagi-Sugeno básico para un péndulo invertido. Para calcular la función $f(x,y)$ de la inferencia se utilizara control en variables de estado. Primero se obtiene la ecuación en variables de estado del péndulo invertido*

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}x_2\\ cu+asen(x_1)\end{bmatrix} \qquad c=\frac{1}{mr^2},\ a=\frac{g}{r},\ m=1[Kg],\ r=1[m],\ g=9{,}72[\frac{m}{s}] \tag{4.49}$$

Después se cálcula el modelo linealizado mediante:

$$A=\begin{bmatrix}\frac{\partial f_1(x_1,x_2,u)}{\partial x_1} & \frac{\partial f_1(x_1,x_2,u)}{\partial x_2}\\ \frac{\partial f_2(x_1,x_2,u)}{\partial x_1} & \frac{\partial f_2(x_1,x_2,u)}{\partial x_2}\end{bmatrix} \qquad con \qquad x_{eq}=\begin{bmatrix}x_{1eq}\\ x_{2eq}\end{bmatrix} \tag{4.50}$$

$$B=\begin{bmatrix}\frac{\partial f_1(x_1,x_2,u)}{\partial u}\\ \frac{\partial f_2(x_1,x_2,u)}{\partial u}\end{bmatrix} \qquad con\ u=u_{eq} \tag{4.51}$$

Para utilizar el modelo lineal en variables de estados los puntos de equilibrio se relacionan con el máximo de las funciones de membresía. Entonces primero definimos estas funciones de membresía como se muestra en la Figura 4.7.

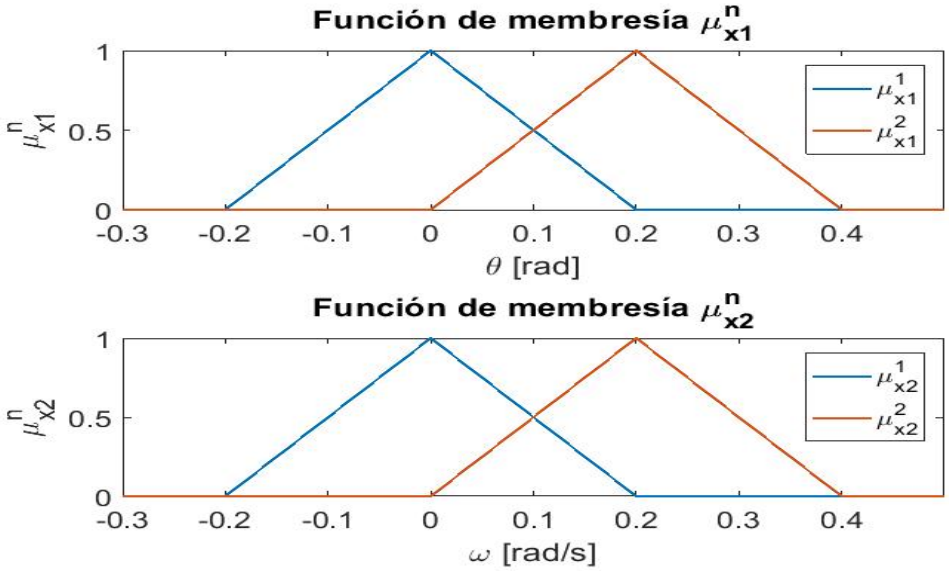


Figura 4.7: funciones de membresía relacionadas con las variables de estados x_1 y x_2

En base a las gráficas de la Figura 4.7 se tienen 4 puntos de equilibrio:

$$x^1_{eq}=\begin{bmatrix}0\\0\end{bmatrix},\qquad x^2_{eq}=\begin{bmatrix}0\\0{,}2\end{bmatrix},\qquad x^3_{eq}=\begin{bmatrix}0{,}2\\0\end{bmatrix},\qquad x^4_{eq}=\begin{bmatrix}0{,}2\\0{,}2\end{bmatrix} \tag{4.52}$$

De estos puntos de equilibrio se tienen que obtener 4 sistemas lineales diferentes pero se reducen a dos ya que la variable de velocidad en la matriz A es igual a uno el punto de equilibrio x^n_{2eq} no se puede sustituir por lo que los modelos están dados de la siguiente forma:

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}}=\underbrace{\begin{bmatrix}0 & 1\\ 9{,}72 & 0\end{bmatrix}}_{A_1}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x}+\underbrace{\begin{bmatrix}0\\1\end{bmatrix}}_{B} \qquad con\ A_2=A_1 \tag{4.53}$$

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}}=\underbrace{\begin{bmatrix}0 & 1\\ 9{,}5262 & 0\end{bmatrix}}_{A_3}\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x}+\underbrace{\begin{bmatrix}0\\1\end{bmatrix}}_{B} \qquad con\ A_3=A_4 \tag{4.54}$$

Con estos sistemas y con ayuda de la función ***lqr()*** *de* ***Matlab*** *se cálcula las ganancias K óptimas para el control:*

$$K_1\begin{bmatrix}19{,}9415 & 7{,}0628\end{bmatrix},\qquad K_2=K_1 \tag{4.55}$$

$$K_3\begin{bmatrix}19{,}5636 & 7{,}0091\end{bmatrix},\qquad K_4=K_3 \tag{4.56}$$

El cálculo de la inferencia se realiza mediante:

$$u = \frac{\sum_{i=1}^{n} w_i f_i(x_1, x_2)}{\sum_{i=1}^{n} w_i} \tag{4.57}$$

donde w_i son los niveles de activación y estos se pueden calcular de la siguiente forma:

$$w_i = \prod_{j=1}^{m} \mu_{x_j}^{i} \tag{4.58}$$

Por lo que los niveles de activación se calculan como:

$$w_1 = \mu_{x_1}^{1}\mu_{x_2}^{1}, \qquad w_2 = \mu_{x_1}^{1}\mu_{x_2}^{2}, \qquad w_3 = \mu_{x_1}^{2}\mu_{x_2}^{1}, \qquad w_4 = \mu_{x_1}^{2}\mu_{x_2}^{2} \tag{4.59}$$

entonces la señal de control es:

$$u = \frac{(w_1 + w_2)(-K_1)x + (w_3 + w_4)(-K_3)x}{w_1 + w_2 + w_3 + w_4} \tag{4.60}$$

En las Figura 4.8 se muestra el desempeño del control difuso y en la Figura 4.9 la variación de los niveles de activación.

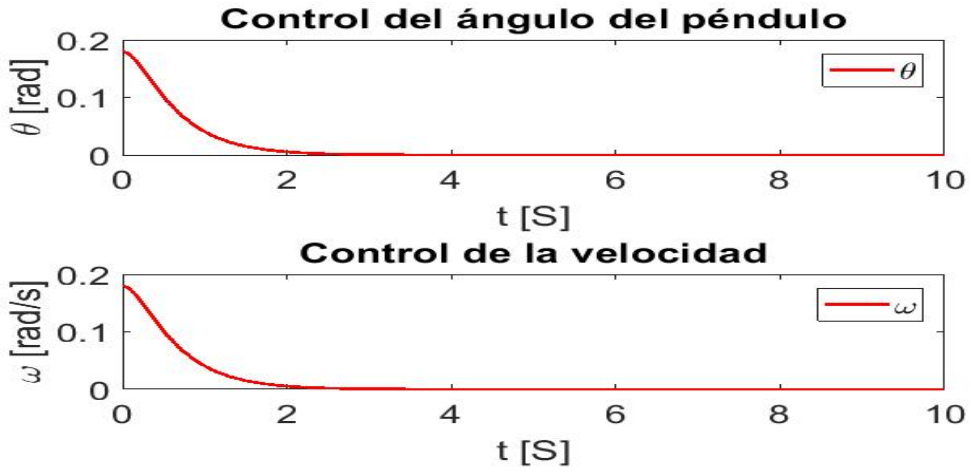


Figura 4.8: Señal de control para el ángulo y la velocidad del péndulo.

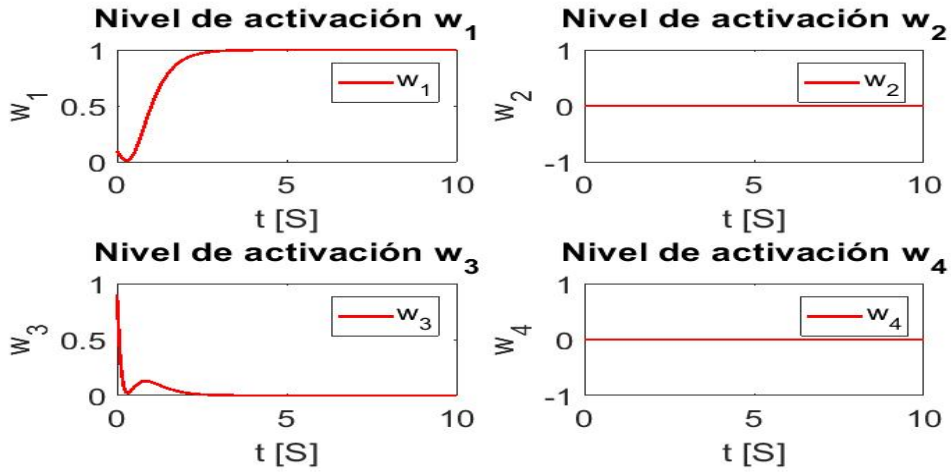


Figura 4.9: Niveles de activación

Capítulo 5

Comunicación y adquisición de datos programados en ROS sobre Raspberry Pi3

5.1. Tarjeta ssc-32

La tarjeta ssc-32 es un servo-driver que permite controlar 32 servomotores, las instrucciones son enviadas a través de el puerto serial (TX, RX) de la Raspberri Pi3 a el puerto serial de la tarjeta ssc-32, además se tiene la ventaja de cambiar la configuración de la tasa de baudios lo cual se puede aumentar el envío de la información de la Raspberri Pi3 hacia la tarjeta ssc-32.
Descripción del Hadware ssc-32

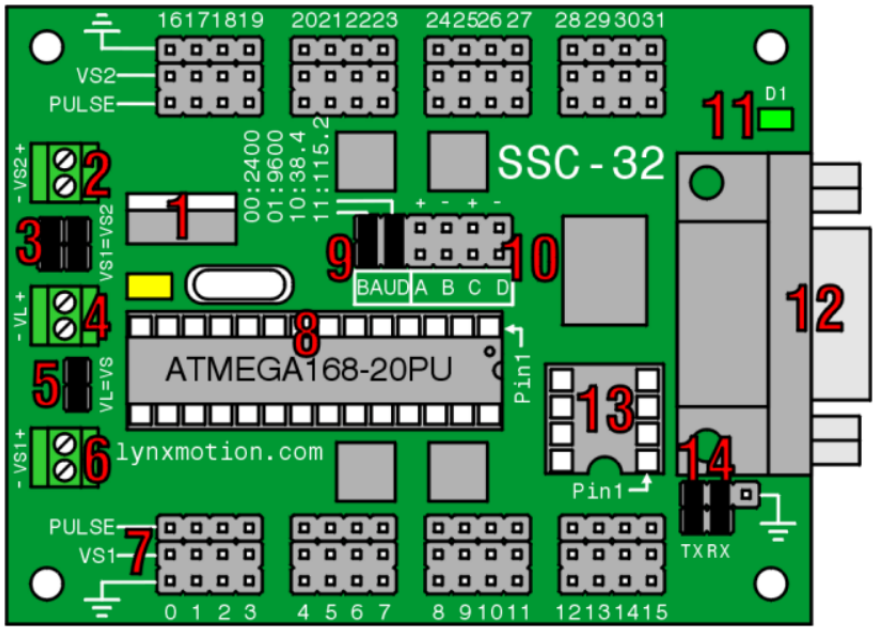


Figura 5.1: Tarjeta scc-32 [11]

1. Es un regulador de voltaje que proporciona una salida de 5 $[V]$ CC, este regulador solo admite voltajes de entrada de 5.5-9 $[V]$ CC. El regulador soporta una corriente máxima de 500 $[mA]$ es recomendable que la corriente sea de 250 $[mA]$ para evitar que el regulador se sobrecaliente.

2. VS2 es una terminal para la alimentación de las terminales 16-31, por lo regular los servomotores se alimentan con 4.8-6 $[V]$ CC, en el caso de utilizar servomotores HSR-5980, HSR-5990 o CYS-S8218 se pueden alimentar con 7.2-7.4 $[V]$ CC.

3. Los jumpers (VS1=VS2) son utilizados para conectar a VS1 y VS2 de la misma batería o fuente de alimentación, si se desea alimentar por separado solo se retiran los jumpers de los headers.

4. VL es la terminal para la alimentación de la electrónica de la placa, para el buen funcionamiento de los circuitos es necesario que se alimente con 6-9 $[V]$ CC. Para revisar el buen funcionamineto de la placa esta debe consumir 35 $[mA]$ sin tener nada conectado a las salidas.

5. El jumper (VL=VS) es utilizado para conectar la elctrónica de la tarjeta y los servomotores a la misma fuente de alimentación. Si la tarjeta se resetea cuando varios servomotores esten en movimiento es recomendable alimetar por separado VS y VL además de que se tiene que retirar el jumper de VL=VS.

6. Al igual que VS2, VS1 es una terminal para la alimentación de las terminales 0-15, por lo regular los servomotores se alimentan con 4.8-6 $[V]$ CC, en el caso de utilizar servomotores HSR-5980, HSR-5990 o CYS-S8218 se pueden alimentar con 7.2-7.4 $[V]$ CC.

7. Estas son las terminales para conectar los servomotores deacuerdo a la Tabla 5.1. Cuando se conecte un bus se debe desactivar la alimentación para no causar algún corto circuito.

Placa	**Cable**
Pulso	Amarillo, Blanco o Anaranjado
Alimentación	Rojo (en algunos casos Anaranjado)
Tierra	Negro o Marrón

Tabla 5.1: Identificación de los cables de conexión para un servomotor

8. Chip IC de Atmel (**ATMEGA168-20PU**)

9. Con estos jumper se puede configurar los baudios según la Tabla 5.2 y en la Figura 5.2 se muestra como conectar los jumpers

Jumpers	**Tasa de baudios**
0 0	2400
0 1	9600
1 0	38.4K
1 1	115.2K

Tabla 5.2: Configuración de la tasa de baudios

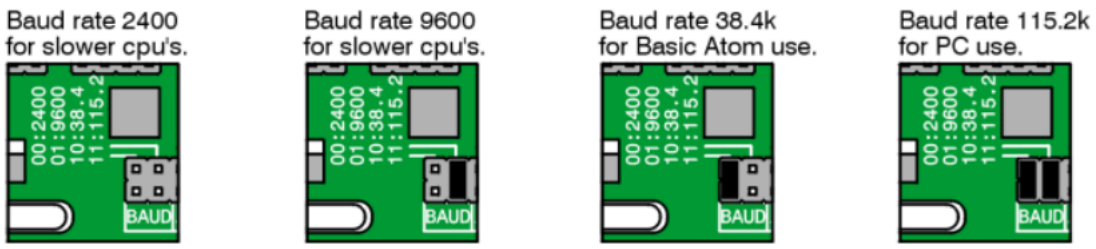


Figura 5.2: Configuración del baud rate [11]

10. Las entradas **A B C D** tienen soportes biestables. Las entradas tienen resistencias internas de tipo pullup de 50 $K\Omega$ [11], Se recomienda utilizar un interruptor normalmente abierto.
11. Es un led indicador, cuando esta encendido continuamente la tarjeta esta funcionado correctamente al recibir un dato serial válido este comenzará a parpadear.
12. Es un conector DB9 para poder comunicar la tarjeta ssc-32 con la computadora ya que a través de este se puede mandar instrucciones de las posiciones del los servomotores.
13. Es un zócalo para una **EEPROM** de 8 pines, esta es compatible con el firmware 2.01GP [11]
14. Es el puerto para la comunicación serie a nivel TTL. Si se desea habilitar el puerto DB9 debe instalar los jumpers como en la Figura 5.3 o en la Figura 5.4 se muestran las conexiones para la comunicación serial TTL con un microcontrolador.

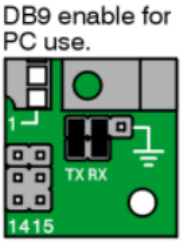


Figura 5.3: Habilita el puerto seri DB9 para comunicar la PC con la tarjeta ssc-32 [11]

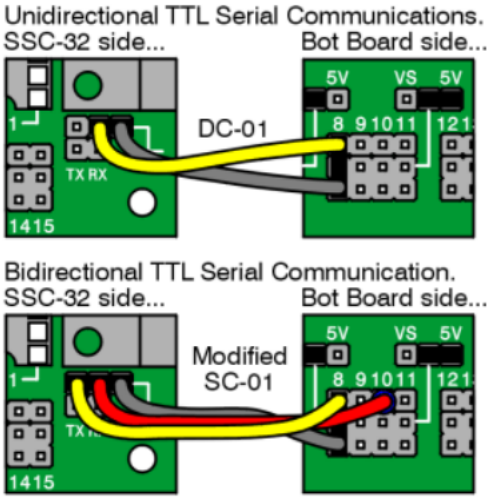


Figura 5.4: Conexión del puerto serie TTL con un microcontrolador [11]

Comandos para los movimientos de los servomotores

Las instrucciones para el movimiento de los servomotores deben enviarse por medio de la comunicación serial ya sea desde la PC o un microcontrolador, las instrucciones deben tener la siguiente estructura.

$$\#<ch>P<pwm>S<spd>\cdots\#<ch>P<pwm>S<spd>T<time><cr>$$

donde:

- $<ch>$: Número del canal en formato decimal 0-31
- $<pwm>$: Ancho de pulso en microsegundos, 500-2500 $[\mu S]$
- $<spd>$: Velocidad de movimiento en μS por segundo para cada canal (opcional)

- $< time >$: Tiempo en μS para el movimiento completo, afecta a todos los canales, 65535 máximo
- $< cr >$: Carácter de retorno de carro, ASCII 13 (necesario para iniciar la acción)

Ejemplo **5.1** *Un movimiento básico de un servomotor esta dado por la siguiente instrucción:*

$$\#5P1600S750\backslash r$$

En la instrucción se activara el servomotor que se encuetra en el canal 5 de la tarjeta scc-32 a la posición 1600 con una velocidad de 750 [μS] por segundo.
Con la velocidad de 750 [μS] el servomotor tardara 2.13 segundos en llegar a la posición 1600.

Ejemplo **5.2** *En este ejemplo se presenta una forma de como mover 2 servo motores y esta estructura también sirve para mover cadenas de servomotores.*

$$\#5P2500\#10P1500T1000\backslash r$$

La unica diferencia, es el valor del tiempo $T < time >$ en lugar de la velocidad, el valor del tiempo afecta a todos los servomotores además de que el valor asignado para el tiempo es lo que tardara en ejecutar la acción.

5.2. Sensor MPU6050

El sensor MPU6050 es un dispositivo que permite medir la orientación, para lograr esto tiene un sistema de tres ejes para el acelérometro, tres ejes para el giroscopio y un procesador de movimiento digital (DMP) [12]. La comunicación mediante I^2C tiene una velocidad de 400 KHz, además el muestreo del convertidor analógico-digital acepta 3.9-8,000 muestras por segundo permitiendo una amplia gama de frecuencias de corte [13]. La orientación de los ejes del sensor se muestran en la Figura 5.5.

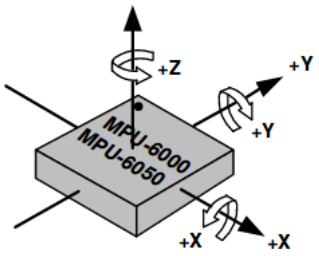


Figura 5.5: Orientación de la sensibilidad de los ejes y la polaridad de la rotación [13]

5.3. Comunicación hacia la tarjeta ssc-32

Antes de explicar la comunicación hacia la targeta ssc-32 primero se deben introducir conceptos básicos de ROS. Primero se debe entender que es un nodo y un tópico.

1. **Nodos**: son los programas basados en c++ o python con instrucciones para ejecutar ROS en donde se realizán procesos de cálculo [14]

2. **Tópicos**: son buses que permiten el intercambio de mensajes entre nodos [15]

La comunicación en ROS esta basada en **publicador-suscriptor** o **cliente-servidor** en este trabajo solo se utilizara la comunicación **publicador-suscriptor**, para realizar esto se requiere programar los **publicadores** y **suscriptores** ya sea en python o C++.
Cuando se realiza un nodo en python ya sea un publicador o un suscriptor en la primeria linea siempre se debe escribir *#/usr/bin/env pytho*, esto sirve para que ROS entienda que el código es un script de python. A continuación se explican las instrucciones básicas y bibliotecas para crear **publicadores** o **suscriptores** en python y C++.

Instrucciones y bibliotecas para publicador python:

1. **import rospy**: con esto se importa todo lo necesario para que python entienda las instrucciones de ROS
2. **from std_ msgs.msg import** <tipo de mensaje>: con esto importamos los tipos de mensajes que ROS entiende, como por ejemplo:
   - String
   - Float32MultiArray
   - Float64MultiArray
3. **pub = rospy.Publisher**('tópico', <tipo de mensaje>, queue_ size=10): declaración del publicador, en el tópico se escribe el nombre del bus al que se desea publicar por ejemplo del diagrama de la Figura 5.6 el tópico es */ssc/*32 , queue_ size es el limitador de mensajes en espera.
4. **rospy.init_ node('topico', anonymous=True)**: esta instrucción es muy importante ya que es el nombre del nodo, si rospy no tiene esta información no puede permitir la comunicación.
5. **rate = rospy.Rate(10) # 10** [Hz]: velocidad con la que se desea publicar los datos.
6. **while not rospy.is_ shutdown():** verifica las salidas del programa por ejemplo al ejecutar Ctrl-C el programa se detiene.
7. **pub.publish(dato)**: esta instrunstrucción es importante, con esta ROS sabe los datos que se deben publicar al tópico.
8. **rate.sleep()**: es una pausa que permite mantener la velocidad deseada establecida en el **rospy.Rate()**.

Instrucciones y bibliotecas para publicador C++:
Los comandos y blibliotecas tienen el mismo fin que las de python, por medio de la tabla 5.3 se muestran las instrucciones en C++ y la similitud con Python.

C++	Python
#include "std_ msgs/String.h"	import rospy
#include "std_ msgs/<tipo de mensaje>"	from std_ msgs.msg import <tipo de mensaje>
ros::init(argc, argv, "nodo");	rospy.init_ node('topico', anonymous=True)
ros::NodeHandle n;	python no lo necesita
ros::Publisher chatter_ pub = n.advertise<std_ msgs::String>("tópico", 10);	pub = rospy.Publisher('tópico', <tipo de mensaje>, queue_ size=10
ros::Rate loop_ rate(10);	rate = rospy.Rate(10) # 10 [Hz]
while (ros::ok())	while not rospy.is_ shutdown():
chatter_ pub.publish(dato);	pub.publish(dato)
loop_ rate.sleep();	rate.sleep()

Tabla 5.3: Comparación de instrucciones entre Python y C++

Nota: ros::NodeHandle n solo se usa en C++, porque se necesita inicializar el nodo y borrar el último nodo que se estaba usando.
Instrucciones y bibliotecas para suscriptor Python:
Solo se explicaran las instrucciones principales para el suscriptor ya que algunos se repiten como por ejemplo: import rospy, from std_ msgs.msg import <tipo de mensaje> y rospy.init_ node('tópico', anonymous=True).

1. **def callback(data)**: es una función muy importante por que es donde el suscriptor almacenar los datos enviados por el tópico y los guarda en data para posteriormente utilizarlos en el programa.
2. **rospy.Subscriber('tópico',** <**tipo de mensaje**> **, callback)**: al igual que en el publicador con esta instrucción declaramos el suscriptor,además manda a llamar a la función callback para almacenar los datos.
3. **rospy.spin()**: evita que el nodo se cierre a destiempo.

Instrucciones y bibliotecas para suscriptor en C++:
En C++ también hay instrucciones que se repiten ya que son parte de la estructura del programa como por ejemplo: #include "std_ msgs/String.h", #include "std_ msgs/<tipo de mensaje>", ros::init(argc, argv, "nodo") y ros::NodeHandle n. Por lo que solo se explicaran las instrucciones importantes de un suscriptor.

1. **void chatterCallback(const std_ msgs::<tipo de mensaje>::ConstPtr& data)**: al igual como con python esta función sirve para almacenar los datos en la varible data.
2. **ros::Subscriber sub = n.subscribe("tópico", 1000, chatterCallback)**: sirve para declar al suscriptor, poder conectar al tópico y también manda a llamar la función chatterCallback para guardar los datos enviados através del tópico.

Ya entendidos los conceptos básicos de ROS podemos pasar a la comunicación por puerto serial de la Raspberry Pi3 con la tarjeta ssc-32. A nivel software la comunicación serial se programa en ROS sobre Raspberry Pi3. La comunicación serial tiene diferentes tasa de baudios por lo que la tarjeta ssc-32 y en la programación del nodo suscriptor se debe especificar o configurar para que no se tenga problemas con la comunicación, la tasa de baudios elegida es de 38,4K esta por encima de la comunicacion serial de un microcontrolador y por debajo de una PC, por lo que el tiempo de espera entre simbolos para enviarlos es de 26 [μS]. En el diagrama de la Figura 5.6 se muestran los nodos y topicos para enviar instrucciones a la tarjeta ssc-32.
Nodo "robotbb_ serialnewrv"
El Programa 11.1 es un nodo suscriptor donde se encuentra programada la comunicación serial y los calculos para relacionar los ángulos obtenidos que el publicador envia, con las posiciones en μS. Obteniendo los datos en μS se envian a la tarjeta ssc-32 y en concecuencia se podrán mover los servomotores.

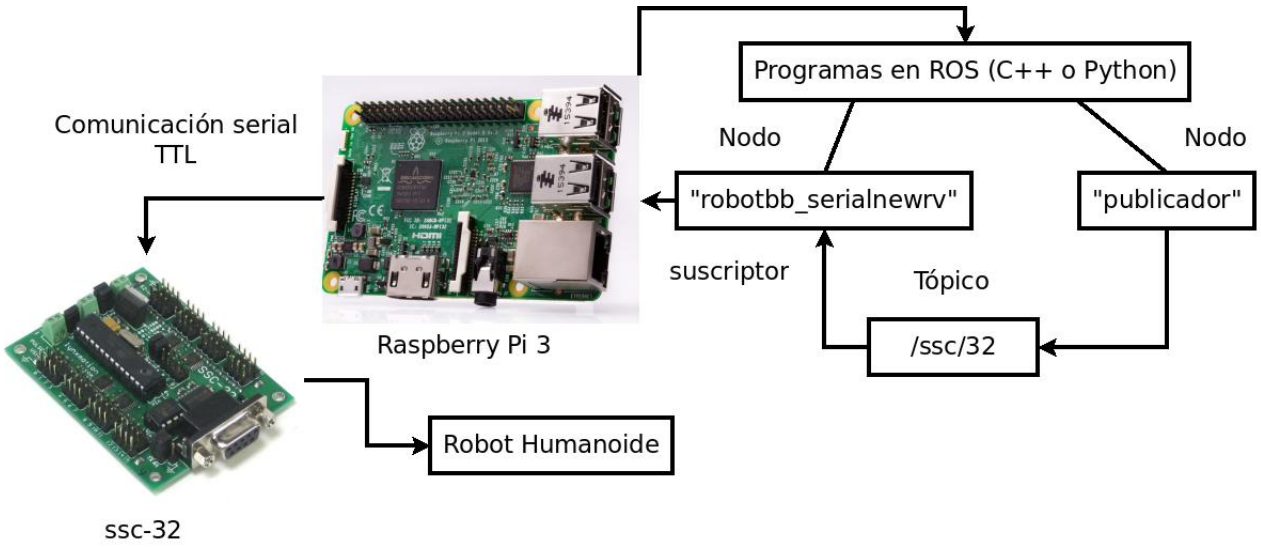


Figura 5.6: Nodos y Topicos en ROS para la comunicación serial de Raspberry Pi3 y la tarjeta ssc-32

Ecuaciones para relacionar los ángulos con los μS
Estas ecuaciones son importantes porqué de esta forma la tarjeta ssc-32 puede entender la información cálculada. Los datos de la relación describen curvas no lineales por lo que se aproximó por medio de una regresión polinomial (Figura 5.7) y están definidas por los polinomios:

$$P_1(\theta) = -0{,}00036\theta^3 + 14{,}07\theta + 1500 \tag{5.1}$$

$$P_2(\theta) = 0{,}00036\theta^3 - 14{,}07\theta + 1500 \tag{5.2}$$

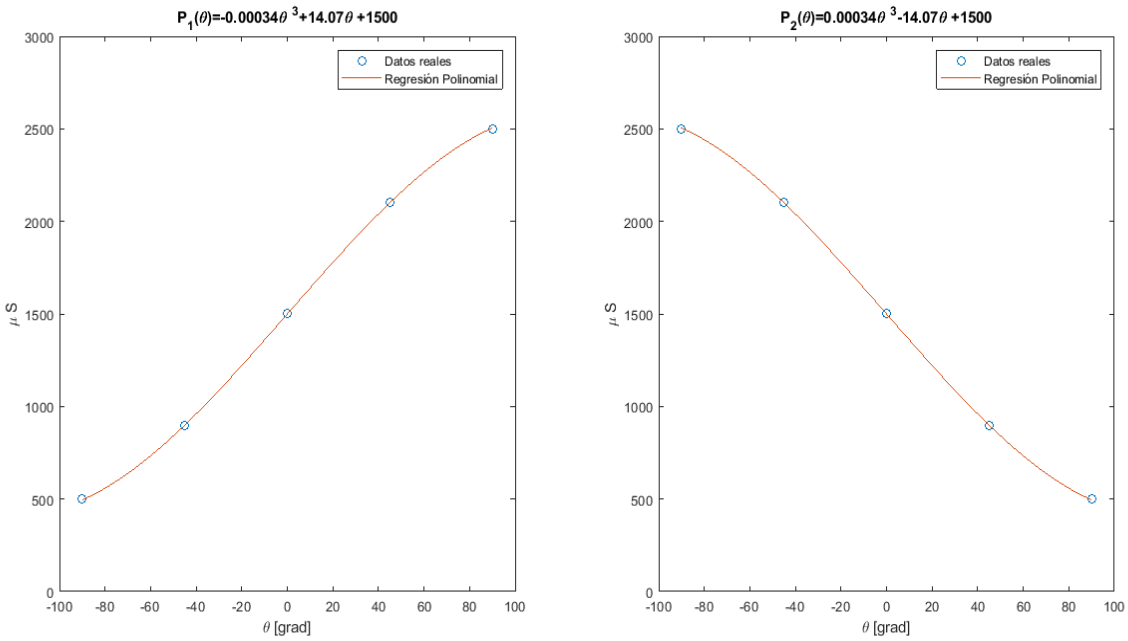


Figura 5.7: Comparación de las muestras de los datos reales con la regresión polinomial

5.4. Medición de ángulos con el sensor MPU6050

El sensor **MPU6050** tiene un acelérometro con el cual se miden fuerzas de aceleración, estas fuerzas pueden ser estáticas como la fuerza de gravedad o dinámicas las cuales son causadas por los movimientos o vibraciones del dispositivo, considrando a estas fuerzas como a_x, a_y y a_z en dirección de x, y y z [16], se pueden calcular los ángulos de cabeceo (θ_x) y balanceo (θ_y) con las siguientes ecuaciones:

$$| a |= \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{5.3}$$

$$\theta_x = tan^{-1}\left(\frac{a_y}{| a |}\right) \tag{5.4}$$

$$\theta_y = tan^{-1}\left(\frac{a_x}{| a |}\right) \tag{5.5}$$

Debido a que las señales obtenidas por el sensor tienen ruido, se utilizará un filtro complementario este consta de dos filtros, uno paso bajas discreto (LPFD) y un filtro paso altas discreto (HPFD). LPFD es aplicado a los ángulos obtenidos por el acelérometro porque las lecturas son más afectadas o menos confiables en tiempos cortos y los datos son más confiables en tiempos largos. HPFD es aplicado a los datos obtenidos de la integral numérica de las señales del giroscopio porque los datos son afectados por los tiempos largos, las salidas de cada filtro son sumadas y se obtienen los ángulos, en la siguiente Figura 5.8 se muestra el diagrama de bloques del filtro complementario.

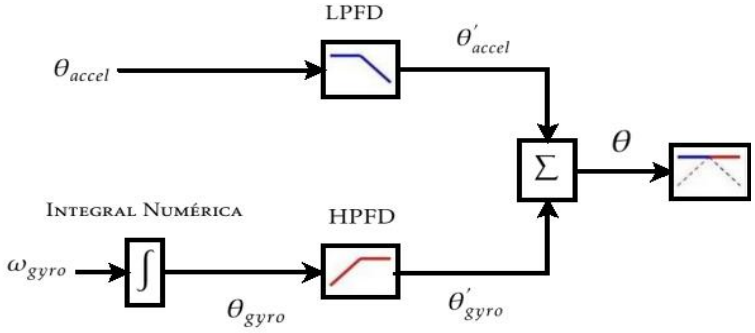


Figura 5.8: Filtro Complementario Discreto [16]

Ecuaciones del LPFD

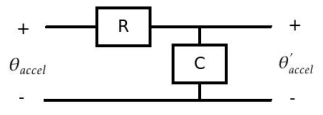


Figura 5.9: Circuito RC para LPF de 1^{er} orden

Si suponemos que θ_{accel} es la señal de entrada y θ'_{accel} la señal de salida con lo cual podemos obtener la ecuación diferencial de 1^{er} orden de un circuito RC (Figura 5.9):

$$\theta_{accel} = \tau \frac{d\theta'_{accel}}{dt} + \theta'_{accel} \qquad \tau = RC \tag{5.6}$$

donde τ es la constante de tiempo. Pero se necesita obtener las ecuaciones discretas del fitro por lo que se aproxima a la derivada como una diferencia hacia atras:

$$\frac{d\theta'_{accel}}{dt} \approx \frac{\theta'_{accel\,i} - \theta'_{accel\,i-1}}{\triangle t} \tag{5.7}$$

Sustituyendo esta aproximación de la derivada (5.7) en la ecuación (5.6) obtenemos una ecuación discreta:

$$\theta'_{accel\,i} = \alpha_{LPF}\theta_{accel\,i} + (1-\alpha_{LPF})\theta'_{accel\,i-1}, \qquad \alpha_{LPF} = \frac{\triangle t}{\tau + \triangle t} \tag{5.8}$$

donde α_{LPF} es una constante del filtro paso bajas y $\triangle t$ es la tasa de muestreo.

Ecuaciones del HPFD

Mediante el circuito CR mostrado en la Figura 5.10 se obtendra la ecuación diferencial de 1^{er} orden para el HPF, esta ecuación es de la forma:

$$\theta'_{gyro} = \tau \frac{d\theta_{gyro}}{dt} - \tau \frac{d\theta'_{gyro}}{dt} \tag{5.9}$$

Nuevamente se utiliza la aproximación de la derivada por diferencias hacia atrás:

$$\frac{d\theta'_{gyro}}{dt} \approx \frac{\theta'_{gyro\,i} - \theta'_{gyro\,i-1}}{\triangle t} \tag{5.10}$$

El filtro complementario hace uso de una integral numérica para obtener $\theta_{gyro\,i}$ apartir de $\omega_{gyro\,i}$ obtenido del giroscopio:

$$\theta_{gyro\,i} = \int \omega_{gyro} dt \approx \omega_{gyro\,i} \triangle t \tag{5.11}$$

Para obtener la ecuación discreta de un HPF sustituimos la aproximación (5.10) y la integral numérica (5.11) en la ecuación diferencial (5.9) y se obtiene:

$$\theta'_{gyro\,i} = \alpha_{HPF}\omega_{gyro\,i} \triangle t + \alpha_{HPF}\theta'_{gyro\,i-1}, \qquad \alpha_{HPF} = \frac{\tau}{\tau + \triangle t} \tag{5.12}$$

Para obtener la ecuación de la salida (θ) del Filtro Complementario solo se tienen que sumar $\theta'_{accel\,i}$ con $\theta'_{gyro\,i}$ de la siguiente forma:

$$\theta = \theta'_{accel\,i} + \theta'_{gyro\,i} \tag{5.13}$$

Debido a la complejidad de los cálculos la ecuación discreta del filtro simplificada [16] esta dada por:

$$\theta_i = \alpha(\theta_{i-1} + \omega_{gyro\,i} \triangle t) + (1-\alpha)\theta_{accel\,i} \tag{5.14}$$

Donde α es un valor entre 0 y 1 el cual es deteriminado después de observar los datos obtenidos del filtrado de las señales del sensor [12], el filtro tiene un buen desempeño con $\alpha = 0{,}8$ por lo cual se obtiene una ecuación para cada ángulo θ_x y θ_y de la forma:

$$\theta_{x,\,y\,i} = 0{,}8(\theta_{x,\,y\,i-1} + \omega_{gyro_{x,\,y}\,i} \triangle t) + 0{,}2\theta_{accel_{x,\,y}\,i} \tag{5.15}$$

En la Figura 10.1 se muestra el desempeño del fitro complementario mediante la ecuación (5.15).

Filtrado de la señal del giroscopio

El giroscopio entrega la medición de la velocidad ángular en los 3 ejes, debido a que los datos tienen ruido es necesario programar un filtro. El filtro esta basado en un filtro digital Butterworth.
El filtro digital Butterworth tiene la siguiente función de transferencia en el espacio z:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2} + \ldots + b_{n+1} z^{-n}}{a_1 + a_2 z^{-1} + a_3 z^{-2} + \ldots + a_{n+1} z^{-n}} \tag{5.16}$$

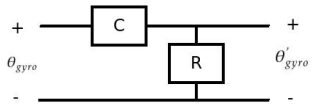


Figura 5.10: Circuito CR para HPF de 1^{er} orden

si realizamos lo siguiente para la ecuación (5.16)

$$Y(z)(a_1+a_2z^{-1}+a_3z^{-2}+\ldots+a_{n+1}z^{-n}) = X(z)(b_1+b_2z^{-1}+b_3z^{-2}+\ldots+b_{n+1}z^{-n}) \quad (5.17)$$

Aplicando la transformada z inversa para la ecuación (5.17)

$$\mathcal{Z}^{-1}\{Y(z)(a_1+a_2z^{-1}+a_3z^{-2}+\ldots+a_{n+1}z^{-n})\} = \mathcal{Z}^{-1}\{X(z)(b_1+b_2z^{-1}+b_3z^{-2}+\ldots+b_{n+1}z^{-n})\}$$

se obtiene una ecuación en diferencias como se muestra a continuación:

$$a_1y_n + a_2y_{n-1} + \ldots + a_{n_0+1}y_{n-n_0} = b_1x_n + b_2x_{n-1} + \ldots + b_{n_0+1}x_{n-n_0} \quad (5.18)$$

La salida del filtro digital es y_n por lo que se necesita despejar de la ecuación (5.18) de la siguiente forma:

$$y_n = \frac{1}{a_1}\left(b_1x_n + b_2x_{n-1} + \ldots + b_{n_0+1}x_{n-n_0} - a_2y_{n-1} - \ldots - a_{n_0+1}y_{n-n_0}\right) \quad (5.19)$$

El orden del filtro que se aplica a la señal es de orden 2, con la siguiente ecuación se calcula y_n para este filtro en particular:

$$y_n = \frac{1}{a_1}(b_1x_n + b_2x_{n-1} + b_3x_{n-2} - a_2y_{n-1} - a_3y_{n-2}) \quad (5.20)$$

Los coeficientes $a_1, a_2, \ldots, a_{n_0+1}$ y $b_1, b_2, \ldots, b_{n_0+1}$ se pueden calcular utilizando la función **[b,a]=butter(n,Wn)** en **Matlab**, dónde n es el orden del filtro digital y Wn el tiempo de muestreo el cual es 0,1 $[seg]$.
Los valores de los coeficientes son:

$$a = [1 \quad -1{,}561 \quad 0{,}6414] \qquad b = [0{,}0201 \quad 0{,}0402 \quad 0{,}0201] \quad (5.21)$$

En la Figura 10.2 se muestra el desempeño del filtro digital Butterworth basado en los coeficientes calculados (5.21).

Nodo "mpu6050_sensor_node"

El programa 11.3 es un nodo publicador. En este nodo se encuentra programada la comunicación I^2C para poder obtener los datos de el giroscopio y el acelerómetro, además se realiza el cálculo para el filtro complementario de estos datos y obtener mediciones sin ruido. Debido a que el nodo es un publicador se necesita especificar la frecuencia a la cual se quieren publicar los datos de los ángulos filtrados, la frecuencia elegida es de 10 Hz. En la Figura 5.6 se encuentra un diagrama a bloques de los nodos y topicos para obtener las lecturas del sensor MPU6050 mediante la comunicación I^2C.

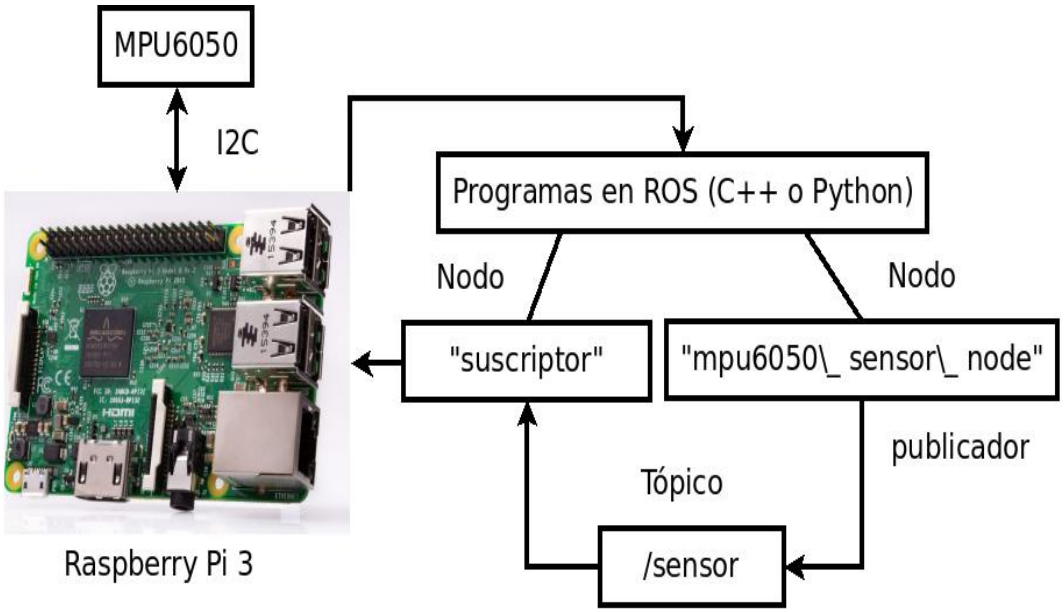


Figura 5.11: Nodos y Topicos en ROS para la comunicación I2C de Raspberry Pi3 con el sensor MPU6050

Capítulo 6

Obtención de la cinemática inversa para las piernas del Robot Humanoide

6.1. Cinemática inversa de un robot planar de 2 grados de libertad (RP2GL)

En la Figura 6.1 se muestra un robot planar de 2 grados de libertad del cual se obtendrá la cinemática inversa mediante el método geométrico [18]. La asignación de los ejes se realizó mediante la convención de Denavit-Hartenberg (**DH**). Menciona que los ejes se deben asignar de la siguiente forma:

1. **(DH1)** los ejes x_i son perpendiculares a los ejes z_{i-1} [18].
2. **(DH2)** los ejes x_i intersectan a los ejes z_{i-1} [18].

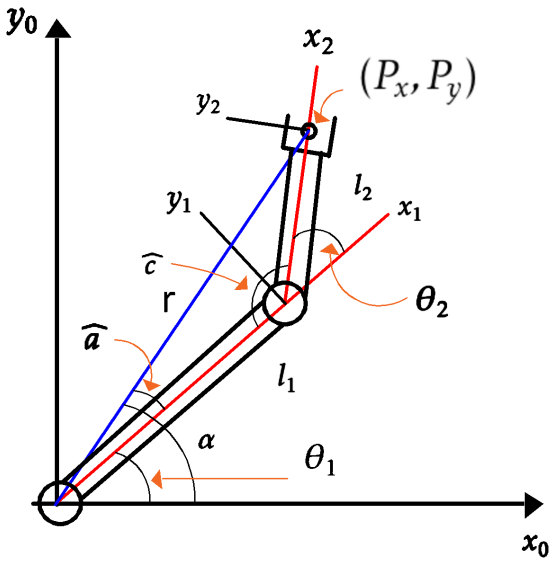


Figura 6.1: Robot planar de dos grados de libertad

Primero obtendremos la expresión para calcular el ángulo θ_1 de la Figura 6.1 obtenemos que:

$$r^2 = P_x^2 + P_y^2 \tag{6.1}$$

$$\alpha = atan2(P_y, P_x) \tag{6.2}$$

Del triángulo de apoyo de la Figura 6.2 se obtiene:

$$y = l_2 sen(\theta_2) \tag{6.3}$$

$$x = l_2 cos(\theta_2) \tag{6.4}$$

$$S_{x\widehat{a}} = l_1 + x \qquad S_{y\widehat{a}} = y \tag{6.5}$$

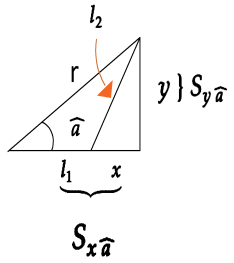


Figura 6.2: Triángulo de apoyo

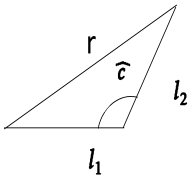


Figura 6.3: Triángulo de apoyo para la ley de cosenos

El cálculo de $\widehat{a}$ esta dado de la siguiente forma:

$$\widehat{a} = atan2(S_{y\widehat{a}}, S_{x\widehat{a}}) \tag{6.6}$$

Sustituyendo (6.3) y (6.4) en (6.5) se obtiene:

$$S_{x\widehat{a}} = l_1 + l_2 cos(\theta_2) \qquad S_{y\widehat{a}} = l_2 sen(\theta_2) \tag{6.7}$$

La forma final de $\widehat{a}$ se obtiene al sustituir (6.7) en (6.6)

$$\widehat{a} = atan2(l_2 sen(\theta_2), l_1 + l_2 cos(\theta_2)) \tag{6.8}$$

De la Figura 6.1 se observa que $\alpha = \widehat{a} + \theta_1$ despejando a θ_1 tenemos que:

$$\theta_1 = \alpha - \widehat{a} \tag{6.9}$$

Finalmente sustituyendo (6.2) y (6.8) en (6.9) obtenemos la ecuación para calcular θ_1:

$$\theta_1 = atan2(P_y, P_x) - atan2(l_2 sen(\theta_2), l_1 + l_2 cos(\theta_2)) \tag{6.10}$$

Ahora se obtendra la ecuación para calcular θ_2. De la Figura 6.3 y por medio de la ley de cosenos obtenemos:

$$r^2 = l_1^2 + l_2^2 - 2 l_1 l_2 cos(\widehat{c}) \tag{6.11}$$

Para obtener $\widehat{c}$ se observa en la Figura 6.1 que:

$$\widehat{c} = \pi - \theta_2 \tag{6.12}$$

Sustituyendo (6.12) en (6.11) se obtiene:

$$r^2 = l_1^2 + l_2^2 + 2 l_1 l_2 cos(\theta_2) \tag{6.13}$$

Despejando $cos(\theta_2)$:

$$cos(\theta_2) = \frac{r^2 - (l_1^2 + l_2^2)}{2 l_1 l_2} = M \tag{6.14}$$

Utilizando la identidad trigonométrica $cos^2(\theta_2) + sen^2(\theta_2) = 1$, despejando $sen(\theta_2)$ y sustituyendo (6.14):

$$sen(\theta_2) = \pm\sqrt{1 - cos^2(\theta_2)} = \pm\sqrt{1 - M^2} \tag{6.15}$$

Por último para obtener la ecuación que permitira calcular θ_2 es la siguiente:

$$\theta_2 = atan2(\pm\sqrt{1 - M^2}, M) \tag{6.16}$$

Las ecuaciones finales para calcular θ_1 y θ_2 son las siguientes:

$$\boxed{\theta_2 = atan2(\pm\sqrt{1 - M^2}, M), \qquad M = \frac{r^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}}$$

$$\boxed{\theta_1 = atan2(P_y, P_x) - atan2(l_2 sen(\theta_2), l_1 + l_2 cos(\theta_2))}$$

6.2. Comparación de la pierna fija con un RP2GL

En esta parte del capítulo se obtendra la cinemática inversa para controlar la pierna fija al suelo del Robot Humanoide (Figura 6.4) utilizando los calculos de la cinemática inversa de un RP2GL. Para utilizar las ecuaciones de RP2GL es necesario hacer algunas suposiciones.

- El radio r_{cf} debe ser constante para mantener el radio del péndulo invertido imaginario constante.
- Cuando el radio r_{cf} es constante entonces α_c y θ_2 también deben ser constantes

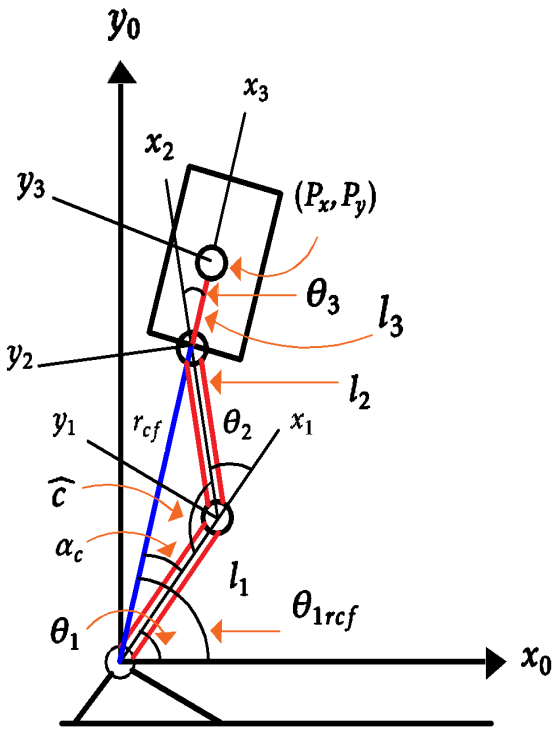


Figura 6.4: Robot Humanoide

Se realizarán los calculos necesarios para poder obtener las ecuaciones de r_{cf} y α_c en función de un valor constante de θ_2.

El triángulo formado por l_1, l_2 y r_{cf} de la Figura 6.4 se observa que:

$$\widehat{c} = \pi - \theta_2 \tag{6.17}$$

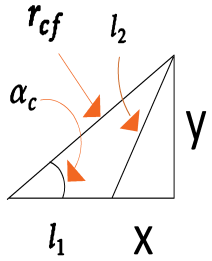


Figura 6.5: Triángulo de apoyo para el cálculo de α_c y r_{cf}

Utilizando la ley de los cosenos se puede obtener la siguiente igualdad:

$$r_{cf}^2 = l_1^2 + l_2^2 + 2l_1 l_2 cos(\theta_2) \tag{6.18}$$

si despejamos r_{cf} de (6.18) el resultado es:

$$r_{cf} = \sqrt{l_1^2 + l_2^2 + 2l_1 l_2 cos(\theta_2)} \tag{6.19}$$

Para obtener α_c se utiliza el triángulo de la Figura 6.5, de la cual se observa:

$$x = l_2 cos(\theta_2), \qquad y = l_2 sen(\theta_2) \tag{6.20}$$

$$S_{x\,\alpha_c} = l_1 + x, \qquad S_{y\,\alpha_c} = y \tag{6.21}$$

Entonces α_c esta dada de la siguiente forma:

$$\alpha_c = atan2(S_{y\,\alpha_c}, S_{x\,\alpha_c}) = atan2(l_2 sen(\theta_2), l_1 + l_2 cos(\theta_2)) \tag{6.22}$$

El RP2GL de la Figura 6.6 esta formado por líneas imaginarias que fueron obtenidas del la Figura 6.4. Con esto se podra controlar las piernas y el movimiento de la cadera del robot.

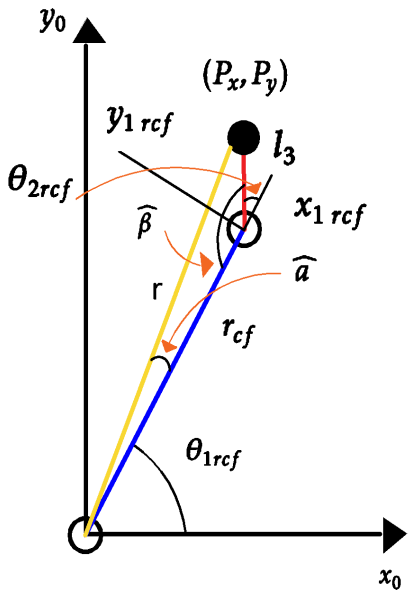


Figura 6.6: RP2GL formado por líneas imaginarias

El cálculo de la cinemática inversa del robot de la Figura 6.6 ya se realizo anteriormente rescribiendo las ecuaciones (6.10) y (6.16) deacuerdo a el RP2GL imaginario de la Figura 6.6:

$$\theta_{2\,rcf} = atan(\pm\sqrt{1 - M_{rcf}^2}, M_{rcf}), \qquad M_{rcf} = \frac{P_x^2 + P_y^2 - (r_{rcf}^2 + l_3^2)}{2 r_{rcf} l_3} \tag{6.23}$$

$$\theta_{1\,rcf} = atan2(P_y, P_x) - atan2(l_3 sen(\theta_{2\,rcf}), r_{cf} + l_3 cos(\theta_{2\,rcf})) \tag{6.24}$$

Para poder controlar el tobillo se requiere obtener la expresión para calcular el ángulo θ_1 y el ángulo θ_3 para la cadera. Entonces de la Figura 6.4 se puede obtener que θ_1 es igual a:

$$\theta_1 = \theta_{1\,rcf} - \alpha_c \tag{6.25}$$

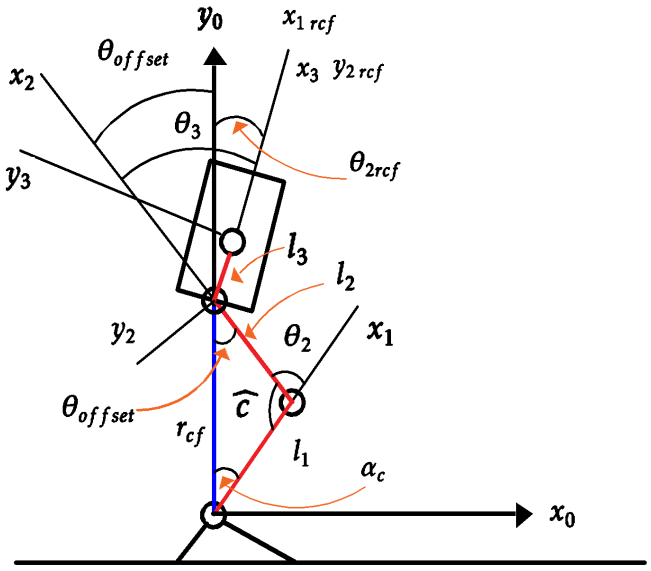


Figura 6.7: Diagrama de apoyo para el cálculo de θ_{offset}

Con base a la Figura 6.7 se observa que θ_3 esta dado por:

$$\theta_3 = \theta_{2\,rcf} - \theta_{offset} \tag{6.26}$$

El ángulo θ_{offset} es una constande que permite llevar al ańgulo θ_3 a su posicion correcta ya que el eslabón l_3 tiene dos sistemas de referencia x_3, y_3, z_3 y $x_{2\,rcf}$, $y_{2\,rcf}$, $z_{2\,rcf}$. Entonces para cálcular la constante θ_{offset} con base a la Figura 6.7 se obtiene:

$$\theta_{offset} = \pi - (\alpha_c + \widehat{c}) \tag{6.27}$$

En resumen el RP2GL formado por las lineas imaginarias tomadas de la Figura 6.4 y plasmadas en la Figura 6.6 es utilizado para el cálculo de θ_1 y θ_3 ya que con estos ángulos se podra manipular o controlar el tobillo y la cadera, θ_2 esta relacionado con el ángulo de la rodilla y se considero constante. Finalmente las ecuaciones de cinemática inversa basadas en RP2GL para mover la pierna fija al suelo con θ_2 constante son:

- **Constantes**

$$\boxed{r_{cf} = \sqrt{l_1^2 + l_2^2 + 2l_1 l_2 cos(\theta_2)}}$$

$$\boxed{\alpha_c = atan2(S_{y\,\alpha_c}, S_{x\,\alpha_c}) = atan2(l_2 sen(\theta_2), l_1 + l_2 cos(\theta_2))}$$

$$\boxed{\theta_{offset} = \pi - (\alpha_c + \widehat{c})}$$

- **Variables**

$$\boxed{\theta_{2\,rcf} = atan(\pm\sqrt{1 - M_{rcf}^2}, M_{rcf}), \qquad M_{rcf} = \frac{P_x^2 + P_y^2 - (r_{rcf}^2 + l_3^2)}{2 r_{rcf} l_3}}$$

$$\boxed{\theta_{1\,rcf} = atan2(P_y, P_x) - atan2(l_3 sen(\theta_{2\,rcf}), r_{cf} + l_3 cos(\theta_{2\,rcf}))}$$

$$\boxed{\theta_1 = \theta_{1\,rcf} - \alpha_c}$$

$$\boxed{\theta_3 = \theta_{2\,rcf} - \theta_{offset}}$$

Capítulo 7

Trayectorias basadas en el polinomio de 5^{to} orden

7.1. Obtención de la forma general de los coeficientes del polinomio de 5^{to} orden

Para calcular los coeficientes del polinomio primero se deben realizar las siguientes suposiciones:

- La trayectoria basada en un polinomio de 5^{to} orden debe tener un punto inicial $P_i(t_i)$ en un tiempo inicial t_i y un punto final $P_f(t_f)$ para un tiempo final t_f [19].
- La velocidad de las trayectorias debe ser cero en la velocidad inicial ($\dot{P}_i(t_i)$) y final $\dot{P}_f(t_f)$)
- La aceleración de las trayectorias deben ser cero en la aceleración inicial ($\ddot{P}_i(t_i)$) y final ($\ddot{P}_f(t_f)$)

Primero se define un polinimio variante en el tiempo general de 5^{to} orden:

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{7.1}$$

Con base en el polinomio (7.1) definiremos las condiciones iniciales para la trayectoria donde $P_i(t_i)$ es el punto inicial, $\dot{P}_i(t)$ velocidad inicial y $\ddot{P}_i(t)$ aceleración inicial mediante:

$$P_i(t_i) = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 + a_4 t_i^4 + a_5 t_i^5 \tag{7.2}$$

$$\dot{P}_i(t_i) = a_1 + 2a_2 t_i + 3a_3 t_i^2 + 4a_4 t_i^3 + 5a_5 t_i^4 \tag{7.3}$$

$$\ddot{P}_i(t_i) = 2a_2 + 6a_3 t_i + 12a_4 t_i^2 + 20a_5 t_i^3 \tag{7.4}$$

y también se define las condiciones finales de la trayectoria, donde $P_f(t_f)$ es el punto final, $\dot{P}_f(t_f)$ velocidad final y $\ddot{P}_i(t_f)$ aceleración final mediante:

$$P_f(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \tag{7.5}$$

$$\dot{P}_f(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \tag{7.6}$$

$$\ddot{P}_f(t_f) = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \tag{7.7}$$

Las condiciones iniciales y finales son valores constantes y conocidos por lo que se debe hacer el cálculo de los coeficientes del polinomios (7.1) y se puede realizar mediante una ecuación matricial de la siguiente manera:

$$\underbrace{\begin{bmatrix} P_i(t_i) \\ \dot{P}_i(t_i) \\ \ddot{P}_i(t_i) \\ P_f(t_f) \\ \dot{P}_f(t_f) \\ \ddot{P}_f(t_f) \end{bmatrix}}_{VP_{6\times1}} = \underbrace{\begin{bmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5 \\ 0 & 1 & 2t_i^2 & 3t_i^2 & 4t_i^3 & 5t_i^4 \\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f^2 & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}}_{MP(t_i,t_f)_{6\times6}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}}_{Va_{6\times1}} \tag{7.8}$$

El vector columna $VP_{6\times1}$ es constante al igual que la matriz $MP(t_i,t_f)_{6\times6}$ el vector columna $Va_{6\times1}$ no es conocido ya que este vector esta conformado por los coeficientes del polinomio. De la ecuación (7.8) se despeja $Va_{6\times1}$ y resulta:

$$Va_{6\times1} = MP(t_i,t_f)^{-1}_{6\times6} VP_{6\times1} \tag{7.9}$$

El resultado de realizar las operaciones de la ecuación (7.9) se obtiene:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \frac{1}{(t_f - t_i)^5} \begin{bmatrix} P_i t_f^3(t_f^2 - 5t_f t_i + 10t_i^2) - P_f t_i^3(10t_f^2 - 5t_f t_i + t_i^2) \\ 30t_f^2 t_i^2(P_f - P_i) \\ -30t_f t_i(t_f + t_i)(P_f - P_i) \\ 10(P_f - P_i)(t_f^2 + 4t_f t_i + t_i^2) \\ -15(P_f - P_i)(t_f + t_i) \\ 6(P_f - P_i) \end{bmatrix} \tag{7.10}$$

Ejemplo **7.1** *Se desea diseñar una trayectoria basada en un polinomio de* 5^{to} *orden con un tiempo inicial* $t_i = 0$*, tiempo final* $t_f = 10$ *y el siguiente vector columna:*

$$VP_{6\times1} = \begin{bmatrix} 0 & 0 & 0 & 0{,}1745 & 0 & 0 \end{bmatrix}^T$$

Mediante la igualdad matricial (7.10) *obtenemos el vector columna de coeficientes para el polinomio.*

$$Va_{6\times1} = \begin{bmatrix} 0 & 0 & 0 & 0{,}0017 & -261{,}8\times10^{-6} & 10{,}4\times10^{-6} \end{bmatrix}^T$$

El polinomio resultante es $P(t) = 0{,}0140t^3 - 0{,}0042t^4 + 335{,}1\times10^{-6}t^5$*. En la Figura se muestran las gráficas del polinomio y sus derivadas.*

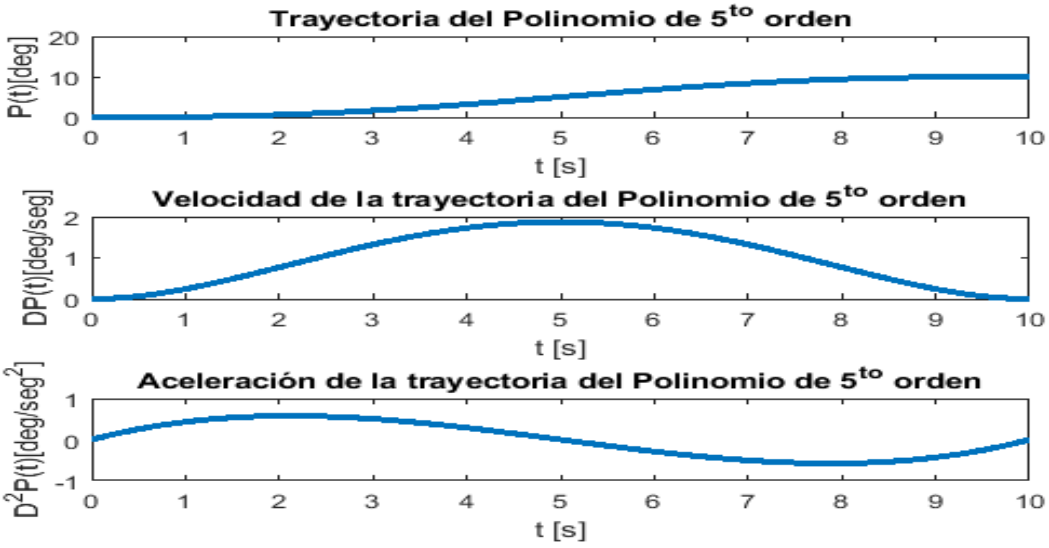


Figura 7.1: perfil de la trayectoria polinomica y sus derivadas

7.2. Diseño de las trayectorias para la marcha bípeda

Diseño de las trayectorias para la inclinación del robot humanoide suponiendo un movimiento basado en el péndulo invertido.

Una parte del control para generar la caminata bípeda es hacer que el robot realice seguimiento de trayectorias ($\theta = P(t), \dot{\theta} = \dot{P(t)}$), en dónde el movimento basado en el péndulo invertido se debe intercalar para generar los movimientos de las piernas para obtener un patrón como se muestra en la Figura 7.2.

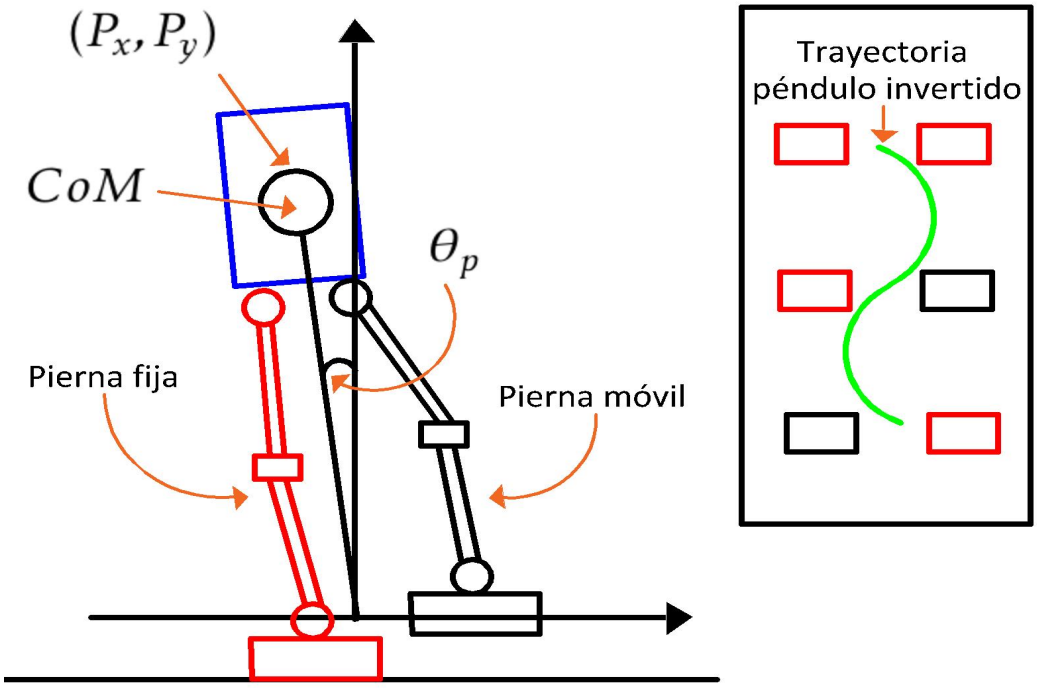


Figura 7.2: Movimientos de inclinación del péndulo invertido (θ_p) para el robot humanoide

Primera etapa de la trayectoria del péndulo invertido para la caminata bípeda.

El robot humanoide parte de un punto inicial, dónde teóricamente se encuetra en $\theta = 0$, $\dot{\theta} = 0$ por lo que se tiene que llevar a un ángulo de inclinación en un cierto tiempo lo cual se realizara mediante una trayectoria. Con base en el polinomio (7.1) y a la ecuación para obtener los coeficientes del polinomio (7.10) se cálculara la trayectoria. La trayectoria tiene un punto inicial en $\theta_{P_i} = 0$, $\dot{\theta}_{P_i} = 0$ y un punto final en $(\theta_{P_f}, \dot{\theta}_{P_f})$.

Segunda etapa de la trayectoria del péndulo invertido para la caminata bípeda.

En esta etapa el robot humanoide una de sus piernas tiene una inclinación basada en los ángulos obtenidos mediante la cinemática inversa del último valor de la trayectoria del péndulo invertido por lo que se tiene que conmutar el control de seguimiento de trayectorias para la otra pierna.

Las trayectorias (Figura 7.3) se realiza mediante los siguiente polinomios:

$$P_n(t) = \begin{bmatrix} 0 & 0 & 0 & 1{,}7453 & -2{,}6180 & 1{,}0472 \\ 5{,}5859 & -20{,}944 & 31{,}4159 & -22{,}6893 & 7{,}854 & -1{,}0472 \\ -89{,}3609 & 188{,}4956 & -157{,}0796 & 64{,}5772 & -13{,}09 & 1{,}0472 \\ 513{,}8249 & -753{,}9822 & 439{,}8230 & -127{,}4090 & 18{,}3260 & -1{,}0472 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \\ t^4 \\ t^5 \end{bmatrix}, con \quad n = 1,2,3,4 \tag{7.11}$$

$$P_1(t) \quad 0 \le t < t_1, \quad P_2(t) \quad t_1 \le t < t_2, \quad P_3(t) \quad t_2 \le t < t_3, \quad P_4(t) \quad t_3 \le t \le t_4 \tag{7.12}$$

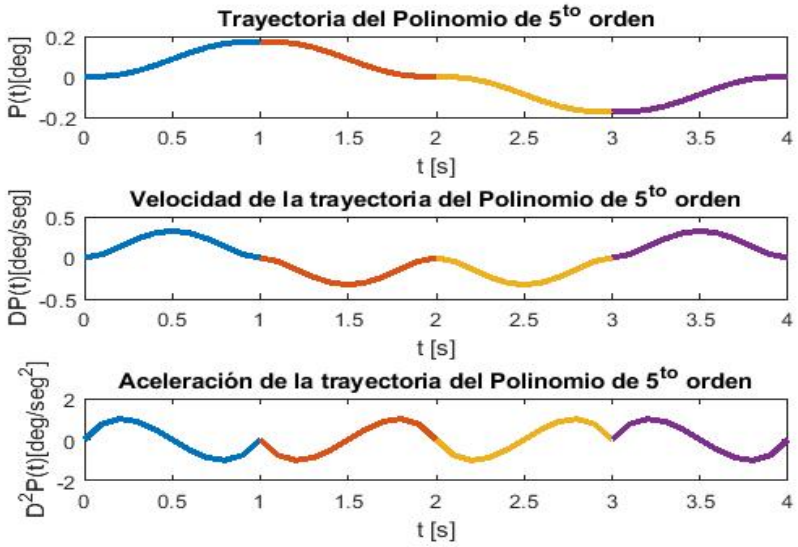


Figura 7.3: perfil de la trayectoria polinomica y sus derivadas formada por los polinomios ($P_n(t)$, con $n = 1, 2, 3, 4$)

En Figura 7.4 se muestra el algoritmo para activar las trayectorias y conmutar las piernas para inclinar el robot.

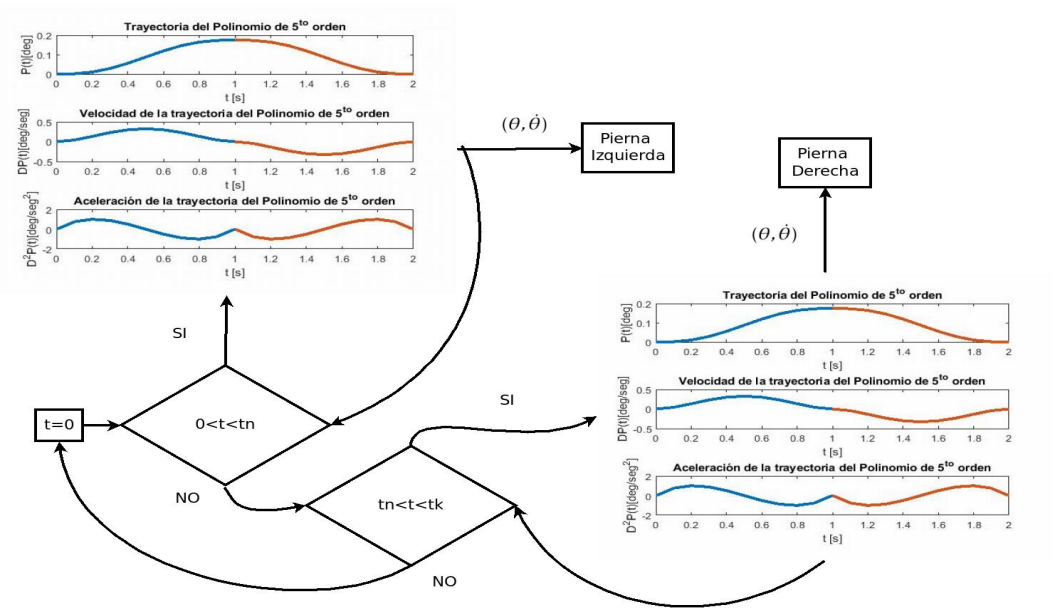


Figura 7.4: Diagrama a bloques para el segimiento de trayectorias

Trayectorias para la generación de los pasos del robot humanoide.

Para generar estas trayectorias primero se supone que el ángulo de la rodilla y del tobillo (θ_2,θ_1) se mantienen constantes, por lo que solo se cálculan las trayectorias para la cadera (θ_3). En la Figura 7.5 se muestra el recorrido de una de las piernas, por lo tanto se deben generar las trayectorias para cada ángulo el cual nos permitira realizar este movimiento mostrado.

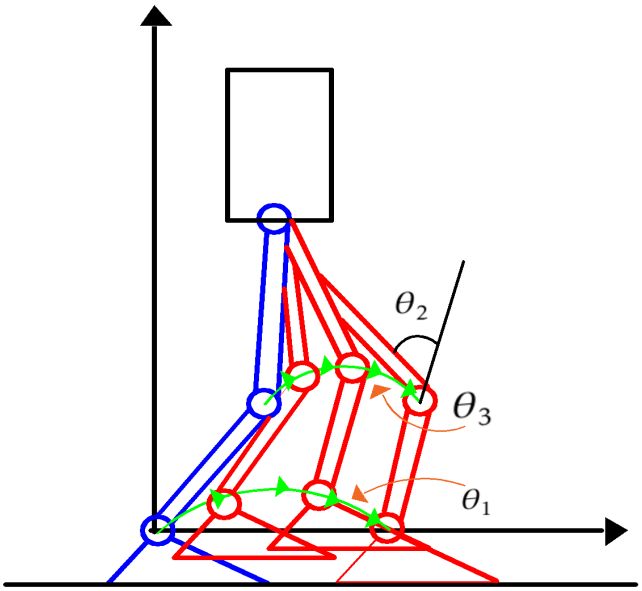


Figura 7.5: Movimientos para generar los pasos del robot humanoide (θ_1= Tobillo, θ_2= Rodilla, θ_3= Cadera)

Para generar estas trayectorias se utilizan las referencias ya mostradas en la Figura 6.4 de los ángulos que toma θ_3 y con base a la ecuación matricial (7.10) para cálcular los coeficientes del polinomio y la trayectoria se debe realizar en el mismo tiempo que las trayectorias de inclinación del robot humanoide.

Los polinomios para la trayectoria de θ_3 son los siguientes:

$$P_n(t) = \begin{bmatrix} -0{,}1746 & 0 & 0 & -1{,}7453 & 2{,}6180 & -1{,}0472 \\ -5{,}5859 & 20{,}944 & -31{,}4159 & 22{,}6893 & -7{,}854 & 1{,}0472 \\ 89{,}3609 & -188{,}4956 & 157{,}0796 & -64{,}5772 & 13{,}09 & -1{,}0472 \\ -513{,}8249 & 753{,}9822 & -439{,}8230 & 127{,}4090 & -18{,}3260 & 1{,}0472 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \\ t^4 \\ t^5 \end{bmatrix}, con \quad n = 1,2,3,4 \tag{7.13}$$

$$P_1(t) \quad 0 \leq t < t_1, \quad P_2(t) \quad t_1 \leq t < t_2, \quad P_3(t) \quad t_2 \leq t < t_3, \quad P_4(t) \quad t_3 \leq t \leq t_4 \tag{7.14}$$

La gráfica de los polinomios en conjunto para generar la trayectoria se muestran en la Figura 7.6.

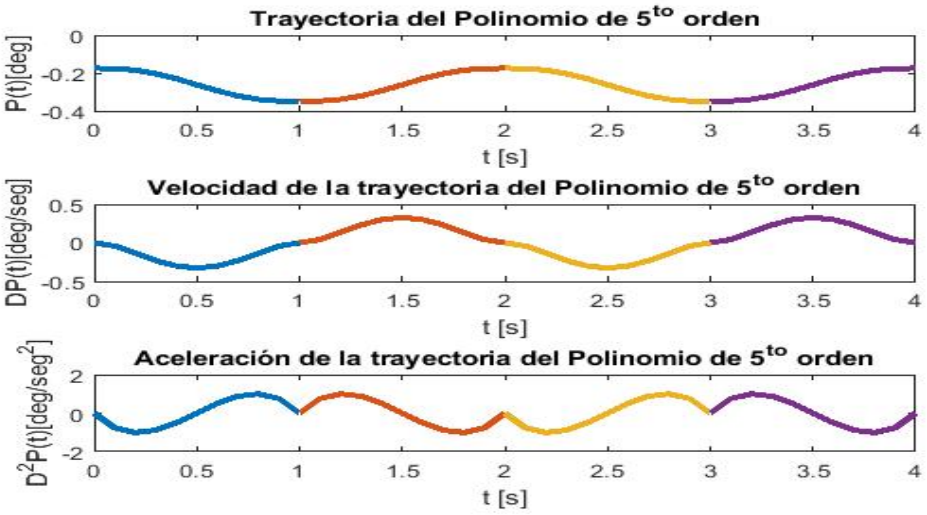


Figura 7.6: Perfil de la trayectoria ($P_1(t)$, $P_2(t)$, $P_3(t)$, $P_4(t)$) para la cadera y sus derivadas.

Las trayectoria para la cadera también debe iniciar y terminar en el tiempo de la trayectoria para las inclinaciones del péndulo invertido ya que esto da sincrónia a los movimientos del robot humanoide.

Capítulo 8

Control difuso de postura

8.1. Diseño del controlador difuso de postura

El diseño del algoritmo para el control difuso principalmente esta basado en las variables que se desean controlar, las cuales son la posición angular (θ) y la velocidad angular ($\dot{\theta}$) suponiendo que el robot humanoide tiene el movimiento parecido al de un péndulo invertido en la Figura 10.3 se muestra el diagrama de control para garantizar la estabilidad del robot humanoide en un punto de referencia dado ($\theta_d, \dot{\theta}_d$).

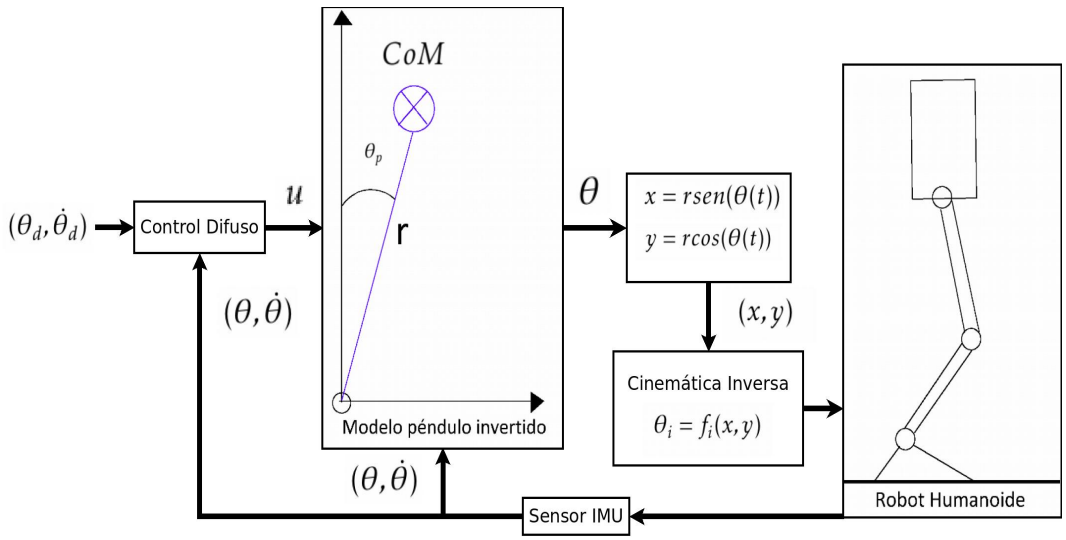


Figura 8.1: Diagrama de control para la postura

El control difuso utilizado esta basado en la inferencia difusa de Takagi-Sugeno, este controlador nos permite combinar el control moderno o alguna otra ley de control con la lógica difusa para agrupar controles lineales diseñados para diferentes sistemas lineales. Los controladores lineales están basados en realimentación de estados por lo que se requiere del modelo matemático del péndulo invertido el cual se obtiene mediante las ecuaciones de Euler-Lagrange.

Modelo matemático del péndulo invertido

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = \tau, \qquad Ecuación \quad de \quad Euler - Lagrange \tag{8.1}$$

$$\mathcal{L} = \mathcal{E}_c - \mathcal{E}_p, \qquad dónde \quad \mathcal{E}_c, \quad \mathcal{E}_p \quad son \quad la \quad Energía \quad cinética \quad y \quad potencial \tag{8.2}$$

$$\mathcal{E}_c = \frac{1}{2}mv^2, \qquad \mathcal{E}_p = mgh \tag{8.3}$$

La posición y velocidad del péndulo invertido están dadas por:

$$x = rsen(\theta), \qquad y = rcos(\theta) = h \tag{8.4}$$

$$\dot{x} = r\dot{\theta}cos(\theta), \qquad \dot{y} = -r\dot{\theta}sen(\theta), \qquad v = \sqrt{\dot{x}^2 + \dot{y}^2} = r\dot{\theta} \tag{8.5}$$

Sustituyendo (8.5) y (8.4) en (8.3) se obtiene:

$$\mathcal{E}_c = \frac{1}{2}mr^2\dot{\theta}^2, \qquad \mathcal{E}_p = mgrcos(\theta) \tag{8.6}$$

Ahora sustituyendo (8.6) en (8.2):

$$\mathcal{L} = \frac{1}{2}mr^2\dot{\theta}^2 - mgrcos(\theta) \tag{8.7}$$

Por último sustituyendo (8.7) en (8.1) y resolviendo se obtiene la ecuación diferencial del péndulo invertido simple:

$$\ddot{\theta} - \frac{g}{r}sen(\theta) = \frac{1}{mr^2}\tau \tag{8.8}$$

Para el diseño de los controladores lineales se requiere el sistema lineal el cual se obtiene mediante la linealización de la ecuación de estados asociada a (8.8), entonces la ecuación de estados del péndulo invertido esta dada por:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \qquad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{g}{r}sen(x_1) + \frac{1}{mr^2}\tau \end{bmatrix} \tag{8.9}$$

Linealización de sistemas no lineales

Considerando un sistema autónomo en lazo abierto $\dot{x} = f(x,u)$, se asume que $f(x,u)$ es continua y diferenciable con $f(x_{eq}, u_{eq}) = 0$ [5]. Se puede escribir:

$$\dot{x} = \left(\frac{\partial f(x,u)}{\partial x}\right)_{x=x_{eq},u=u_{eq}} \cdot x + \left(\frac{\partial f(x,u)}{\partial u}\right)_{x=x_{eq},u=u_{eq}} \cdot u + f_h(x,u) \tag{8.10}$$

dónde $f_h(x,h)$ son los términos de orden superior de x y u. Se denotará la matriz jacobiana de f con respecto a x en $x = x_{eq}, u = u_{eq}$ con A y se denota con B la matriz jacobiana de f con respecto a u y en el mismo punto $(x = x_{eq}, u = u_{eq})$, entonces

$$A = \left(\frac{\partial f(x,u)}{\partial x}\right)_{x=x_{eq},u=u_{eq}} \qquad B = \left(\frac{\partial f(x,u)}{\partial u}\right)_{x=x_{eq},u=u_{eq}} \tag{8.11}$$

omitiendo $f_h(x,u)$, se obtiene la ecuación:

$$\dot{x} = A \cdot x + B \cdot u \tag{8.12}$$

Con base en lo anterior se obtiene el modelo matemático lineal del péndulo invertido

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} f_1(x_1,x_2,u) \\ f_2(x_1,x_2,u) \end{bmatrix} \qquad con \quad f_1(x_1,x_2,u) = x_2,\ f_2(x_1,x_2,u) = \frac{g}{r}sen(x_1) + \frac{1}{mr^2}u \tag{8.13}$$

Con (8.11) y (8.13) se obtendrá la matriz A y B asociada al modelo lineal del péndulo invertido

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}_{x_{eq},u_{eq}} = \begin{bmatrix} 0 & 1 \\ \frac{g}{r}cos(x_{1eq}) & 0 \end{bmatrix}_{x_{eq},u_{eq}} \qquad B = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix}_{x_{eq},u_{eq}} = \begin{bmatrix} 0 \\ \frac{1}{mr^2} \end{bmatrix}_{x_{eq},u_{eq}} \tag{8.14}$$

Con lo cual se obtiene el modelo lineal del péndulo invertido:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{r}cos(x_{1eq}) & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{mr^2} \end{bmatrix} u \tag{8.15}$$

Puntos de equilibrio

Los puntos de equilibrio (x^p_{eq}) para los sistemas linelaes están relacionados con el máximo grado de pertenencia que se obtienen de las funciones de mémbresia propuestas $(\mu^p_r(x_n))$. Las funciones de membresía triangulares contienen un valor único real donde $max\{\mu^p_r(x^p_c)\} = 1$ por lo cual $x^p_c = x^p_{eq}$, entonces esto se puede expresar como:

$$\begin{bmatrix} x^1_{eq} \\ x^2_{eq} \\ \vdots \\ x^p_{neq} \end{bmatrix} = \begin{bmatrix} max\{(\mu^1_1(x_1))^{-1}\} & \cdots & max\{(\mu^1_1(x_n))^{-1}\} \\ max\{(\mu^2_1(x_1))^{-1}\} & \cdots & max\{(\mu^2_2(x_n))^{-1}\} \\ \vdots & \ddots & \vdots \\ max\{(\mu^p_r(x_1))^{-1}\} & \cdots & max\{(\mu^p_r(x_n))^{-1}\} \end{bmatrix} = \begin{bmatrix} x^1_{1eq} & \cdots & x^1_{neq} \\ x^2_{1eq} & \cdots & x^2_{neq} \\ \vdots & \ddots & \vdots \\ x^p_{1eq} & \cdots & x^p_{neq} \end{bmatrix} \tag{8.16}$$

Dónde n es el número de variables de estados, r es el número de funciones de membresía, p esta relacionada con el número de sistemas lineales y también con la cantidad de variables de estado difusas, $(\mu^p_r(x_n))^{-1}$ es la función inversa de $\mu^p_r(x_n)$, entonces p esta definida de la siguiente forma:

$$p = r^n \tag{8.17}$$

Las variables de estado difusas pueden expresarse de la siguiente forma:

$$\begin{bmatrix} \mathcal{FX}^1 \\ \mathcal{FX}^2 \\ \vdots \\ \mathcal{FX}^j \\ \vdots \\ \mathcal{FX}^p \end{bmatrix} = \begin{bmatrix} \mu_1^1(x_1) & \mu_1^1(x_2) & \cdots & \mu_1^1(x_n) \\ \mu_1^2(x_1) & \mu_1^2(x_2) & \cdots & \mu_2^2(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_1^j(x_1) & \mu_1^j(x_2) & \cdots & \mu_r^j(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_r^p(x_1) & \mu_r^p(x_2) & \cdots & \mu_r^p(x_n) \end{bmatrix} \tag{8.18}$$

Funciones de membresía utilizadas para el control del robot humanoide
Las funciones de membresía asociadas a las variable de estados x_1 y x_2 son definidas por funciones triangulares como se muestran en las Figuras 8.2 y 8.3.

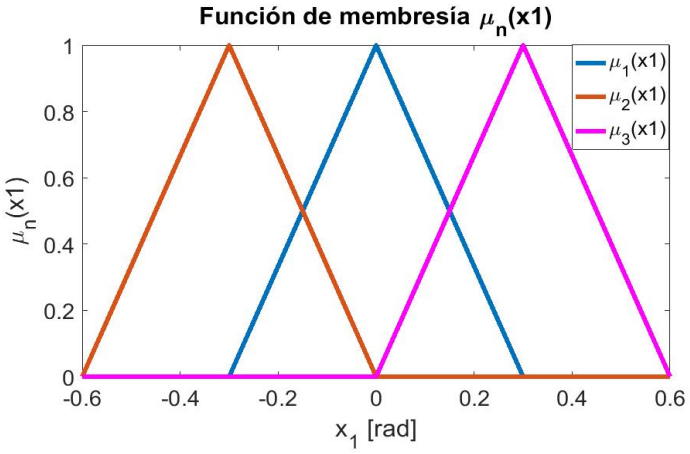


Figura 8.2: Funciones de membresía asociadas a la variable x_1

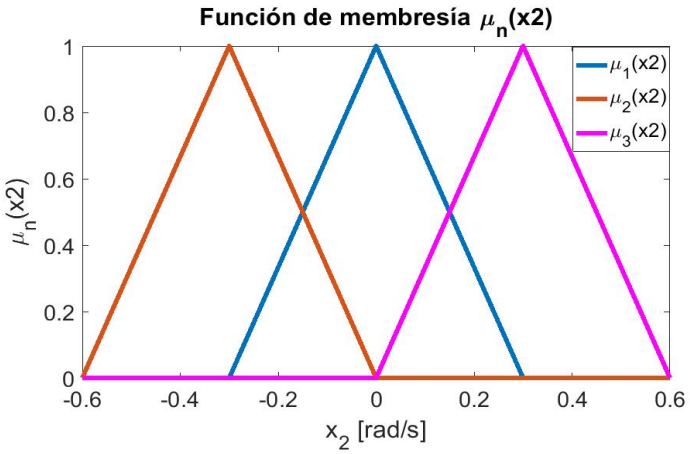


Figura 8.3: Funciones de membresía asociadas a la variable x_2

En la siguiente Figura 8.4 se muestra a las funciones de membresía asociadas a las varibales de estado y a la región difusa que se genera.

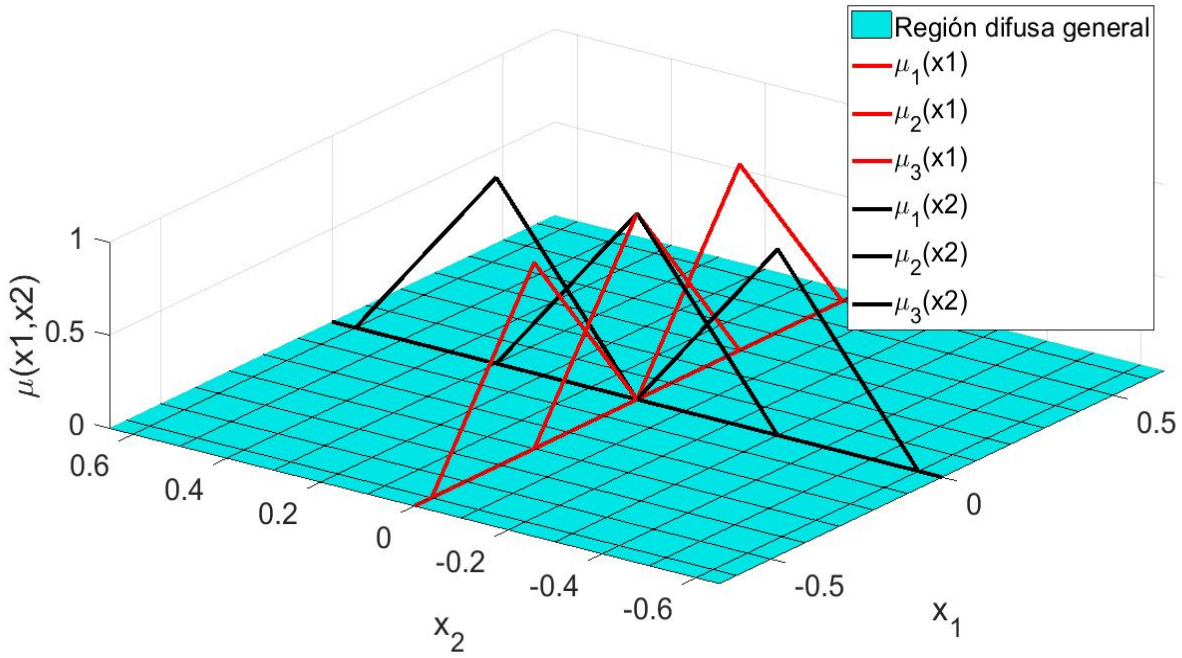


Figura 8.4: Funciones de membresía asociadas a las variables x_1, x_2

Con base en las Figuras 8.2, 8.3 y en la ecuación (8.18) se obtienen las variables de estados difusas como se muestra a continuación:

$$\begin{bmatrix} \mathcal{FX}^1 \\ \mathcal{FX}^2 \\ \mathcal{FX}^3 \\ \mathcal{FX}^4 \\ \mathcal{FX}^5 \\ \mathcal{FX}^6 \\ \mathcal{FX}^7 \\ \mathcal{FX}^8 \\ \mathcal{FX}^9 \end{bmatrix} = \begin{bmatrix} \mu_1^1(x_1) & \mu_1^1(x_2) \\ \mu_1^2(x_1) & \mu_2^2(x_2) \\ \mu_1^3(x_1) & \mu_3^3(x_2) \\ \mu_2^4(x_1) & \mu_1^4(x_2) \\ \mu_2^5(x_1) & \mu_2^5(x_2) \\ \mu_2^6(x_1) & \mu_3^6(x_2) \\ \mu_3^7(x_1) & \mu_1^7(x_2) \\ \mu_3^8(x_1) & \mu_2^8(x_2) \\ \mu_3^9(x_1) & \mu_3^9(x_2) \end{bmatrix} \tag{8.19}$$

Debido a que se tienen nueve variables de estados difusas por lo tanto existen nueve puntos de equilibrio, con base a la ecuación (8.16) y a las Figuras 8.2, 8.3 se obtienen los siguientes puntos de equilibrio:

$$\begin{bmatrix} x_{eq}^1 \\ x_{eq}^2 \\ x_{eq}^3 \\ x_{eq}^4 \\ x_{eq}^5 \\ x_{eq}^6 \\ x_{eq}^7 \\ x_{eq}^8 \\ x_{eq}^9 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}3 \\ 0 & 0{,}3 \\ -0{,}3 & 0 \\ -0{,}3 & -0{,}3 \\ -0{,}3 & 0{,}3 \\ 0{,}3 & 0 \\ 0{,}3 & -0{,}3 \\ 0{,}3 & 0{,}3 \end{bmatrix} \tag{8.20}$$

Sustituyendo los puntos de equilibrio (8.20) en el sistema lineal (8.15) se obtienen nueve sistemas lineales de la forma:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ \frac{g}{r}cos(x^p_{1eq}) & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{mr^2}\end{bmatrix}u \qquad \forall\ p=1,2,3,4,\ldots,9 \tag{8.21}$$

Sustituyendo las constantes $r = 0{,}525\ [m]$, $6{,}77\ [Kg]$, $g = 9{,}72\ \left[\frac{m}{s^2}\right]$ y los puntos de equilibrio x^p_{eq}, se obtienen las siguientes matrices A_n y B:

$$A_1=\begin{bmatrix}0 & 1\\ 18{,}5143 & 0\end{bmatrix} \quad con \quad A_2=A_3=A_1 \tag{8.22}$$

$$A_4=\begin{bmatrix}0 & 1\\ 17{,}6874 & 0\end{bmatrix} \quad con \quad A_5=A_6=A_7=A_8=A_9=A_4 \tag{8.23}$$

$$B=\begin{bmatrix}0\\ 0{,}5359\end{bmatrix} \tag{8.24}$$

Por lo tanto los nueve sistemas lineales en variables de estados se reducen a dos sistemas y se pueden expresar de la sigiente forma:

$$\dot{x}=A_1x+Bu \qquad \dot{x}=A_4x+Bu \tag{8.25}$$

Con base en los sistemas lineales (8.25) se calculan los controladores LQR para la realimentación de estado, para la implementación se debe calcular un control discreto LQR por lo que los sistemas se deben discretizar mediante:

$$A^j_d=e^{A_jTs} \qquad con \quad j=0,1,\ldots,p \tag{8.26}$$

$$B^j_d=\int_0^{Ts}e^{A_j\tau}B_jd\tau \qquad con \quad j=0,1,\ldots,p \tag{8.27}$$

si A es invertible entonces:

$$B^j_d=A_j^{-1}(A^j_d-I)B_j \tag{8.28}$$

El sistema discreto [20] esta dado por:

$$x_{k+1}=A^j_dx_k+B^j_du_k \tag{8.29}$$

Análisis de estabilidad para sistemas discretos lineales.

Teorema 8.1 (Estabilidad para tiempo discreto [20]) *Si la solución de la ecuación de Lyapunov*

$$(A^j_d)^TP^j_dA^j_d-P^j_d+Q_d=\overline{0} \tag{8.30}$$

existe y es única con $P^j_d=(P^j_d)^T>0$ (Matriz Hermitiana), $\forall\ Q_d>0$ (Simétrica y Positiva definida), entonces el origen (x_{eq}) del sistema $x_{k+1}=A^j_dx_k$ es asintoticamente estable en el sentido de Lyapunov, y además implica que los valores característicos de A^j_d cumplen que $\| \lambda(A^j_d) \|<1$

La discretización se cálcula mediante la función **c2d(Sistema_continuo,Ts)** de Matlab basadas en las ecuaciones (8.26) y (8.28) dado esto se obtienen las matrices discretas A^j_d:

$$A^1_d=\begin{bmatrix}1{,}094 & 0{,}1031\\ 1{,}909 & 1{,}094\end{bmatrix} \quad con \quad A^2_d=A^3_d=A^1_d \tag{8.31}$$

$$A^4_d=\begin{bmatrix}1{,}09 & 0{,}103\\ 1{,}821 & 1{,}09\end{bmatrix} \quad con \quad A^5_d=A^6_d=A^7_d=A^8_d=A^9_d=A^4_d \tag{8.32}$$

La ecuación de Lyapunov (8.30) se puede resolver mediante la función **dlyap(Ad,Qd)** de matlab, la variable a encontrar es P^j_d, y con base a los sistemas cálculados anteriormente se obtiene:

$$P^1_d=1\times10^5\begin{bmatrix}-0{,}2613 & 0\\ 0 & 4{,}8387\end{bmatrix} \quad con \quad P^2_d=P^3_d=P^1_d \tag{8.33}$$

$$\lambda(A_d^1) = \{1{,}5376, 0{,}6504\} \qquad con \qquad \lambda(A_d^2) = \lambda(A_d^3) = \lambda(A_d^1) \tag{8.34}$$

$$P_d^4 = 1 \times 10^4 \begin{bmatrix} -0{,}0878 & -0{,}0003 \\ -0{,}0003 & 1{,}5535 \end{bmatrix} \quad con \quad P_d^5 = P_d^6 = P_d^7 = P_d^8 = P_d^9 = P_d^4 \tag{8.35}$$

$$\lambda(A_d^4) = \{1{,}5231, 0{,}6569\} \qquad con \qquad \lambda(A_d^5) = \lambda(A_d^6) = \lambda(A_d^7) = \lambda(A_d^8) = \lambda(A_d^9) = \lambda(A_d^1) \tag{8.36}$$

Con base en los cálculos anteriores $P_d^j \ \forall \ j = 1, 2, 3, \ldots, 9$ no es positiva definida y $\| \lambda(A_d^j) \| \ \forall \ j = 1, 2, 3, \ldots, 9$ no cumple que $\lambda(A_d^j) < 0$ por lo que los sistemas son inestables.

Análisis de la controlabilidad para los sistemas lineales discretos

Teorema 8.2 (Controlabilidad para sistemas lineales [20]) *El par (A_d^j,B_d^j) se dice que es controlable si se cumple cualquiera de las siguientes condiciones:*

1. *La matriz de controlabilidad* $\mathcal{C}^j = \begin{bmatrix} B_d^j & A_d^j B_d^j & (A_d^j)^2 B_d^j & \ldots & (A_d^j)^{n-1} B_d^j \end{bmatrix}$ *es de rango completo* $\rho(\mathcal{C}^|) = n$ *donde n es el número de estados.*
2. *La matriz de funciones*

$$w_c^j(t) = \int_0^\infty e^{A_d^j t} B_d^j (B_d^j)^T e^{(A_d^j)^T \tau} d\tau$$

   es invertible $\forall \ t > 0$.
3. *La matriz* $\begin{bmatrix} \lambda^j I - A_d^j & | & B_d^j \end{bmatrix}$ *es de rango completo* $\forall \ \lambda^j - \lambda_i^j(A_d^j)$, $i = 1, \ldots, n$

Aplicando el Teorema 8.2 a los sistemas discretos $x_{k+1} = A_d^j x_k + B_d^j u_k \ \forall \ j = 1, 2, \ldots, 9$. Las matrices de controlabilidad estan dadas por:

$$\mathcal{C}^j = \begin{bmatrix} B_d^j & A_d^j B_d^j \end{bmatrix} = \begin{bmatrix} 0{,}0027 & 0{,}0087 \\ 0{,}0553 & 0{,}0657 \end{bmatrix}, \quad \forall \quad j = 1, 2, 3 \tag{8.37}$$

$$\mathcal{C}^j = \begin{bmatrix} B_d^j & A_d^j B_d^j \end{bmatrix} = \begin{bmatrix} 0{,}0027 & 0{,}0086 \\ 0{,}0552 & 0{,}0651 \end{bmatrix}, \quad \forall \quad j = 4, \ldots, 9 \tag{8.38}$$

Con base en las matrices de controlabilidad (8.37) y (8.38) el rango es igual a:

$$\rho(\mathcal{C}^j) = 2 \quad \forall \quad j = 1, 2, \ldots, 9 \tag{8.39}$$

De la ecuación (9.17) podemos decir que el rango para las matrices de controlabilidad es igual al número de estados que componen a los sistemas lineales discretos y como se cumple alguna de las condiciones del Teorema (8.2) entonces se puede llegar a la conclusión de que los sistemas $x_{k+1} = A_d^j x_k + B_d^j u_k \ \forall \ j = 1, \ldots, 9$ son controlables.

Cálculo de las ganancias para el controlador LQR basado en modelo [20].

Para determinar las ganancias del control discreto se debe resolver la ecuación de Riccati la cual es:

$$P_d^j = Q_d^j + (A_d^j)^T P_d^j (I + B_d^j (R_d)^{-1} (B_d^j)^{-1} P_d^j)^{-1} A_d^{-1} \tag{8.40}$$

La ganancia discreta se cálcula mediante:

$$K_d^j = (R_d + (B_d^j)^T P_d^j B_d^j)^{-1} (B_d^j)^T P_d^j A_d^j \tag{8.41}$$

Dónde Q_d y R_d son los pesos de importancia que se le asignan a las variables de estados. La función **lqrd(A,B,Q,R,Ts)** en matlab cálcula la ganancia óptima para el controlador apartir del sistema continuo. Para el diseño de las ganancias se asignan los pesos de importancia Q y R como sigue:

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \qquad R = 1 \tag{8.42}$$

El valor del tiempo de muestreo se asignó basado en la publicación de los datos en **ROS** la cual es: Ts=0.1 $[seg]$. Las ganancias cálculadas son:

$$K_d^1 = \begin{bmatrix} 56{,}9585 & 13{,}3858 \end{bmatrix} \qquad K_d^2 = \begin{bmatrix} 54{,}6276 & 13{,}1427 \end{bmatrix} \tag{8.43}$$

entonces la ley de control difusa se cálcula mediante:

$$f_1(x_k^1, x_k^2) = \begin{bmatrix} 56{,}9585 & 13{,}3858 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} \tag{8.44}$$

$$f_2(x_k^1, x_k^2) = \begin{bmatrix} 54{,}6276 & 13{,}1427 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} \tag{8.45}$$

$$u_k = \frac{(w_1 + w_2 + w_3) f_1(x_k^1, x_k^2) + (w_4 + w_5 + w_6 + w_7 + w_8 + w_9) f_2(x_k^1, x_k^2)}{w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9} \tag{8.46}$$

dónde x_k^1 y x_k^2 son las variable de estados discretas, u_k es la entrada discreta y w_n son los niveles de activación los cuales se pueden cálcular mediante:

$$w_p = \mu_r^p(x_k^1)\mu_r^p(x_k^2)\ldots\mu_r^p(x_k^n) \qquad \forall\ p = 1, 2, 3, \ldots \tag{8.47}$$

8.2. Simulaciones

Para realizar las simulaciones se requiere de modelos matemáticos que permitan predecir los movimientos del robot. El modelo continuo del péndulo invertido permite predecir el movimiento del balanceo del robot, debido a que este modelo tiene que ser programado en un nodo de **ROS** para predecir los movimientos esto implica que no se puede utilizar el modelo continuo del péndulo invertido por eso se utiliza el método número rico de Euler para reslover ecuaciones diferenciales.

Método de Euler [21]

Cosiderando un problema de aproximación de una función continua $x = f(t)$ para $t \geq 0$ la cual satisface a la ecuación diferencial:

$$\dot{x} = F(t, x, u(t)) \qquad x(0) = \alpha \tag{8.48}$$

la definición de la derivada es:

$$\dot{x} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h} \tag{8.49}$$

si se supone que h es pequeña pero $h > 0$ entonces se puede aproximar a la derivada como:

$$\dot{x} \approx \frac{f(t+h) - f(t)}{h} \tag{8.50}$$

Sustituyendo (8.50) en (8.48) se obtiene:

$$\frac{f(t+h) - f(t)}{h} = F(t, x, u(t)) \tag{8.51}$$

como se desea obtener el valor siguiente se debe despejar $f(t+h)$

$$f(t+h) = f(t) + hF(t, x, u(t)) \tag{8.52}$$

reescribiendo la ecuación en terminos de x se obtiene:

$$x_{t+h} = x + hF(t, x, u(t)) \tag{8.53}$$

Ahora expresamos la ecuación (8.53) de forma recursiva con la siguiente expresión:

$$x_{k+1} = x_k + hF(t_k, x_k, u_k) \qquad k = 0, 1, 2, \ldots, n-1 \tag{8.54}$$

comenzando en $x_0 = \alpha$.

Con base en lo anterior se obtiene el modelo matemático del péndulo invertido de la forma recursiva mediante la siguiente ecuación:

$$\begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \end{bmatrix} = \begin{bmatrix} x^1_k + hx^2_k \\ x^2_k + h\left(\frac{g}{r} sen(x^1_{k+1}) + \frac{1}{mr^2} u_k\right) \end{bmatrix} \qquad x^1_0 = \alpha \quad x^2_0 = 0 \quad h = 0{,}1 \tag{8.55}$$

dónde x^1_{k+1} y x^1_k están asociadas a la variable de estado x_1, x^2_{k+1} y x^2_k a la variable de estado x_2, u_k está asociada con la entrada u de la ecuación diferencial.

Para obtener la solución numérica en lazo abierto de la ecuación diferencial del péndulo invertido suponiendo que las condiciones iniciales son $x^1_0 = 0{,}1745\ [rad]$, $x^2_0 = 0\ [rad/seg]$ y $u_k = 0$ se muestra con la gráfica de la Figura 8.5

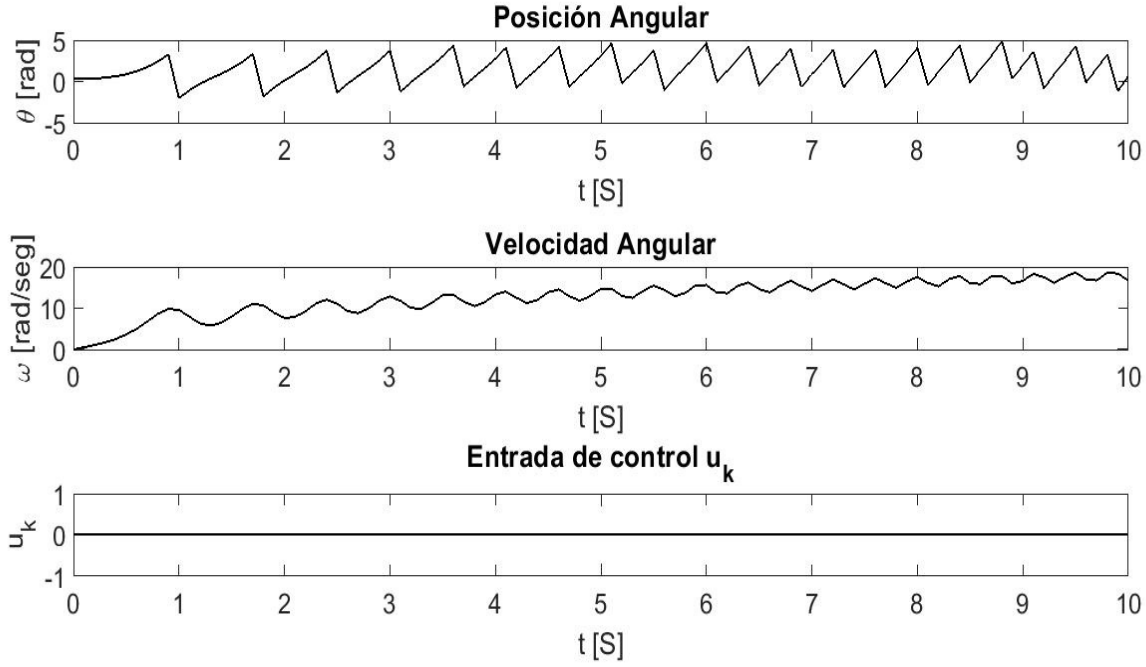


Figura 8.5: Gráfica de la solución numérica de la ecuación diferencial del péndulo invertido

Para simular la solución de la ecuación diferencial numérica con el control difuso ya previamente cálculado utilizamos las mismas condiciones iniciales y suponemos que el sensor es ideal en la Figura 8.6 se muestra el digrama de control discreto para este caso y en las Figuras 8.7 y 8.8 observamos las gráficas resultantes del control y de los niveles de activación del control difuso.

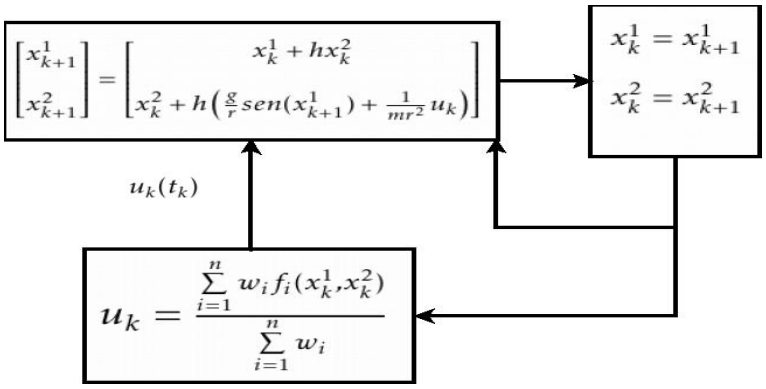


Figura 8.6: Diagrama de control discreto

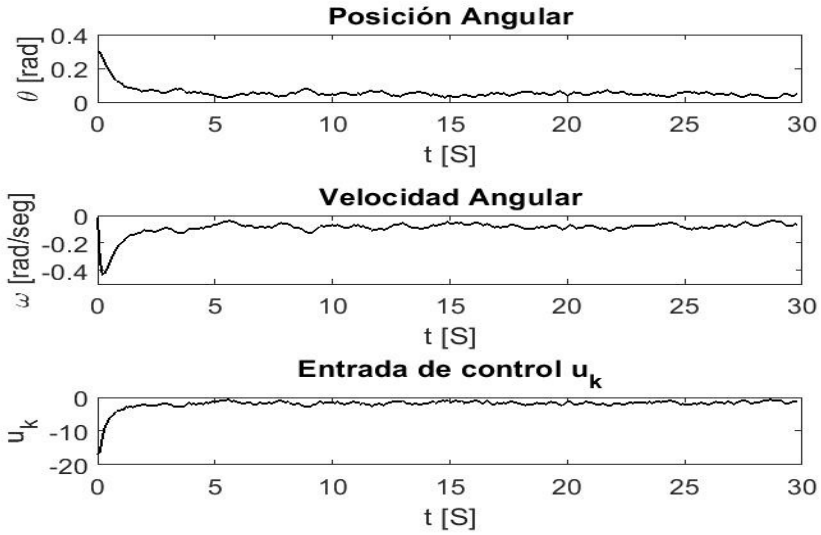


Figura 8.7: Gráfica del control difuso con la aproximación numérica del péndulo invertido

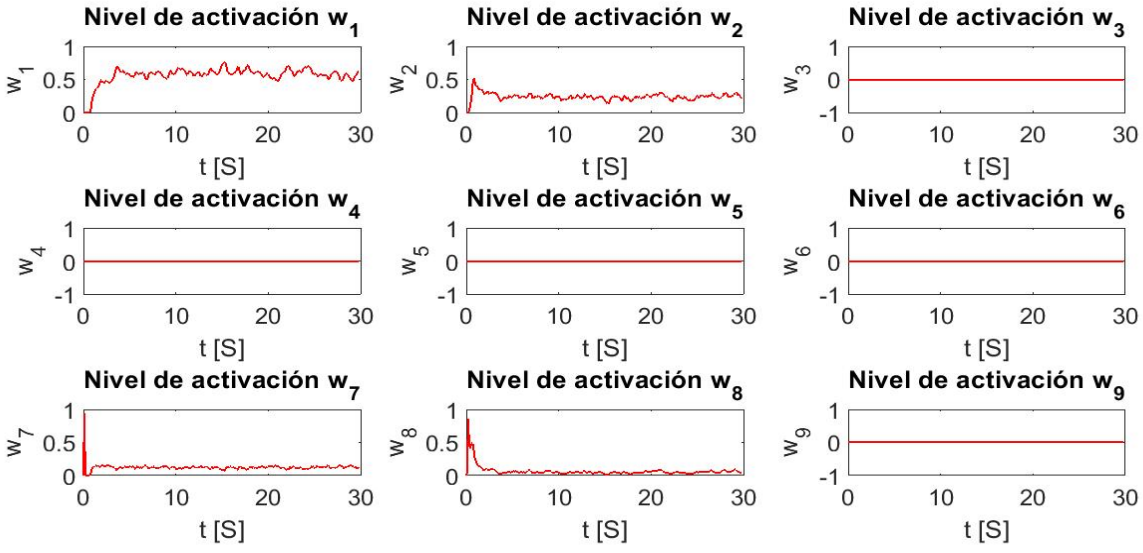


Figura 8.8: Gráficas de los niveles de activación para el control difuso

Capítulo 9

Control difuso para caminata

9.1. Diseño del controlador difuso para la marcha bípeda

EL diseño del control difuso para la caminata esta basado fundamentalmente en el segimiento de trayectorias para ángulos de inclinación del robot humanoide. El control difuso solo controla el movimiento de la pierna fija al suelo, para controlar la pierna movil o la pierna que iniciara con el paso se cálcula trayectorias las cuales generaran los moviminetos de esta (**Capítulo 7**). En la Figura 9.1 se muestra un diagrama a bloques para generar la caminata bípeda basada en el control de un péndulo invertido.

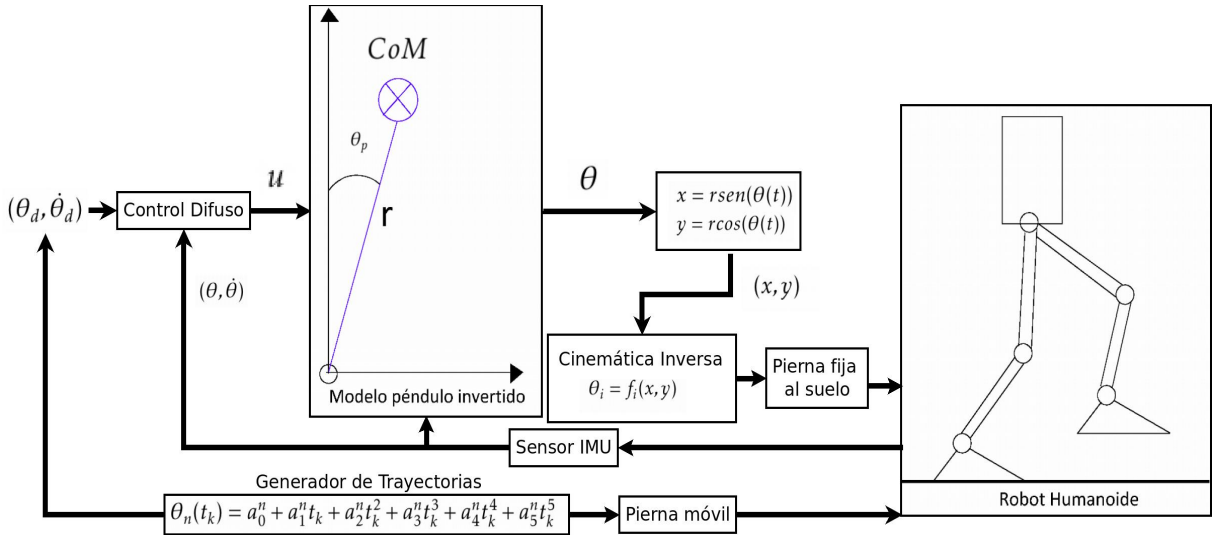


Figura 9.1: Diagrama de control para la caminta

Para el segimiento de trayectorias ($\theta = P(t)$) se requiere un sistema de control con entrada de referencia [22].

El control de seguimiento de trayectorias está basado en el error de los estados por lo que se debe reescribir el modelo matemático discreto del péndulo invertido suponiendo que:

$$x_{ek}^1 = r(t_k) - x_k^1 \qquad x_{ek}^2 = \dot{r}(t_k) - x_k^2 \tag{9.1}$$

El modelo discreto mediante la solución de Euler está dado de la siguiente forma:

$$\begin{bmatrix} x_{e(k+1)}^1 \\ x_{e(k+1)}^2 \end{bmatrix} = \begin{bmatrix} x_{ek}^1 + hx_{ek}^2 \\ x_{ek}^2 + h\left(\frac{g}{r} sen(x_{e(k+1)}^1) + \frac{1}{mr^2} u_k\right) \end{bmatrix} \qquad x_0^1 = \alpha \quad x_0^2 = 0 \quad h = 0{,}1 \tag{9.2}$$

El diagrama de control difuso propuesto para el seguimiento de las trayectorias se muestra en la Figura 9.2.

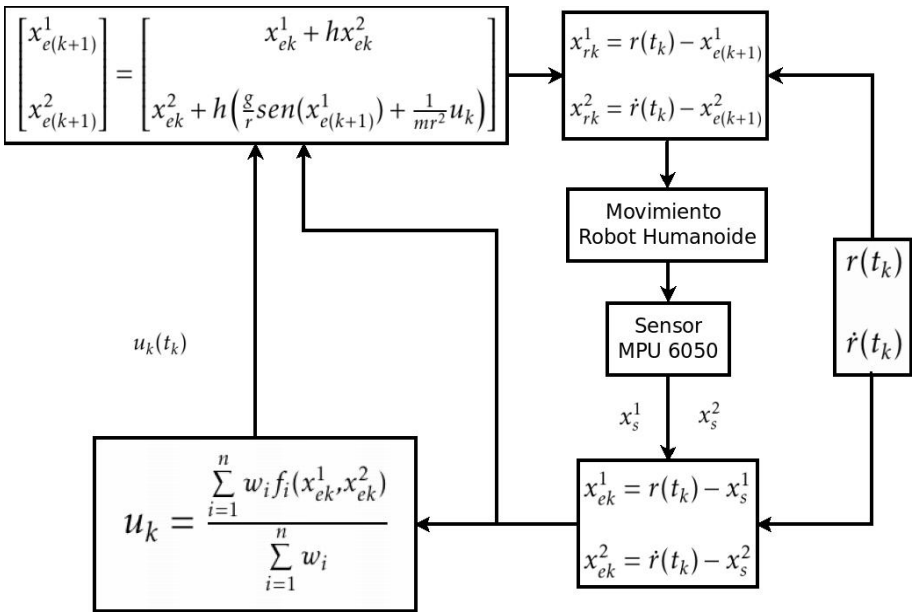


Figura 9.2: Diagrama de control difuso discreto con referencia basado en el error de los estados

Se definirán las funciones de membresía para cada variable de estado basadas en el error como se muestran en las Figuras 9.3 y 9.4.

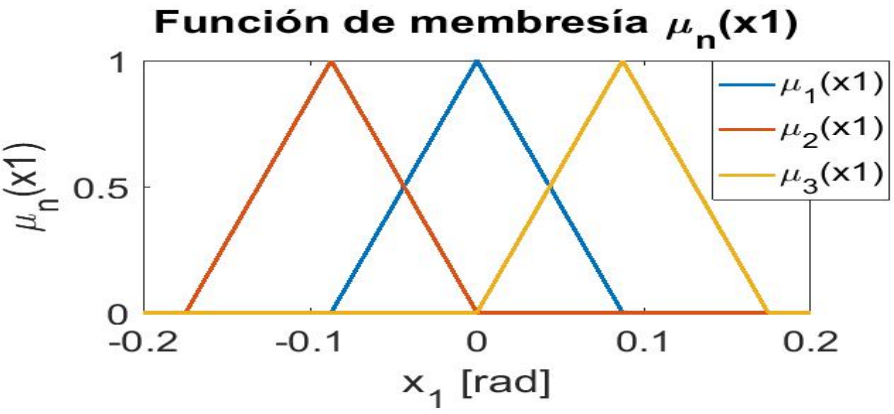


Figura 9.3: Funciones de membresía asociadas a la variable x_e^1

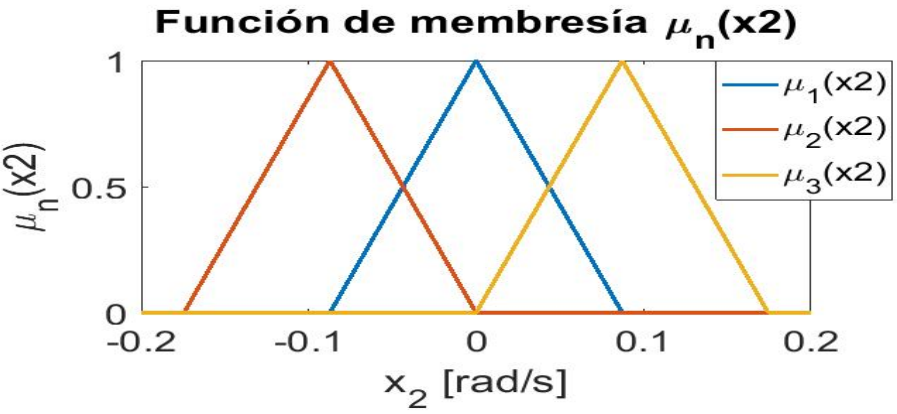


Figura 9.4: Funciones de membresía asociadas a la variable x_e^2

En la Figura 9.2 se muestran las funciones de membresía para las variables de estados x_e^1, x_e^2 y la región difusa que se forma al unirlas.

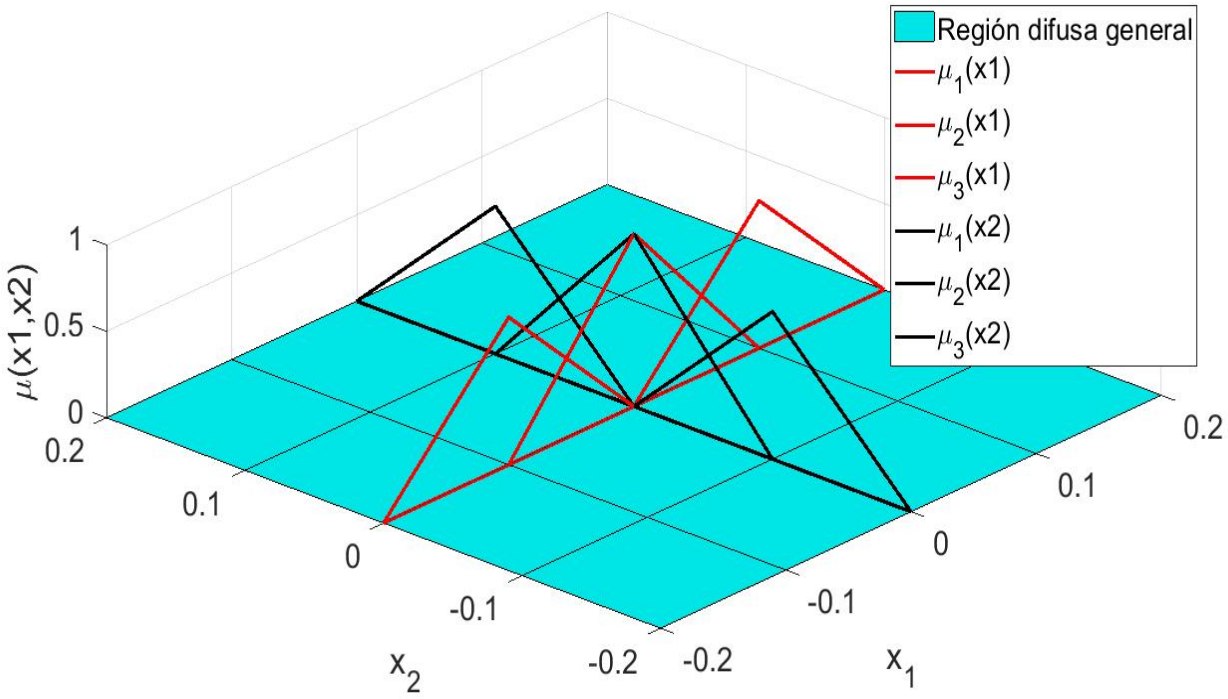


Figura 9.5: Funciones de membresía asociadas a las variables x_e^1, x_e^2

Con base en las funciones membresía ya definidas en las Figuras 9.3, 9.4 y en la ecuación (8.16), se obtienen los puntos de equilibro siguientes:

$$\begin{bmatrix} x_{eq}^1 \\ x_{eq}^2 \\ x_{eq}^3 \\ x_{eq}^4 \\ x_{eq}^5 \\ x_{eq}^6 \\ x_{eq}^7 \\ x_{eq}^8 \\ x_{eq}^9 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -0{,}0873 \\ 0 & 0{,}0873 \\ -0{,}0873 & 0 \\ -0{,}0873 & -0{,}0873 \\ -0{,}0873 & 0{,}0873 \\ 0{,}0873 & 0 \\ 0{,}0873 & -0{,}0873 \\ 0{,}0873 & 0{,}0873 \end{bmatrix} \tag{9.3}$$

Los puntos de equilibrio son sustituidos en la ecuación de estados lineal, ya que el modelo esta basado en el error de los estados se rescribe la ecuación lineal de la siguiente forma:

$$\begin{bmatrix} \dot{x}_e^1 \\ \dot{x}_e^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{r} cos(x_{1eq}^p) & 0 \end{bmatrix} \begin{bmatrix} x_e^1 \\ x_e^2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{mr^2} \end{bmatrix} u \qquad \forall\, p = 1, 2, 3, 4, \ldots, 9 \tag{9.4}$$

Sustituyendo las constantes $r = 0{,}525\ [m]$, $6{,}77\ [Kg]$, $g = 9{,}72\ \left[\frac{m}{s^2}\right]$ y los puntos de equilibrio x_{eq}^p, se obtienen las siguientes matrices A y B para el sistema basado en el error:

$$A_1 = \begin{bmatrix} 0 & 1 \\ 18{,}5143 & 0 \end{bmatrix} \quad con \quad A_2 = A_3 = A_1 \tag{9.5}$$

$$A_4 = \begin{bmatrix} 0 & 1 \\ 18{,}4438 & 0 \end{bmatrix} \quad con \quad A_5 = A_6 = A_7 = A_8 = A_9 = A_4 \tag{9.6}$$

$$B = \begin{bmatrix} 0 \\ 0{,}5359 \end{bmatrix} \tag{9.7}$$

Por lo tanto los nueve sistemas lineales en variables de estados se reducen a dos sistemas y se pueden expresar de la sigiente forma:

$$\dot{x}_e = A_1 x_e + Bu \qquad \dot{x}_e = A_4 x_e + Bu, \qquad con \quad x_e = r(t) - x \tag{9.8}$$

Análisis de estabilidad para sistemas discretos lineales.

Discretización de los sistemas continuos para el péndulo invertido.

$$A^1_{ed} = \begin{bmatrix} 1{,}094 & 0{,}1031 \\ 1{,}909 & 1{,}094 \end{bmatrix} \quad con \quad A^2_{ed} = A^3_{ed} = A^1_{ed} \tag{9.9}$$

$$A^4_{ed} = \begin{bmatrix} 1{,}094 & 0{,}1031 \\ 1{,}902 & 1{,}094 \end{bmatrix} \quad con \quad A^5_{ed} = A^6_{ed} = A^7_{ed} = A^8_{ed} = A^9_{ed} = A^4_{ed} \tag{9.10}$$

Cálculo de la estabilidad de los sistemas con la ecuación de Lyapunov (8.30):

$$P^1_{ed} = 1 \times 10^5 \begin{bmatrix} -0{,}2613 & 0 \\ 0 & 4{,}8387 \end{bmatrix} \quad con \quad P^2_{ed} = P^3_{ed} = P^1_{ed} \tag{9.11}$$

$$\lambda(A^1_{ed}) = \{1{,}5376, 0{,}6504\} \qquad con \qquad \lambda(A^2_{ed}) = \lambda(A^3_{ed}) = \lambda(A^1_{ed}) \tag{9.12}$$

$$P^4_{ed} = 1 \times 10^4 \begin{bmatrix} -0{,}0639 & -0{,}0003 \\ -0{,}0003 & 1{,}1797 \end{bmatrix} \quad con \quad P^5_{ed} = P^6_{ed} = P^7_{ed} = P^8_{ed} = P^9_{ed} = P^4_{ed} \tag{9.13}$$

$$\lambda(A^4_{ed}) = \{1{,}5368, 0{,}6512\} \qquad con \qquad \lambda(A^5_{ed}) = \lambda(A^6_{ed}) = \lambda(A^7_{ed}) = \lambda(A^8_{ed}) = \lambda(A^9_{ed}) = \lambda(A^1_{ed}) \tag{9.14}$$

Con los cálculos anteriores debido a que P^j_{ed} no son positivas definidas y además los valores característicos no cumplen con $\| \lambda(A^j_{ed}) \| < 0$, por lo que los sistemas no son estables.

Análisis de la controlabilidad para los sistemas lineales discretos

Con base al teorema (8.2) para la controlabilidad de sistemas discretos se obtienen las siguientes matrices de controlabilidad:

$$\mathcal{C}^j_e = \begin{bmatrix} B^j_{ed} & A^j_{ed} B^j_{ed} \end{bmatrix} = \begin{bmatrix} 0{,}0027 & 0{,}0087 \\ 0{,}0553 & 0{,}0657 \end{bmatrix}, \quad \forall \quad j = 1, 2, 3 \tag{9.15}$$

$$\mathcal{C}^j_e = \begin{bmatrix} B^j_{ed} & A^j_{ed} B^j_{ed} \end{bmatrix} = \begin{bmatrix} 0{,}0027 & 0{,}0086 \\ 0{,}0552 & 0{,}0651 \end{bmatrix}, \quad \forall \quad j = 4, \ldots, 9 \tag{9.16}$$

utilizando las matrices de controlabilidad (9.15) y (9.16) se cálcula el rango que es igual a:

$$\rho(\mathcal{C}^j_e) = 2 \quad \forall \quad j = 1, 2, \ldots, 9 \tag{9.17}$$

con lo cual se puede concluir que los sistemas lineales son controlables.

Cálculo de las ganancias para el controlador LQR basado en modelo.

Al igual que en capítulo 8 se utiliza la función **lqrd(A,B,Q,R,Ts)** en matlab para cálcular un controlador linel óptimo discreto (L.Q.R.D.) para cada sistema lineal definido en las regiones asociadas a las funciones membresía y el tiempo de muestreo de 0.1 [seg]. Las ganancias calculadas para los sistemas lineales (9.8) son:

$$K^1_d = \begin{bmatrix} 56{,}9585 & 13{,}3858 \end{bmatrix} \qquad K^2_d = \begin{bmatrix} 56{,}7602 & 13{,}3653 \end{bmatrix} \tag{9.18}$$

La ley decontrol basada en la inferencia de Takagi-Sugeno para el error de los estados está dada por:

$$f_1(x^1_{ek}, x^2_{ek}) = \begin{bmatrix} 56{,}9585 & 13{,}3858 \end{bmatrix} \begin{bmatrix} x^1_{ek} \\ \\ x^2_{ek} \end{bmatrix} \tag{9.19}$$

$$f_2(x^1_{ek}, x^2_{ek}) = \begin{bmatrix} 56{,}7602 & 13{,}3653 \end{bmatrix} \begin{bmatrix} x^1_{ek} \\ x^2_{ek} \end{bmatrix} \tag{9.20}$$

$$u_k = \frac{(w_1 + w_2 + w_3) f_1(x^1_{ek}, x^2_{ek}) + (w_4 + w_5 + w_6 + w_7 + w_8 + w_9) f_2(x^1_{ek}, x^2_{ek})}{w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9} \tag{9.21}$$

donde x^1_{ek} y x^2_{ek} son las variable de estados discretas basadas en el error , u_k es la entrada discreta y w_n son los niveles de activación los cuales se pueden calcular mediante:

$$w_p = \mu^p_r(x^1_{ek})\mu^p_r(x^2_{ek}) \ldots \mu^p_r(x^n_{ek}) \qquad \forall\ p = 1, 2, 3, \ldots \tag{9.22}$$

9.2. Simulaciones

Con base en los datos calculados para el diseño del controlador anteriormente y en el diagrama a bloques del control de la Figura 9.2 y suponiendo que el sensor es ideal. El diagrama de control se puede reducir como se muestra en la Figura 9.6.

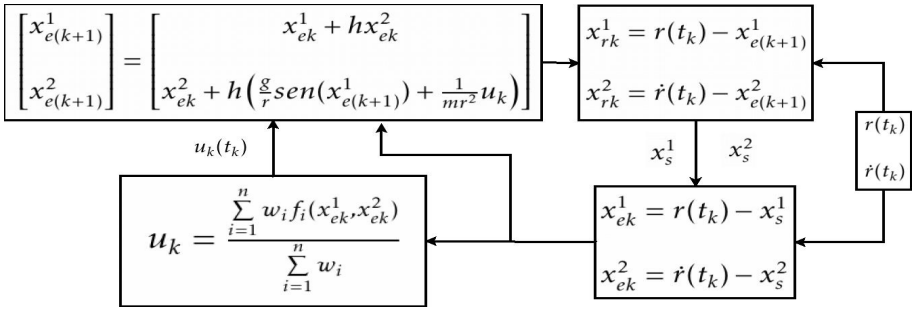


Figura 9.6: Diagrama de control difuso discreto con referencia basado en el error de los estados y sensor ideal

El diagrama de la Figura 9.6 esta programado en un script de **Matlab** (Programa 11.7) para realizar la simulación la cual arroja las siguientes gráficas mostradas en las Figuras 9.7, 9.8 y 9.9.

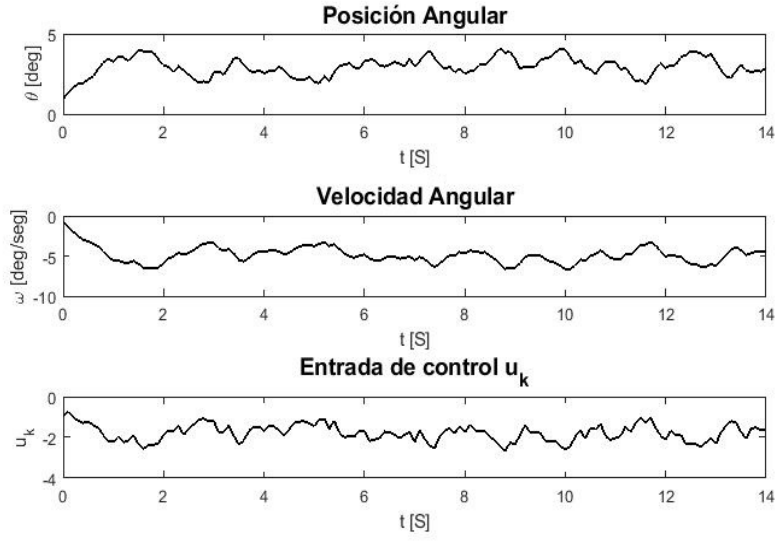


Figura 9.7: Gráfica del control difuso para el error con la aproximación númerica del péndulo invertido

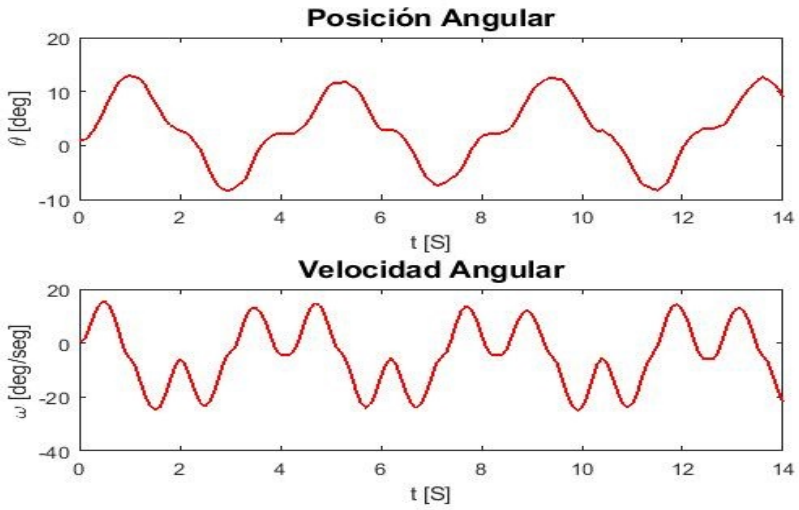


Figura 9.8: Gráfica del control difuso del seguimiento de las trayectorias

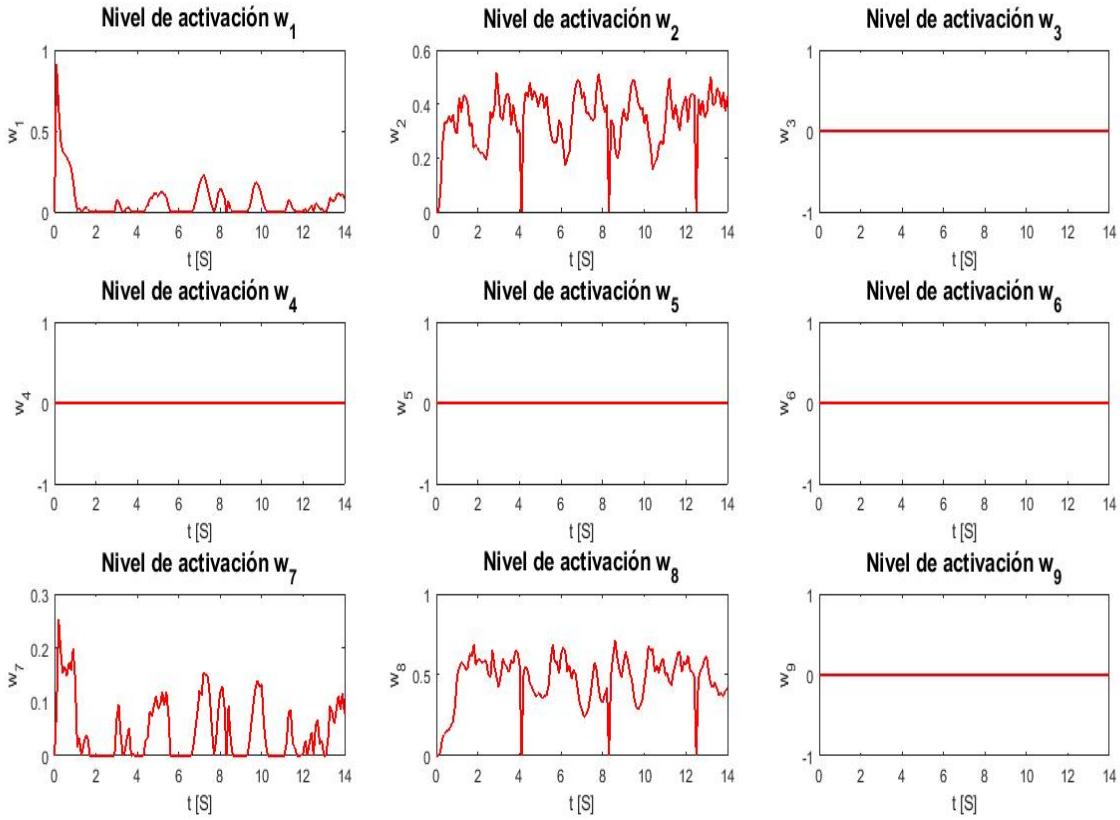


Figura 9.9: Gráficas de los niveles de activación para el control difuso basado en el error

Capítulo 10

Resultados y Conclusiones

10.1. Resultados

Desempeño del filtro Complementario

Las señales adquiridas de los sensores contienen ruido, para quitarlo se necesitan de filtros en el **Capítulo 5** ya se han calculado los filtros. En el caso para la señal digital de los ángulos medidos ($\theta_{x,y}$) se aplica un filtro complementario especificamente para este caso esta dado por la ecuación:

$$\theta_{x,\,y\,i} = 0{,}8(\theta_{x,\,y\,i-1} + \omega_{gyro_{x,\,y}\,i}\,\triangle\, t) + 0{,}2\theta_{accel_{x,\,y}\,i}$$

En la gráfica de la Figura 10.1 se muestran los datos no filtrados (Azul) contra los filtrados (Rojo) con el objetivo de mostrar el funcionamiento del filtro programado en ROS.

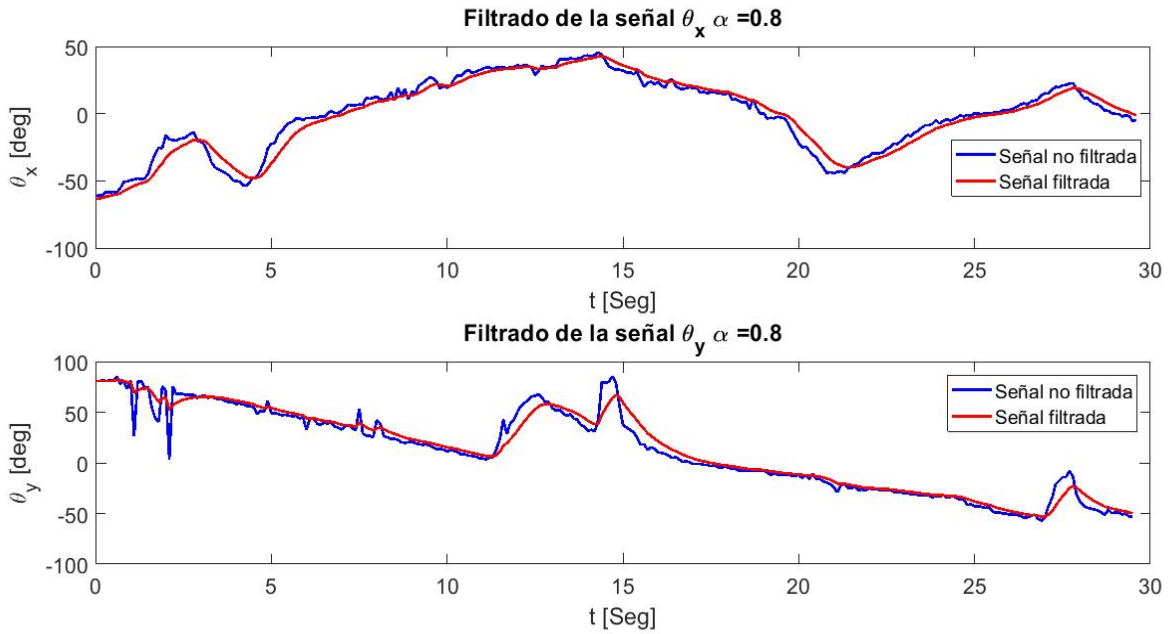


Figura 10.1: Desempeño del filtro complementario para los ángulos θ_x y θ_y

Desempeño del filtro Butterworth

El filtro Butterworth digital esta programado en un nodo de ROS junto con el filtro complementario para reducir el ruido de la señal digital del giroscopio como se mostró en el **Capítulo 5**, el cual tiene una ecuación en diferencias finitas recursiva:

$$y_n = \frac{1}{a_1}\left(b_1 x_n + b_2 x_{n-1} + b_3 x_{n-2} - a_2 y_{n-1} - a_3 y_{n-2}\right)$$

$$\begin{aligned} y_{n-2} &= y_{n-1}, \qquad y_{n-1} = y_n \\ x_{n-2} &= x_{n-1}, \qquad x_{n-1} = x_n \end{aligned} \tag{10.1}$$

Con coeficientes:

$$a = [1 \quad -1{,}561 \quad 0{,}6414] \qquad b = [0{,}0201 \quad 0{,}0402 \quad 0{,}0201]$$

En la Figura 10.2 se muestra el comportamiento del filtro, la señal del giroscopio (Rojo) es comparada contra la señal filtrada (Negra) esto se hace con el fin de mostrar el funcionamiento del filtro.

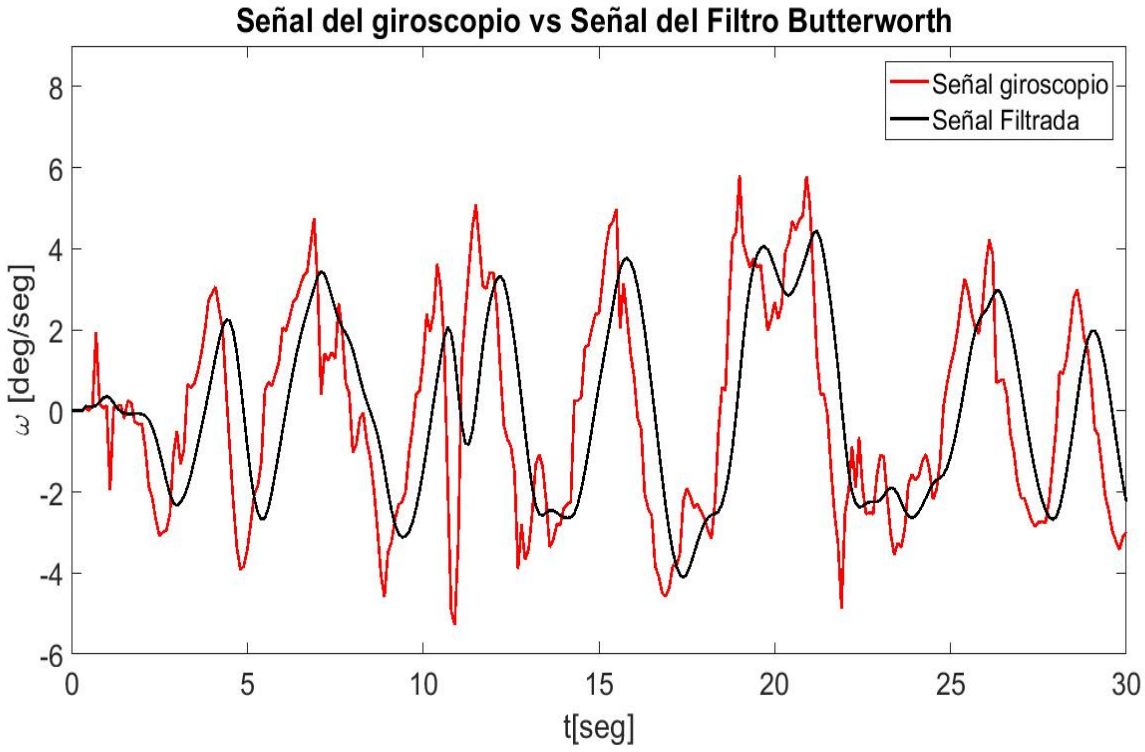


Figura 10.2: Desempeño del filtro Butterworth para el giroscopio (velocidad ángular ω)

Datos obtenidos de ROS para mostrar el comportamiento del control difuso de postura contra las simulaciones.

En esta sección se presentan los resultados del comportamiento del control difuso ya aplicado al robot humanoide, además estos resultados se comparan con los datos obtenidos de las simulación realizada en en **Capítulo 8** y de esta forma comparar la parte teórica con la aplicada. Los datos del control difuso teórico y aplicado se muestran en las gráficas de las Figuras 10.3, 10.4.

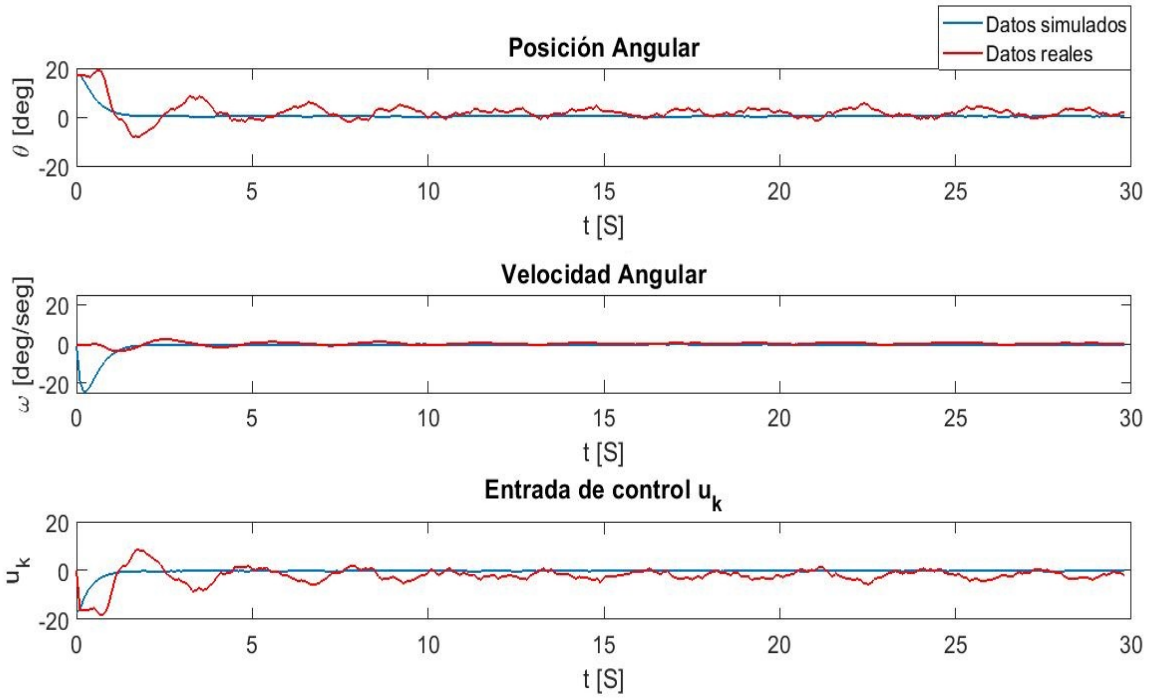


Figura 10.3: Gráficas del control difuso para el robot humanoide

Los niveles de activación (Figura 10.4) en pocas palabras son los que toman las deciciones basadas en la lógica difusa para saber que región de control difusa utilizar para llevar al robot humanoide a una posición estable deseada.

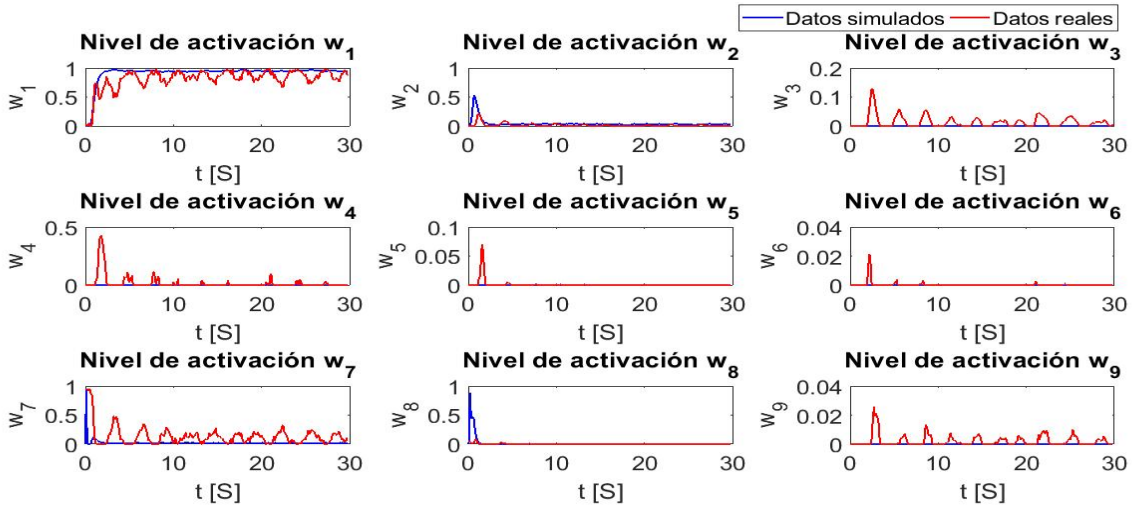


Figura 10.4: Gráficas de los niveles de activación w_n para el control difuso

Con base en los datos mostrados en las gráficas de la Figura 10.3, se obtiene la estabilidad de los datos reales los cuales se muestran en la Figura 10.5 dónde se grafican estos datos contra los simulados que las variables de estados adquieren con la aplicación del control difuso.

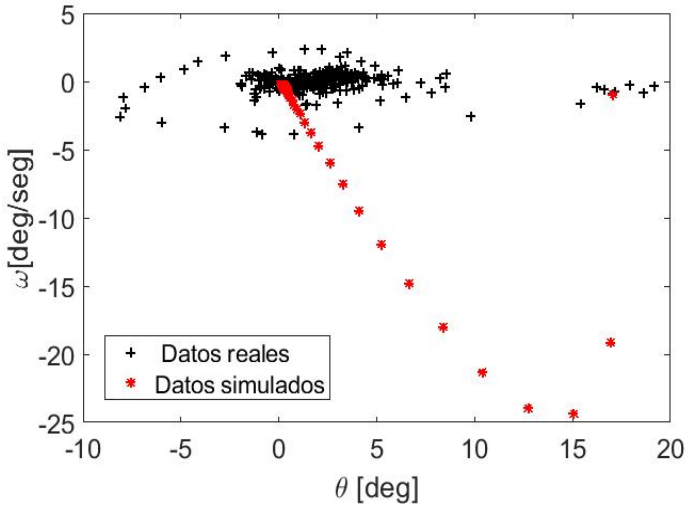


Figura 10.5: Gráfica de $\theta[deg]$ *vs* $\omega[deg/seg]$

En las gráficas que se muestran en la Figura 10.6 son los ángulos obtenidos de la aplicación de la cinemática inversa que el tobillo y la cadera del robot humanoide deben adquirir para ser estables.

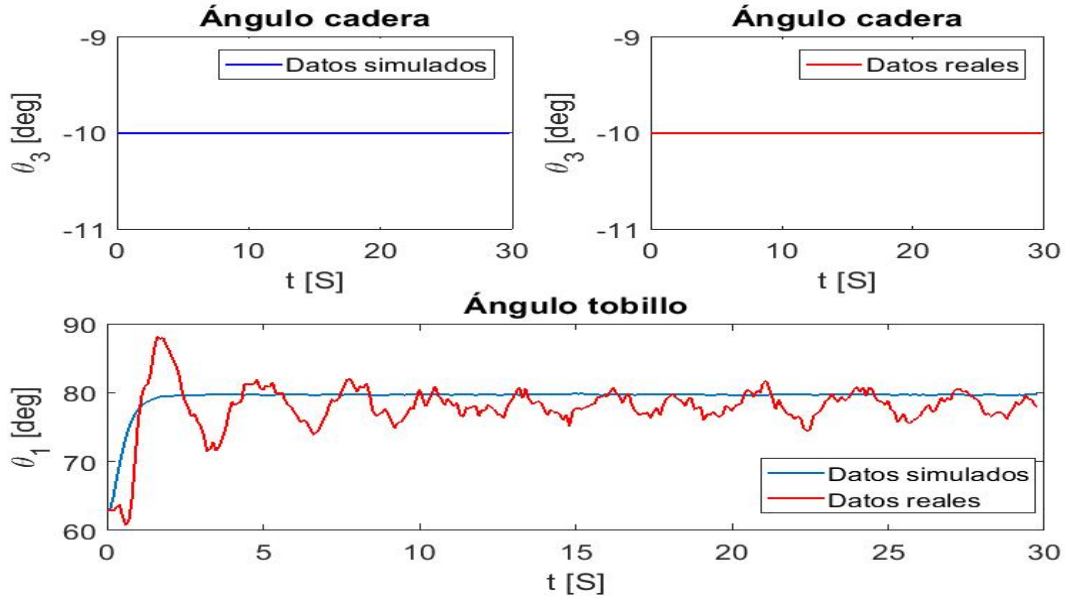


Figura 10.6: Gráfica de los movimientos teóricos y reales para los servomotores del tobillo y cadera

Programa del control difuso en ROS

El código del Programa 11.8 es un nodo suscriptor-publicador el cual es utilizado para realizar el control difuso de la postura, además se utiliza el programa 11.9 para calcular las funciones de membesía.

Datos obtenidos de ROS para mostrar el funcionamiento del control difuso para el seguimeinto de trayectorias contra las simulaciones.

Se muestran las gráficas en las Figuras 10.7 y 10.8 el comportamiento del control difuso basado en el error de los estados del péndulo invertido contra las simulación realizada en en **Capítulo 9** y de esta forma poder observar el desempeño del control difuso teórico y aplicado al robot humanoide.

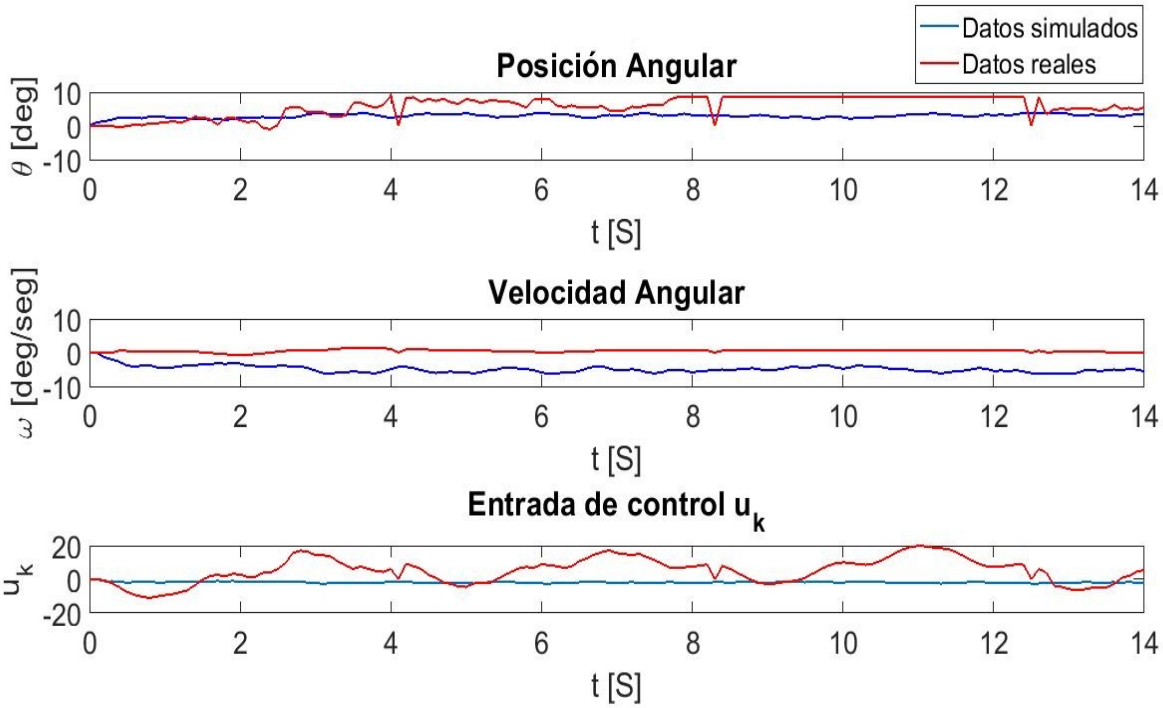


Figura 10.7: Gráfica del control difuso para el error con la aproximación númerica del péndulo invertido

Al igual que en el control difuso de postura los niveles de activación son los que determinan automáticamete las regiones de control.

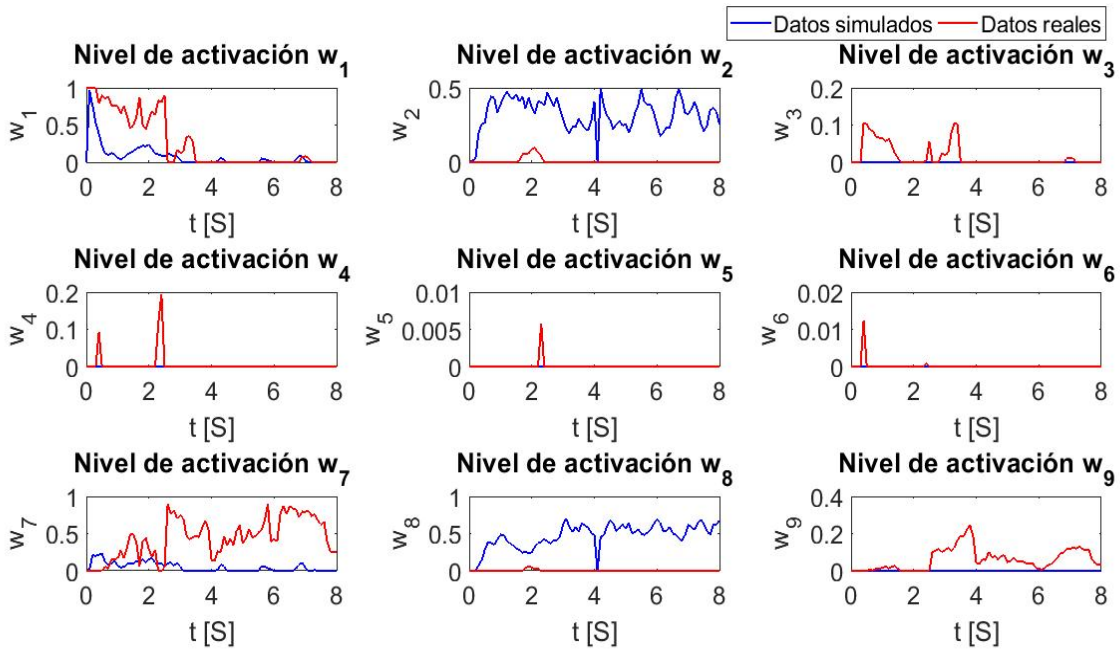


Figura 10.8: Gráficas de los niveles de activación para el control difuso basado en el error

En la gráfica de la Figura 10.9 se muestran las trayectorias simuladas de los estados del péndulo invertido que el robot humanoide debe seguir y las experimentales que al igual están basadas en los estados del péndulo invertido que el robot humanoide ha realizado.

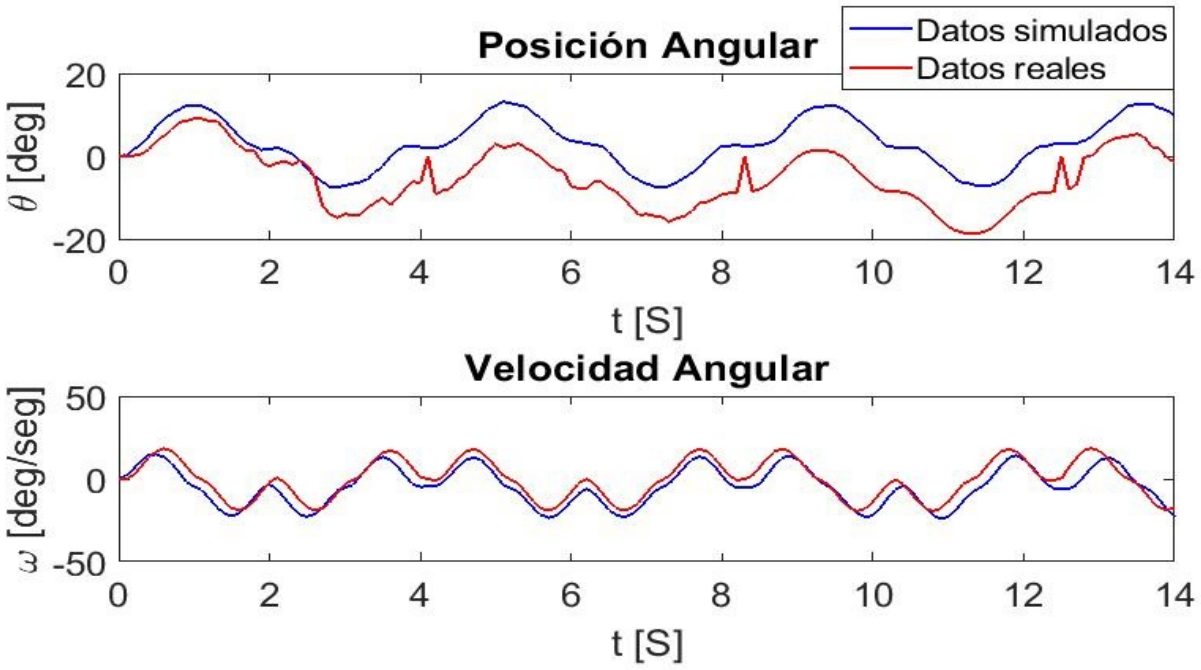


Figura 10.9: Gráfica del control difuso del seguimiento de las trayectorias

Con base a los datos teóricos y experimentales mostrados en las gráficas de la Figura 10.10 se realizó par mostrar la estabilidad del sistma bajo el control difuso téorico y experimental.

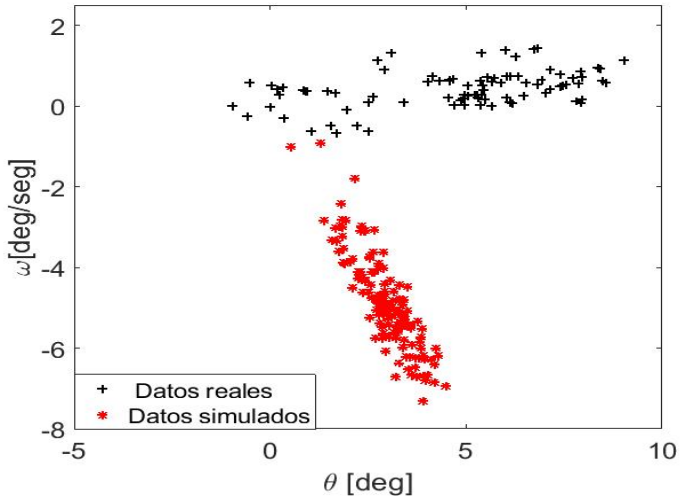


Figura 10.10: Gráfica de $\theta[deg]$ *vs* $\omega[deg/seg]$

En las gráficas de la Figura 10.11 se muestran las trayectorias que los ángulos del tobillo y la cadera de la pierna fija con base en la aplicación de la cinemática inversa y el cotrol difuso para el robot humanoide.

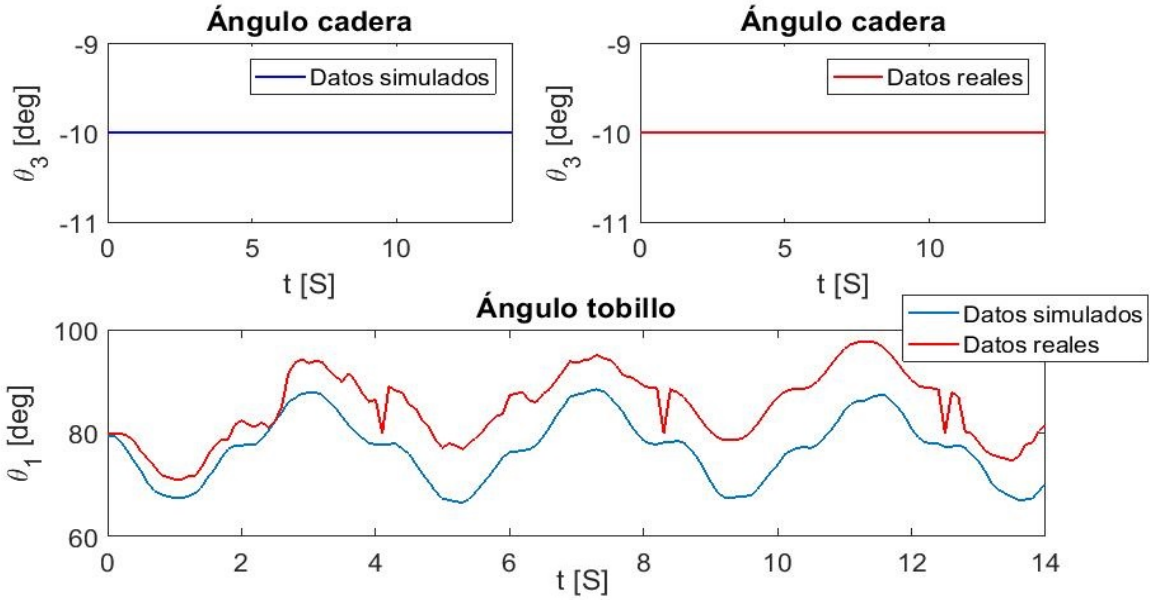


Figura 10.11: Gráfica de los movimientos teóricos y reales para los servomotores del tobillo y cadera

Programa del control difuso en ROS

El código del Programa 11.10 es un nodo suscriptor-publicador para realizar el control difuso de seguimiento de trayectorias para el péndulo invertido y generar las inclinaciones del robot para la caminata bípeda. Al igual que el control difuso de postura se utiliza el programa 11.9 para las funciones de membesía.

10.2. Conclusiones

Las señales digitales que se obtienen del sensor MPU-6050 contienen ruido por lo que se realizaron filtros digitales recursivos basados en el filtro complementario y el filtro Butterword. Estos filtros son utilizados para que la serie de datos obtenida tengan como caraterística principal la disminución de valores aleatorios y estos sean estables, con lo cual los datos de la posición y velocidad angular tienen menor variación ante un cambio pequeño. El filtro complementario solo depende de un parámetro $0 < \alpha < 1$ el cual se obtiene experimentalmente variando α y observando el desempeño del filtro, en este trabajo $\alpha = 0{,}8$ debido a las observaciones experimentales presentó resultados con menor variación con respecto a la señal no filtrada, y por lo contrario para filtro Butterworth se tienen que calcular los coeficientes de la ecuación en diferencias finitas con base en el tiempo de muestreo (0.1[seg]). La aplicación de estos filtros se muestran en las gráficas de las Figuras 10.1, 10.2 con lo cual se concluye que los filtros cumplen con lo antes mencionado.

Con base en los polinomios de 5^{to} orden y las ecuaciones generadas para calcular los coeficientes de pendiendo de los puntos y los tiempos que se desean alcanzar se obtienen trayectorias suaves obligando a que las velocidades y aceleraciones sean cero al inicio y final de estas con lo cual el robot humanoide no tenga movimientos bruscos, tanto para el control difuso de seguimiento de trayectorias como para el movimiento de la segunda pierna con lo cual se podrá generar la caminata bípeda.

Mediante el análisis de la cinemática inversa se obtuvieron las ecuaciones basadas en un robot planar de dos grados de libertad, estos cálculos son importantes ya que sirve para poder mover las piernas del robot humanoide y con base en los resultados del control difuso de la postura y la caminata ya sean teóricos y experimentales se puede mostrar que los movimientos son los esperados.

Los controladores difusos basados en la inferencia difusa de Tagaki-Sugeno permiten combinar controladores lineales con la lógica difusa haciendo que el control sea robusto, con los datos obtenidos en las simulaciones y los reales obtenidos en los experimentos del control difuso de postura se puede mostrar que el enfoque de controlar un péndulo invertido ayuda a que el robot humanoide tenga estabilidad.

Con base en las simulaciones y experimentos realizados para el generador de las trayectorias de la pierna movil y el seguimiento del trayectorias del péndulo invertido, las trayectorias reales que el robot humanoide adquiere estan desfasadas al rededor de 5 grados esto es debido a las vibraciones de los servomoteres ópticos las cuales se suman a las mediciones del sensor **MPU-6050** por lo cual el robot humanoide tarda 4 segundos para completar las trayectorias que generan los pasos.

Los objetivos específicos han sido logrados esto implica que el objetivo general también tendría que ser alcanzado pero la caminata bípeda del robot humanoide se complica debido a factores como el tamaño, peso de la estructura de las piernas porque pueden doblarse cuando el robot se inclina haciendo que se tropiece y no realice correctamente la caminata bípeda y por último la adherecia de la planta de los pies con el suelo ya que si esta no existe el robot humanoide se desliza y no podrá avanzar como se esperaba.

Es evidente siempre se debe tender a mejorar el trabajo. Por lo que se tiene en primer lugar garantizar que el sensor IMU y con ayuda de filtros los datos sean confiables (que no presenten ruido), los sensores MPU-9150 [23] y Sparkfun Razor IMU 9DOF [24] especialmente con este sensor, ROS ya tiene un paquete para adquirir los datos del acelerómetro y giroscopio. El enfoque para generar el control se propone utilizar un péndulo invertido con radio variable y la utilización del ***Zero Moment Point*** (**ZMP**) [3]. El **ZMP** es un punto de contacto con lo cual los momentos producidos por la gravedad equilibran los momentos debidos con el contacto con el suelo del Robot Humanoide [25]. El movimiento de la pierna que se genera mediante la cinemática inversa ahora como el radio del péndulo es variable la rodilla también va tener que moverse por lo cual podemos conciderar un modelo de cinemática inversa para un robot planar de tres grados de libertad, debido a que los cálculos aumentan el número de ecuaciones no lineales se puede obtener mediante aproximaciones númericas iterativas [26].

Capítulo 11

Programas

11.1. Comunicación serial y sensor MPU6050 en ROS

Comunicación serial

Nodo suscriptor "robotbb_serialnewrv" en Python.

```
import math
import serial
import RPi.GPIO as GPIO
import os, time
import time
import rospy
from std_msgs.msg import Float32MultiArray
msg=Float32MultiArray
#comunicacion serial
port=serial.Serial("/dev/ttyAMA0",baudrate=38400)
#Pierba Izquierda y derecha
PID=[0,0,0,0,0,0,0,0,0,0,0,0]
#Brazo Izquierdo
BI= [0,0,0,0,0,0,0]
#Brazo Derecho
BD=[0,0,0,0,0,0,0]
#Movimientos de la Camara
C =[0,0,0,0,0,0,0]
    #--------------------
     #Pierna Izquierda
    PID2=PID[2]-90 #por la referencia  de los angulos
    PID3=PID[3]-90
    #cadera
    S0=int(0.00036*PID[0]*PID[0]*PID[0]-14.07*PID[0]+1500)
    #Rodilla
    S1=int(-0.00036*PID[1]*PID[1]*PID[1]+14.07*PID[1]+1500)
    #tobillo
    S2=int(0.00036*PID2*PID2*PID2-14.07*PID2+1500)
    #tobillo
    S3=int(0.00036*PID3*PID3*PID3-14.07*PID3+1500)
    #cadera
    S4=int(0.00036*PID[4]*PID[4]*PID[4]-14.07*PID[4]+1500)
    #cadera
    S5=int(0.00036*PID[5]*PID[5]*PID[5]-14.07*PID[5]+1500)
    #Pierna derecha
    PID8=PID[8]-90 #por la referencia  de los angulos
    PID9=PID[9]-90
    S16=int(-0.00036*PID[6]*PID[6]*PID[6]+14.07*PID[6]+1500)
    S17=int(0.00036*PID[7]*PID[7]*PID[7]-14.07*PID[7]+1500)
    S18=int(-0.00036*PID8*PID8*PID8+14.07*PID8+1500)
    S19=int(-0.00036*PID9*PID9*PID9+14.07*PID9+1500)
    S20=int(-0.00036*PID[10]*PID[10]*PID[10]+14.07*PID[10]+1500)
    S21=int(-0.00036*PID[11]*PID[11]*PID[11]+14.07*PID[11]+1500)
    #Brazo izquierdo
    S6=int(0.0003657*BI[0]*BI[0]*BI[0]-14.074*BI[0]+1500)
    S7=int(0.0003657*BI[1]*BI[1]*BI[1]-14.074*BI[1]+1500)
    S8=int(0.0003657*BI[2]*BI[2]*BI[2]-14.074*BI[2]+1500)
    S9=int(0.0003657*BI[3]*BI[3]*BI[3]-14.074*BI[3]+1500)
    #Brazo derecha
```

```
    S22=int(-0.0003657*BD[0]*BD[0]*BD[0]+14.074*BD[0]+1500)
    S23=int(-0.0003657*BD[1]*BD[1]*BD[1]+14.074*BD[1]+1500)
    S24=int(-0.0003657*BD[2]*BD[2]*BD[2]+14.074*BD[2]+1500)
    S25=int(-0.0003657*BD[3]*BD[3]*BD[3]+14.074*BD[3]+1500)
    #Cabeza (camara)
    C0=C[0]-90;
    S30=int(-0.0003657*C0*C0*C0+14.074*C0+1500)
    S31=int(-0.0003657*C[1]*C[1]*C[1]+14.074*C[1]+1500)
    #Comunicacion serial Movimiento de los servomotores
    #Pierba Izquierda y derecha
    port.write("#0P%s#1P%s#2P%s#3P%s#4P%s#5P%s#16P%s#17P%s#18P%s#19P%s#20P%s#21P%sT1000\
    r"%(str(S0),str(S1),str(S2),str(S3),str(S4),str(S5),str(S16),str(S17),str(S18),str(S19),str(S20)
    ,str(S21))) # PID
    #Brazo Izquierdo
    port.write("#6P%s#7P%s#8P%s#9P%sT1000\r"%(str(S6),str(S7),str(S8),str(S9))) #BI
    #Brazo Derecho
    port.write("#22P%s#23P%s#24P%s#25P%sT1000\r"%(str(S22),str(S23),str(S24),str(S25))) #BD
    #MOvimientos Camara
    port.write("#30P%s#31P%sT1000\r"%(str(S30),str(S31))) #C

#_________________fin___funcion_______

#____Funcion__subscriber_______
def listener():
    # Nodo
    rospy.init_node('robotbb_serialnewr')
    # Topico
    rospy.Subscriber('/ssc/32',Float32MultiArray,callback)
    rospy.spin()

#__________Fin__subscriber______________

#__int__main
if __name__ == '__main__':
     listener()
```

Programa 11.1: robotbb_ serialnewrv.py

Sensor MPU6050

Nodo publicador "mpu6050_ sensor_ node" en Python.
Programa de comunicación I^2C obtenido de [17].

```
import rospy
from std_msgs.msg import Float32MultiArray
import smbus
from time import sleep
import math

#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47
# Constantes globales para el Filtro Complementario
alphax=0.8
alphay=0.8
#---------------
def MPU_Init():
  #write to sample rate register
  bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)

  #Write to power management register
  bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)

  #Write to Configuration register
```

```
  bus.write_byte_data(Device_Address, CONFIG, 0)

  #Write to Gyro configuration register
  bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)

  #Write to interrupt enable register
  bus.write_byte_data(Device_Address, INT_ENABLE, 1)

def read_raw_data(addr):
  #Accelero and Gyro value are 16-bit
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)

        #concatenate higher and lower value
        value = ((high << 8) | low)

        #to get signed value from mpu6050
        if(value > 32768):
                value = value - 65536
        return value


bus = smbus.SMBus(1)  # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68   # MPU6050 device address

def talker():
    pub = rospy.Publisher('/sensor', Float32MultiArray, queue_size=1)
    rospy.init_node('mpu6050_sensor_node', anonymous=True)
    hz=10
    Ts=1/hz
    rate = rospy.Rate(hz) # 10hz
    MPU_Init()
    Angxp=0
    Angyp=0
    xk=0
    xkm1=0
    xkm2=0
    yk=0
    ykm1=0
    ykm2=0

    print (" Reading Data of Gyroscope and Accelerometer")
    while not rospy.is_shutdown():
       # hello_str = "hello world %s" % rospy.get_time()
       # rospy.loginfo(hello_str)

        #Read Accelerometer raw value
        acc_x = read_raw_data(ACCEL_XOUT_H)
        acc_y = read_raw_data(ACCEL_YOUT_H)
        acc_z = read_raw_data(ACCEL_ZOUT_H)
        #Read Gyroscope raw value
        gyro_x = read_raw_data(GYRO_XOUT_H)
        gyro_y = read_raw_data(GYRO_YOUT_H)
        gyro_z = read_raw_data(GYRO_ZOUT_H)

        #Full scale range +/- 250 degree/C as per sensitivity scale factor
        Ax = acc_x/16384.0
        Ay = acc_y/16384.0
        Az = acc_z/16384.0

        Gx = gyro_x/131.0
        Gy = gyro_y/131.0
        Gz = gyro_z/131.0
#       print ("Gx=%.2f" %Gx, u'\u00b0'+ "/s", "\tGy=%.2f" %Gy, u'\u00b0'+ "/s", "\tGz=%.2f" %Gz, u
    '\u00b0'+ "/s", "\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)

        #angulo
        Accx=math.atan2(Ay,math.sqrt((Ax*Ax)+(Az*Az)))*(180.0/3.14)
        Accy=-math.atan2(Ax,math.sqrt((Ay*Ay)+(Az*Az)))*(180.0/3.14)
        #filtro complementario
        Angx=alphax*(Angxp+Gx*Ts)+(1-alphax)*Accx
        Angy=alphay*(Angyp+Gy*Ts)+(1-alphay)*Accy
        Angxp=Angx
        Angyp=Angy
        #filtro digital  gyroscopio
        yk=0.0201*Gy+0.0402*xkm1+0.0201*xkm2+1.56*ykm1-0.6414*ykm2
```

```
106         print("giroscopio(deg/s)=%.2f " %yk, "\tAng_y(deg)=%.2f" %Angy, "\tgiroscopio(rad/s)=%.3f"
    %(yk*(3.14/180)), "\tAngy(rad)=%.3f" %(Angy*(3.14/180)) )
107         #publica valores
108         pub.publish(arreglo)
109         rate.sleep()
110
111 if __name__ == '__main__':
112     try:
113         talker()
114     except rospy.ROSInterruptException:
115         pass
```

Programa 11.2: robotbb_ mpu6050_ sensor_ node.py

11.2. Cálculo de la cinemática inversa y trayectorias

Cálculo de la cinemática inversa en ROS

Nodo publicador " ci_ robot_ pf " en C++.

```
1  //ProgramaciC3n de la Cinematica inversa del RP2GL para
2  //Las piernas del Robot BigBrother
3  //este programa sirve para verificar que los puntos de las trayectorias
4  //sean los correctos y poder corregirlos
5  // ademas tambien funcina para comunicarse con las salida del modelo del
6  // pendulo invertido
7  #include "ros/ros.h"
8  #include "std_msgs/Float32MultiArray.h"
9  #include "std_msgs/Float32.h"
10 #include <iostream>
11 #include <cmath>
12 #include <ctime>
13 using namespace std;
14 //muestreo
15 float cont=0;
16 float hz=10;
17 float Ts=1/hz;
18 //parametros de la CI
19 //medidas de las distancias de los eslabones en metros
20 float l1=0.175, l2=0.175, l3=0.175; //[m]
21 float dr=0.525;
22 float rcf=0.344683;//radio constante de la pierna fija
23 //angulos constantes
24 float pi=3.14;
25 float th2=0.349067;
26 float alpha_c=0.174533;
27 float aC=2.79253;
28 float theta_offset=0.174533;
29 float vc=10;
30 float theta_p=0;
31 //angulos varibles rad
32 float theta_2rcf=0,theta_1rcf=1.5708;
33 float theta_1=1.39626,theta_3=-0.174533;
34 //Medidas lineales variables
35 float x=0, y=0.525,M_rcf=0;
36 //__funcion para obtener los datos del angulo para el pendulo invertido________
37 void obtencionangulo(const std_msgs::Float32::ConstPtr& angulo)
38 {
39  theta_p=angulo->data;
40  //x2=msgs->data[1];
41 }
42
43 int main(int argc, char **argv)
44 {
45   ros::init(argc, argv,"ci_robot_pf");//inicializacion del nodo publicador
46   ros::NodeHandle n;
47   //nodo suscriptor
48   ros::Subscriber subS=n.subscribe("/Estado/x1",1,obtencionangulo);
49   std_msgs::Float32MultiArray Vector;
50   Vector.data.resize(13);
51   ros::Publisher pubvec=n.advertise<std_msgs::Float32MultiArray>("/ssc/32",1);
52   ros::Rate loop(hz);
53   while(ros::ok())
```

```
  {
     //posicion del pendulo invertido
     x=dr*sin(theta_p);
     y=dr*cos(theta_p);
     std::cout<<"x="<<x<<std::endl;
     std::cout<<"y="<<y<<std::endl;
     // cinematica inversa para el robot humanoide
     M_rcf=((x*x+y*y-(rcf*rcf+l3*l3))/(2*rcf*l3));
     if(M_rcf>1)
     {
          std::cout<<"M_rcf="<<M_rcf<<std::endl;
          M_rcf=1;
     }
     else
     {
        M_rcf;
        std::cout<<"M_rcf="<<M_rcf<<std::endl;
     }
     cout<<"M_rcf="<<M_rcf<<endl;
     cout<<"theta_2rcf="<<theta_2rcf<<" theta_2rcf="<<theta_2rcf*(180/pi)<<endl;
     cout<<"theta_1rcf="<<theta_1rcf<<" theta_1rcf="<<theta_1rcf*(180/pi)<<endl;
     cout<<"theta_1="<<theta_1<<" theta_1="<<theta_1*(180/pi)<<endl;
     cout<<"theta_3="<<theta_3<<" theta_3="<<theta_3*(180/pi)<<endl;
     theta_2rcf=atan2(-sqrt(1-M_rcf*M_rcf),M_rcf);
     theta_1rcf=atan2(y,x)-atan2(l3*sin(theta_2rcf),rcf+l3*cos(theta_2rcf));
     theta_1=theta_1rcf-alpha_c;
     theta_3=theta_2rcf-theta_offset;
     //vector para publicar los movimientos del Robot Humanoide
     pubvec.publish(Vector);
     ros::spinOnce();
     loop.sleep();
  }
  return 0;
}
```

Programa 11.3: robotbb_cirhbb_node.cpp

Cálculo de las trayectorias

Programa en Matlab para el cálculo de los coeficientes de los polinomios de las trayectorias.

```
%% Calculo de coeficientes para el polinomio
qi1=0*(pi/180);%%para x
qf1=10*(pi/180);
ti=0;
tf=1;
t=ti:0.1:tf;
a10=(qi1*tf^3*(tf^2 - 5*tf*ti + 10*ti^2))/(tf - ti)^5 - (qf1*ti^3*(10*tf^2 - 5*tf*ti + ti^2))/(tf -
    ti)^5;
a11=(30*tf^2*ti^2*(qf1 - qi1))/(tf - ti)^5;
a12=-(30*tf*ti*(tf + ti)*(qf1 - qi1))/(tf - ti)^5;
a13=(10*(qf1 - qi1)*(tf^2 + 4*tf*ti + ti^2))/(tf - ti)^5;
a14=-(15*(tf + ti)*(qf1 - qi1))/(tf - ti)^5;
a15=(6*(qf1 - qi1))/(tf - ti)^5;
%% polinomio 1
qd1=a10+a11*t+a12*t.^2+a13*t.^3+a14*t.^4+a15*t.^5;
dqd1=a11+2*a12*t+3*a13*t.^2+4*a14*t.^3+5*a15*t.^4;
ddqd1=2*a12+6*a13*t+12*a14*t.^2+20*a15*t.^3;
%% Calculo de coeficientes para el polinomio
ti2=1;
tf2=2;
t2=ti2:0.1:tf2;
qi2=10*(pi/180);
qf2=0*(pi/180);
a20=(qi2*tf2^3*(tf2^2 - 5*tf2*ti2 + 10*ti2^2))/(tf2 - ti2)^5 - (qf2*ti2^3*(10*tf2^2 - 5*tf2*ti2 +
    ti2^2))/(tf2 - ti2)^5;
a21=(30*tf2^2*ti2^2*(qf2 - qi2))/(tf2 - ti2)^5;
a22=-(30*tf2*ti2*(tf2 + ti2)*(qf2 - qi2))/(tf2 - ti2)^5;
a23=(10*(qf2 - qi2)*(tf2^2 + 4*tf2*ti2 + ti2^2))/(tf2 - ti2)^5;
a24=-(15*(tf2 + ti2)*(qf2 - qi2))/(tf2 - ti2)^5;
a25=(6*(qf2 - qi2))/(tf2 - ti2)^5;
%% polinomio 2
qd2=a20+a21*t2+a22*t2.^2+a23*t2.^3+a24*t2.^4+a25*t2.^5;
```

```
dqd2=a21+2*a22*t2+3*a23*t2.^2+4*a24*t2.^3+5*a25*t2.^4;
ddqd2=2*a22+6*a23*t2+12*a24*t2.^2+20*a25*t2.^3;
%% Calculo de coeficientes para el polinomio 3
qi3=0*(pi/180);%%para x
qf3=10*(pi/180);
ti3=2;
tf3=3;
te2=ti3:0.1:tf3;
a30=(qi3*tf3^3*(tf3^2 - 5*tf3*ti3 + 10*ti3^2))/(tf3 - ti3)^5 - (qf3*ti3^3*(10*tf3^2 - 5*tf3*ti3 +
    ti3^2))/(tf3 - ti3)^5;
a31=(30*tf3^2*ti3^2*(qf3 - qi3))/(tf3 - ti3)^5;
a32=-(30*tf3*ti3*(tf3 + ti3)*(qf3 - qi3))/(tf3 - ti3)^5;
a33=(10*(qf3 - qi3)*(tf3^2 + 4*tf3*ti3 + ti3^2))/(tf3 - ti3)^5;
a34=-(15*(tf3 + ti3)*(qf3 - qi3))/(tf3 - ti3)^5;
a35=(6*(qf3 - qi3))/(tf3 - ti3)^5;
%% polinomio 3
qd3=a30+a31*te2+a32*te2.^2+a33*te2.^3+a34*te2.^4+a35*te2.^5;
dqd3=a31+2*a32*te2+3*a33*te2.^2+4*a34*te2.^3+5*a35*te2.^4;
ddqd3=2*a32+6*a33*te2+12*a34*te2.^2+20*a35*te2.^3;
%% Calculo de coeficientes para el polinomio 4
qi4=10*(pi/180);%%para x
qf4=0*(pi/180);
ti4=3;
tf4=4;
te4=ti4:0.1:tf4;
a40=(qi4*tf4^3*(tf4^2 - 5*tf4*ti4 + 10*ti4^2))/(tf4 - ti4)^5 - (qf4*ti4^3*(10*tf4^2 - 5*tf4*ti4 +
    ti4^2))/(tf4 - ti4)^5;
a41=(30*tf4^2*ti4^2*(qf4 - qi4))/(tf4 - ti4)^5;
a42=-(30*tf4*ti4*(tf4 + ti4)*(qf4 - qi4))/(tf4 - ti4)^5;
a43=(10*(qf4 - qi4)*(tf4^2 + 4*tf4*ti4 + ti4^2))/(tf4 - ti4)^5;
a44=-(15*(tf4 + ti4)*(qf4 - qi4))/(tf4 - ti4)^5;
a45=(6*(qf4 - qi4))/(tf4 - ti4)^5;
%% polinomio 4
qd4=a40+a41*te4+a42*te4.^2+a43*te4.^3+a44*te4.^4+a45*te4.^5;
dqd4=a41+2*a42*te4+3*a43*te4.^2+4*a44*te4.^3+5*a45*te4.^4;
ddqd4=2*a42+6*a43*te4+12*a44*te4.^2+20*a45*te4.^3;

figure(1)
subplot(3,1,1);
plot(t,qd1,t2,qd2,'linewidth',3)
title('Trayectoria del Polinomio de 5^{to} orden')
ylabel('P(t)[deg]')
xlabel('t [s]')

subplot(3,1,2);
plot(t,dqd1,t2,dqd2,'linewidth',3)
title('Velocidad de la trayectoria del Polinomio de 5^{to} orden')
ylabel('DP(t)[deg/seg]')
xlabel('t [s]')

subplot(3,1,3);
plot(t,ddqd1,t2,ddqd2,'linewidth',3)
title('Aceleracion de la trayectoria del Polinomio de 5^{to} orden')
ylabel('D^{2}P(t)[deg/seg^{2}]')
xlabel('t [s]')

figure(2)
subplot(3,1,1);
plot(te2,-qd3,te4,-qd4,'linewidth',3)
title('Trayectoria del Polinomio de 5^{to} orden')
ylabel('P(t)[deg]')
xlabel('t [s]')

subplot(3,1,2);
plot(te2,-dqd3,te4,-dqd4,'linewidth',3)
title('Velocidad de la trayectoria del Polinomio de 5^{to} orden')
ylabel('DP(t)[deg/seg]')
xlabel('t [s]')

subplot(3,1,3);
plot(te2,-ddqd3,te4,-ddqd4,'linewidth',3)
title('Aceleracion de la trayectoria del Polinomio de 5^{to} orden')
ylabel('D^{2}P(t)[deg/seg^{2}]')
xlabel('t [s]')
```

Programa 11.4: trayectoriaRP2GL.m

11.3. Simulación control difuso discreto de la postura y seguimiento de trayectorias

Simulación control difuso discreto de la postura

Programa principal del control difuso discreto de la postura.

```
%% Prueba de la estabilidad para cada ganancia
%% discreta de control moderno y con control Difuso
%% por el metodo de solucion de Euler
%% Parametros del Sistema no lineal discreto del pendulo
clear;
clc;
m=6.77; %Kg
r=0.5250; %m
g=9.72;
a=1/(m*r^2);
c=g/r;
%Condicion inicial
i=1;
x1=0.348062;
x2=0.003862;
x1k=x1;
x2k=x2;
x1km1=0;
x2km1=0;
x1km2=0;
x2km2=0;
uk=0; %condicion inicial del controlador
%condiciones de tiempo de muestreo y referencia del controlador
h=0.1; %paSos o tiempo de muestreo
ref=0;
%% inicializacion niveles de activacion para el control difuso
w1=0;
w2=0;
w3=0;
w4=0;
w5=0;
w6=0;
w7=0;
w8=0;
w9=0;
%% codigo para el control DIFUSO con solucion del Sistema por EULER
while i<300
  %% guarda los datos de las variables de estados
  xk(1,i)=x1k;
  xk(2,i)=x2k;
  ukd(1,i)=uk;
  %% la posicion limitada de 0<=xik<=2pi
  x1k=atan2(sin(x1k),cos(x1k));
  %% Funciones de Membresia
  %Membresias para la variable x1
  %          a     c    b  X
  mu1x1=FT(-0.3,  0  ,0.3,x1k);
  mu2x1=FT(-0.6,-0.3,0   ,x1k);
  mu3x1=FT( 0   , 0.3,0.6,x1k);
  %Membresias para la variable x2
  %          a     c    b  X
  mu1x2=FT(-0.3,  0  ,0.3,x2k);
  mu2x2=FT(-0.6,-0.3,0   ,x2k);
  mu3x2=FT( 0   , 0.3,0.6,x2k);
  %----Niveles de Activacion
  w(1,i)=w1;
  w(2,i)=w2;
  w(3,i)=w3;
  w(4,i)=w4;
  w(5,i)=w5;
  w(6,i)=w6;
  w(7,i)=w7;
  w(8,i)=w8;
  w(9,i)=w9;
  w1=mu1x1*mu1x2;
  w2=mu1x1*mu2x2;
  w3=mu1x1*mu3x2;
  w4=mu2x1*mu1x2;
```

```
  w5=mu2x1*mu2x2;
  w6=mu2x1*mu3x2;
  w7=mu3x1*mu1x2;
  w8=mu3x1*mu2x2;
  w9=mu3x1*mu3x2;
  %% Controles lineales e inferencia difusa T-S
  ukc1=-[56.9585    13.3858]*[x1k;x2k];
  ukc2=-[54.6276    13.1427]*[x1k;x2k];
  Un=(w1+w2+w3)*ukc1+(w4+w5+w6+w7+w8+w9)*ukc2;
  Ud=w1+w2+w3+w4+w5+w6+w7+w8+w9;
  uk=Un/Ud;
  %% Ecuacion de Estados solucion Euler
  x1km1=h*x2k+x1k;
  x2km1=h*(c*sin(x1k)+a*uk)+x2k;
  x1k=x1km1;
  x2k=x2km1;
  i=i+1;
end
```

Programa 11.5: Pruebas_de_la_estabilidad_de_CL.m

Programa para la función triangular.

```
%% Funcion para generar las funciones de membresia triangulares
function mu = FT(a,c,b,x)
if x<a
    mu=0;
elseif x>b
    mu=0;
elseif (x>=a) && (x<c)
    m1=1/(c-a);
    mu=m1*(x-a);
elseif (x>=c) && (x<=b)
    m2=-1/(b-c);
    mu=m2*(x-c)+1;
else
    mu=0;
end

end
```

Programa 11.6: FT.m

Simulación del control difuso discreto para el seguimiento de trayectorias

Programa principal del control difuso discreto para el seguimiento de las trayectorias.

```
%%Prueba de la estabilidad para cada ganancia
%%discreta de control moderno y con control Difuso
%%por el metodo de solucion de Euler
%%Utiliza la funcion FT.m
%% tiempo inicial y final
ti=0;
tf=4;
%% tiempos para las trayectorias
ttf1=1; %ttf1=tti2 ttfn=>tiempo de trayectoria final, ttin=>tiempo de trayectoria inicial
ttf2=2; %ttf2=tti3
ttf3=3; %ttf3=tti4
ttf4=tf;
%% Parametros del Sistema no lineal discreto del pendulo
% clear;
% clc;
m=6.77;%Kg
r=0.5250;%m
g=9.72;
a=1/(m*r^2);
c=g/r;
%coeficientes del polinomio trayectoria 1 posicion angular pendulo
%invertido
%Coeficientes del polinomio para las trayectorias
a1n=[a10 a11 a12 a13 a14 a15];
a2n=[a20 a21 a22 a23 a24 a25];
a3n=[a30 a31 a32 a33 a34 a35];
```

```
a4n=[a40 a41 a42 a43 a44 a45];
%Condicion inicial
i=1;
j=1;
p=0; %Perturbaciones
rk=0*rand*(pi/180); %referencia
drk=0*(pi/180); %velocidad de la referencia
x1=1*rand*(pi/180); %0.067815;
x2=1*rand*(pi/180);%-0.006020;
x1ek=x1-rk;
x2ek=drk-x2;
x1r=x1;
x2r=x2;
x1ekm1=0;
x2ekm1=0;
uk=-x1*(180/pi); %condicion inicial del controlador
%% condiciones de tiempo de muestreo y referencia del controlador
h=0.1;%%paos o tiempo de muestreo
%% inicializacion niveles de activacion para el control difuso
w1=0;
w2=0;
w3=0;
w4=0;
w5=0;
w6=0;
w7=0;
w8=0;
w9=0;
%% codigo para el control DIFUSO con solucion del Sistema por EULER
while j<142
  %% guarda los datos de las variables de estados
  xek(1,j)=x1ek;
  xek(2,j)=x2ek;
  ukd(1,j)=uk;
  xr(1,j)=x1r;
  xr(2,j)=x2r;
  if i<10*(tf)+2
  %% la posicion limitada de 0<=xik<=2pi
  x1ek=atan2(sin(x1ek),cos(x1ek));
  %% Funciones de Membresia
  %Membresias para la variable x1
  %        a       c      b      X
  mu1x1=FT(-0.0873, 0     ,0.0873,x1ek);
  mu2x1=FT(-0.1745,-0.0873,0     ,x1ek);
  mu3x1=FT( 0     , 0.0873,0.1745,x1ek);
  %Membresias para la variable x2
  %        a       c      b      X
  mu1x2=FT(-0.0873, 0     ,0.0873,x2ek);
  mu2x2=FT(-0.1745,-0.0873,0     ,x2ek);
  mu3x2=FT( 0     , 0.0873,0.1745,x2ek);
  %-----Niveles de Activacion
  w(1,j)=w1;
  w(2,j)=w2;
  w(3,j)=w3;
  w(4,j)=w4;
  w(5,j)=w5;
  w(6,j)=w6;
  w(7,j)=w7;
  w(8,j)=w8;
  w(9,j)=w9;
  w1=mu1x1*mu1x2;
  w2=mu1x1*mu2x2;
  w3=mu1x1*mu3x2;
  w4=mu2x1*mu1x2;
  w5=mu2x1*mu2x2;
  w6=mu2x1*mu3x2;
  w7=mu3x1*mu1x2;
  w8=mu3x1*mu2x2;
  w9=mu3x1*mu3x2;
  %% Controles lineales e inferencia difusa T-S y trayectroria
  %% Trayectoria
  tk=h*i;
  if tk>=0 && tk<ttf1
        rk=fgdt(1,tk,a1n); %Trayectoria
        drk=fgdt(2,tk,a1n); %derivada de la trayectoria
```

```
        %estado=0; %(0 + , 1 -) conmuta el valor del angulo del pendulo de + a - para cambiar la
    referencia
        pierna(1,i)=0; %pendulo Pierna Izq. (PI)
  elseif tk>=ttf1 && tk<ttf2
        rk=fgdt(1,tk,a2n); %Trayectoria
        drk=fgdt(2,tk,a2n); %derivada de la trayectoria
        %estado=1;
        pierna(1,i)=1; %pendulo Pierna Der. (PD)
  elseif tk>=ttf2 && tk<ttf3
        rk=-fgdt(1,tk,a3n); %Trayectoria
        drk=-fgdt(2,tk,a3n); %derivada de la trayectoria
        %estado=0;
        pierna(1,i)=0; %pendulo Pierna Der. (PD)
  elseif tk>=ttf3 && tk<=ttf4
        rk=-fgdt(1,tk,a4n); %Trayectoria
        drk=-fgdt(2,tk,a4n); %derivada de la trayectoria
        %estado=1;
        pierna(1,i)=1; %pendulo Pierna Izq. (PI)
  end
  %% control
  ukc1=-[56.9585    13.3858]*[x1ek;x2ek];
  ukc2=-[56.7602    13.3653]*[x1ek;x2ek];
  Un=(w1+w2+w3)*ukc1+(w4+w5+w6+w7+w8+w9)*ukc2;
  Ud=w1+w2+w3+w4+w5+w6+w7+w8+w9;
  uk=Un/Ud;
  %% Ecuacion de Estados solucion Euler
  x1ekm1=h*x2ek+x1ek;
  x2ekm1=h*(c*sin(x1ek)+a*uk)+x2ek;
  %x1ek=x1ekm1;
  %x2ek=x2ekm1;
  %% Variables reales
  x1r=x1ekm1+rk;
  x2r=x2ekm1+drk;
  %% Sensor ideal
  theta=atan2(sin(x1r),cos(x1r));
  %%-----Calculo del error--------
  x1ek=theta-rk+1*(pi/180)*rand; %el rand simula el ruido
  x2ek=x2r-drk;%+0.01*(pi/180)*rand;
  i=i+1;
  else
      i=1;
  end
  j=j+1;
end
```

Programa 11.7: Pruebas_de_la_estabilidad_de_CL3.m

11.4. Control difuso postura y caminata bípeda en ROS

Control difuso postura en ROS

Programa principal del control difuso discreto de la postura.

```
#include "ros/ros.h"
#include "std_msgs/Float32MultiArray.h"
#include "std_msgs/Float32.h"
#include <iostream>
#include <fstream>
#include "FT.h"
using namespace std;
using namespace Funcion_Triangular;
int i=0;
float Hz=10;
float Ts=1/Hz;
float x=0,y=0;
float r=0.5250;//[m]
float g=9.72;//[m/s^2]
float m=6.77;//[Kg]
float a=1/(m*r*r);//Constante
float c=g/r;//Constante
//Varibles de estados discretas y sus deferencias
float x1k=0*(3.1416/180),x2k=0,x1km1=0,x2km1=0,x1=x1k,x2=x2k;
//variable de entrada o de control
```

```
21 float uk=0;
22 //variables de control difuso
23 float ukc1=0,ukc2=0;
24 float Un=0,Ud=0;
25 //funciones de membresia
26 float mu1x1=0,mu2x1=0,mu3x1=0;//Funcion de membresia para la variable de estado x1
27 float mu1x2=0,mu2x2=0,mu3x2=0;//Funcion de membresia para la variable de estado x2
28 // Niveles de activacion para el control difuso
29 float w1=0,w2=0,w3=0; //niveles de activacion para el control de los sistemas dx=A1x+Bu,dx=A2x+Bu,dx
       =A3x+Bu
30 float w4=0,w5=0,w6=0,w7=0,w8=0,w9=0;// nivels de activacion para el control de los sistemas dx=An+Bu
       , n=4,5,..,9
31 void datosObtenidos(const std_msgs::Float32MultiArray::ConstPtr& msgs)
32 {
33  x1=msgs->data[0];
34  x2=msgs->data[1];
35 }
36
37 int main(int argc, char **argv)
38 {
39   ros::init(argc, argv,"solucion_mpe_y_control");
40   ros::NodeHandle n;
41   ros::Subscriber subS=n.subscribe("/sensor",1,datosObtenidos);
42   std_msgs::Float32 EstadosX12;
43   //EstadosX12.data.resize(2);
44   ros::Rate loop(Hz);
45   FILE*pFile;
46   pFile=fopen("/home/ubuntu/Big_Brother/catkin_ws/src/robotbb_cdyp/src/var_de_estados.txt","w");
47   while(ros::ok())
48   {
49    //cout<<"x1="<<x1<<", x2="<<x2<<", u="<<uk<<endl;
50     if(i<=302)
51     {
52       fprintf(pFile,"%f,%f,%f;\n",x1,x2,uk);
53     }
54     else
55     {
56     }
57 //-------------------------------------------------------------------
58     //Control para los sistemas A1, A2, A3
59     //uk=-(56.9585*x1k+13.3858*x2k);
60 //-------------------------------------------------------------------
61     //Control para los sistemas A4,A5,...,A9
62     //uk=-(54.6276*x1k+13.1427*x2k);
63 //-------------------------------------------------------------------
64     //Control Difuso
65     //Funciones de membresia para la variable de estados x1k
66     //       inicio centro  final  X
67     mu1x1=ft(-0.3,     0,    0.3, x1);
68     mu2x1=ft(-0.6,  -0.3,      0, x1);
69     mu2x1=ft(   0,   0.3,    0.6, x1);
70     //Funciones de membresia para la variable de estados x2k
71     //       inicio centro  final  X
72     mu1x2=ft(-0.3,     0,    0.3, x2);
73     mu2x2=ft(-0.6,  -0.3,      0, x2);
74     mu2x2=ft(   0,   0.3,    0.6, x2);
75     //Calculo de los niveles de activacion
76     w1=mu1x1*mu1x2;
77     w2=mu1x1*mu2x2;
78     w3=mu1x1*mu3x2;
79     w4=mu2x1*mu1x2;
80     w5=mu2x1*mu2x2;
81     w6=mu2x1*mu3x2;
82     w7=mu3x1*mu1x2;
83     w8=mu3x1*mu2x2;
84     w9=mu3x1*mu3x2;
85     //Calculos de las leyes de control ukc1, ukc2 para el control difuso basado en modelo
86     ukc1=-(56.9585*x1+13.3858*x2);
87     ukc2=-(54.6276*x1+13.1427*x2);
88     //Calculo de la Inferencia Difusa
89     Un=(w1+w2+w3)*ukc1+(w4+w5+w6+w7+w8+w9)*ukc2;
90     Ud=w1+w2+w3+w4+w5+w6+w7+w8+w9;
91     uk=Un/Ud;
92     //Solucion numerica por el metodo de euler
93     //Ecuacion de estados del pendulo invertido
94     //x1k=atan2(sin(x1),cos(x1));
```

```
    x1km1=Ts*x2+x1;
    x2km1=Ts*(c*sin(x1)+a*uk)+x2;
    //x1k=x1km1;
    //x2k=x2km1;
    cout<<"i="<<i<<endl;
    i++;
    pubvec.publish(EstadosX12);
    ros::spinOnce();
    loop.sleep();
  }
  fclose(pFile);
  return 0;
}
```

Programa 11.8: robotbb_ cdyp_ node.cpp

Programa para generar las funciones de membresía triangulares.

```
#include <iostream>
using namespace std;
namespace Funcion_Triangular
{ //       inicial   centro    final       X
  float ft(float a, float c, float b, float x)
  {
    float ut;
    float m1=1/(c-a);
    float m2=-(1/(b-c));
    if(x<a)
    {
       ut=0;
    }
    else
    {
       if(x>b)
       {
          ut=0;
       }
       else
       {
          if(x>=a && x<c)
          {
             ut=m1*(x-a);
          }
          else
          {
            if(x>=c && x<=b)
            {
                ut=m2*(x-c)+1;
            }
            else
            {
                ut=0;
            }
          }
       }
    }
    return ut;
  }
}
```

Programa 11.9: FT.h

Control difuso caminata bípeda en ROS

Programa principal del control difos discreto para la caminata bípeda.

```
#include "ros/ros.h"
#include "std_msgs/Float32MultiArray.h"
//#include "std_msgs/Float32.h"
#include <iostream>
#include <fstream>
#include "FT.h"
#include "GDT.h"
using namespace std;
using namespace Funcion_Triangular;
using namespace Generador_De_Trayectorias;
int i=0;
float Hz=10;
float Ts=1/Hz;
float x=0,y=0;
float r=0.5250;//[m]
float g=9.72;//[m/s^2]
float m=6.77;//[Kg]
float a=1/(m*r*r);//Constante
float c=g/r;//Constante
float pi=3.14;
//cambio de pierna para generar el movimiento del pendulor
float pierna=0; //0->Pierna Izq 1->Pierna Der
//Varibles de estados del error discretas y sus deferencias
float x1ek=0*(3.1416/180),x2ek=0,x1ekm1=0,x2ekm1=0,x1=x1ek,x2=x2ek;
//variable de entrada o de control
float uk=0;
//Tiempo final de las trayectorias
float tf=4,tk=0; //tf=29.8 [seg]
float ttf1=1,ttf2=2,ttf3=3,ttf4=tf;//ttin+1=ttfn ttfn->tiempo de la trayectoria final ttin->tiempo
    de la trayectoria inicial
//trayectoria pierna movil
float tc=-10.0051*(pi/180);
//Variables de las referencias
float rk=0*(3.1416/180),drk=0;
//
    ------------------------------------------------------------------------------------------------------------------------

//Coeficientes del polinomio de la trayectoria para el pendulo
//trayectoria de 0 a 10 grados de 0 a 4 seg
float a10=0.000000E+00,a11=0.000000E+00,a12=-0.000000E+00,a13=1.744444E+00,a14=-2.616667E+00,a15
    =1.046667E+00;
float a20=5.582222E+00,a21=-2.093333E+01,a22=3.140000E+01,a23=-2.267778E+01,a24=7.849999E+00,a25
    =-1.046667E+00;
float a30=-8.931555E+01,a31=1.884000E+02,a32=-1.570000E+02,a33=6.454444E+01,a34=-1.308333E+01,a35
    =1.046667E+00;
float a40=5.135645E+02,a41=-7.536000E+02,a42=4.396000E+02,a43=-1.273444E+02,a44=1.831667E+01,a45
    =-1.046667E+00;

//
    ------------------------------------------------------------------------------------------------------------------------

//coeficientes de las trayectorias para movimiento de la pierna movil, y con esto generar el paso
//trayectorias de -10.0051 a -20 grados de 0 a 4 seg
float tc10=-1.745334E-01,tc11=-0.000000E+00,tc12=0.000000E+00,tc13=-1.743555E+00,tc14=2.615332E+00,
    tc15=-1.046133E+00;
float tc20=-5.753908E+00,tc21=2.092266E+01,tc22=-3.138398E+01,tc23=2.266621E+01,tc24=-7.845996E+00,
    tc25=1.046133E+00;
float tc30=8.909547E+01,tc31=-1.883039E+02,tc32=1.569199E+02,tc33=-6.451152E+01,tc34=1.307666E+01,
    tc35=-1.046133E+00;
float tc40=-5.134771E+02,tc41=7.532156E+02,tc42=-4.393758E+02,tc43=1.272795E+02,tc44=-1.830732E+01,
    tc45=1.046133E+00;

//
    ------------------------------------------------------------------------------------------------------------------------

//variables de control difuso
float ukc1=0,ukc2=0;
float Un=0,Ud=0;
//funciones de membresia
float mu1x1=0,mu2x1=0,mu3x1=0;//Funcion de membresia para la variable de estado x1
float mu1x2=0,mu2x2=0,mu3x2=0;//Funcion de membresia para la variable de estado x2
```

```
57 // Niveles de activacion para el control difuso
58 float w1=0,w2=0,w3=0; //niveles de activacion para el control de los sistemas dx=A1x+Bu,dx=A2x+Bu,dx
       =A3x+Bu
59 float w4=0,w5=0,w6=0,w7=0,w8=0,w9=0;// nivels de activacion para el control de los sistemas dx=An+Bu
       , n=4,5,..,9
60 void datosObtenidos(const std_msgs::Float32MultiArray::ConstPtr& msgs)
61 {
62  x1=-msgs->data[2];
63  x2=-msgs->data[3];
64 }
65
66 int main(int argc, char **argv)
67 {
68   ros::init(argc, argv,"solucion_mpe_y_control");
69   ros::NodeHandle n;
70   ros::Subscriber subS=n.subscribe("/sensor",1,datosObtenidos);
71   //std_msgs::Float32 EstadosX12;
72   std_msgs::Float32MultiArray EstadosX12;
73   EstadosX12.data.resize(4);
74   ros::Rate loop(Hz);
75   FILE*pFile;
76   pFile=fopen("/home/ubuntu/Big_Brother/catkin_ws/src/robotbb_cdyp5/src/var_de_estados.txt","w");
77   while(ros::ok())
78   {
79    cout<<"x1="<<x1<<", x2="<<x2<<", x1ek="<<x1ek<<", x2ek="<<x2ek<<", uk="<<uk<<", tc="<<tc<<", tk="
       <<tk<<endl;
80     if(i<=402)
81     {
82       fprintf(pFile,"%f,%f,%f,%f,%f;\n",x1,x2,x1ek,x2ek,uk,tc);
83     }
84     else
85     {
86     }
87 //-------------------------------------------------------------------
88     //Control para los sistemas A1, A2, A3
89     //uk=-(56.9585*x1k+13.3858*x2k);
90 //-------------------------------------------------------------------
91     //Control para los sistemas A4,A5,...,A9
92     //uk=-(54.6276*x1k+13.1427*x2k);
93 //-------------------------------------------------------------------
94    //tiempo
95    tk=Ts*i;
96 //-------------------------------------------------------------------
97    //Calculo de las trayectorias
98    //rk(tk) trayectoria posicion drk(tk) trayectoria velocidad
99    if(tk>=0 && tk<ttf1)
100   {//1
101      rk=fgdt(1,tk,a10, a11, a12, a13, a14, a15);//funcion calcula las trayectorias
102     drk=fgdt(2,tk,a10, a11, a12, a13, a14, a15);
103      tc=fgdt(1,tk,tc10, tc11, tc12, tc13, tc14, tc15); //trayectoria cadera
104     pierna=0;
105   }//1
106   else
107   {//1
108     if(tk>=ttf1 && tk<ttf2)
109     {//2
110        rk=fgdt(1,tk,a20, a21, a22, a23, a24, a25);//funcion calcula las trayectorias
111       drk=fgdt(2,tk,a20, a21, a22, a23, a24, a25);
112        tc=fgdt(1,tk,tc20, tc21, tc22, tc23, tc24, tc25); //trayectoria cadera
113       pierna=0;
114     }//2
115     else
116     {//2
117       if(tk>=ttf2 && tk<ttf3)
118       {//3
119          rk=-fgdt(1,tk,a30, a31, a32, a33, a34, a35);//funcion calcula las trayectorias
120         drk=-fgdt(2,tk,a30, a31, a32, a33, a34, a35);
121          tc=fgdt(1, tk,tc30,tc31, tc32, tc33, tc34, tc35); //trayectoria cadera
122         pierna=1;
123       }//3
124       else
125       {//3
126         if(tk>=ttf3 && tk<=ttf4)
127         {//4
128             rk=-fgdt(1,tk,a40, a41, a42, a43, a44, a45);//funcion calcula las trayectorias
129            drk=-fgdt(2,tk,a40, a41, a42, a43, a44, a45);
```

```
130             tc=fgdt(1,tk,tc40, tc41, tc42, tc43, tc44, tc45); //trayectoria cadera
131             pierna=1;
132           }//4
133           else{}//4
134         }//3
135       }//2
136     }//1
137 //---------------------------------------------------------------------
138     //Calculo del error
139     x1ek=rk-x1;
140     x2ek=drk-x2;
141 //---------------------------------------------------------------------
142      //Control Difuso
143      //Funciones de membresia para la variable de estados x1k
144      //       inicio   centro      final   X
145      mu1x1=ft(-0.1745,   0     ,   0.1745, x1ek);
146      mu2x1=ft(-0.3491,  -0.1745,   0     , x1ek);
147      mu3x1=ft( 0     ,   0.1745,   0.3491, x1ek);
148      //Funciones de membresia para la variable de estados x2k
149      //       inicio   centro      final   X
150      mu1x2=ft(-0.1745,   0     ,   0.1745, x2ek);
151      mu2x2=ft(-0.3491,  -0.1745,   0     , x2ek);
152      mu3x2=ft( 0     ,   0.1745,   0.3491, x2ek);
153      //Calculo de los niveles de activacion
154      w1=mu1x1*mu1x2;
155      w2=mu1x1*mu2x2;
156      w3=mu1x1*mu3x2;
157      w4=mu2x1*mu1x2;
158      w5=mu2x1*mu2x2;
159      w6=mu2x1*mu3x2;
160      w7=mu3x1*mu1x2;
161      w8=mu3x1*mu2x2;
162      w9=mu3x1*mu3x2;
163      //Calculos de las leyes de control ukc1, ukc2 para el control difuso basado en modelo
164      ukc1=-(56.9585*x1ek+13.3858*x2ek);
165      ukc2=-(56.1673*x1ek+13.3039*x2ek);
166      //Calculo de la Inferencia Difusa
167      Un=(w1+w2+w3)*ukc1+(w4+w5+w6+w7+w8+w9)*ukc2;
168      Ud=w1+w2+w3+w4+w5+w6+w7+w8+w9;
169      uk=Un/Ud;
170      //Solucion numerica por el metodo de euler
171      //Ecuacion de estados del pendulo invertido
172      //x1k=atan2(sin(x1),cos(x1));
173      x1ekm1=Ts*x2ek+x1;
174      x2ekm1=Ts*(c*sin(x1ek)+a*uk)+x2ek;
175      //cout<<"i="<<i<<endl;
176      if(i==tf/Ts)
177      {
178        //i=tf/Ts;
179          i=0;
180      }
181      else
182      {
183         i++;
184      }
185      pubvec.publish(EstadosX12);
186      ros::spinOnce();
187      loop.sleep();
188    }
189    fclose(pFile);
190    return 0;
191 }
```

Programa 11.10: robotbb_cdyp5_node.cpp

Bibliografía

[1] H. Hirukawa, "Walking biped humanoids that perform manual labour," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2007.

[2] H. O. Lim, Y. Ogura, and A. Takanishi, "Locomotion pattern generation and mechanisms of a new biped walking machine," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2008.

[3] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics.* Springer tracts in advanced robotics: 101, Berlin : Springer, [2014], 2014.

[4] M. S. Mahmoud, *Fuzzy Control, Estimation and Diagnosis : Single and Interconnected Systems.* Cham : Springer International Publishing : Imprint: Springer, 2018., 2018.

[5] R. Palm, D. Driankov, and H. Hellendoorn, *Model based fuzzy control : fuzzy gain schedulers and sliding mode fuzzy controllers.* Berlin : Springer Verlag, c1997, 1997.

[6] A. Tapia Villegas, *Diseño estructural de un robot humanoide para las competencias tipo RoboCup soccer.* 2017.

[7] RoboCup, "Robocup." `https://www.robocup.org/photos`, 2011. [Online; accesed 04-Marzo-2019].

[8] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic : theory and applications.* Upper Saddle River, New Jersey : Prentice Hall, c1995, 1995.

[9] J. J. Buckley and E. Eslami, *An introduction to fuzzy logic and fuzzy sets.* Advances in soft computing, Heidelberg : Physica-Verlag, c2002, n.d.

[10] B. Bede, *Mathematics of fuzzy sets and fuzzy logic.* Studies in fuzziness and soft computing: 295, Berlin : Springer, 2013, 2013.

[11] J. Frye, "Ssc-32 manual." `http://www.trcom.com.ar/img/product/A043_BRAZO_ROBOT_MANIPULADOR_PC_0_ARDUINO/SSC32_Manual_Ingles.pdf`, 2009. [Online; accesed 6-Febrero-2019].

[12] V. Tripathi, A. Bansal, and R. Gupta, "Development of self-stabilizing platform using mpu-6050 as imu," in *Advances in Signal Processing and Communication* (B. S. Rawat, A. Trivedi, S. Manhas, and V. Karwal, eds.), (Singapore), pp. 373–382, Springer Singapore, 2019.

[13] InvenSenese, "Mpu-6000 and mpu-6050 product specification revision 3.4." `https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf`, 2013. [Online; accesed 18-Febrero-2019].

[14] HabibOladepo, "Nodes." `http://wiki.ros.org/Nodes`, 2018. [Online; accesed 20-Febrero-2019].

[15] TullyFoote, "Topics." `http://wiki.ros.org/Topics`, 2019. [Online; accesed 20-Febrero-2019].

[16] R. Urdhwareshe, M. Bakshi, P. Naiknavare, and S. Naik, "Design and implementation of imu sensor fusion and pid control in quadrotor," *Int. J. Electron. Commun*, vol. 2, pp. 55–63, 2014.

[17] ElectronicWings, "Mpu6050 (accelerometer+gyroscope) interfacing with raspberry pi." `https://www.electronicwings.com/raspberry-pi/mpu6050-accelerometergyroscope-interfacing-with-raspberry-pi`, 2018. [Online; accesed 27-Febrero-2019].

[18] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control.* Hoboken, New Jersey : J. Wiley, c2006, 2006.

[19] M. G. LOPEZ RODRIGUEZ and J. D. CASTRO DÍAZ, *Apuntes de Control de Robots Industriales Teoría y Laboratorio, Facultad de Ingeniería, UNAM.* 2018.

[20] G. O. Dr. Marcos Angel, *Apuntes de Control Avanzado, Facultad de Ingeniería, UNAM.* 2018.

[21] D. Greenspan, *Numerical solution of ordinary differential equations : for classical, relativistic and nano systems.* Physics textbook, Weinheim : Wiley-VCH, [2006], 2006.

[22] K. Ogata, J. G. Aranda Perez, F. J. Rodríguez Ramírez, G. Sanchez Garcia, and K. Ogata, *Sistemas de control en tiempo discreto.* México : Prentice Hall, c1996, 1996.

[23] InvenSenese, "Mpu-9150 product specification revision 4.3." https://www.invensense.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf, 2013. [Online; accesed 19-Agosto-2019].

[24] P. Bartz, "razor imu 9dof." http://wiki.ros.org/razor_imu_9dof, 2018. [Online; accesed 19-Agosto-2019].

[25] S. Castro, "Walking robot control: From pid to reinforcement learning." https://blogs.mathworks.com/racing-lounge/2019/04/24/walking-robot-control/, April 24, 2019. [Online; accesed 03-Septiembre-2019].

[26] R. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition).* Springer US, 2010.