PONTIFICIA UNIVERSIDAD
CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

DESARROLLO DE UN CONTROLADOR DE POSICIÓN AVANZADO PARA ENDOSCOPIO BLANDO EN CIRUGÍA LAPAROSCÓPICA

Tesis para optar el grado académico de Maestro en
Ingeniería de Control y Automatización que presenta:

Renzo Rogger Acosta Gonzales

Asesor:

Ph.D., Ing. Julio César Tafur Sotelo

Co Asesora:

Ph.D., Ing. Ruth Vanessa Canahuire Cabello

Lima, 2023

Informe de Similitud

Yo, Julio César Tafur Sotelo, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesor de la tesis de investigación titulado “Desarrollo de un Controlador de Posición Avanzado para Endoscopio Blando en Cirugía Laparoscópica”, del autor Acosta Gonzales Renzo Rogger, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 15%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 21/06/2023.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:
Lima, 16 de septiembre del 2023

Apellidos y nombres del asesor: Tafur Sotelo, Julio César	
DNI: 06470028	
ORCID: 0000-0003-3415-1969	Firma:

Agradecimientos

El agradecimiento a PROCIENCIA por el apoyo financiero en el desarrollo de esta tesis mediante el proyecto titulado “Sistema Robótico Cooperativo para Cirugía Asistida con Funcionalidades de Control de Corte y Endoscopio Blando”, con CONTRATO 142-2020-FONDECYT, a los docentes asesores por su guía constante, a la colaboración constante del equipo de trabajo del proyecto, a mi familia por el apoyo incondicional, y a Dios por darme la oportunidad y la salud para continuar desarrollándome personalmente y profesionalmente.

Resumen

El presente estudio se desarrolla en el marco de brindar asistencia al cirujano en la laparoscopia, la cual es una cirugía utilizada para tratar problemas de salud en la zona abdominal. El procedimiento utiliza una cámara conectada a un tubo delgado flexible llamado endoscopio, el cual permite observar al interior de la zona abdominal del paciente; las imágenes obtenidas por el instrumento son utilizadas por el cirujano durante el tratamiento del paciente. Para garantizar un procedimiento correcto, se debe mover correctamente el endoscopio en el interior del abdomen, siendo esta tarea específicamente la que se busca facilitar su control y con ello dar apertura a una serie de posibilidades como el movimiento asistido, la operación remota y la automatización completa de la tarea.

Con el fin de proponer una solución, actualmente, con el avance en el campo de la robótica blanda se han diseñado y fabricados manipuladores o actuadores blandos que puedan ser usados como endoscopios, los cuales tienen la capacidad de deformarse y ser forzados a moverse para alcanzar diferentes posiciones deseadas dentro de sus límites de operación. El cuerpo del manipulador o actuador blando en estudio presenta cuatro cámaras internas, las cuales pueden ser deformadas regulando la cantidad de presión de aire al interior de cada cámara. Para controlar y alcanzar la posición deseada del efector final del endoscopio, donde una cámara será conectada, en el presente trabajo se realiza el modelamiento de la dinámica del cuerpo del endoscopio y el diseño del controlador de posición.

La tarea de modelamiento consiste en definir las características de la estructura de una red neuronal recurrente con realimentación a la salida y luego realizar su entrenamiento usando el algoritmo DBP (Dynamic Back-Propagation) para obtener los pesos de conexión entre las neuronas de la red. El diseño del controlador consiste de dos etapas. En la primera etapa se definen las características de la estructura de una red neuronal prealimentada (feed-forward). Para el entrenamiento de la red se utiliza el algoritmo DBP bajo un enfoque dinámico donde se considera el sistema en lazo cerrado, el cual comprende tanto al controlador como al modelo del sistema. El controlador de posición obtenido es válido solamente dentro de un rango de movimiento; por ello, se definen un conjunto de controladores para cada rango de operación. En la segunda etapa, se utiliza el método difuso Takagi Sugeno para la integración de los controladores locales y la obtención de un controlador global valido en todo el rango de operación. El controlador obtenido se implementa y prueba mediante simulación con el objetivo de validar su desempeño para diferentes posiciones deseadas del endoscopio.

Palabras claves: robótica blanda, actuador neumático blando, redes neuronales, lógica difusa

Abstract

The present study is carried out within the framework of providing assistance to the surgeon in laparoscopy, which is a surgery used to treat health problems in the abdominal area. The procedure uses a camera connected to a thin flexible tube called endoscope, which allows seeing inside the patient's abdominal area; the images obtained are used by the surgeon during the patient's treatment. An essential and correct procedure consists of moving the endoscope correctly inside the abdomen. This task is seeking to facilitate its control and open up a series of possibilities such as assisted movement, remote operation and complete automation of tasks.

In order to propose a solution, currently, with advances in the field of soft robotics, soft manipulators or actuators have been designed and manufactured to be used as endoscopes, which have the ability to deform and be forced to move and reach different desired positions within its operation limits. The soft manipulator body or actuator under study has four internal chambers, which can be deformed by regulating air pressure of each chamber. In order to control and reach the desired position of endoscope final effector, where a camera will be connected, in the current work the endoscope body modeling and the position controller design are carried out as main tasks.

The modeling task consists of defining the characteristics of a recurrent neural network with output feedback and then training it using the DBP (Dynamic Back-Propagation) algorithm to obtain its connection weights between network neurons. The controller design consists of two stages. In the first stage, the characteristics of a feed forward neural network are defined. For network training, the DBP algorithm is used under a dynamic approach where the closed-loop system is considered, which includes both the controller and the system model. The obtained position controller is valid only within a range of motion; therefore, a set of controllers is defined for each range of operation. In the second stage, the fuzzy Takagi Sugeno method is used to integrate the local controllers and obtain a global controller for complete endoscope operating range. The controller obtained is implemented and tested by simulation in order to validate its performance for different desired positions of endoscope final effector.

Key words: soft robotic, soft pneumatic actuator, neural networks, data driven modeling and control techniques, fuzzy logic controllers

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

Dentro de los antecedentes del presente estudio, se considera a la laparoscopia como un procedimiento que ha revolucionado el campo de la cirugía por los beneficios que ofrece al paciente como un rápido tiempo de recuperación, la reducción de costos por periodo menores de estancia en el centro hospitalario, reducción de tamaño y número de cortes que mejora la estética post intervención. Sin embargo, estos procedimientos son más difíciles de llevar a cabo para los cirujanos respecto a la cirugía clásica (abierta e invasiva), debido a la necesidad de manipular remotamente los instrumentos de cirugía en la zona de intervención del paciente. Además, el campo de visión es reducido generalmente a dos dimensiones y se requiere de un alto entrenamiento y experiencia para el buen control de fuerza y posición durante la manipulación de los instrumentos en el interior del cuerpo del paciente. Por ello, para aliviar las desventajas mencionadas anteriormente, a la actualidad se han desarrollado sistemas robóticos quirúrgicos como el Da Vinci, que, según la revista de la Sociedad de Cirujanos Laparoscópicos y Robóticos, se ha utilizado en diferentes tipos de cirugía como la histerectomía, prostatectomía, colecistectomía y cirugía transoral (Tovah W. et al, 2022).

En dicha línea, el continuo desarrollo de la robótica en el campo de laparoscopia, busca no solamente limitarse a la interacción de forma externa con el paciente, como es el caso del robot Da Vinci con quién se puede realizar desde tarea sencillas como colocar hilo a una aguja hasta realizar varias suturas con diferentes niveles de precisión, ya que existe la necesidad de los cirujanos de ser asistidos con instrumentos que interactúan al interior del paciente como es el caso de los endoscopios blandos, el cual permite observar las cavidades internas de la persona durante una operación. El endoscopio blando es caracterizado principalmente por su alta capacidad de deformación, la cual permite al cuerpo del endoscopio tener una alta capacidad de maniobrabilidad en las cavidades internas del paciente, las cuales son de tamaño reducido y dependiente de la situación clínica de cada persona. Otra característica importante es asegurar la compatibilidad del instrumento al entrar en contacto con los diferentes tejidos y órganos del cuerpo humano de forma segura. El diseño y construcción de los endoscopios blandos inteligentes, cuyo control de movimiento se realiza de manera asistida, se basa en la robótica blanda, la cual es una rama multidisciplinaria que integra la ciencia de los materiales, los circuitos electrónicos, los algoritmos inteligentes y mecanismos de actuación como la neumática y electromecánica principalmente. Donde el presente trabajo se centra en el diseño e implementación de un controlador de posición avanzado para un endoscopio blando con accionamiento neumático.

La importancia de desarrollar un controlador de posición para el endoscopio blando radica en aprovechar su alta capacidad de maniobrabilidad en entornos de operación reducidos y no estructurados. Está característica se debe a que el cuerpo del endoscopio está fabricado de un material hiperelástico normalmente de silicona brindándole infinitos grados de libertad al cuerpo. Debido a la característica del material, el estudio del cuerpo del endoscopio está basado en los manipuladores blandos o continuos, donde las investigaciones tanto teóricas y aplicadas en el área de control son relativamente modernas y en actual desarrollo. Por ello, mediante el presente trabajo se pretende aportar con la investigación en el desarrollo del controlador de posición avanzado para un endoscopio blando en la cirugía laparoscópica, de manera que permita asistir al cirujano y disminuir el tiempo de recuperación del paciente.

El documento está estructurado en seis capítulos donde se desarrollan los temas correspondientes al estado del arte, características del endoscopio blando, modelamiento del endoscopio blando, diseño del controlador de posición para el endoscopio blando, resultados e implementación mediante simulación y conclusiones. En el primer capítulo, se muestran los principales conceptos y técnicas correspondientes al modelamiento y control de manipuladores blandos, donde la aplicación del presente trabajo es el cuerpo de un endoscopio blando. En el segundo capítulo, se describen las características del diseño mecánico del cuerpo del endoscopio blando considerado en el presente trabajo y la instrumentación requerida para el desarrollo experimental de la metodología propuesta en la presente tesis. En el tercer capítulo, se desarrolla la metodología para el modelamiento de la dinámica del cuerpo del endoscopio basado en una estructura de red neuronal recurrente con retroalimentación a la salida, donde se describe las características de la red neuronal y el algoritmo implementado para su entrenamiento, luego se presenta la definición y los resultados de los indicadores de desempeño correspondientes a la etapa de entrenamiento y validación del modelo. En el cuarto capítulo, se presenta el diseño del controlador de posición avanzado, el cual se divide en dos etapas. En la primera etapa, se define las características de la estructura del controlador de posición local basado en una red neuronal prealimentada; así como la estrategia y algoritmo de entrenamiento de la red. En la segunda de etapa, se describe el análisis de estabilidad del sistema en lazo en cerrado, a partir del cual se determina la validez solamente local del controlador obtenido. A partir de ello, se desarrolla el método difuso de Takagi Sugeno para la integración de controladores locales y la obtención de un controlador global válido para todo el rango de operación del endoscopio. En el quinto capítulo, se describe la implementación del controlador y sus respectivas pruebas de desempeño mediante simulación. En el sexto capítulo, se presentan las conclusiones que se llegaron durante el desarrollo del presente trabajo.

Capítulo I: ESTADO DEL ARTE

1.1. Introducción

La cirugía laparoscópica es un tipo de cirugía mínimamente invasiva realizada en el vientre de la persona, la cual por sus características permite reducir de forma favorable el tiempo de recuperación del paciente. Durante el procedimiento, los cirujanos requieren el uso del endoscopio que le permita examinar internamente la zona de intervención del paciente; por ello, tener la capacidad de poder controlar la posición del endoscopio dentro del cuerpo de la persona es fundamental para lograr una cirugía exitosa. En este primer capítulo, se revisan las técnicas actuales de control de posición de los endoscopios, lo cual involucran el estudio de las metodologías de modelamiento, las estrategias de control, los algoritmos de control y la introducción de ciertos conceptos y descripciones correspondiente al campo de la robótica blanda. Finalmente, se presenta el objetivo general y los objetivos específicos del presente trabajo, los cuales ocupan su desarrollo en gran medida en los capítulos posteriores.

1.2 Conceptos preliminares de diseño y construcción

Se presenta de forma preliminar para un mejor entendimiento las características principales físicas del cuerpo de un endoscopio blando, representado en la Figura 1.1 (a) dentro del abdomen del paciente. El diseño y la construcción del cuerpo del endoscopio está basado en las características de los manipuladores (o robots) blandos; por ello, el material utilizado para su fabricación son materiales hiperelásticos como la silicona o el caucho. Su geometría y diseño está orientada a sufrir deformaciones funcionales tanto axiales como de flexión dentro de su entorno de operación y con ello alcanzar diferentes posiciones y orientaciones en el efector final de todo el cuerpo, donde se instala una pequeña cámara para permitir la visión al cirujano al interior del paciente. En la parte (b) y (c) de la Figura 1.1 se muestran el cuerpo del endoscopio sin deformación y con deformación respectivamente.

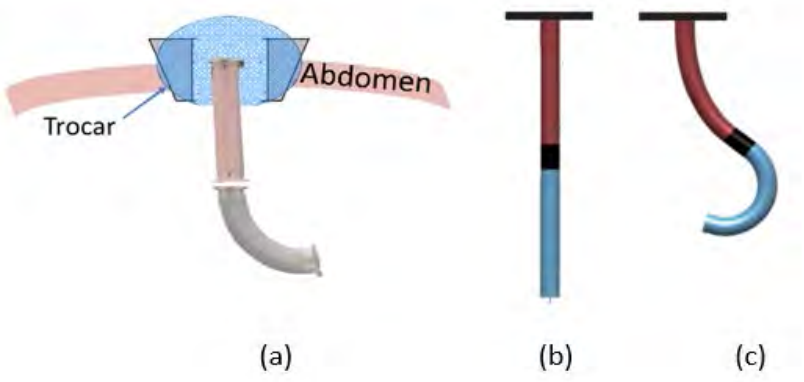


Figura 1.1. Endoscopio blando de dos secciones con deformaciones funcionales

Los manipuladores blandos comúnmente utilizados como endoscopios son accionados por mecanismo de tendones o cámaras neumáticas. En el presente trabajo se utilizará un endoscopio con accionamiento neumático debido a sus principales ventajas como dinámica de respuesta rápida, desacoplamiento de subsistemas, alta densidad de potencia, bajo costo de fabricación y facilidad de miniaturización; en contraparte, se resalta la inclusión de no linealidades, debido a la naturaleza del mecanismo de accionamiento y el material hiperelástico empleado en la fabricación.

Las características generales presentadas en el párrafo anterior permiten al cuerpo del endoscopio interactuar de forma segura y compatible en el interior del paciente, donde el entorno de operación se caracteriza por tener espacios no estructurados y reducidos, siendo en ese sentido necesario que el cuerpo del endoscopio presente una alta capacidad de deformación e infinitos grados de libertad de movimiento (Cianchetti M. et al, 2018) que asegure su navegación al interior de las cavidades humanas. Las ventajas anteriores, en contraparte, exigen un reto en el planteamiento y desarrollo de técnicas de control de posición. Dentro de los retos más relevantes, se encuentra la reducida capacidad de accionamiento forzado, debido a la presencia de infinitos grados de libertad en el cuerpo del endoscopio; la dificultad de implementar sensores y medir de manera completa toda la configuración espacial del cuerpo endoscopio, debido a la existencia de infinitos puntos de deformación y un entorno de operación cerrado dentro del paciente; la presencia de redundancia en el accionamiento del endoscopio, debido a un mayor número de accionamiento independientes en manipulador respecto la cantidad de grados de libertad en la posición del efector final.

1.3. Espacios de control de un manipulador blando

En un manipulador blando (robot blando), como en este caso, se definen generalmente tres espacios de control. Cada espacio queda definido por un conjunto de variables que describen el comportamiento del manipulador en el tiempo. La Figura 1.2 muestra una representación gráfica de los tres espacios de control, los cuales se describirán a continuación. El espacio de uniones o actuadores se define a través de las variables que pueden ser manipuladas para el control de posición del cuerpo del endoscopio como la longitud y las presiones neumáticas de las cámaras internas del endoscopio. El espacio de configuración espacial describe la deformación de la columna vertebral del endoscopio a través de variables geométricas como el radio de curvatura, ángulo del plano de flexión, longitud de curvatura, etc. El espacio de tareas define las posiciones cartesianas y las orientaciones espaciales de una posición de interés del cuerpo del endoscopio, el cual es usualmente la ubicación del efector final (Thuruthel, T., et al, 2018). La dimensión de los espacios de control anteriores; es decir, el

número de variables que se define en cada espacio dependerá del diseño del cuerpo del endoscopio y las técnicas de modelamiento y control seleccionadas de acuerdo a los objetivos de control. Además, la Figura 1.2 representa dos mapeos o transformaciones f existentes entre cada par de espacios de control. El primer mapeo f_{esp} entre el espacio de uniones/actuadores y configuración espacial es específico; es decir, depende del diseño del cuerpo del endoscopio; por ejemplo, no es igual la relación existente en un endoscopio accionado por la deformación de un mecanismo de tendones que otro accionado por la deformación de cámaras internas del cuerpo. El segundo mapeo f_{ind} entre el espacio de configuración espacial y el espacio de tareas es independiente; por ejemplo, la relación geométrica entre la configuración espacial del cuerpo del endoscopio y las posiciones del efector final no depende del diseño del endoscopio. El mapeo global desde el espacio de uniones/actuadores al espacio de tarea define la cinemática directa y en sentido contrario define la cinemática inversa.

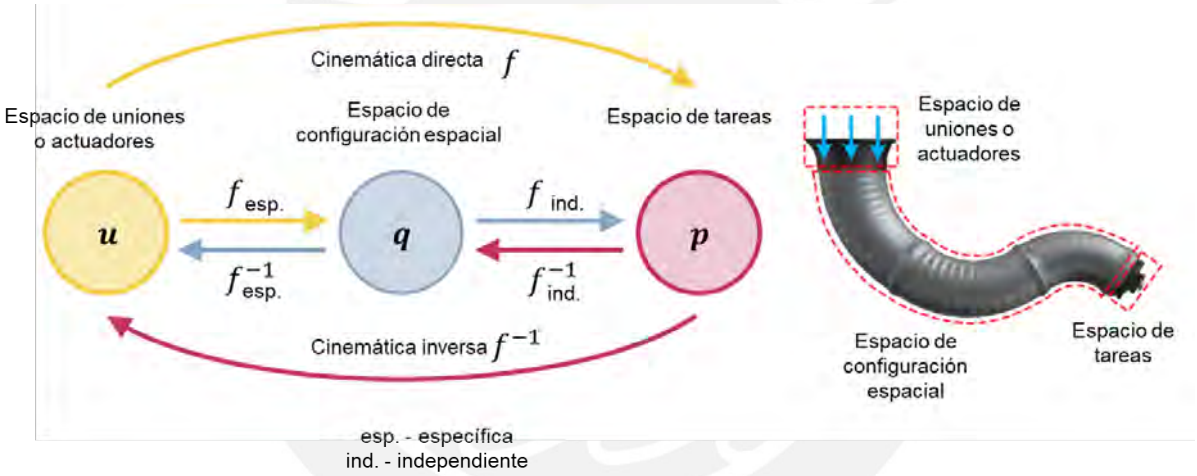


Figura 1.2. Espacio de control en el cuerpo del endoscopio blando. Fuente: Adaptado de (Wang X., et al., 2021).

1.4. Modelamiento del endoscopio blando

El modelamiento del cuerpo de un endoscopio blando puede estar basado en modelos analíticos y experimentales. Independiente a ello, el modelo define la cinemática directa del sistema; es decir, los mapeos existentes entre los tres espacios de control mencionados anteriormente. En el presente apartado se presenta y analiza los modelos estudiados para predecir el comportamiento dinámico del cuerpo del endoscopio.

El modelo de radio de curvatura constante es un modelo analítico ampliamente utilizado para describir el comportamiento de los manipuladores blandos. La suposición del modelo se basa

en asumir que la deformación del cuerpo del endoscopio se puede describir por una curva de radio constante a lo largo de toda la columna vertebral del manipulador o a lo largo de la longitud de una sección. En ese sentido, en el espacio de configuración espacial se definen como variables al radio de curvatura (r), la curvatura (K), la longitud de arco (l) y el ángulo del plano de flexión ($\emptyset$) como se muestra en la Figura 1.3. Debemos notar que es común definir el plano X-Z para un ángulo de flexión igual a cero, donde el eje vertical corresponde al eje z y el plano X-Y coincide con la base del cuerpo del manipulador.

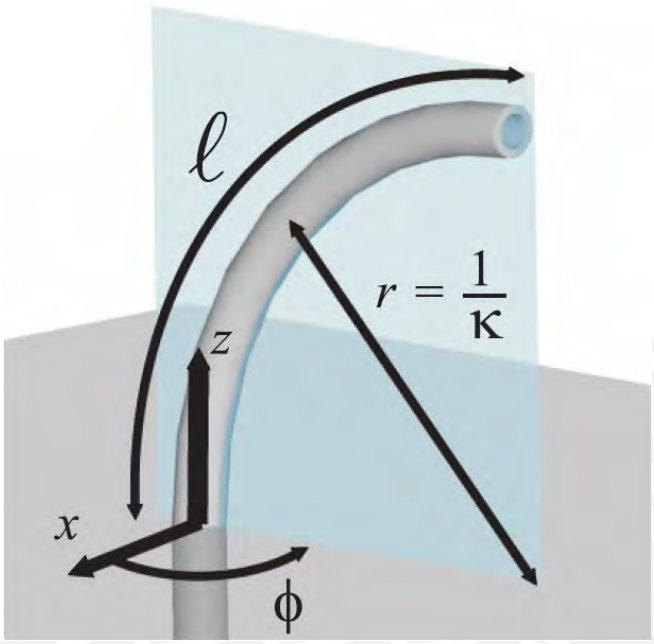


Figura 1.3. Variable del espacio en la configuración espacial del modelo de radio de curvatura constante. Fuente: (Webster, et al.,2016)

Tomando en cuenta las variables anteriores, se pueden definir los dos mapeos entre los espacios de control del manipulador. El mapeo entre el espacio de uniones/actuadores hacia el espacio de configuración espacial es un mapeo dependiente y el mapeo entre este último con el espacio de tarea es un mapeo independiente. El primer mapeo al ser dependiente está sujeto al diseño del cuerpo del manipulador blando mientras que el segundo mapeo no depende de ello. Respecto a las variables en cada espacio de control, en el de uniones/actuadores, se define como variables las longitudes en cada una de las cámaras internas del cuerpo del endoscopio. Donde típicamente la dimensión del espacio es de tres o cuatro, debido a la presencia de la misma cantidad de cámaras internas en el cuerpo de un endoscopio dándole una capacidad de movimiento espacial (omnidireccional). En el espacio de tareas, se define como variables las posiciones cartesianas y ángulos de orientación del

efector final del cuerpo del endoscopio; en ese sentido, la dimensión máxima posible será de seis considerando la descripción completa de todos los grados de libertad.

Los resultados o las ecuaciones derivadas del modelo de radio de curvatura constante definen el mapeo dependiente entre la longitud de arco (l), el ángulo del plano de flexión ($\emptyset$) y la curvatura (k) respecto a la longitud interna de cada cámara (l_i), donde $i \in \{1, 2, 3, 4\}$ y d es el diámetro externo del endoscopio, las cuales se muestran en las ecuaciones (1.1), (1.2) y (1.3) respectivamente. El mapeo independiente, también conocido como la matriz de transformación homogénea (T_w), donde s es igual a longitud de arco (l) se muestra en la ecuación (1.4). Las ecuaciones, del (1.1) al (1.3), derivan del análisis geométrico considerando las suposiciones del modelo de radio de curvatura constante y la matriz de transformación homogénea (T_w), puede ser obtenida por varios métodos como Denavit-Hartenberg, Frenet-Serret, etc (Webster R., et al, 2010).

$$l(q) = \frac{l_1 + l_2 + l_3 + l_4}{4} \dots (1.1) \quad \emptyset(q) = \tan^{-1}\left(\frac{l_4 - l_2}{l_3 - l_1}\right) \dots (1.2)$$

$$k(q) = \frac{(l_1 - 3l_2 + l_3 + l_4)\sqrt{(l_4 - l_2)^2 + (l_3 - l_1)^2}}{d(l_1 + l_2 + l_3 + l_4)(l_4 - l_2)} \dots (1.3)$$

$$T_W = \begin{bmatrix} \cos^2(\emptyset)\,(\cos(ks) - 1) + 1 & \sin(\emptyset)\cos(\emptyset)(\cos(ks) - 1) & \cos(\emptyset)\sin(ks) & \frac{\cos(\emptyset)(1 - \cos(ks))}{k} \\ \sin(\emptyset)\cos(\emptyset)(\cos(ks) - 1) & \cos^2(\emptyset)\,(1 - \cos(ks)) + \cos(ks) & \sin(\emptyset)\sin(ks) & \frac{\sin(\emptyset)(1 - \cos(ks))}{k} \\ -\cos(\emptyset)\sin(ks) & -\sin(\emptyset)\sin(ks) & \cos(ks) & \frac{\sin(ks)}{k} \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (1.4)$$

Los modelos experimentales se basan en el uso de un conjunto de datos (Kim D., et al, 2021), dentro de los cuales se tiene como alternativa ser obtenidos a partir de pruebas experimentales o simulaciones donde se fuerza al cuerpo del endoscopio blando alcanzar diferentes posiciones en todo su rango de operación. El requerimiento principal para la primera alternativa es disponer de al menos un prototipo físico donde se pueda ensayar las pruebas, mientras que para la segunda alternativa se debe disponer de un modelo digital 3D del cuerpo del endoscopio que pueda ser simulado usando un software FEM (Finite Element Method - Análisis de Elementos Finitos) como Abaqus u otro (Wang X., et al, 2021). Por otro lado, la primera alternativa presenta como ventaja la obtención de datos del sistema a partir de un entorno de operación real, donde se considera el efecto de diversos factores de desviación respecto al comportamiento de un modelo digital 3D como los errores en la fabricación del cuerpo del endoscopio, la dinámica de sensores y actuadores y perturbaciones en el entorno de operación, los cuales no son incluidas completamente en las simulaciones.

Una estrategia para la recolección de datos que permita aprovechar las ventajas de cada alternativa para la obtención del modelo es hacer uso de los datos por etapas; es decir, considerar en primer lugar los resultados recolectados por simulaciones y en segundo lugar lo obtenido a través de las pruebas experimentales.

El objetivo de los datos generados es ser usados en el entrenamiento del modelo basado en redes neuronales, el cual permita relacionar las variables del espacio de actuadores con las correspondientes al espacio de configuración espacial o tareas. Dentro de los modelos basados en redes neuronales para el modelamiento de sistemas existen diferentes estructuras, cuyo arreglo fundamental consiste en una capa de neuronas de entradas, una o más capas de neuronas intermedias y una capa de neuronas de salidas como se muestra en la Figura 1.4. Las relaciones de conexión entre las neuronas de capas contiguas se definen a través de los coeficientes de pesos, como por ejemplo los coeficientes v_{ij} y w_{jk}, los cuales definen el comportamiento de la red neuronal y sus respectivos valores son obtenidos durante la etapa de entrenamiento del modelo a partir de los datos disponibles. A su vez, cada neurona de las diferentes capas define un comportamiento entre su entrada y salida a través de una función de activación, $f_j(m_j)$ de correspondencia no lineal, donde las más usadas son las funciones sigmoideas y gaussianas. A partir de la estructura descrita, es posible definir una relación matemática entre las variables de entrada y de salida de la red neuronal, las cuales se definen a través de las ecuaciones (1.5), (1.6) y (1.7).

Considerando la estructura base de una red neuronal su entrenamiento se plantea como un problema de optimización donde se busca minimizar el error existente entre la variable de salida medida ($y1_i^*, y2_i^*$) del sistema a modelar y la estimada por la red neuronal ($y1_i, y2_i$) encontrando los valores óptimos de los pesos de los coeficientes de conexión entre las neuronas (v_{ij} y w_{jk}) e incluso los valores óptimos de las pendientes (a_j) y los centros (c_j) correspondientes a las funciones de activación (f_j), que se pueden definir por las ecuaciones (1.8), (1.9) o (1.10), para cada una de las neuronas de la capa intermedia. El planteamiento matemático, de lo anteriormente mencionado, se muestra en la Figura 1.5, donde se considera el método de la gradiente descendente para la obtención de los valores de v_{ij} y w_{jk} que minimicen la función de costo J definida como el error cuadrático entre las salidas calculadas por el modelo ($y1_i, y2_i$) y las salidas deseadas ($y1_i^*, y2_i^*$). El algoritmo a utilizar para el cálculo de las derivadas parciales ($\frac{\partial J}{\partial v_{ij}}, \frac{\partial J}{\partial w_{jk}} \frac{\partial J}{\partial a_j} \frac{\partial J}{\partial c_j}$) de la función de costo J respecto a cada uno de los parámetros (v_{ij}, w_{ij}, a_i y c_i) dependerá del tipo de estructura de la red neuronal seleccionada para modelar el sistema como se presentará más adelante.

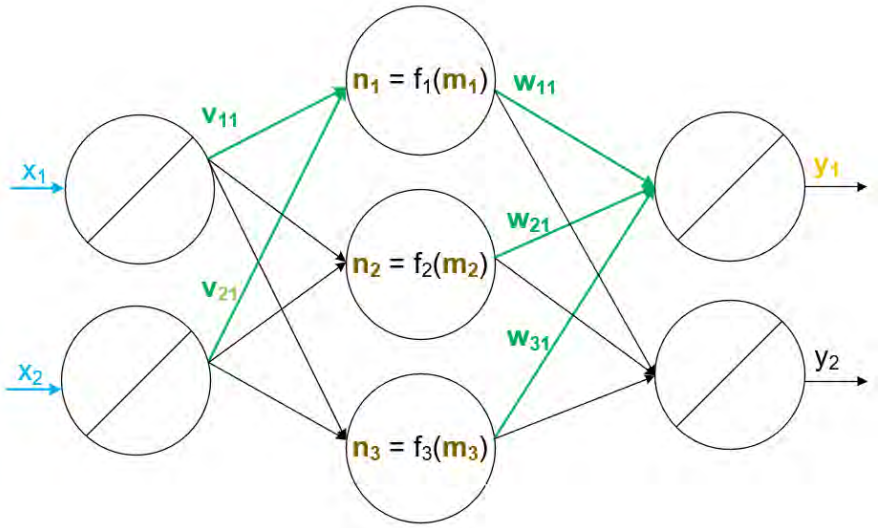


Figura 1.4. Estructura y ecuaciones básicas de una red neuronal

$$m_j = \sum_{i=1}^{I} v_{ij} * x_i \,, \forall j = 1 \dots J \dots (1.5)$$

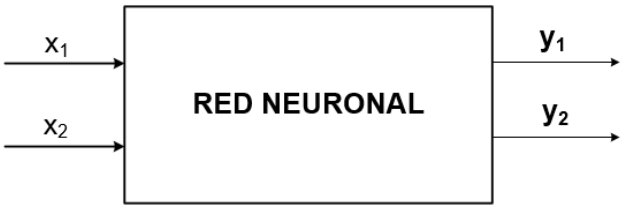
$$n_j = f_j(m_j) \,, \forall j = 1 \dots J \dots (1.6)$$

$$y_k = \sum_{j=1}^{J} w_{jk} * n_j \,, \forall k = 1 \dots K \dots (1.7)$$

Sigmoidea Tipo 1: $n_j = \frac{1}{1+e^{-\left(\frac{m_j - c_j}{a_j}\right)}} \dots (1.8)$

Sigmoidea Tipo2: $n_j = \frac{2}{1+e^{-\left(\frac{m_j - c_j}{a_j}\right)}} - 1 \dots (1.9)$

Gaussiana: $n_j = e^{-\left(\frac{m_j - c_j}{a_j}\right)^2} \dots (1.10)$



DATOS DE ENTRENAMIENTO

x1	x2	y1*	y2*
$x1_1$	$x2_1$	$y1^*_1$	$y2^*_1$
$x1_1$	$x2_2$	$y1^*_2$	$y2^*_2$
…	…	…	…
$x1_{n-1}$	$x2_{n-1}$	$y1^*_{n-1}$	$y2^*_{n-1}$
$x1_n$	$x2_{n-1}$	$y1^*_n$	$y2^*_n$

i) Función de costo:

$$J = J_1 + \dots + J_n$$

$$J = 0.5(y1_1 - y1_1^*)^2 + 0.5(y2_1 - y2_1^*)^2 + \dots + 0.5(y1_i - y1_i^*)^2 + 0.5(y2_i - y2_i^*)^2$$

ii) Cálculo de las derivadas parciales:

$$\frac{\partial J}{\partial v_{ij}} = \frac{\partial J_1}{\partial v_{ij}} + \dots + \frac{\partial J_n}{\partial v_{ij}}, \frac{\partial J}{\partial w_{jk}} = \frac{\partial J_1}{\partial w_{jk}} + \dots + \frac{\partial J_n}{\partial w_{jk}}, \frac{\partial J}{\partial a_j} = \frac{\partial J_1}{\partial a_j} + \dots + \frac{\partial J_n}{\partial a_j}, \frac{\partial J}{\partial c_j} = \frac{\partial J_1}{\partial c_j} + \dots + \frac{\partial J_n}{\partial c_j}$$

iii) Actualización de pesos (método gradiente descendente):

$$v_{ij} = v_{ij} - \eta \frac{\partial J}{\partial v_{ij}}, w_{jk} = w_{jk} - \eta \frac{\partial J}{\partial w_{jk}}, a_j = a_j - \eta \frac{\partial J}{\partial a_j}, c_j = c_j - \eta \frac{\partial J}{\partial c_j}$$

Figura 1.5. Planteamiento matemático para el entrenamiento de la red neuronal

Dentro del modelamiento de sistemas usando redes neuronales se puede considerar dos grandes tipos de estructuras la red neuronal recurrente (o dinámica) y la red neuronal estática (o directa), donde la diferencia principal entre las dos estructuras es la presencia o ausencia de realimentación a la salida del modelo de la red neuronal hacia su entrada respectivamente como se muestra en la Figura 1.6. Para el caso de la red neuronal recurrente sí se considera en su estructura la realimentación, en cambio para la red neuronal estática no se considera realimentación (Daekyum K., et al, 2021). Las ecuaciones generales que describen el comportamiento mencionado para cada tipo de estructura de red neuronal se muestran en el lado derecho de la Figura 1.6.

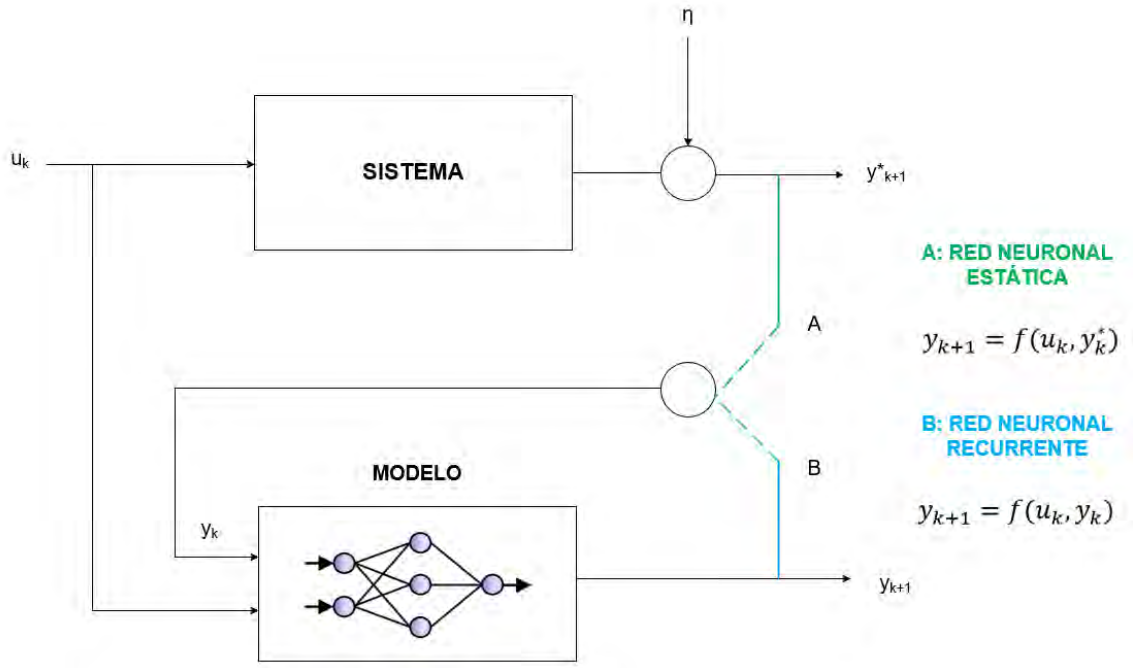


Figura 1.6. Comparación entre la estructura de una red neuronal estática y recurrente

En el caso de la red neuronal estática, debe considerarse la creación de una dinámica artificial que permita modelar al sistema dinámico; para ello es posible incorporar tanto las entradas (u) como las salidas (y) del sistema actuales (u_k, y_k) y pasadas $(u_{k-1} \dots u_{k-m-1}, y_{k-1} \dots y_{k-n-1})$, cuya representación se muestra en la Figura 1.7. Debido a la naturaleza estática de la red neuronal, en la etapa de entrenamiento se utiliza el algoritmo de retropropagación de errores para el cálculo de las derivadas parciales $(\frac{\partial J}{\partial v_{ij}}, \frac{\partial J}{\partial w_{jk}})$ de la función de costo J, donde las ecuaciones a considerar se muestran en el lado derecho de la Figura 1.7 (Werbos P.,1974). En las ecuaciones, $\bar{e}_1 \dots \bar{e}_m$ representan los errores retropropagados correspondientes a las neuronas de la capa intermedia, e representa el error retropropagado correspondiente a la neurona de la capa de salida y $f'(m_1) \dots f'(m_m)$ representan las derivadas de las funciones de activación de las neuronas de la capa intermedia.

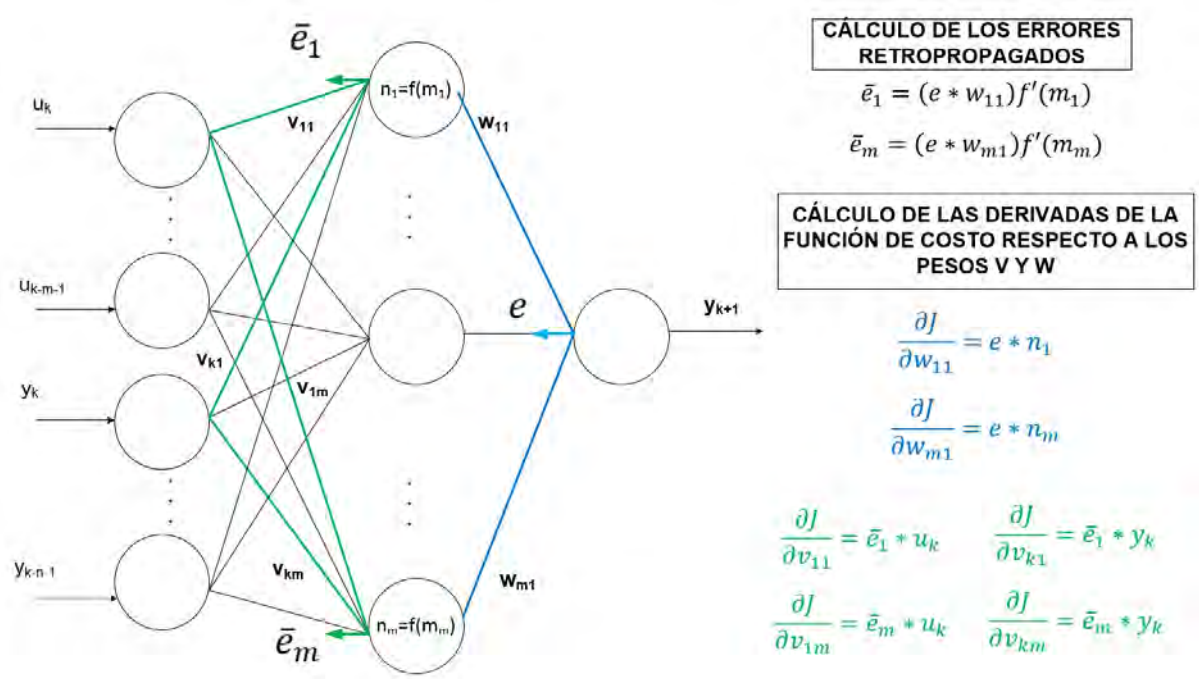


Figura 1.7. Red neuronal estática y algoritmo de propagación de errores

En el caso de la red neuronal dinámica, debe considerarse el efecto de la realimentación de la salida de la red neuronal hacia su entrada, para una comprensión clara de ello se realiza el despliegue temporal de la red neuronal representado en la Figura 1.8, donde la entrada de la red neuronal de la etapa siguiente corresponde a la salida de la red de la etapa anterior; por ejemplo, y_1 entre la etapa 1 y la etapa 2. Para el cálculo de las derivadas parciales de la función de costo J respecto a los coeficientes de conexión v_{ij} y w_{jk}, se requiere obtener las derivadas parciales de las salidas correspondiente a cada etapa; por ejemplo, $\frac{\partial y_1}{\partial w_{ij}}$ en la etapa 1 y $\frac{\partial y_2}{\partial w_{ij}}$ en la etapa 2 como se muestran en la parte inferior de la Figura 1.8. Se puede notar que numéricamente la derivada parcial de las salidas de la red neuronal en cada etapa respecto a los coeficientes de los pesos depende no solamente de la derivada parcial en su correspondiente etapa sino también de las derivadas parciales de las etapas anteriores, esto debido a la dependencia temporal en el comportamiento de la red neuronal dinámica. Para el entrenamiento de la red neuronal, se tiene como primera alternativa el uso del algoritmo de retropropagación de errores a través del tiempo (Back Propagation Through Time - BPTT por sus siglas en inglés) y como segunda alternativa el algoritmo de retropropagación dinámica (Dynamic Back Propagation - DBP por sus siglas en inglés), siendo este último algoritmo más eficiente en memoria al ser naturaleza recursiva respecto al primer algoritmo (Jin, Y. et al, 1993). El resultado del algoritmo DBP recursivo se muestra en la Figura 1.8, donde el cálculo de la derivada parcial total ($\overline{\frac{\partial y_{k+1}}{\partial w}}$) se requiere el cálculo derivada parcial en la etapa actual,

$\frac{\partial y_{k+1}}{\partial w}$, la derivada parcial total de la etapa anterior, $\frac{\overline{\partial y_k}}{\overline{\partial w}}$, y la gradiente de la salida de la red neuronal respecto a la salida de la etapa anterior, $\frac{\partial y_{k+1}}{\partial y_k}$.

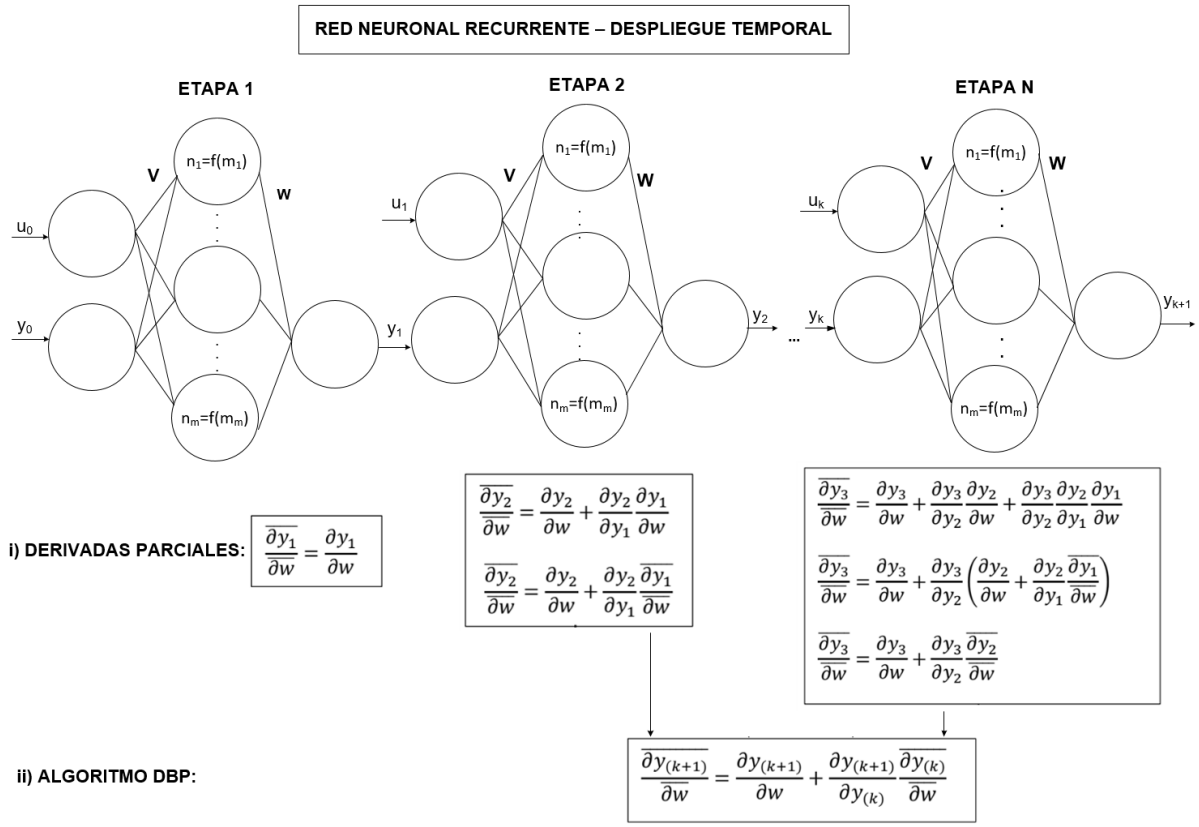


Figura 1.8. Estructura y despliegue temporal de la red neuronal dinámica y algoritmo dinámico de retropropagación

1.5. Técnicas de control

Existen diversas técnicas que se aplican para controlar la posición y orientación de un actuador blando accionado neumáticamente, dentro de las cuáles el diseño del controlador puede estar basado en el modelo del sistema o por el contrario no ser considerado. Además, como parte de la estrategia de control es posible considerar un lazo de control interno de presión o flujo neumático de las cámaras internas del actuador y un lazo de control externo de posición u orientación del cuerpo del actuador blando.

A pesar de la complejidad del modelado de los actuadores blandos, debido a la presencia de no linealidades originadas por el material hiperelástico utilizado en su fabricación y el mecanismo de accionamiento neumático, se han propuesto controladores bajo el enfoque convencional de diseño basado en el modelo matemático del sistema. Por ejemplo, la técnica de control de modelado de energía basado en el modelo dinámico de eslabones rígidos (Franco, E. et al, 2021) y la técnica de control de linealización exacta basado en el modelo de

curvatura constante y masas concentradas (Falkenhahn, V. et al, 2017). Sin embargo, los modelos teóricos mencionados requieren un alto cómputo y presentan una baja precisión lo que dificulta el adecuado diseño del controlador.

Teniendo en cuenta los inconvenientes presentados en el diseño de controladores basados en un modelo analítico, se han propuesto algoritmos de control cuyo diseño no están basado en el modelo del sistema. Por ejemplo, el controlador PID, donde la sintonización de sus respectivos parámetros se realiza manualmente utilizando el método de Ziegler – Nichols o de manera automática utilizando el algoritmo de descenso de coordenadas (Khan, A. et al, 2020). A parte del controlador PID clásico, se ha utilizado variantes como el PID con ganancia programada y el PID difuso, lo cual permite mejorar el desempeño del controlador adaptando sus respectivos parámetros con respecto a las posiciones deseadas. Sin embargo, el diseño de estos controladores al no estar basado en el modelo del sistema resulta difícil analizar su estabilidad y garantizar su convergencia.

Considerando las limitaciones de las dos propuestas anteriores, la alternativa del diseño del controlador basado en un modelo experimental se presenta como una solución bastante atractiva, en el cual el modelo es obtenido a partir de la medición de un conjunto de datos que describan el comportamiento de la dinámica del sistema. Diferentes técnicas de control están basadas en el modelo empírico de segundo orden de parámetros concentrados y sus respectivas variantes, que se describen a continuación. La versión más básica considera un modelo lineal de segundo orden, a partir cual se diseña un controlador de regímenes deslizantes (Skorina, E. et al., 2015). Con el objetivo de superar el modelado de las no linealidades del sistema, la cual no es considerada en el método de control anterior se plantean dos enfoques. El primer enfoque sintetiza las no linealidades considerando un rango de incertidumbres en los parámetros del modelo; a partir de ello, se diseña un controlador robusto utilizando el método backstepping (Wang, T. et al., 2018). Un segundo enfoque consiste en representar las no linealidades como perturbación del modelo, a partir del cual se diseña el controlador usando la técnica de regímenes deslizantes (Khan, AH. et al., 2020). Los controladores basados en los modelos empíricos descritos permiten analizar la estabilidad del sistema y garantizar su convergencia bajo las condiciones de diseño. Para considerar adicionalmente las incertidumbres del modelado y ampliar el alcance de operación del sistema, las cuales no son tomadas en cuenta en los métodos anteriores, se ha diseñado un controlador robusto adaptativo basado en el modelo lineal de segundo orden con incertidumbres en sus parámetros (Chen, C. et al., 2020). Así mismo, una versión mejorada del método anterior consiste en utilizar en el diseño del controlador un modelo no lineal de segundo orden, donde sus respectivos parámetros son dependientes de la posición del

sistema. Las técnicas de control presentadas basadas en el modelo de segundo orden, en general, presentan la limitación que están aplicadas a sistemas del tipo una entrada y una salida.

Otra técnica propone como estructura del controlador una red neuronal, la cual es entrenada utilizando el modelo del sistema. En la etapa de entrenamiento, se obtienen los parámetros de la red neuronal; por ejemplo, los pesos de conexión entre las capas de neuronas de entrada, salida e intermedia y los coeficientes de las funciones de activación (Centurelli, A., et al, 2022). Esta técnica si permite controlar un sistema con múltiples entradas y salidas. Debido a que las redes neuronales vienen siendo utilizado en diferentes campos de la ingeniería, en el presente trabajo se realiza un estudio de aplicación en la robótica blanda. A continuación, se presenta una descripción preliminar de algunos conceptos fundamentales respecto a esta técnica de control. El entrenamiento del controlador se puede plantear bajo un enfoque estático o dinámico, donde el objetivo en ambos casos es lograr que el sistema integrado (controlador más modelo) alcance el comportamiento deseado. En un entrenamiento estático se considera al controlador más el modelo del sistema en lazo abierto, en cambio en un entrenamiento dinámico se considera al controlador más el modelo del sistema en lazo cerrado, lo dos enfoques descritos se representa en la Figura 1.9.

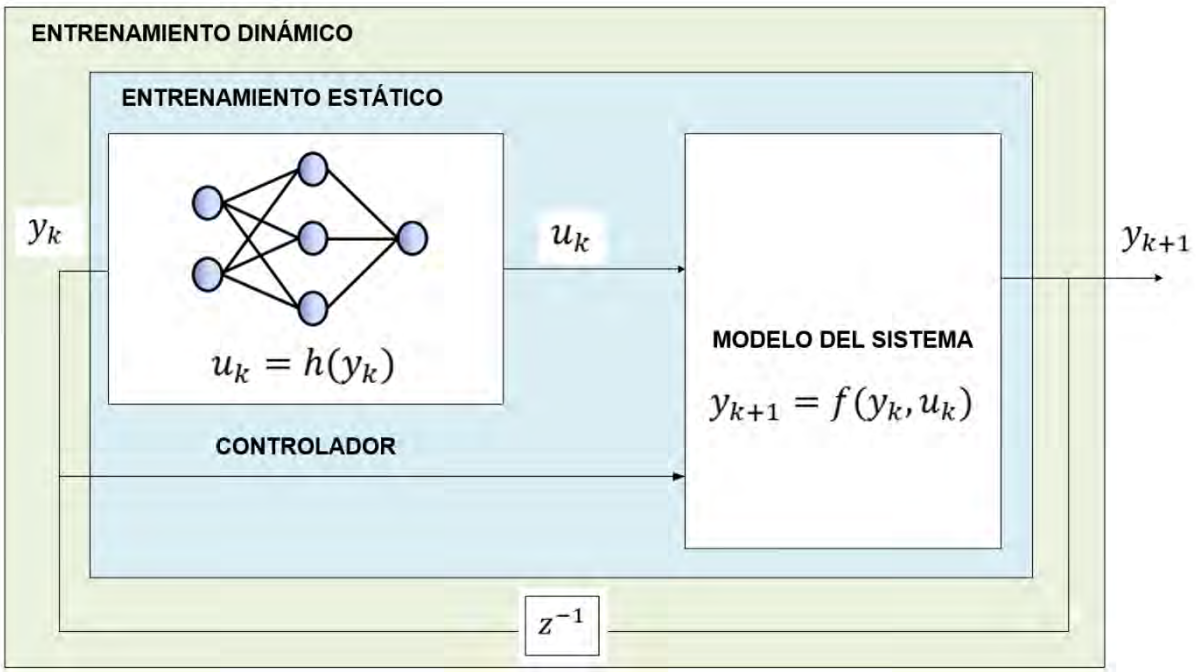


Figura 1.9. Entrenamiento del controlador bajo el enfoque estático y dinámico

El controlador obtenido bajo un enfoque de entrenamiento dinámico posee mejores prestaciones, ya que permite considerar la dinámica del sistema en lazo cerrado, así como el

control de sistema inestables; en cambio, el entrenamiento estático no considera la realimentación del sistema y no es posible estabilizar sistemas inestables. Para ambos tipos de entrenamiento, se busca minimizar la función de costo J definida como la suma de los errores cuadráticos entre la salida del modelo en lazo cerrado y_{k+1} con la salida deseada y^*_{k+1} más el esfuerzo de control u_k como se muestra en la ecuación (1.11).

$$J = \frac{1}{2}\sum_{k=0}^{N-1}(y_{k+1} - y^*_{k+1})^T Q(y_{k+1} - y^*_{k+1}) + u_k^T R u_k \ldots (1.11)$$

Donde:

$$y^*_{k+1} = A_m y_k$$

Respecto al cálculo de las derivadas parciales de la función costo $\frac{\overline{\partial J}}{\overline{\partial v}}$ y $\frac{\overline{\partial J}}{\overline{\partial w}}$ para determinar las matrices óptimas de conexión v y w de la red neuronal, en el caso del entrenamiento estático se requiere el cálculo de las derivadas simples $\frac{\overline{\partial y_{k+1}}}{\overline{\partial v}}$ y $\frac{\overline{\partial y_{k+1}}}{\overline{\partial w}}$, donde se hace uso del algoritmo de retropropagación de errores; en el caso del entrenamiento dinámico se requiere el cálculo de las derivadas totales $\frac{\overline{\partial y_{k+1}}}{\overline{\partial v}}$ y $\frac{\overline{\partial y_{k+1}}}{\overline{\partial w}}$, donde hace uso del algoritmo de retropropagación dinámica. Los resultados de los algoritmos para cada enfoque de entrenamiento se muestran en la Figura 1.10.

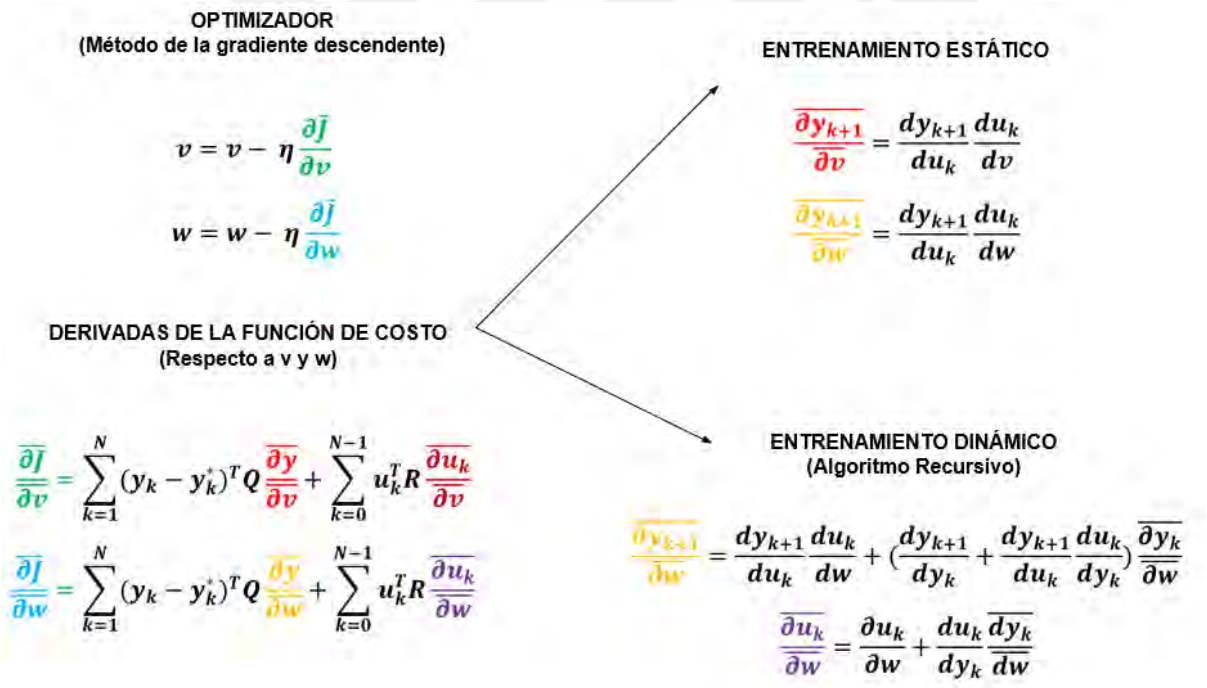


Figura 1.10. Algoritmos de entrenamiento estático y dinámico del controlador. Fuente: Propia.

Respecto a la estrategia de entrenamiento del controlador, debe considerarse un aprendizaje incremental, el cual consiste en ir seleccionado de manera progresiva posiciones iniciales cada más alejadas a la posición deseada.

1.6. Objetivo general y específicos

En la presente sección se listan el objetivo general y los objetivos específicos del presente trabajo, los cuales serán considerados durante el desarrollo de la presente tesis.

Objetivo general

Desarrollar un controlador de posición avanzado para un endoscopio blando empleado en cirugía laparoscópica para facilitar al cirujano la manipulación del instrumento al interior del paciente.

Objetivos específicos

Para alcanzar el objetivo es necesario realizar las siguientes actividades de investigación y desarrollo:

- Realizar el estudio del proceso y las tecnologías actuales en la cirugía laparoscópica asistido por endoscopio blando.
- Entender y describir el modelo que representa la cinemática/dinámica del endoscopio blando para el diseño del controlador.
- Validar el desempeño del modelo del endoscopio blando a través de simulaciones.
- Diseñar el controlador de posición avanzado del endoscopio blando.
- Validar el desempeño del controlador de posición diseñado a través de simulaciones considerando sus condiciones de operación.
- Analizar resultados y proponer mejoras a partir de los resultados obtenidos de la investigación.

Capítulo II: CARACTERÍSTICAS DEL ENDOSCOPIO BLANDO

2.1. Introducción

En el presente capítulo, se exponen las características técnicas relacionadas al cuerpo del endoscopio blando, el cual es el sistema bajo estudio en el presente trabajo. En ese sentido, para lograr una correcta comprensión, se empieza con la descripción de los criterios vinculados al diseño del cuerpo del endoscopio, así como el diseño mecánico resultante. Luego, se describe la propuesta de la instrumentación para el cuerpo del endoscopio, la cual es necesaria para la aplicación experimental de la metodología de modelamiento y control mostradas en los capítulos posteriores; además, se presentan la arquitectura del sistema, las especificaciones técnicas de los instrumentos para la medición, adquisición, control y regulación de las variables de operación del sistema, así como su respectivo esquema neumático. Finalmente, se realiza la descripción de las condiciones necesarias vinculadas al movimiento del cuerpo del endoscopio.

2.2. Criterios de diseño mecánico del cuerpo del endoscopio

La forma general de un endoscopio blando consiste en un cuerpo cilíndrico con cámaras y cavidades internas adicionales, las cuales son utilizadas tanto para el accionamiento interno del cuerpo del endoscopio y la instalación de uno o más componentes de tamaño reducido como una cámara o herramienta de corte, una representación de lo mencionado se muestra en el lado derecho de la Figura 2.1.

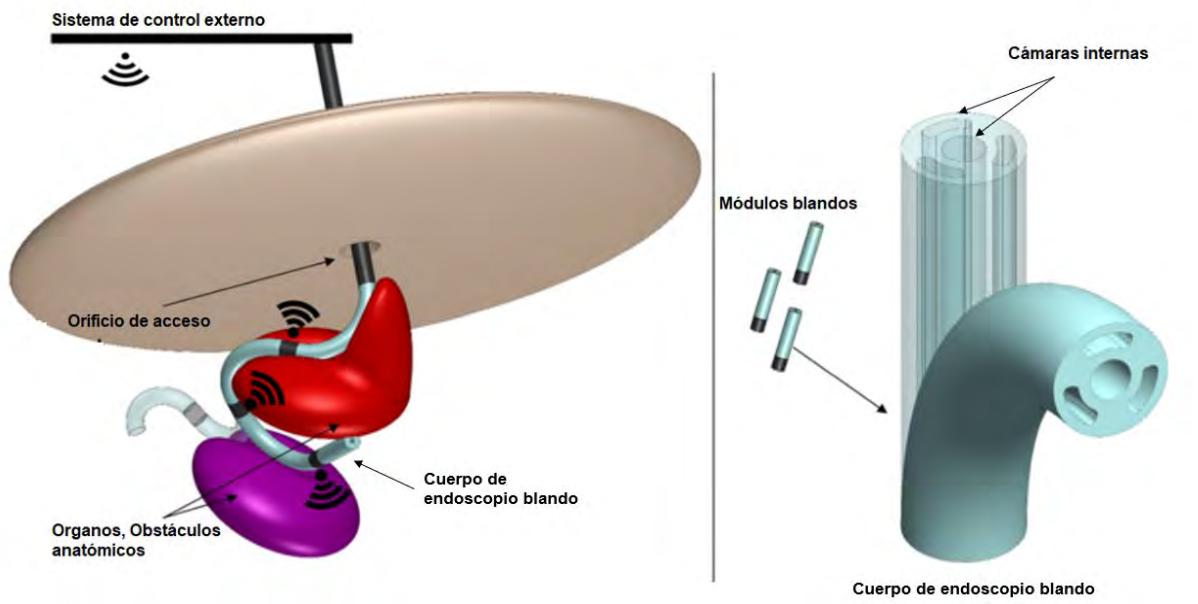


Figura 2.1. Cuerpo de endoscopio blando. Fuente: Adaptado de (Gerboni G., et al, 2015)

Debido a que la ubicación de trabajo del endoscopio en la cirugía laparoscópica es en el interior del vientre del paciente, la característica de su entorno de operación es no estructurado (dinámico o cambiante), siendo muy importante en ese sentido que el cuerpo del endoscopio posea la capacidad de tener un desplazamiento omnidireccional y asegurar una completa cobertura en diferentes zonas del cuerpo, como se muestra en lado izquierdo de la Figura 2.1. Lo anterior está relacionado con las siguientes especificaciones de diseño: limitación de la deformación radial del cuerpo del endoscopio, número de cámaras internas para el accionamiento interno y el diseño volumétrico de las correspondientes cámaras internas, las cuáles serán detalladas a continuación (Lenssen J., et al, 2019).

El cuerpo del manipulador al estar fabricado de un material hiperelástico y al ser accionado neumáticamente de manera interna a través de sus cámaras sufre deformaciones en diferentes direcciones espaciales; sin embargo, funcionalmente se requiere limitar las cámaras solamente a deformaciones longitudinales y restringir todo tipo de deformación radial. Lo anterior permite tener la capacidad de controlar el movimiento de flexión y longitudinal del cuerpo del endoscopio y con ello alcanzar diferentes posiciones en el efector final del cuerpo regulando las presiones en las cámaras internas del sistema hasta alcanzar una determinada posición espacial. Para lograr ello, existe la posibilidad de cubrir de manera interna y/o externa el cuerpo del endoscopio con un material de mayor módulo de elasticidad como por ejemplo una tela, fibra, hilo o malla metálica. Respecto a la ubicación del recubrimiento, se puede considerar en el diseño realizarlo alrededor de la superficie cilíndrica externa del cuerpo del endoscopio y/o en la superficie interna de cada una de sus respectivas cámaras.

Figura 2.2. Deformación radial del endoscopio: Adaptado de (Gerboni G., et al, 2015)

La Figura 2.2 muestra una comparación del efecto de la presencia y ausencia de recubrimiento externo en el cuerpo del endoscopio respecto a la deformación radial, la cual no es deseada por las razones anteriormente detalladas.

Normalmente, por un tema de seguridad durante la operación del endoscopio el rango promedio de operación de la presión neumática en cada una de cámaras internas está entre 0 a 2 bar; además, las dimensiones externas típicas promedios del cuerpo cilíndrico presentan un diámetro y longitud de 12 mm y 50 mm respectivamente. Bajo las restricciones anteriores, la distribución, el número, el tamaño y la forma de la sección de cada una de las cámaras internas debe permitir cubrir todo el rango de posiciones de trabajo del efector final del endoscopio. La distribución más común es la radial, donde las cámaras se ubican de forma concéntrica alrededor del centro geométrico de la sección del endoscopio. Para alcanzar no solamente movimientos en el plano para el efector final, sino en todo el espacio de trabajo (omnidireccional) es necesario definir al menos tres cámaras internas en el cuerpo del endoscopio. Un número de tres o cuatro cámaras son valores comúnmente usados, debido a que tener una mayor cantidad de cámaras significa ocupar un mayor espacio volumétrico y con ello reducir el espacio para la ubicación de otros dispositivos como una cámara u otro instrumento. Respecto al tamaño de la sección, un mayor tamaño permite reducir el rango de presión de trabajo a costa de una reducción en el espacio volumétrico disponible en el cuerpo del endoscopio y viceversa. Respecto a la forma de la sección, es típico definir la forma de media luna o mitad de circunferencia, lo cual permite una deformación homogénea de las cámaras cuándo se encuentran presurizadas (Elsayed Y., et al, 2014). La Figura 2.3 muestra una comparativa respecto a los criterios mencionados entre un diseño de sección con tres y cuatro cámaras internas. En la imagen (A), se muestra una sección con tres cámaras internas sin presurizar y en la imagen (B) presurizada; de la misma manera, se muestra los escenarios correspondientes para una sección con cuatro cámaras en las imágenes (C) y (D). Los vectores $\overrightarrow{r_1}$ y $\overrightarrow{r_2}$, mostrados en la Figura 2.3, representan la dirección de las fuerzas resultantes correspondientes a las cámaras neumáticas presurizadas. A continuación, se analiza el efecto del diseño en el movimiento de flexión del cuerpo endoscópico considerando dos cámaras presurizadas en simultáneo. El resultado se muestra en las imágenes (B) y (D), donde se muestra el vector de fuerza de flexión resultante ($\overrightarrow{r_1}$+$\overrightarrow{r_2}$) en la sección, donde se observa que la magnitud es mayor en una configuración de cuatro cámaras, debido a la propia distribución radial de los centros geométricos de las cámaras. Como resultado de ello, se obtiene una mayor longitud de deformación y ángulo de flexión en una configuración de cuatro cámaras considerando una misma cantidad de cámaras activas y una misma presión. Lo anterior, permite considerar la elección de la cantidad de cámaras en función al rango de deformación requerido por el cuerpo del endoscopio dentro de su entorno de operación.

El diseño volumétrico de las cámaras internas es otro parámetro a tomar en consideración, ya que permite optimizar el espacio volumétrico, así como ajustar el rango de deformación en el cuerpo del endoscopio. Dentro de este criterio, se define la longitud, así como el perfil de la sección a lo largo del eje de cada cámara interna del endoscopio. La reducción del espacio volumétrico en cada cámara de accionamiento permite tener mayor disponibilidad de espacio para agregar componentes adicionales, así como aumentar la rigidez del sistema y en consecuencia reducir su rango de deformación y movimiento en el efector final. Normalmente, el diseño de la sección es de área constante; sin embargo, se podría proponer y ajustar lo anterior de acuerdo al requerimiento de la aplicación. En la Figura 2.4, se puede observar tres propuestas de diseño volumétrico de las cámaras internas del endoscopio, de sección uniforme (A), de perfil parabólico (B) y sección constante por tramos (C).

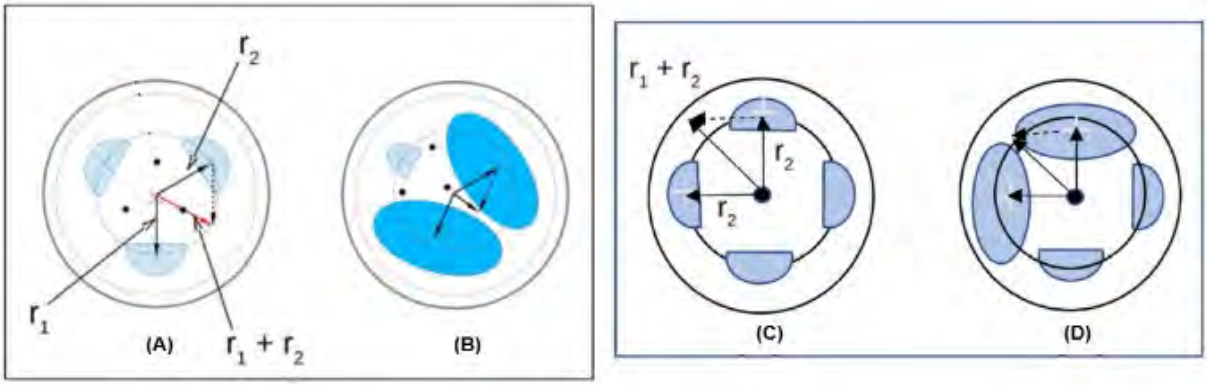


Figura 2.3. Comparativa del diseño de la sección del cuerpo del endoscopio de tres y cuatro cámaras. Fuente: (Frás J., et al., 2015)

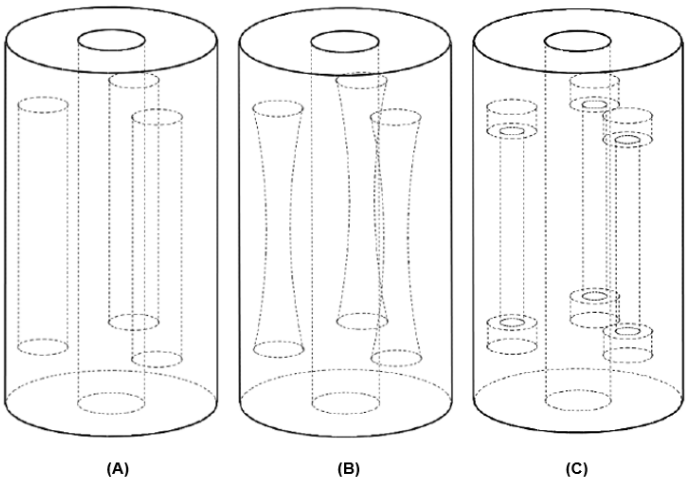


Figura 2.4. Propuestas de diseño volumétrico en el cuerpo de un endoscopio con cuatro cámaras internas. Fuente: (Lenssen J., et al., 2019)

2.2. Diseño mecánico

El diseño mecánico del cuerpo del endoscopio blando (Connolly F., et al, 2016), mostrado en la Figura 2.5., se define considerando los criterios de diseño descritos en la sección anterior. Las partes principales del cuerpo son el cuerpo medio (1), la base (2) y la tapa (3). Todo el cuerpo cilíndrico posee un diámetro de 10 mm y una longitud de 60 mm. A lo largo de todo el eje del endoscopio, presenta un agujero de 2.5 mm de diámetro, donde se puede alojar un pequeño dispositivo como una cámara, un sensor de flexión o alguna otra herramienta en particular. El cuerpo medio del endoscopio presenta cuatro cámaras con sección transversal uniforme de media circunferencia con un diámetro de 2 mm. El material utilizado para la fabricación del cuerpo es el caucho Ecoflex 00-50.

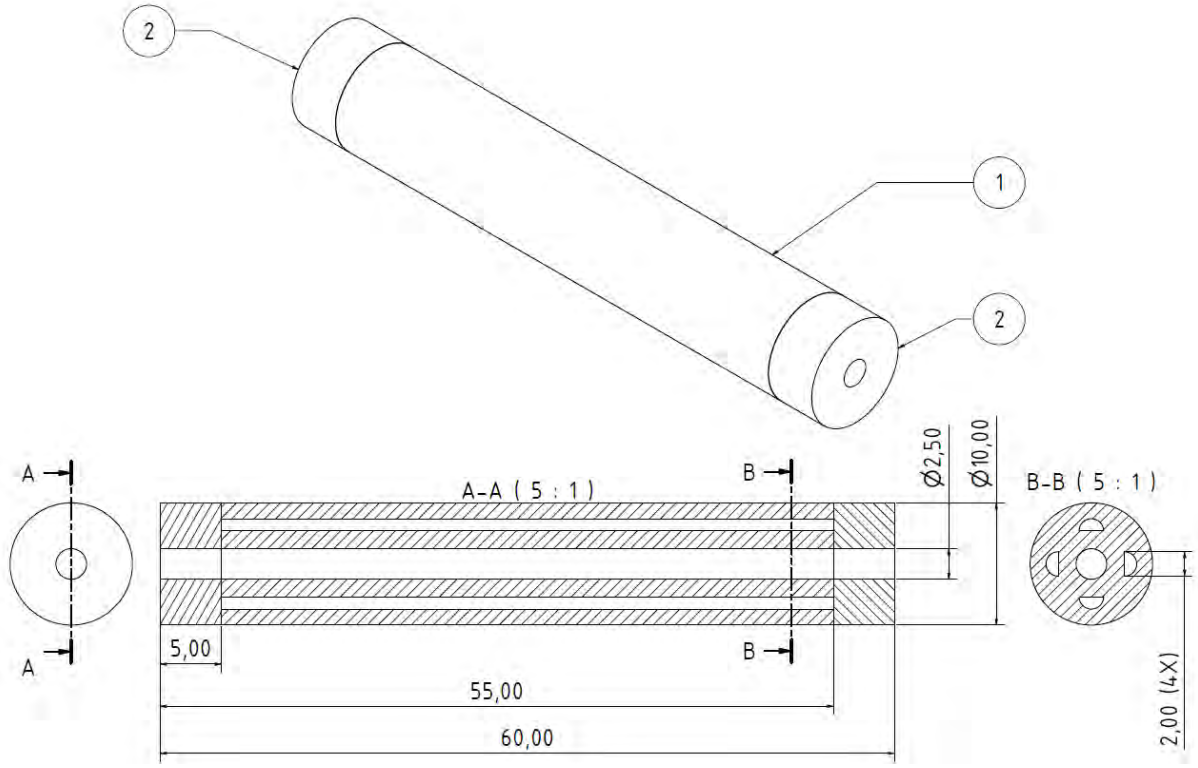


Figura 2.5. Diseño mecánico del cuerpo del endoscopio blando

2.3. Instrumentación

En la presente sección, se presenta la propuesta de la instrumentación para el cuerpo del endoscopio blando, en la cual se describen las características técnicas principales de los sensores, actuadores y tarjetas electrónicas que pueden ser consideradas para su implementación. Esta propuesta es compatible con la metodología para la obtención del modelo y el diseño del controlador que se presentarán en los capítulos 3 y 4. En la Figura 2.6, se observa el cuerpo del endoscopio blando y la instrumentación necesaria para el control de posición. En el efector final del cuerpo para medir su posición, se instala un sensor magnético

(6), el cual funciona en conjunto con un módulo generador de campo magnético y módulo de procesamiento y adquisición de datos (7) ubicados al entorno del cuerpo del endoscopio. El cuerpo es accionado neumáticamente a través de la deformación de sus cuatro cámaras internas, donde cada línea de aire es medida por un sensor de presión (3) y regulada por una electroválvula proporcional (5). Las variables medidas son leídas por una tarjeta electrónica programable Arduino Due (1), donde se procesan y obtienen las señales de control para las electroválvulas proporcionales. Es necesario contar con una etapa de potencia (2) para la conexión entre la placa Arduino Due y las electroválvulas.

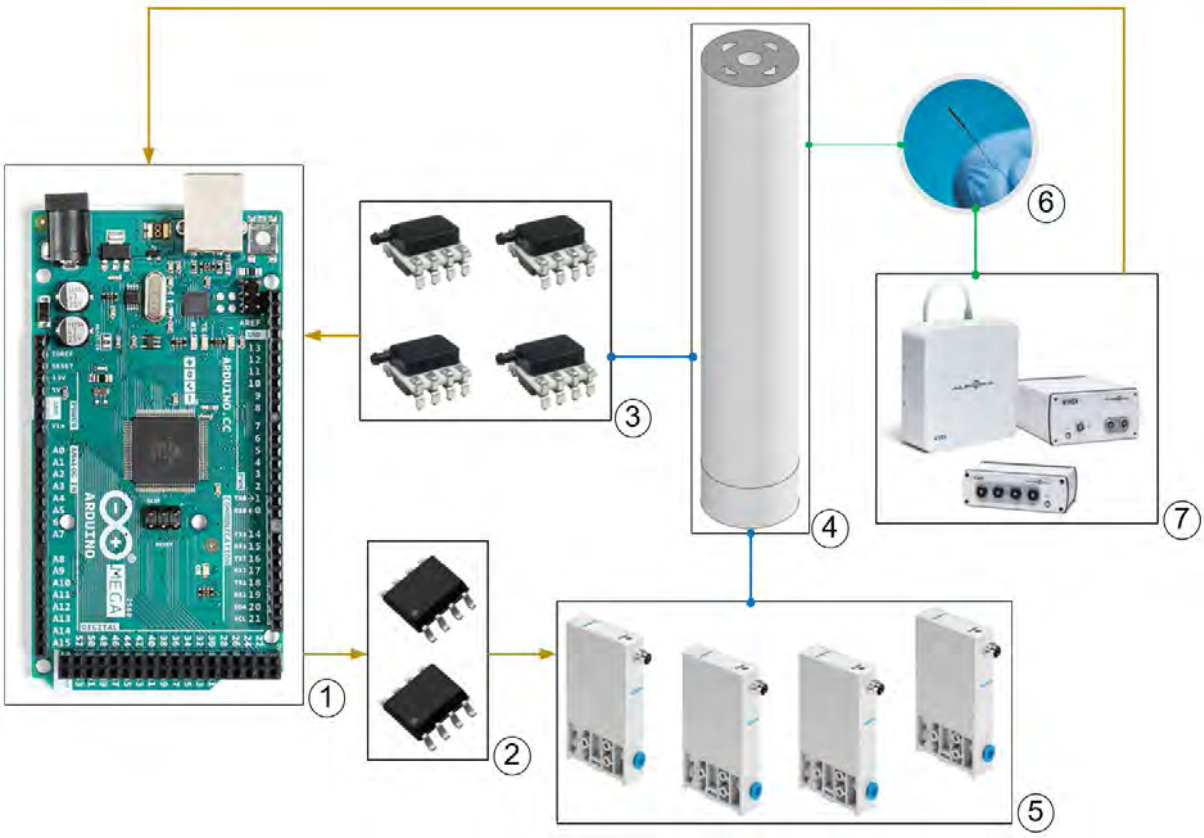


Figura 2.6 Arquitectura del prototipo de endoscopio blando

A continuación, se describen las características técnicas más importantes de los componentes anteriormente mencionados.

El sistema de medición de posición del efector final permite obtener sus respectivas coordenadas de posición (x, y). El principio de medición se basa en la lectura y análisis de las corrientes inducidas en el sensor magnético, instalado en el cuerpo del endoscopio, al desplazarse dentro un volumen de campo magnético. Para ello, el sistema cuenta con tres módulos (generador de campo magnético, adquisición de señales y procesamiento), los cuales se muestran en la Figura 2.7; así como, las especificaciones técnicas principales del sistema.

Parámetro	Valor
Sensor DOF (Grados de libertad)	5 o 6
Número máximo de sensores	16 (5 DOF) / 8 (6 DOF)
Frecuencia de medición	40 Hz
Dimensión / Peso	20 x 200 x 70 mm / 3.2 kg
Error Máximo RMS	1.20 mm / 0.7 °

Figura 2.7. Especificaciones técnicas principales del sistema de medición de posición

El sensor de presión permite la medición de la presión neumática absoluta en cada una de las cámaras internas del endoscopio. Sus principales especificaciones técnicas se muestran en la Figura 2.8. La relación es lineal entre la presión medida por el sensor con la señal analógica de voltaje del sensor dentro de su rango de operación.

Parámetro	Valor
Voltaje de alimentación (Vss)	3.0 - 3.63VDC
Tipo de presión	Absoluta
Rango de presión	0 - 1.6 bar
Tipo de salida	Analógica
Rango de salida	0 - Vss

Figura 2.8. Especificaciones técnicas principales del sensor de presión

Respecto a la tarjeta electrónica programable Arduino Due, se muestran sus principales características técnicas en la Figura 2.9. Las señales del sistema de medición de posición y el sensor de presión son leídas directamente por el Arduino Due. Por otro lado, será necesario considerar una etapa de potencia para controlar y regular el voltaje en los solenoides

proporcional de las electroválvulas, debido a que la corriente de operación máxima de los pines digitales del Arduino Due no debe superar corriente promedio máxima de 5 mA.

Parámetro	Valor
Microcontrolador	Atmel SAM3X8E ARM Cortex-M3
Memoria Flash/SRAM	512 KB/96KB
Velocidad de reloj	84 MHz
Voltaje de pines E/S	3.3 V
N° de pines PWM	12
N° de entradas analógicas	12
N° de puertos SPI	2

Figura 2.9. Especificaciones técnicas de la placa electrónica Arduino Due

Cada de una las electroválvulas reguladoras de presión, cuyas especificaciones técnicas se muestran en la Figura 2.10, permiten variar la presión neumática de cada una de las cámaras neumáticas internas del endoscopio dentro de un rango de presión de 0 a 2 bar, con lo cual se puede obtener diferentes deformaciones funcionales y movimientos del cuerpo del endoscopio blando.

Parámetro	Valor
Cuerpo de válvula	3 vías proporcional
Rango de presión de entrada	0 - 11 bar
Rango de presión de salida	0.01 - 2 bar
Rango de voltaje de referencia	0 - 5 v
Rango de voltaje de alimentación	19 - 29 v
Máximo consumo de potencia	1 W

Figura 2.10. Especificaciones técnicas principales de la electroválvula proporcional reguladora de presión

La etapa de potencia está basada en cuatro transistores MOSFET para cada respectiva electroválvula. En el encapsulado TD9944, está integrado dos transistores, cuyas características técnicas se muestran en la Figura 2.11. Cabe resaltar que estos transistores son compatibles con las salidas digitales del controlador Arduino Due.

Parámetro	Valor
Voltaje de control ($V_{GS(th)}$)	0.6 - 2 V
Corriente máxima ($I_{D(ON)}$)	1.9 A
Voltaje máximo ($V_{DS(OFF)}$)	240 V
Temperatura de operación	-55°C – 150°C

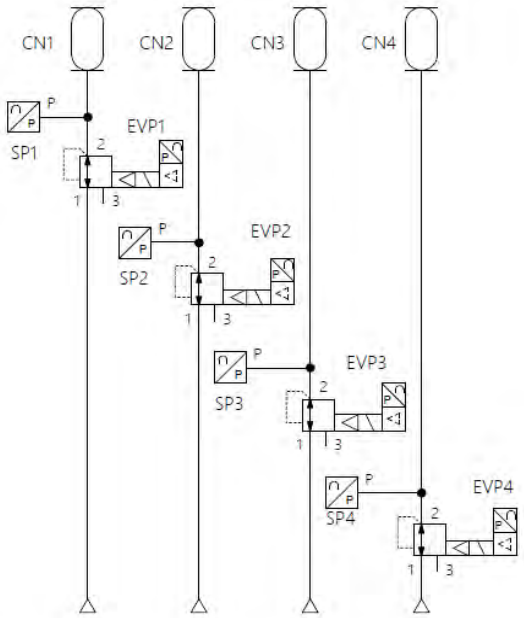
Figura 2.11. Especificaciones técnicas de los transistores MOSFET

2.4. Diseño neumático

El diseño del circuito neumático y sus componentes principales del cuerpo del endoscopio se presenta en la Figura 2.12. Para cada cámara interna (CNx), podemos observar una línea neumática independiente, cuya presión es medida y controlada por sus respectivos sensores de presión (SPx) y electroválvula proporcional (EVPx). El rango de operación de la presión para cada cámara interna se encuentra desde 0 a 2 bar.

2.5. Condiciones del movimiento omnidireccional del cuerpo del endoscopio

En la presente sección, se describen las condiciones para el movimiento del cuerpo del endoscopio blando dentro de su rango de operación. Como se muestra en lado izquierdo de la Figura 2.13, la posición de referencia fija del sistema de coordenadas XYZ se ubica en el efector final del cuerpo del endoscopio sin deformar; así mismo, hacia el lado derecho, se muestra una vista en sección del plano XY donde se pueden observar las secciones semicirculares correspondiente a las cuatro cámaras internas del cuerpo. La identificación de las cámaras del cuerpo se realiza de acuerdo a su ubicación en el plano XY, donde se definen la cámara $x+$ (eje x positivo), cámara $x-$ (eje x negativo), cámara $y+$ (eje y positivo) y la cámara $y-$ (eje y negativo). Debido a que cada cámara puede ser presurizada de manera independiente, se definen las siguientes presiones $px+$ (presión en la cámara $x+$), $px-$ (presión en la cámara $x-$), $py+$ (presión en la cámara $y+$) y $py-$ (presión en la cámara $y-$). Para desplazar el cuerpo del endoscopio hacia cualquiera de los cuatros cuadrantes del plano XY, se presurizan las cámaras internas en combinaciones de pares.



TAG	DESCRIPCIÓN
EVPX	ELECTROVÁLVULA PROPORCIONAL DE PRESIÓN (X = 1, 2, 3 ,4)
SPX	SENSOR DE PRESIÓN (X = 1, 2, 3, 4)
CNX	CÁMARA NEUMÁTICA (X = 1, 2, 3, 4)

Figura 2.12. Diagrama neumático del endoscopio blando

Se tiene cuatro combinaciones posibles para la presurización de las cámaras internas del endoscopio, las cuales se muestran en la Figura 2.14. Por ejemplo, en la parte (A) se activan las presiones $px+$ y $py+$ generando un desplazamiento del cuerpo hacia el tercer cuadrante del plano XY, en la parte (B) se activan las presiones $px+$ y $py-$ generando un desplazamiento hacia el segundo cuadrante, en la parte (C) se activan las presiones $px-$ y $py+$ generando un desplazamiento hacia el cuarto cuadrante y en la parte (D), se activan las presiones $px-$ y $py-$ generando un desplazamiento hacia el primer cuadrante. Con ello, es posible controlar cualquier posición (x, y) del cuerpo del endoscopio. Bajo estas condiciones de movimiento, se desarrolla el modelamiento de la dinámica del sistema y el diseño del controlador de posición en los capítulos 3 y 4 respectivamente.

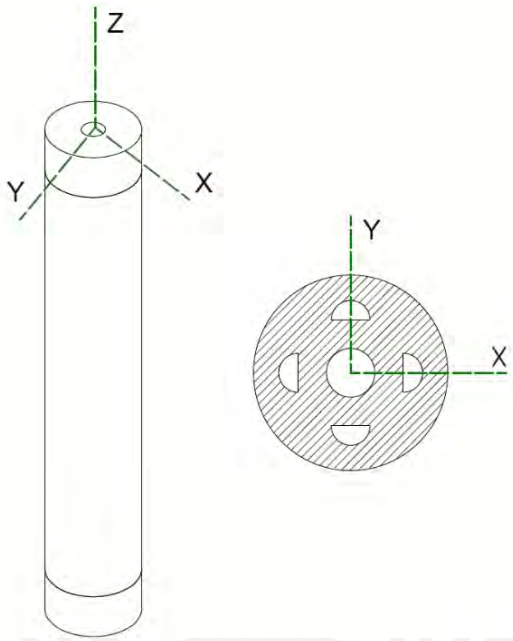


Figura 2.13. Coordenadas de referencia del cuerpo del endoscopio

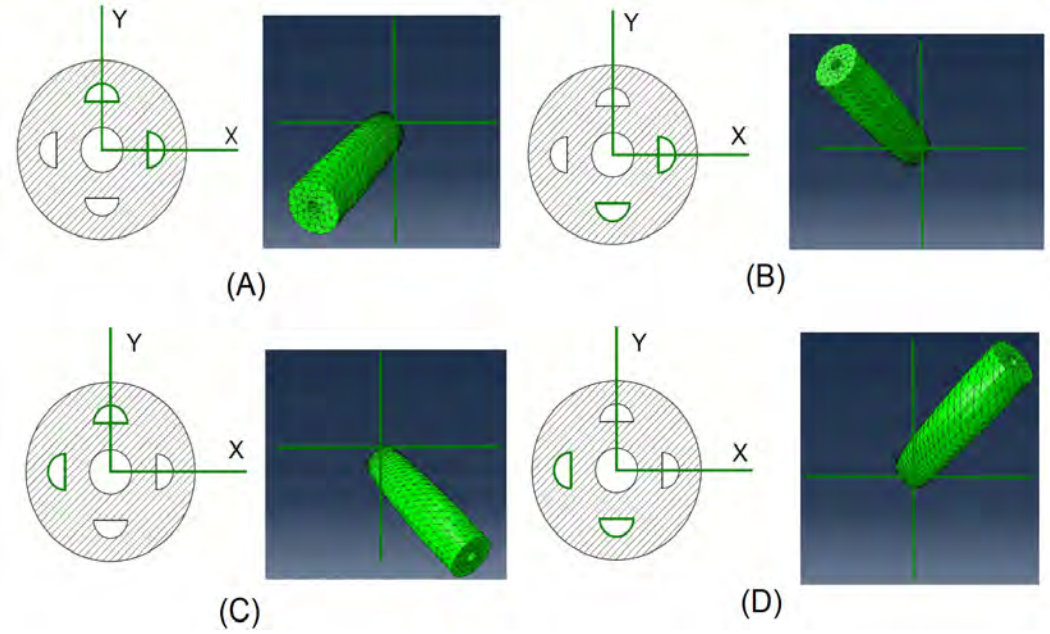


Figura 2.14. Condiciones para el movimiento omnidireccional del cuerpo

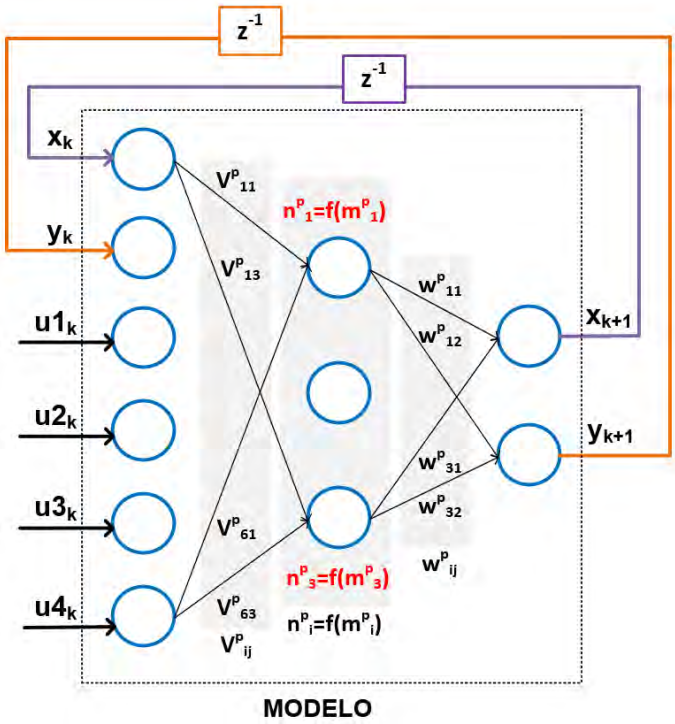
Capítulo III: MODELAMIENTO DEL ENDOSCOPIO BLANDO

En el presente capítulo, se desarrolla la metodología para la obtención del modelo de la dinámica del cuerpo del endoscopio blando basado en una estructura de red neuronal con realimentación. El capítulo inicia con la descripción de las características de la estructura de red neuronal que se utiliza para modelar la dinámica del sistema. Luego, se muestran las consideraciones tomadas en cuenta para la generación de los datos de entrenamiento de la red neuronal. A continuación, se describe el algoritmo de entrenamiento de la red neuronal, así como los aspectos relevantes relacionados a su correspondiente implementación. Posteriormente, se presentan los resultados del entrenamiento de la red neuronal y los indicadores de confiabilidad del modelo obtenido. Finalmente, se muestran los datos generados para la etapa de validación de la red neuronal, así como los resultados obtenidos.

3.1. Estructura de la red neuronal

La estructura considerada para el modelamiento de sistema es una red neuronal recurrente con retroalimentación a la salida como se muestra en la Figura 3.1. La capa de entrada de la red neuronal presenta seis neuronas de entrada, la capa intermedia 3 neuronas y la capa de salida 2 neuronas. La función de activación f correspondiente a las neuronas de la capa de entrada y salida es lineal y la correspondiente a la capa intermedia es no lineal (función sigmoidea). La capa de entrada está formada por las neuronas correspondientes a las presiones u_k de las cuatro cámaras del cuerpo más las neuronas correspondientes a las coordenadas x_k e y_k de la posición cartesiana r_k del efector final, las cuales son retroalimentadas desde la capa de salida de la red neuronal. La capa de salida está formada por neuronas correspondientes a las coordenadas x_{k+1} e y_{k+1} de la posición cartesiana r_{k+1}. Las conexiones entre las neuronas de la capa de entrada e intermedia es a través de los coeficientes v_{ij}^p y las conexiones entre la capa intermedia y la capa de salida son a través de los coeficientes w_{ij}^p. Por ello, la estructura del modelo está formada por una matriz de coeficientes v^p de tamaño 6x3 y una matriz de coeficientes w^p de tamaño 3x2, los cuales se muestran en las ecuaciones (3.1) y (3.2) respectivamente. El vector de entradas m^p para las neuronas de la capa intermedia se obtiene por la multiplicación del vector de entradas $[r_k\ u_k]$ con la matriz de coeficientes v^p como se muestra en la ecuación (3.3). El vector de salidas n^p para las neuronas de la capa intermedia se obtiene a partir de la ecuación (3.4) donde $f(m_i^p), \forall i = 1, 2\ y\ 3$ corresponde a la función sigmoidea aplicada a cada elemento del vector m^p. Finalmente, el vector de salidas r_k se obtiene de la multiplicación del vector n^p con la matriz de coeficientes w^p como se muestra en la ecuación (3.5). El conjunto de ecuaciones

anteriormente mencionadas describe las relaciones entrada - salida del modelo del sistema entre cada instante de tiempo t_k y t_{k+1}.



Presión en cámaras internas:

$$u_k = [u1_k \quad u2_k \quad u3_k \quad u4_k]^T$$

Donde:

$1 = x+, 2 = y+, 3 = x-, 4 = y-$

Posición efector final:

$$r_k = [x_k \quad y_k]^T$$

Coeficientes de conexión:

$$v_{ij}^p \quad w_{ij}^p$$

Donde:

i: neurona de partida i

j: neurona de llegada j

Entrada de neurona intermedia:

$$m_i^p$$

Salida de neurona intermedia:

$$n_i^p$$

Función de activación neurona intermedia

$$f(m_i^p)$$

Figura 3.1. Estructura de la red neuronal del modelo

$$v^p = \begin{bmatrix} v_{11}^p & v_{12}^p & v_{13}^p \\ v_{21}^p & v_{22}^p & v_{23}^p \\ v_{31}^p & v_{32}^p & v_{33}^p \\ v_{41}^p & v_{42}^p & v_{43}^p \\ v_{51}^p & v_{52}^p & v_{53}^p \\ v_{61}^p & v_{62}^p & v_{63}^p \end{bmatrix} \ldots (3.1), \quad w^p = \begin{bmatrix} w_{11}^p & w_{12}^p \\ w_{21}^p & w_{22}^p \\ w_{31}^p & w_{32}^p \end{bmatrix} \ldots (3.2)$$

$$m^p = (v^p)^T \times in_{red} \ldots (3.3)$$

Donde:

$$in_{red} = [x_k \quad y_k \quad u1_k \quad u2_k \quad u3_k \quad u4_k]^T$$

$$m^p = [m_1^p \quad m_2^p \quad m_3^p]^T$$

$$n^p = [f(m_1^p) \quad f(m_2^p) \quad f(m_3^p)]^T \ldots (3.4)$$

Donde:

$$f(m_i^p) = \frac{2}{1 + e^{-m_i^p}} - 1$$

$$r_{k+1} = (w^p)^T \times n^p \ldots (3.5)$$

Donde:

$$r_{k+1} = [x_{k+1} \quad y_{k+1} \quad z_{k+1}]^T$$

3.2. Generación de datos de entrenamiento

Para el entrenamiento de la red neuronal, al no disponer de datos experimentales, se utiliza el software de simulación Abaqus CAE, donde se modela solamente el cuerpo del endoscopio considerando el diseño mecánico presentado en el capítulo anterior. Los datos de entrenamiento, la presión en las cámaras internas y la posición del efector final, son registrados durante un tiempo de simulación de 27.8 segundos teniendo en cuenta las condiciones necesarias para el movimiento omnidireccional del endoscopio que se describieron en el capítulo anterior. La planificación de la secuencia de presurización de las cámaras internas del cuerpo del endoscopio se divide en cuatro etapas con un tiempo de duración de 6.8 segundos como se representa en la Figura 3.2.

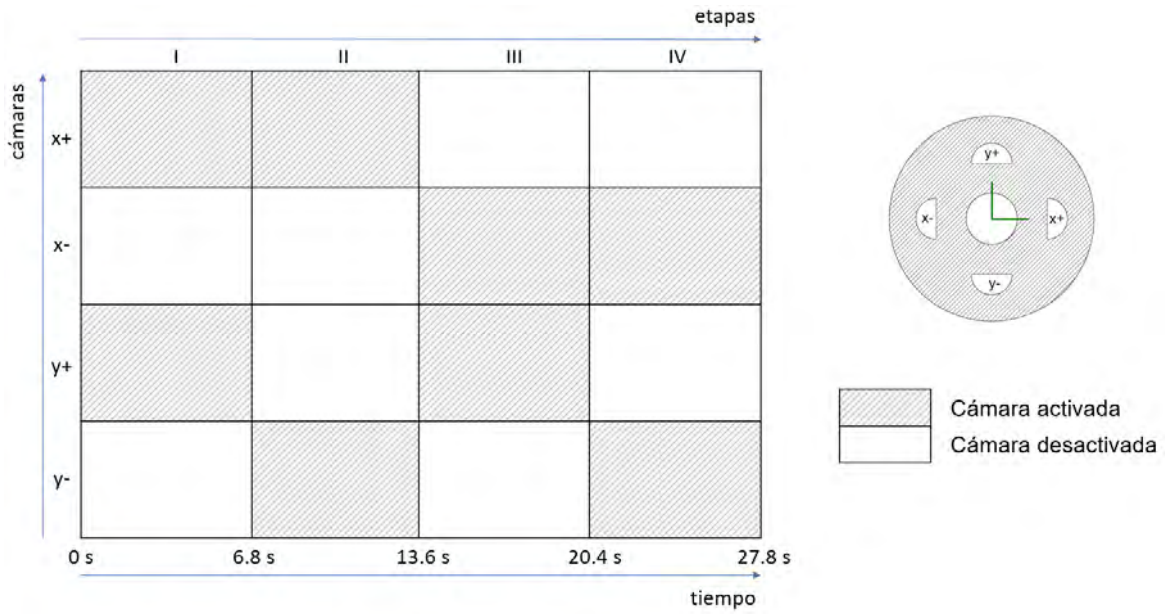


Figura 3.2. Planificación de las presiones para la obtención de datos de entrenamiento

En la primera etapa, se activa la cámara $x+$ e $y+$; en la segunda etapa, se activa la cámara $x+$ e $y-$; en la tercera etapa, se activa la cámara $x-$ e $y+$; y en la cuarta etapa, se activa la cámara $x-$ e $y-$. La configuración de la simulación en Abaqus CAE se realiza considerando la secuencia de activación de las cámaras internas anteriormente descritas y un periodo de cómputo fijo de 1 ms generando 27800 datos durante el tiempo total de simulación.

El patrón de las presiones que se considera en cada una de las cuatro etapas de activación se muestra en la Figura 3.3 para la cámara x (línea azul) y la cámara y (línea roja). Se puede observar que el patrón de las señales de presión es trapezoidal con un rango de amplitud de 0 a 0.3 bar y un periodo de 0.6 segundos. Cabe resaltar que debido a las restricciones propias del modelo digital y el software que se dispone no es posible aumentar la amplitud de la

presión ni tampoco hacer uso de un patrón de señal completamente cuadrado. También, se puede notar que las señales de presión tanto para la cámara x y la cámara y no se traslapan entre sí, debido a que se consideran diferentes tiempos de duración para la rampa de subida, en alta y la rampa de bajada para cada patrón trapezoidal de las presiones tanto en x como en y. Los valores de los tiempos anteriormente mencionados se muestran en la Tabla 3.1.

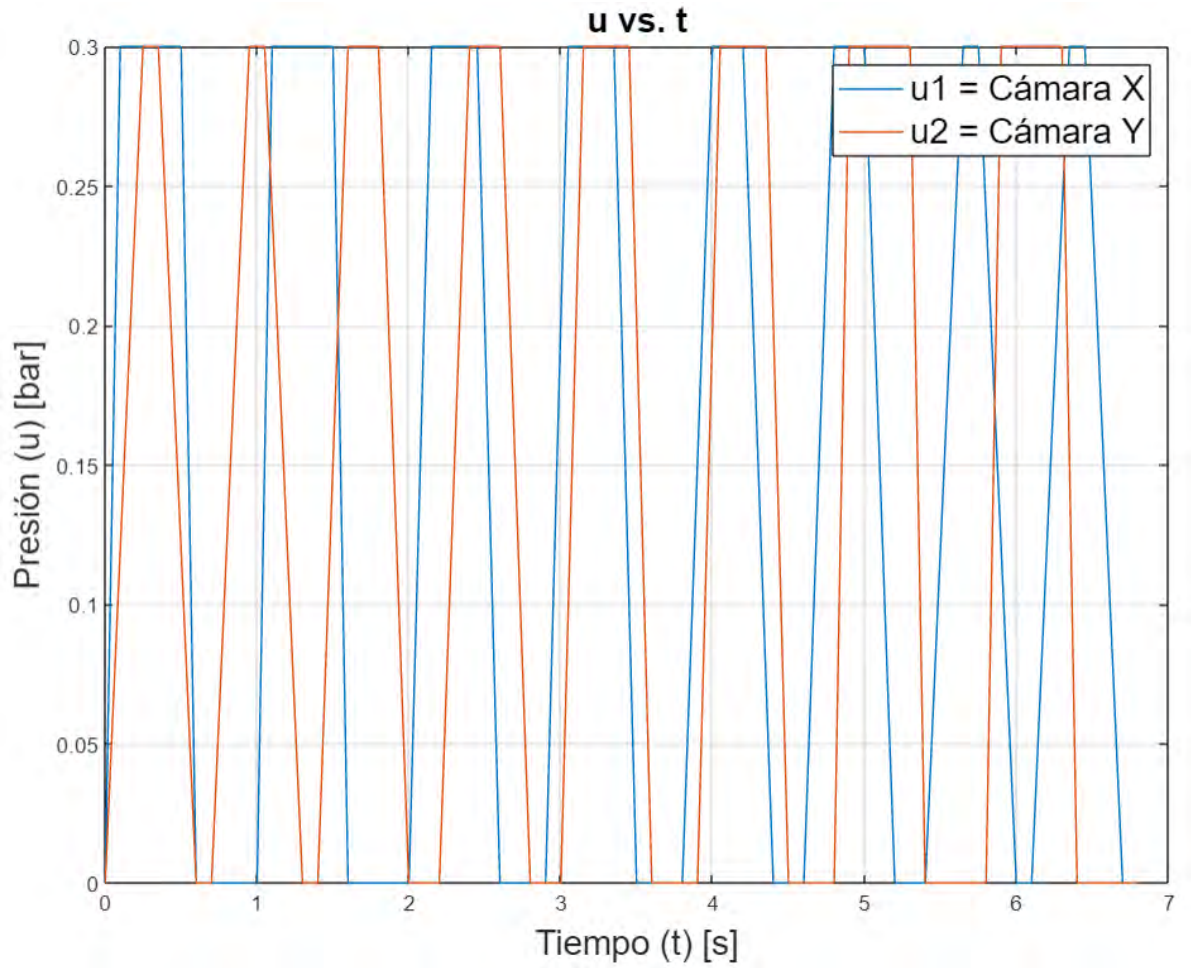


Figura 3.3. Patrón de las presiones por etapa en la cámara x e y

Tabla 3.1. Tiempo de subida, en alta y baja para los patrones trapezoidales

Tiempo	Valor 1	Valor 2	Valor 3	Valor 4
De subida (s)	0.10	**0.15**	**0.20**	0.25
En alta (s)	0.40	**0.30**	**0.20**	0.10
De bajada (s)	0.10	**0.15**	**0.20**	0.25

A partir de la secuencia de presiones de entrada descritas, se obtiene como respuesta o salida del sistema sus respectivas posiciones del efector final mostradas en la Figura 3.4, donde las coordenadas x_k se muestran en línea de color azul y las coordenadas y_k se

muestran en línea de color rojo. Se puede observar que tanto para las coordenadas x_k e y_k su rango de operación se encuentra desde -30 mm hasta 30 mm. De igual manera, se obtienen 27800 datos para cada una de las coordenadas cartesianas. El código utilizado en Matlab para la generación de los datos de entrenamiento se muestra en el Anexo 01 correspondiente al presente documento.

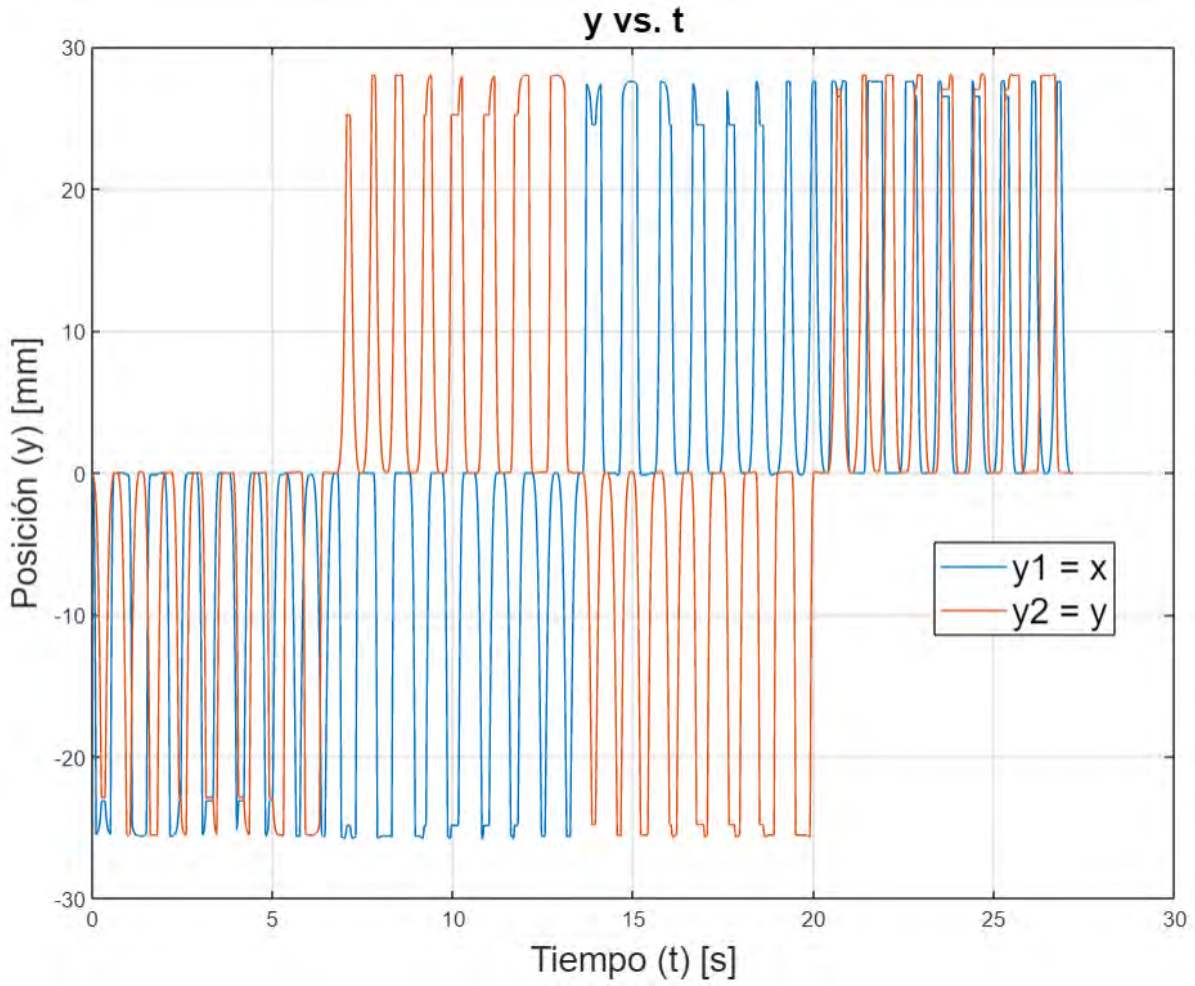


Figura 3.4. Datos de las posiciones cartesianas del efector final para entrenamiento

3.3. Algoritmo de entrenamiento

Para el entrenamiento del modelo basado en redes neuronales se utiliza el algoritmo DBP (Narendra K., et al, 1990). El objetivo del proceso de entrenamiento es encontrar los valores óptimos de los coeficientes de conexión v_{ij} entre las neuronas de la capa de entrada e intermedia y los coeficientes w_{ij} entre las neuronas de la capa intermedia y de salida. Para ello, es necesario minimizar la función de costo J mostrada en la ecuación (3.5), la cual es definida como la suma de los errores cuadráticos entre las posiciones cartesianas medidas r_k^* y las posiciones cartesianas estimadas r_k por la red neuronal correspondientes al efector final del cuerpo del endoscopio. Como optimizador se utiliza el método del gradiente

descendente mostrado en la ecuación (3.6) y (3.7), donde n es la tasa de aprendizaje y $\frac{\partial \bar{J}}{\partial \bar{w}_{ij}^{p}}$, $\frac{\partial \bar{J}}{\partial \bar{v}_{ij}^{p}}$ las derivadas parciales totales de la función de costo J respecto a cada uno de los coeficientes de conexión v_{ij}^{p} y w_{ij}^{p}.

$$J = \frac{1}{2}\sum_{K=1}^{N}(r_k - r_k^*)^T(r_k - r_k^*) \ldots (3.5)$$

Donde:

$$r_k = [x_k \quad y_k \quad z_k]^T,\ r_k^* = [x_k^* \quad y_k^* \quad z_k^*]^T$$

Las derivadas parciales totales $\frac{\partial \bar{J}}{\partial w_{ij}^{p}}$, $\frac{\partial \bar{J}}{\partial v_{ij}^{p}}$ son obtenidas con las ecuaciones (3.8) y (3.9) respectivamente, las cuales son calculadas para todos los instantes de tiempo $t_k\ \forall k = 1, \ldots, N$. Los vectores de posiciones r_k, r_k^* son conocidos y se deben calcular las derivadas parciales totales $\frac{\partial \bar{r}_k}{\partial \bar{w}_{ij}^{p}}$, $\frac{\partial \bar{r}_k}{\partial \bar{v}_{ij}^{p}}$ usando el algoritmo de retropropagación dinámica (Dynamic Back Propagation - DBP por sus siglas en inglés).

$$v_{ij}^{p} = v_{ij}^{p} - n\frac{\bar{\partial J}}{\partial \overline{v_{ij}^{p}}} \ldots (3.6) \qquad w_{ij}^{p} = w_{ij}^{p} - n\frac{\bar{\partial J}}{\partial \overline{w_{ij}^{p}}} \ldots (3.7)$$

$$\frac{\bar{\partial J}}{\partial \overline{w_{ij}^{p}}} = \sum_{k=1}^{N}(r_k - r_k^*)^T\frac{\overline{\partial r_k}}{\partial \overline{w_{ij}^{p}}} \ldots (3.8) \qquad \frac{\bar{\partial J}}{\partial \overline{v_{ij}^{p}}} = \sum_{k=1}^{N}(r_k - r_k^*)^T\frac{\overline{\partial r_k}}{\partial \overline{v_{ij}^{p}}} \ldots (3.9)$$

El algoritmo DBP mostrado en las ecuaciones (3.10) y (3.11), debido a su naturaleza recursiva, es válido entre cada instante de tiempo t_k y t_{k+1} y permite el cálculo de las derivadas parciales totales $\frac{\partial \bar{r}_{k+1}}{\partial w_{ij}^{p}}$, $\frac{\partial \bar{r}_{k+1}}{\partial v_{ij}^{p}}$, donde las derivadas parciales simples $\frac{\partial r_{k+1}}{\partial w_{ij}^{p}}$, $\frac{\partial r_{k+1}}{\partial v_{ij}^{p}}$ y la matriz jacobiana $\frac{\partial r_{k+1}}{\partial r_k}$ se pueden obtener a partir de las ecuaciones de relación entrada - salida del modelo del sistema. La ecuación (3.12) permite obtener la matriz jacobiana $\frac{\partial r_{k+1}}{\partial r_k}$ del modelo del sistema y las ecuaciones desde la (3.13) hasta la (3.16) permiten obtener las componentes de las derivadas simples $\frac{\partial r_{k+1}}{\partial w_{ij}^{p}}, \frac{\partial r_{k+1}}{\partial v_{ij}^{p}}$, donde los términos in_{red}, w^p, v^p, n_i y $f'(m_i^p)$ son conocidos.

El flujo a seguir durante el proceso de entrenamiento del modelo del sistema se muestra en la Figura 3.5, donde las tareas se han agrupado principalmente en tres procesos: inicialización, ejecución y evaluación. Las condiciones necesarias para finalizar el proceso de

entrenamiento consisten en haber superado el número máximo de iteraciones o haber alcanzado el % de error mínimo relativo deseado.

$$\overline{\frac{\partial r_{k+1}}{\partial w_{ij}^p}} = \frac{\partial r_{k+1}}{\partial w_{ij}^p} + \frac{\partial r_{k+1}}{\partial r_k}\overline{\frac{\partial r_k}{\partial w_{ij}^p}} \dots (3.10)$$

$$\overline{\frac{\partial r_{k+1}}{\partial v_{ij}^p}} = \frac{\partial r_{k+1}}{\partial v_{ij}^p} + \frac{\partial r_{k+1}}{\partial r_k}\overline{\frac{\partial r_k}{\partial v_{ij}^p}} \dots (3.11)$$

Donde:

$$\overline{\frac{\partial r_{k+1}}{\partial w_{ij}^p}} = \begin{bmatrix} \overline{\frac{\partial x_{k+1}}{\partial w_{ij}^p}} \\ \overline{\frac{\partial y_{k+1}}{\partial w_{ij}^p}} \end{bmatrix}, \overline{\frac{\partial r_{k+1}}{\partial v_{ij}^p}} = \begin{bmatrix} \overline{\frac{\partial x_{k+1}}{\partial v_{ij}^p}} \\ \overline{\frac{\partial y_{k+1}}{\partial v_{ij}^p}} \end{bmatrix}, \overline{\frac{\partial r_k}{\partial w_{ij}^p}} = \begin{bmatrix} \overline{\frac{\partial x_k}{\partial w_{ij}^p}} \\ \overline{\frac{\partial y_k}{\partial w_{ij}^p}} \end{bmatrix}, \overline{\frac{\partial r_k}{\partial v_{ij}^p}} = \begin{bmatrix} \overline{\frac{\partial x_k}{\partial v_{ij}^p}} \\ \overline{\frac{\partial y_k}{\partial v_{ij}^p}} \end{bmatrix}$$

$$\frac{\partial r_{k+1}}{\partial w_{ij}^p} = \begin{bmatrix} \frac{\partial x_k}{\partial w_{ij}^p} \\ \frac{\partial y_k}{\partial w_{ij}^p} \end{bmatrix}, \frac{\partial r_{k+1}}{\partial v_{ij}^p} = \begin{bmatrix} \frac{\partial x_k}{\partial v_{ij}^p} \\ \frac{\partial y_k}{\partial v_{ij}^p} \end{bmatrix}, \frac{\partial r_{k+1}}{\partial r_k} = \begin{bmatrix} \frac{\partial x_{k+1}}{\partial x_k} & \frac{\partial x_{k+1}}{\partial y_k} \\ \frac{\partial y_{k+1}}{\partial x_k} & \frac{\partial y_{k+1}}{\partial y_k} \end{bmatrix},$$

$$\frac{\partial r_{k+1}}{\partial r_k} = (w^p)^T \times diag\left(f'\left(m_i^p\right)\right) \times \left(v^p(1{:}\,2,:)\right)^T \dots (3.12)$$

$$\frac{\partial x_{k+1}}{\partial w_{ij}^p} = n_i \ \forall j = 1, \frac{\partial x_{k+1}}{\partial w_{ij}^p} = 0 \ \forall j \neq 1 \dots (3.13)$$

$$\frac{\partial y_{k+1}}{\partial w_{ij}^p} = n_i \ \forall j = 2, \frac{\partial y_{k+1}}{\partial w_{ij}^p} = 0 \ \forall j \neq 2 \dots (3.14)$$

$$\frac{\partial x_k}{\partial v_{ij}^p} = in_{red(i)} \times w_{j1}^p \times f'(m_j^p) \dots (3.15)$$

$$\frac{\partial y_k}{\partial v_{ij}^p} = in_{red(i)} \times w_{j2}^p \times f'(m_j^p) \dots (3.16)$$

Donde:

$$f'\left(m_j^p\right) = \frac{1 - n_j^2}{2}$$

El proceso de inicialización empieza con la carga y normalización de los datos de entrenamiento. La normalización consiste en escalar los datos de entrada y salida de entrenamiento calculando sus valores relativos respecto a los valores máximos de entrada y salida del grupo de datos de entrenamiento. Luego, se definen las características de la estructura del modelo asignando el número de neuronas para la capa de entrada, intermedia y salida de la red neuronal. Por último, se ingresan los parámetros asociados al entrenamiento del modelo, donde se considera una tasa de aprendizaje n igual a 0.01, un % de error relativo

total mínimo del 2% y una cantidad máxima de iteraciones de 5000. Bajo estas condiciones se inicializan las matrices v^c y w^c previo al comienzo del entrenamiento.

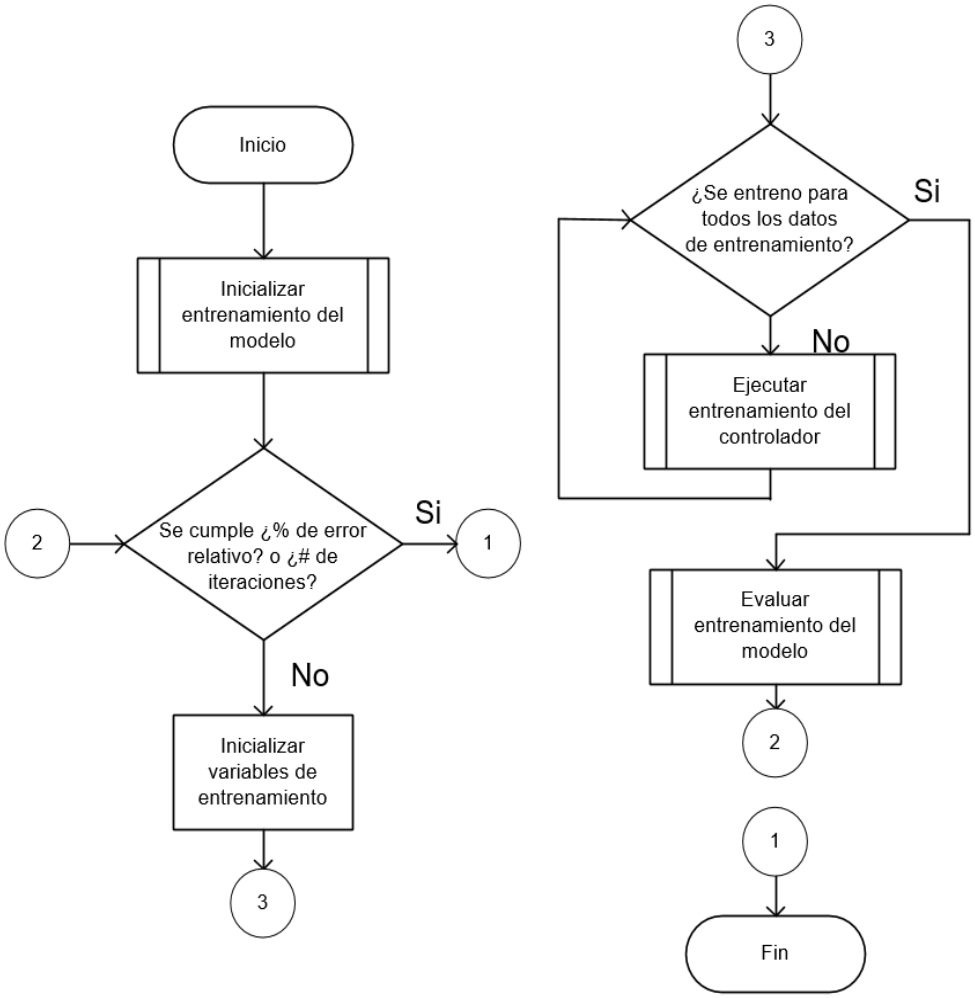


Figura 3.5. Diagrama de flujo del algoritmo de entrenamiento

El proceso de ejecución del entrenamiento se basa principalmente en el algoritmo DBP y el optimizador de gradiente descendente para la obtención de los valores óptimos de las matrices de coeficientes v^p y w^p que permitan minimizar la función de costo definida como el error cuadrático entre la posición del modelo r_k y la posición deseada r_k^* . Para ello, se requiere calcular para el conjunto de datos de entrenamiento su respectiva derivada parcial total de la función de costo J respecto a cada uno de los coeficientes de conexión v_{ij}^p y w_{ij}^p $(\frac{\partial \bar{J}}{\partial \bar{w}_{ij}^p}, \frac{\partial \bar{J}}{\partial \bar{v}_{ij}^p})$, las cuales se pueden obtener calculando el error entre las posiciones estimadas y medidas, r_k y r_k^* respectivamente, y las derivadas parciales totales de la salida estimada r_k respecto a los coeficientes de conexión v_{ij}^p y w_{ij}^p $(\frac{\partial \bar{r}_k}{\partial \bar{w}_{ij}^p}, \frac{\partial \bar{r}_k}{\partial \bar{v}_{ij}^p})$. Estas últimas derivadas son calculadas usando el algoritmo DBP entre cada instante de tiempo t_k y t_{k+1} así como las

derivadas parciales $\frac{\partial r_{k+1}}{\partial w_{ij}^{p}}$, $\frac{\partial r_{k+1}}{\partial v_{ij}^{p}}$ y la matriz jacobiana $\frac{\partial r_{k+1}}{\partial r_{k}}$, las cuales son obtenidas a partir de las ecuaciones de relación entrada y salida de la red neuronal que se mostraron anteriormente.

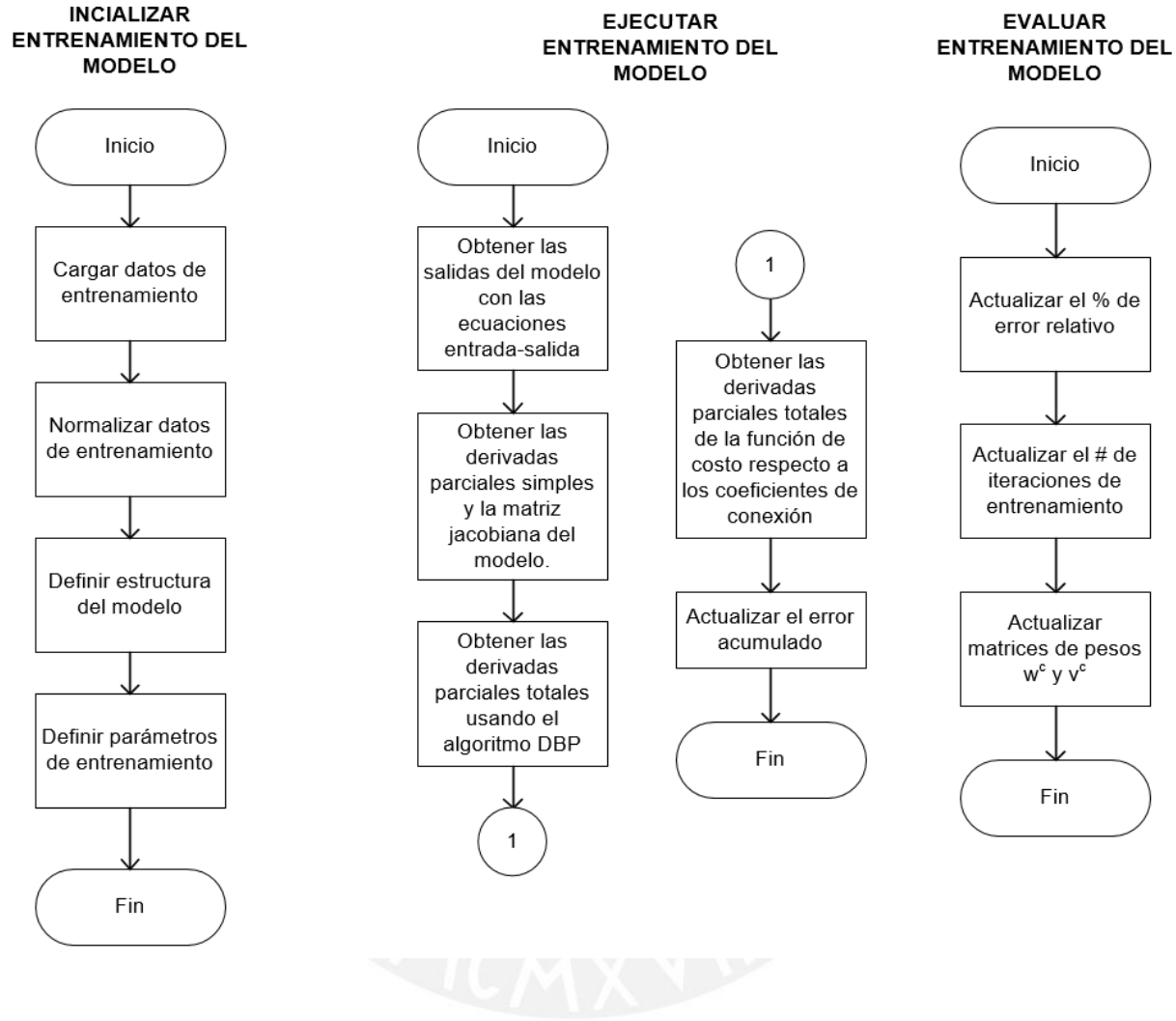


Figura 3.6. Procesos de entrenamiento del modelo del sistema

El proceso de evaluación consiste en verificar si el porcentaje de error relativo total del modelo entrenado es menor al porcentaje de error relativo total mínimo deseado. En caso esta última condición se cumpla, se terminará el entrenamiento caso contrario se continuará hasta que se cumpla cualquiera de las dos condiciones (el número máximo de iteraciones o el porcentaje de error deseado). Finalmente, al término de cada iteración se actualizan los valores de los coeficientes de las matrices v^p y w^p. Los procesos anteriormente descritos se muestran en la Figura 3.6. La implementación del algoritmo de entrenamiento descrito se muestra en el Anexo 02 correspondiente al presente documento.

3.4. Resultados del entrenamiento de la red neuronal

En la presente sección, se muestran los resultados del entrenamiento de la red neuronal que modela la dinámica del cuerpo del endoscopio; además, se define el porcentaje de error relativo como indicador de confiabilidad del modelo obtenido. En las Figuras 3.7 y 3.8 se realiza la comparación entre la posición deseada r_k^* (de línea roja) y la posición estimada por el modelo r_k (de línea azul). De ambas gráficas, se puede observar una similar tendencia en el error de predicción entre las coordenadas x_k^* y x_k y las coordenadas y_k^* e y_k. El error relativo se define en la ecuación (3.17), el cual compara para el conjunto de N datos de entrenamiento el vector de posición deseados r_k^* y el vector posición estimado r_k por el modelo.

$$error\ relativo = \frac{\sqrt{\sum_{k=1}^{N}(r_k^* - r_k)^T(r_k^* - r_k)}}{\sqrt{\sum_{k=1}^{N} r_k^{*2}}} \ldots (3.17)$$

El comportamiento del porcentaje de error relativo respecto a la cantidad de iteraciones en el entrenamiento de la red neuronal se muestra en la Figura 3.9, donde se han considerado un total de 3000 iteraciones. El error relativo total (en línea azul) converge a un valor de 24.50%. El error relativo respecto a la coordenada x (en línea roja) y a la coordenada y (en línea amarilla) convergen a los valores de 24.65% y 24.35% respectivamente. Con ello se cuantifica la confiabilidad del modelo alcanzado bajo las condiciones de entrenamiento anteriormente descritas.

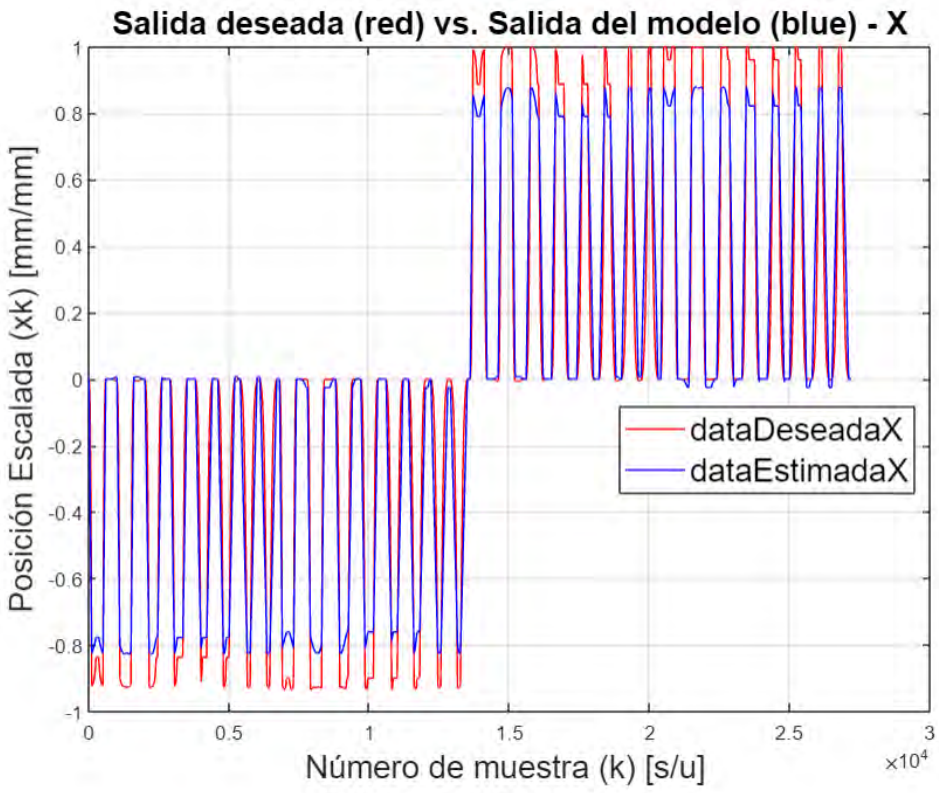


Figura 3.7. Posición X deseada vs. estimada del efector final del endoscopio

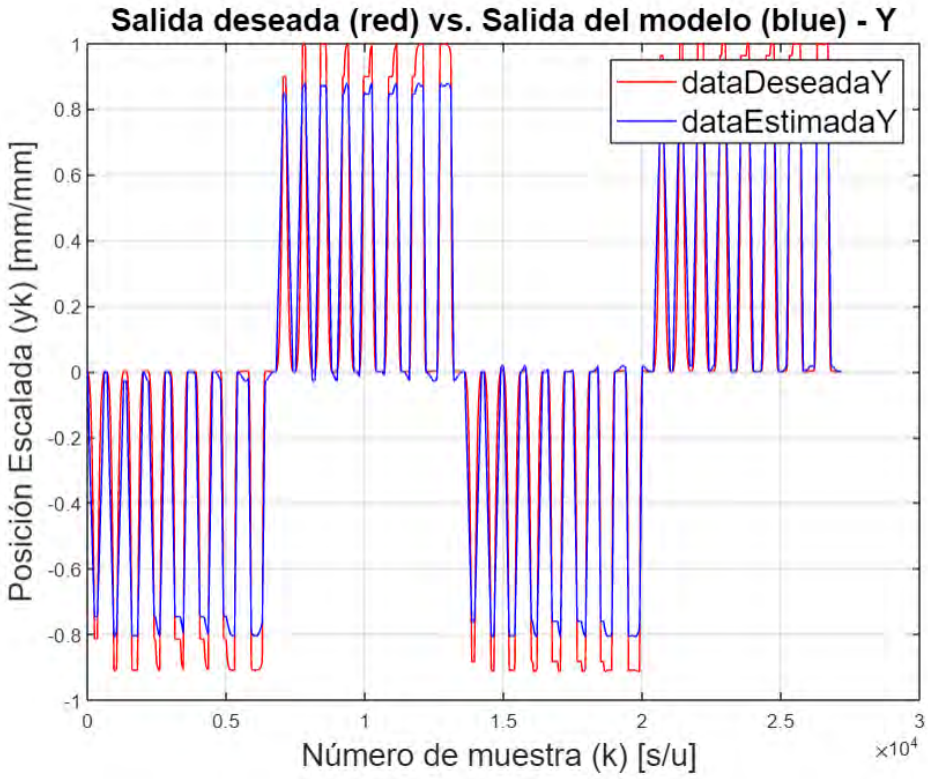


Figura 3.8. Posición Y deseada vs. estimada del efector final del endoscopio

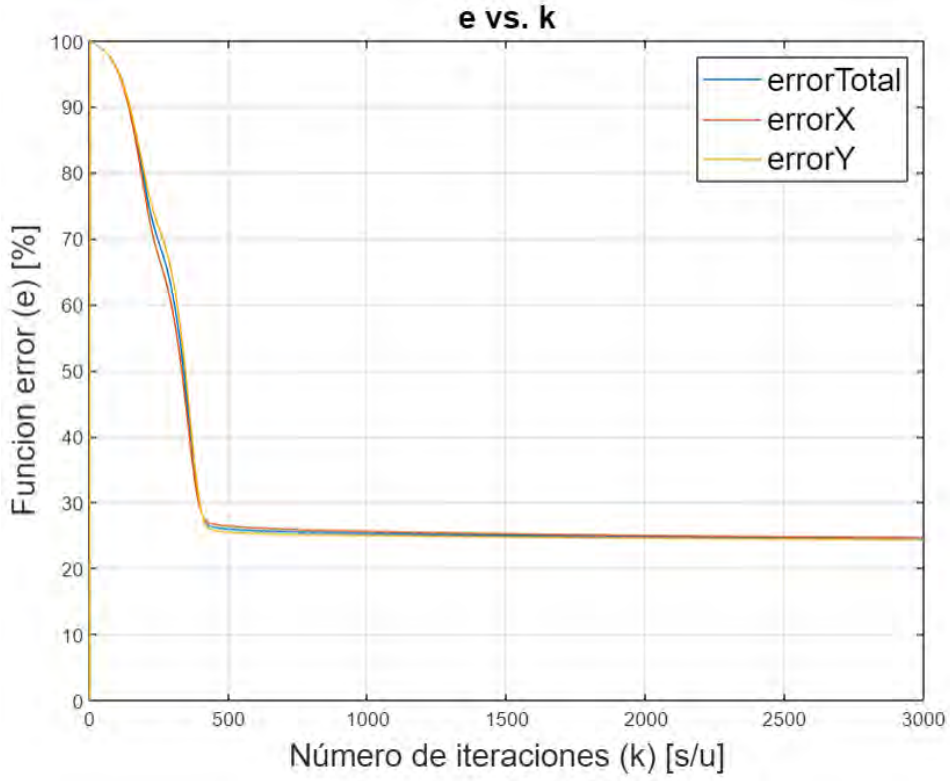


Figura 3.9. Evolución del error relativo durante el entrenamiento de la red neuronal

Se espera que los porcentajes de error en la estimación de las coordenadas x e y disminuyan al aplicar la misma metodología utilizando datos experimentales, debido a la ausencia de las restricciones que se presentan en el modelo digital descritas anteriormente.

3.5. Validación de la red neuronal

En esta etapa se realiza la verificación de la red neuronal a partir de los datos de validación generados usando el modelo digital del endoscopio en el software Abaqus CAE para lo cual se considera la secuencia de activación de las presiones en las cámaras del endoscopio mostrada en la Figura 3.2. El patrón de las señales de presión que se utiliza en esta etapa es el semiperiodo de una señal sinusoidal con un periodo de dos segundos y una amplitud de 0.3 bar. Las señales de entradas con las características descritas se muestran en la Figura 3.10, donde la presión $px+$ se muestra en línea azul, la presión $py+$ en línea roja, la presión $px-$ en línea amarilla y la presión $py-$ en línea violeta. La duración total de las señales de presión es de 4 segundos con un periodo de muestreo de 1 ms correspondiendo para cada variable una cantidad de 4000 datos. Considerando la configuración descrita para las presiones de entrada se generan las diferentes posiciones para el efector final del cuerpo del endoscopio, las cuales se muestran en la Figura 3.11. La coordenada x se muestra de color azul y la coordenada y de color rojo.

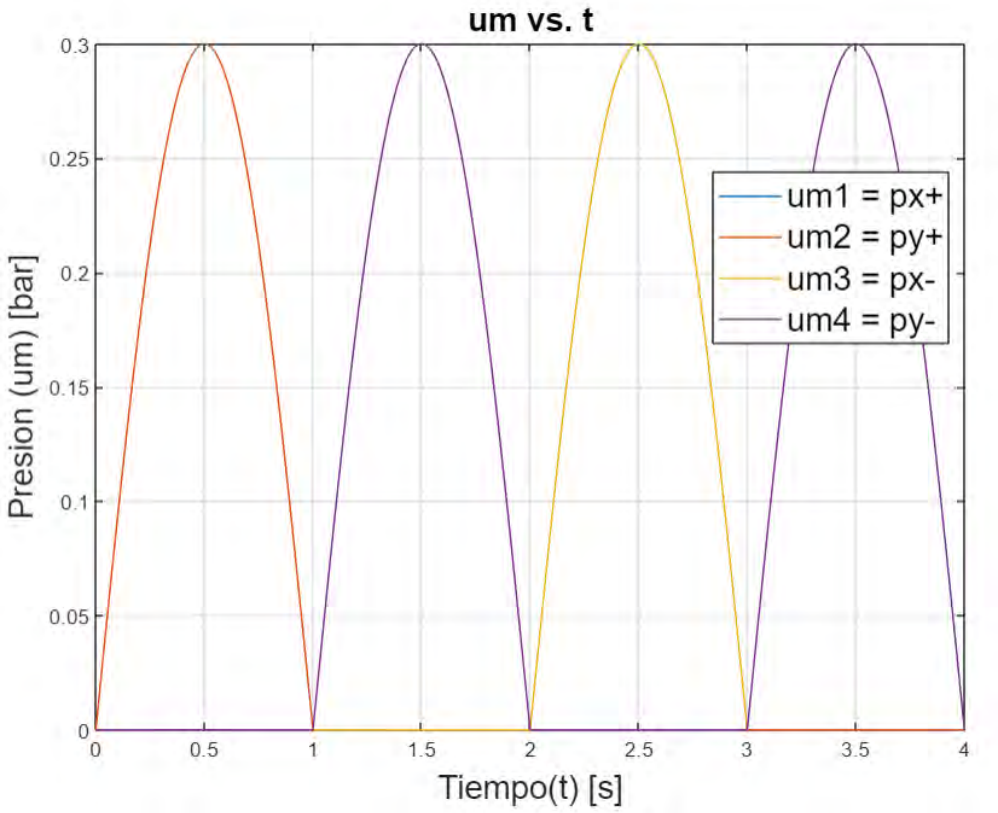


Figura 3.10. Presiones en el cuerpo del endoscopio para la etapa de validación

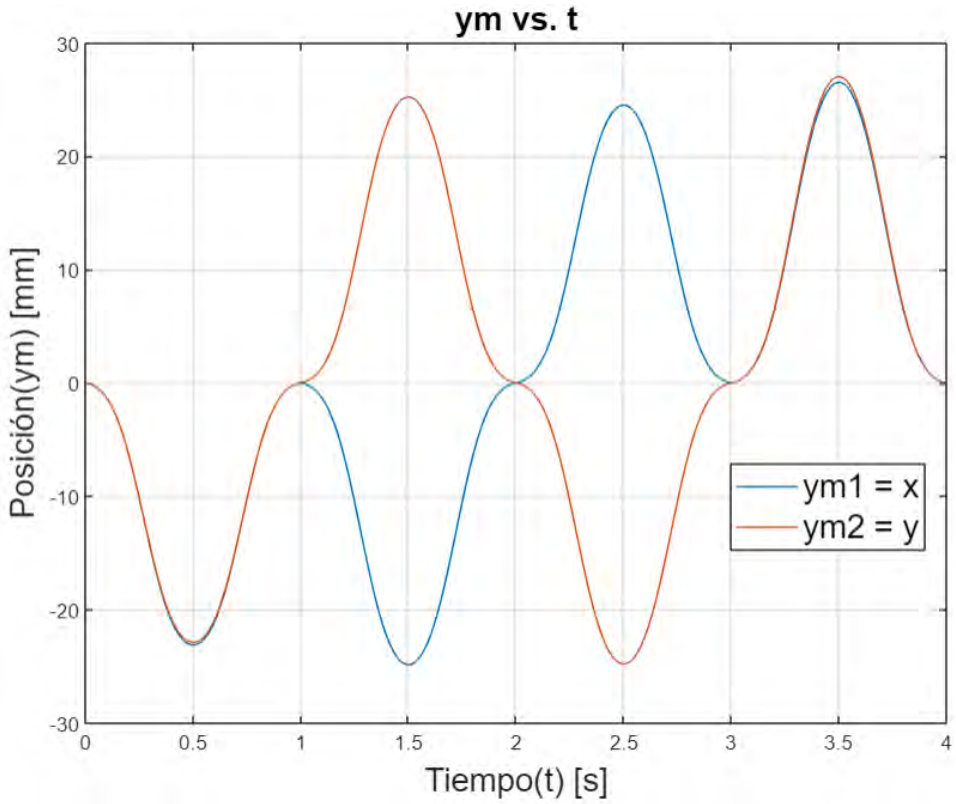


Figura 3.11. Posiciones del efector final del endoscopio para la etapa de validación

La validación de la red neuronal, que modela al sistema, se analiza comparando las respuestas de las coordenadas deseadas x_k^* e y_k^* con las coordenadas estimadas x_k e y_k por el modelo, las cuales se muestran en la Figuras 3.12 y 3.13 respectivamente. Considerando que en la etapa de entrenamiento se alcanzó un porcentaje de error relativo total de 24.50%, existe una propagación de dicho error para la etapa de validación. La presencia del error se debe a la fuente de generación de los datos de entrenamiento y validación; y a la estructura de la red neuronal recurrente seleccionada. Los datos obtenidos del software Abaqus CAE no pueden contener la dinámica completa del sistema (endoscopio) a modelar, debido a las limitaciones propias del modelo digital; por ejemplo, no se puede considerar en el modelado la dinámica del sistema neumático de accionamiento. Por otro lado, la estructura del modelo de red neuronal seleccionado está orientado por naturaleza a modelar un sistema dinámico generándose una pérdida de compatibilidad entre los datos de entrenamiento y validación y el modelo seleccionado disminuyendo así la confiabilidad en el modelo. No se ha optado por modificar la estructura de red neuronal, ya que la metodología está enfocada a ser utilizada haciendo uso de datos experimentales. Para el cálculo del error relativo en la etapa de validación también se utiliza la ecuación (3.16) obteniendo como resultado un error relativo total del 30.48% y error relativo para las coordenadas x e y de 30.49% y 30.46% respectivamente. El código utilizado en Matlab para realizar el proceso de validación del modelo se muestra en el Anexo 03 correspondiente al presente documento.

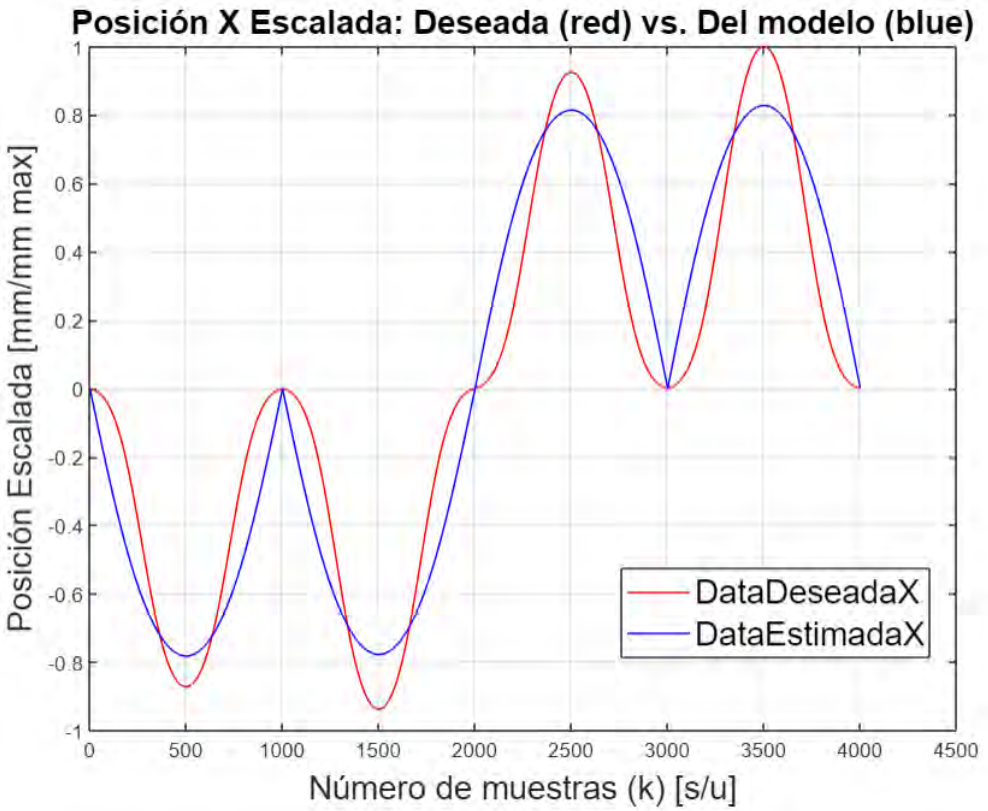


Figura 3.12. Posición X deseada vs. del modelo en la etapa de validación

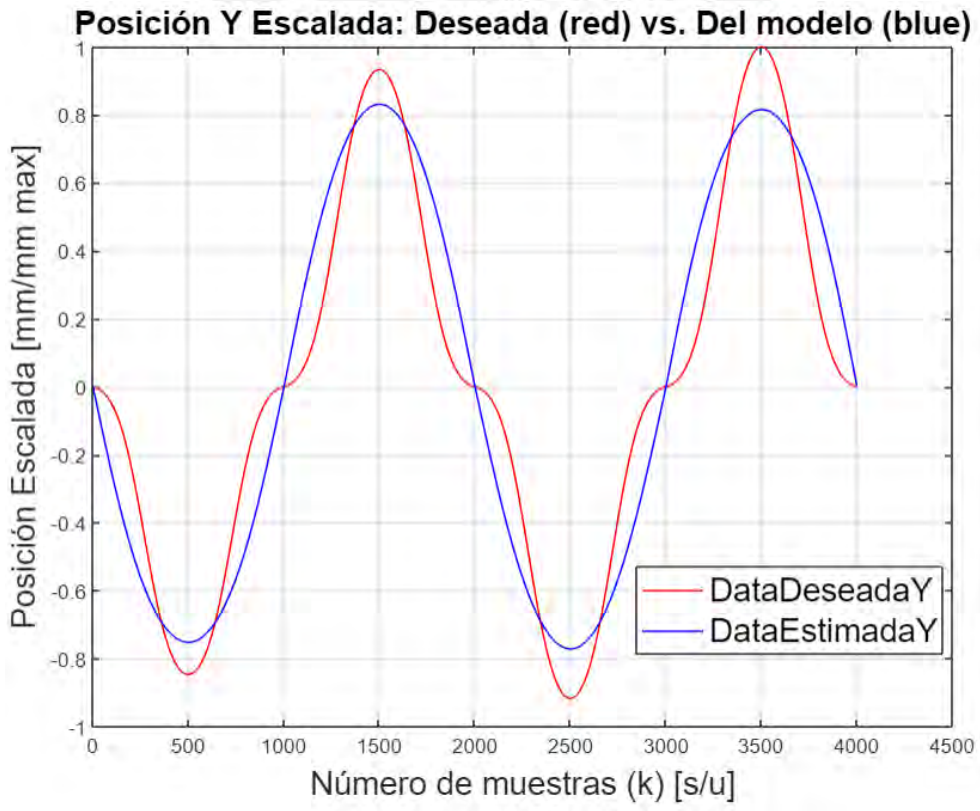


Figura 3.13. Posición Y deseada vs. del modelo en la etapa de validación

Capítulo IV: CONTROLADOR DE POSICIÓN DEL ENDOSCOPIO BLANDO

En el presente capítulo, se presenta el diseño del controlador de posición del efector final del cuerpo del endoscopio y el análisis de estabilidad del sistema en lazo cerrado. Se inicia con la descripción de la estructura del controlador, que está estaba basado en una red neuronal, así como las características del sistema en lazo cerrado. Luego, se detalla el algoritmo y la estrategia de entrenamiento que se utiliza para la obtención del controlador de posición. Posteriormente, se describe el procedimiento para el análisis de estabilidad del sistema en lazo cerrado. Finalmente, se presentan los diagramas de flujo y se describen los aspectos más relevantes de la programación para la implementación del algoritmo de entrenamiento del controlador y los cálculos correspondientes para el análisis de estabilidad del sistema en lazo cerrado.

4.1. Estructura del controlador y características del sistema en lazo cerrado

La estructura del controlador (Hunt K., et al, 1992) corresponde a una red neuronal pre-alimentada (Feedforward Neural Network en inglés) compuesta por tres capas de neuronas como se muestra en la Figura 4.1 dentro del bloque controlador. Las neuronas de la capa de entrada reciben las señales de error obtenidas al comparar las coordenadas x e y del vector de posición del sistema $\mathrm{r}_k = [x_k\ y_k\ ]^T$ con el vector de posición deseado $r_k^* = [x_k^*\ y_k^*\ ]^T$ del efector final del cuerpo del endoscopio en cada instante de tiempo k; adicionalmente, en la capa de entrada se tiene una neurona bias con valor igual a 1. De las neuronas de la capa de salida del controlador se obtiene el vector de presiones $u_k = [u1_k\ u2_k\ u3_k\ u4_k]$ en cada instante de tiempo k para las cuatro cámaras internas del cuerpo del endoscopio.

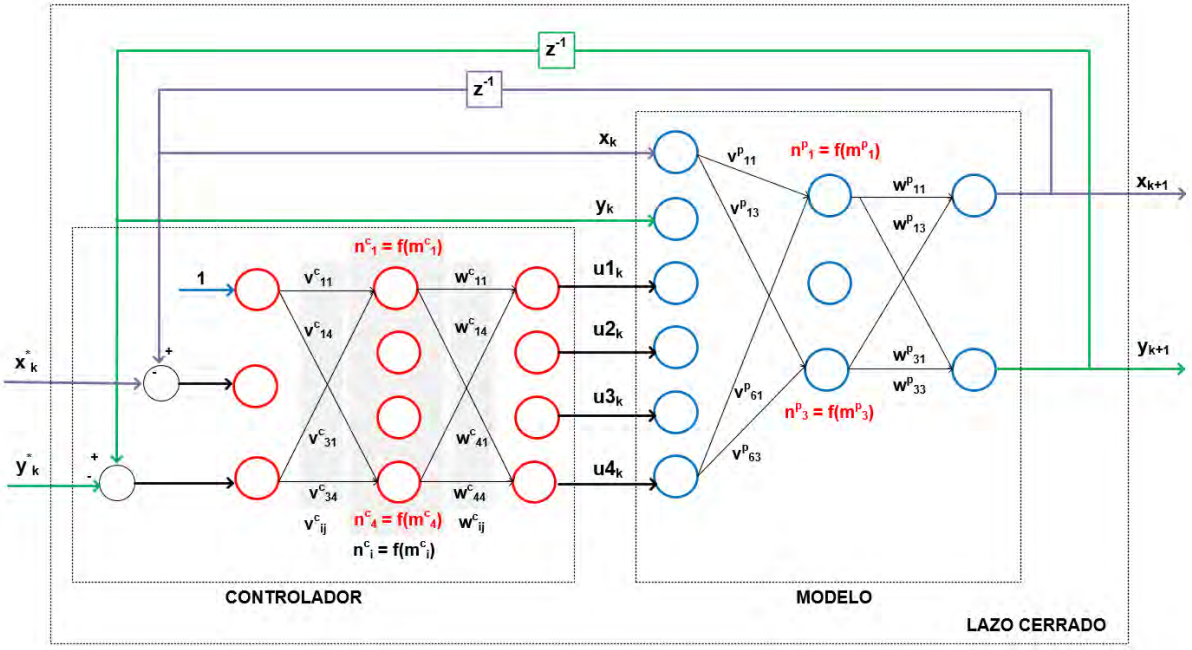


Figura 4.1. Estructura del controlador y del sistema en lazo cerrado

En la capa intermedia se considera cuatro neuronas, ya que con dicha cantidad se logra un buen resultado en el entrenamiento del controlador, así como reducir el costo de cómputo en su implementación como se presentará en el siguiente capítulo. Las funciones de activación $n_i^c = f(m_i^c)$ para cada neurona de la capa intermedia se definen por una función sigmoidea $f(m_i^c) = \frac{2}{1+e^{-m_i^c}} - 1$. Los rangos de los vectores de posición y presión son normalizados de forma tal que $\mathrm{r}_k, r_k^* \epsilon [-1, 1]$ y $u_k \epsilon [0, 1]$. Los pesos de coeficientes de conexión entre las capas de neuronas w^c y v^c se ordenan en dos matrices como se muestran en las ecuaciones (4.1) y (4.2) respectivamente. La matriz v^ccorresponde a la conexión entre las neuronas de la capa de entrada y la capa intermedia; por ejemplo, el coeficiente v_{32}^c corresponde a la conexión entre la neurona tres de la capa de entrada con la neurona dos de la capa intermedia. La matriz w^ccorresponde a la conexión entre las neuronas de la capa intermedia y la capa de salida; por ejemplo, el coeficiente w_{21}^ccorresponde a la conexión entre la neurona dos de la capa intermedia con la neurona uno de la capa de salida.

$$v^c = \begin{bmatrix} v_{11}^c & v_{12}^c & v_{13}^c & v_{14}^c \\ v_{21}^c & v_{22}^c & v_{23}^c & v_{24}^c \\ v_{31}^c & v_{32}^c & v_{33}^c & v_{34}^c \end{bmatrix} \dots (4.1), \qquad w = \begin{bmatrix} w_{11}^c & w_{12}^c & w_{13}^c & w_{14}^c \\ w_{21}^c & w_{22}^c & w_{23}^c & w_{24}^c \\ w_{31}^c & w_{32}^c & w_{33}^c & w_{34}^c \\ w_{41}^c & w_{42}^c & w_{43}^c & w_{44}^c \end{bmatrix} \dots (4.2)$$

Las ecuaciones que describen la relación entre las entradas y las salidas del controlador se pueden obtener a partir del cálculo de los vectores $m^c, n^c, f(m_i^c)$ y las matrices w^c, v^c que almacenan información de la estructura interna de la red neuronal del controlador. Primero de la ecuación (4.3), se obtiene el vector m^c , que contiene los valores de entrada para las neuronas de la capa intermedia, a partir del vector $[1 \ (r_k - r_k^*) \ ]^T$, que contiene los valores de las neuronas de la capa de entrada, y la matriz v^c, que contiene los coeficientes de conexión entre la capa de entrada e intermedia. Luego con la ecuación (4.4), se calcula el vector n_i^c, que contiene los valores de salida de las neuronas de la capa intermedia, a partir su respectiva función de activación $f(m_i^c)$ y señal de entrada m_i^c. Por último, con la ecuación (4.5), se obtiene el vector u_k, que contiene las señales de control, a partir del vector n^c y la matriz w^c, que contiene los coeficientes de conexión entre las neuronas de la capa intermedia y la capa de salida. El modelo del sistema desarrollado en el anterior capítulo, el cual está basado en una estructura de red neuronal recurrente con realimentación a la salida, es utilizado para el entrenamiento del controlador y se muestra dentro del bloque modelo de la Figura 4.1. El sistema en lazo cerrado, que está formado por la integración entre el controlador y el modelo se muestra dentro del bloque lazo cerrado de la Figura 4.1, donde el vector de entrada $r_k^* = [x_k^* \ y_k^*]^T$ corresponde a las coordenadas de posiciones deseadas y el vector de salida $r_{k+1} = [x_{k+1} \ y_{k+1}]^T$ corresponde a las coordenadas de posiciones estimadas por el modelo.

$$m^c = (v^c)^T \times [1 \quad (r_k - r_k^*)]^T \ldots (4.3) \qquad n_i^c = f(m_i^c) \ldots (4.4) \qquad u_k = (w^c)^T \times n^c \ldots (4.5)$$

Donde:

$$r_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, r = \begin{bmatrix} x_k^* \\ y_k^* \end{bmatrix}$$

Donde:

$$f(m_i^c) = \frac{2}{1 + e^{-m_i^c}} - 1$$

4.2. Algoritmo y estrategia de entrenamiento del controlador

La formulación matemática y el desarrollo del proceso de entrenamiento del controlador se pueden agrupar en los siguientes pasos. En primer lugar, se minimiza la función de costo J, mostrada en la ecuación (4.6), la cual está definida como el error cuadrático entre el vector de posición deseado r_k^* y el vector de posición r_k estimado por el modelo para una cantidad N de datos. Debido a que la señal de control u no es considerando en la ecuación, durante la ejecución del entrenamiento debe verificarse que el valor de u no supere su valor máximo. La ecuación (4.7) muestra el desarrollo de la expresión de la función de costo J considerando una matriz de pesos Q igual a la matriz identidad I de orden 2. El valor del vector de posición deseado r_k^* para cada instante de tiempo t_k se define a partir de la ubicación o trayectoria deseada y el valor del vector de posición estimado r_k se obtiene a partir de las ecuaciones de relación entre las entradas y salidas del sistema en lazo cerrado para cada instante de tiempo t_k. El proceso de optimización de la función de costo J tiene como objetivo encontrar los valores óptimos de las componentes de las matrices v^cy w^c, las cuales corresponden a los coeficientes de conexión entre las capas de neuronas del controlador.

$$J = \frac{1}{2}\sum_{k=0}^{N} (r_k - r_k^*)^T Q (r_k - r_k^*) \ldots (4.6) \qquad J = \frac{1}{2}\sum_{k=0}^{N} (x_k - x_k^*)^2 + (y_k - y_k^*)^2 \ldots (4.7)$$

Donde:

$$Q = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

En segundo lugar, se utiliza la gradiente descendente como método de optimización, cuya formulación se muestran en las ecuaciones (4.8) y (4.9) para cada coeficiente de conexión v_{ij}^c y w_{ij}^c respectivamente, donde n es la tasa de aprendizaje y $\frac{\overline{\partial J}}{\partial v_{ij}^c}, \frac{\overline{\partial J}}{\partial w_{ij}^c}$ son las derivadas parciales totales de la función de costo J respecto a cada coeficiente de conexión v_{ij}^c y w_{ij}^c. Estas últimas derivadas son calculadas a partir de las ecuaciones (4.10) y (4.11), en las cuales se considera los N datos de entrenamiento correspondiente a cada instante de tiempo t_k, donde $\frac{\overline{\partial r_k}}{\partial w_{ij}^c}$ es la derivada parcial total en el instante k de la posición estimada r_k respecto al coeficiente de conexión w_{ij}^c y $\frac{\overline{\partial r_k}}{\partial v_{ij}^c}$ es la derivada parcial total en el instante k de r_k respecto a v_{ij}^c.

$$v_{ij}^{c} = v_{ij}^{c} - n\frac{\overline{\partial J}}{\overline{\partial v_{ij}^{c}}} \ldots (4.8) \qquad w_{ij}^{c} = w_{ij}^{c} - n\frac{\overline{\partial J}}{\overline{\partial w_{ij}^{c}}} \ldots (4.9)$$

$$\frac{\overline{\partial J}}{\overline{\partial w_{ij}^{c}}} = \sum_{k=1}^{N} (r_k - r_k^*)^T Q \frac{\overline{\partial r_k}}{\overline{\partial w_{ij}^{c}}} \ldots (4.10) \qquad \frac{\overline{\partial J}}{\overline{\partial v_{ij}^{c}}} = \sum_{k=1}^{N} (r_k - r_k^*)^T Q \frac{\overline{\partial r_k}}{\overline{\partial v_{ij}^{c}}} \ldots (4.11)$$

En tercer lugar, para determinar las derivadas parciales totales $\frac{\overline{\partial r_k}}{\overline{\partial w_{ij}^{c}}}$ y $\frac{\overline{\partial r_k}}{\overline{\partial v_{ij}^{c}}}$ se utiliza el algoritmo de retropropagación dinámica (Dynamic Back Propagation en inglés), cuya formulación se muestran en las ecuaciones (4.12) y (4.13) respectivamente, donde debido a la naturaleza recursiva del algoritmo el cálculo de las derivadas parciales totales $\frac{\overline{\partial r_{k+1}}}{\overline{\partial w_{ij}^{c}}}$ y $\frac{\overline{\partial r_{k+1}}}{\overline{\partial v_{ij}^{c}}}$ en el instante de tiempo $k+1$ depende de las derivadas parciales totales $\frac{\overline{\partial r_k}}{\overline{\partial w_{ij}^{c}}}$ y $\frac{\overline{\partial r_k}}{\overline{\partial v_{ij}^{c}}}$ en el instante de tiempo k. También, se requiere el cálculo de las derivadas parciales $\frac{\partial r_{k+1}}{\partial u_k}$ y $\frac{\partial r_{k+1}}{\partial r_k}$, las cuales se obtienen de las ecuaciones de relación entrada y salida del modelo del sistema y las derivas parciales simples $\frac{\partial u_k}{\partial r_k}$, $\frac{\partial u_k}{\partial w_{ij}^{c}}$ y $\frac{\partial u_k}{\partial v_{ij}^{c}}$, las cuales se obtienen de las ecuaciones de relación entrada y salida del controlador del sistema.

$$\frac{\overline{\partial r_{k+1}}}{\overline{\partial w_{ij}^{c}}} = \frac{\partial r_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial w_{ij}^{c}} + \left(\frac{\partial r_{k+1}}{\partial r_k} + \frac{\partial r_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial r_k}\right)\frac{\overline{\partial r_k}}{\overline{\partial w_{ij}^{c}}} \ldots (4.12)$$

$$\frac{\overline{\partial r_{k+1}}}{\overline{\partial v_{ij}^{c}}} = \frac{\partial r_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial v_{ij}^{c}} + \left(\frac{\partial r_{k+1}}{\partial r_k} + \frac{\partial r_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial r_k}\right)\frac{\overline{\partial r_k}}{\overline{\partial v_{ij}^{c}}} \ldots (4.13)$$

Donde:

$$\frac{\overline{\partial r_{k+1}}}{\overline{\partial w_{ij}^{c}}} = \begin{bmatrix} \frac{\overline{\partial x_{k+1}}}{\overline{\partial w_{ij}^{c}}} \\ \frac{\overline{\partial y_{k+1}}}{\overline{\partial w_{ij}^{c}}} \end{bmatrix}, \frac{\overline{\partial r_{k+1}}}{\overline{\partial v_{ij}^{c}}} = \begin{bmatrix} \frac{\overline{\partial x_{k+1}}}{\overline{\partial v_{ij}^{c}}} \\ \frac{\overline{\partial y_{k+1}}}{\overline{\partial v_{ij}^{c}}} \end{bmatrix}, \frac{\overline{\partial r_k}}{\overline{\partial w_{ij}^{c}}} = \begin{bmatrix} \frac{\overline{\partial x_k}}{\overline{\partial w_{ij}^{c}}} \\ \frac{\overline{\partial y_k}}{\overline{\partial w_{ij}^{c}}} \end{bmatrix}, \frac{\overline{\partial r_k}}{\overline{\partial v_{ij}^{c}}} = \begin{bmatrix} \frac{\overline{\partial x_k}}{\overline{\partial v_{ij}^{c}}} \\ \frac{\overline{\partial y_k}}{\overline{\partial v_{ij}^{c}}} \end{bmatrix}, \frac{\partial u_k}{\partial w_{ij}^{c}} = \begin{bmatrix} \frac{\partial u1_k}{\partial w_{ij}^{c}} \\ \frac{\partial u2_k}{\partial w_{ij}^{c}} \\ \frac{\partial u3_k}{\partial w_{ij}^{c}} \\ \frac{\partial u4_k}{\partial w_{ij}^{c}} \end{bmatrix}, \frac{\partial u_k}{\partial v_{ij}^{c}} = \begin{bmatrix} \frac{\partial u1_k}{\partial v_{ij}^{c}} \\ \frac{\partial u2_k}{\partial v_{ij}^{c}} \\ \frac{\partial u3_k}{\partial v_{ij}^{c}} \\ \frac{\partial u4_k}{\partial v_{ij}^{c}} \end{bmatrix}$$

$$\frac{\partial r_{k+1}}{\partial r_k} = \begin{bmatrix} \frac{\partial x_{k+1}}{\partial x_k} & \frac{\partial x_{k+1}}{\partial y_k} \\ \frac{\partial y_{k+1}}{\partial x_k} & \frac{\partial y_{k+1}}{\partial y_k} \end{bmatrix}, \frac{\partial r_{k+1}}{\partial u_k} = \begin{bmatrix} \frac{\partial x_{k+1}}{\partial u1_k} & \frac{\partial x_{k+1}}{\partial u2_k} & \frac{\partial x_{k+1}}{\partial u3_k} & \frac{\partial x_{k+1}}{\partial u4_k} \\ \frac{\partial y_{k+1}}{\partial u1_k} & \frac{\partial y_{k+1}}{\partial u2_k} & \frac{\partial y_{k+1}}{\partial u3_k} & \frac{\partial y_{k+1}}{\partial u4_k} \end{bmatrix}, \frac{\partial u_k}{\partial r_k} = \begin{bmatrix} \frac{\partial u1_k}{\partial x_k} & \frac{\partial u1_k}{\partial y_k} \\ \frac{\partial u2_k}{\partial x_k} & \frac{\partial u2_k}{\partial y_k} \\ \frac{\partial u3_k}{\partial x_k} & \frac{\partial u3_k}{\partial y_k} \\ \frac{\partial u4_k}{\partial x_k} & \frac{\partial u4_k}{\partial y_k} \end{bmatrix}$$

El cálculo del primer grupo de derivadas $\frac{\partial r_{k+1}}{\partial r_k}$ y $\frac{\partial r_{k+1}}{\partial u_k}$ se muestran en las ecuaciones (3.11) y (4.14), las cuales han sido derivadas a partir de las ecuaciones del modelo (3.3), (3.4) y (3.5) mostradas en el capítulo anterior.

$$\frac{\partial r_{k+1}}{\partial u_k} = (w^p)^T \times diag\left(f'\left(m_i^p\right)\right) \times \left(v^p(3:6,:)\right)^T \ldots (4.14)$$

Donde:

$$f'\left(m_i^p\right) = \frac{1 - \left(n_i^p\right)^2}{2}$$

El cálculo del segundo grupo de derivadas $\frac{\partial u_k}{\partial r_k}$, $\frac{\partial u_k}{\partial w_{ij}^c}$ y $\frac{\partial u_k}{\partial v_{ij}^c}$ se obtienen a partir de la ecuación la ecuación (4.15) hasta la (4.23), las cuales han sido obtenidas a partir de las ecuaciones de relaciones de entrada y salidas del controlador mostradas desde la ecuación (4.1) hasta la (4.3).

$$\frac{\partial u_k}{\partial r_k} = (w^c)^T \times diag\left(f'(m_i^c)\right) \times (v^c(2:3,:))^T \ldots (4.15)$$

Donde:

$$f'(m_i^c) = \frac{1 - (n_i^c)^2}{2}$$

$$\frac{\partial u1_{k+1}}{\partial w_{ij}^c} = n_i \; \forall j = 1, \qquad \frac{\partial u1_{k+1}}{\partial w_{ij}^c} = 0 \; \forall j \neq 1. . (4.16)$$

$$\frac{\partial u2_{k+1}}{\partial w_{ij}^c} = n_i \; \forall j = 2, \qquad \frac{\partial u2_{k+1}}{\partial w_{ij}^c} = 0 \; \forall j \neq 2. . (4.17)$$

$$\frac{\partial u3_{k+1}}{\partial w_{ij}^c} = n_i \; \forall j = 3, \qquad \frac{\partial u3_{k+1}}{\partial w_{ij}^c} = 0 \; \forall j \neq 3. . (4.18)$$

$$\frac{\partial u4_{k+1}}{\partial w_{ij}^c} = n_i \; \forall j = 4, \qquad \frac{\partial u4_{k+1}}{\partial w_{ij}^c} = 0 \; \forall j \neq 4. . (4.19)$$

$$\frac{\partial u1_k}{\partial v_{ij}^c} = in_{red(i)} \times w_{j1}^c \times f'\left(m_j^c\right) \ldots (4.20)$$

$$\frac{\partial u2_k}{\partial v_{ij}^c} = in_{red(i)} \times w_{j2}^c \times f'\left(m_j^c\right) \ldots (4.21)$$

$$\frac{\partial u3_k}{\partial v_{ij}^c} = in_{red(i)} \times w_{j3}^c \times f'\left(m_j^c\right) \ldots (4.22)$$

$$\frac{\partial u4_k}{\partial v_{ij}^c} = in_{red(i)} \times w_{j4}^c \times f'\left(m_j^c\right) \ldots (4.23)$$

Donde:

$$in_{red(i)} = [1 \quad (x_k - x_k^*) \quad (y_k - y_k^*)]^T, \qquad f'\left(m_j^c\right) = \frac{1 - \left(n_j^c\right)^2}{2}$$

La estrategia de entrenamiento está orientada a obtener los coeficientes de conexión w_{ij}^c y v_{ij}^c del controlador que minimicen la función de costo J para ello se realiza la partición del rango de las coordenadas de posición x e y respectivamente como se muestra en la Figura 4.2. Ambos ejes x e y se pueden dividir en una cantidad M y N de particiones respectivamente, lo cual genera un total de $M \times N$ subregiones para toda la región de movimiento del cuerpo del endoscopio; donde, la subregión a_{ij} corresponde a la partición i de la coordenada X y a la partición j de la coordenada Y. Para cada subregión definida se selecciona una posición representativa r_{ij}.

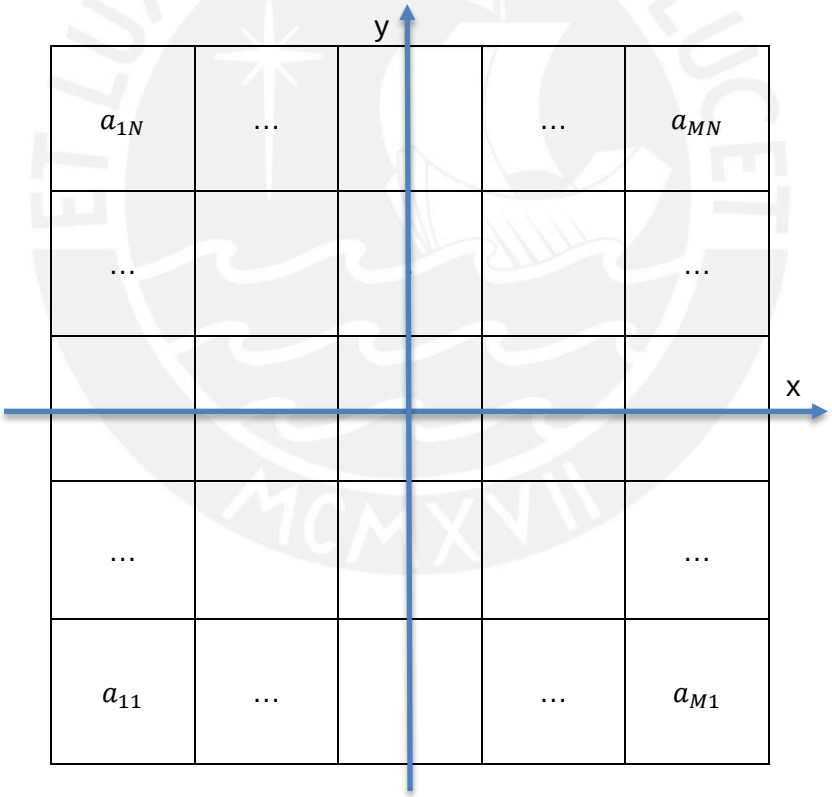


Figura 4.2. Partición del rango de las coordenadas de posición x e y

El entrenamiento del controlador se puede realizar considerando varias posiciones iniciales r_0 y varias posiciones de deseadas r^*, donde sus respectivos valores son asignados a partir de las posiciones representativas r_{ij}. En el próximo capítulo, se presentará la configuración del entrenamiento que permite alcanzar los mejores resultados en el entrenamiento del controlador.

Las especificaciones deseadas del controlador se definen a partir las características de la respuesta deseada del sistema en lazo cerrado en el dominio del tiempo definidas por el tiempo de establecimiento (T_{es}), el máximo sobreimpulso (M_p), el error en estado estacionario (e_s) y el tipo de respuesta más adecuada a la aplicación, cuyos valores se muestran en la Tabla 4.1.

Tabla 4.1. Especificaciones deseadas de la respuesta del sistema en el dominio del tiempo

Parámetro	**Valor**
T_{es}	100 ms
M_p	10%
e_s	2%
Tipo de respuesta	Sobreamortiguada

4.3. Análisis de estabilidad del sistema en lazo cerrado

La evaluación de estabilidad para el sistema en lazo cerrado formado por el controlador más el modelo del sistema ambos basados en redes neuronales (Tanaka K., 1996) se basa en el análisis de las condiciones de estabilidad para una clase de sistema no lineal que puede representarse como un tipo de inclusiones diferenciales lineales, Linear Differential Inclusion (LDI) en inglés, cuya representación se muestra en la ecuación (4.24).

$$x(k+1) = Ax(k)$$

$$A = \sum_{i=1}^{r} h_i A_i \dots (4.24)$$

Donde:

$$r \in z^+, x(k) = [x_1(k)\ x_2(k) \dots\ x_n(k)]^T$$

$$h_i \geq 0, \sum_{i=1}^{r} h_i = 1$$

El equilibrio del sistema descrito por la ecuación (4.24) es asintóticamente estable alrededor de la posición de equilibrio $r_k^* = 0$ sí existe una matriz común positiva definida P que cumpla la condición mostrada en la desigualdad (4.25). Así mismo, otro resultado importante se puede obtener analizando la estabilidad de las matrices $A_i A_J\ \forall\ i, j = 1,2,\dots,r$ que en caso de una de ellas sea inestable se puede concluir la no existencia de una matriz P común positiva definida.

$$A_i^T P A_i - P < 0\ \ \forall i = 1, 2, \dots, r \dots (4.25)$$

Para aplicar las condiciones de estabilidad es necesario obtener tanto para el modelo como para el controlador sus respectivas representaciones LDI. La Figura 4.3 muestra la

representación del sistema en lazo cerrado considerado para el análisis de estabilidad, de donde se derivan las ecuaciones que relacionan las entradas y salidas del modelo y el controlador. Se inicia el proceso con el modelo del sistema, el cual es descrito desde la ecuación (4.26) hasta la ecuación (4.28), donde $x_k, y_k, u1_k, u2_k, u3_k\ y\ u4_k$ son las entradas, x_{k+1} e y_{k+1} son las salidas, w_{1ij}^p y w_{2ij}^p son los coeficientes de conexión, v_{1i}^p son las entradas de las neuronas de la capa intermedia y f_{1i}^p son las funciones de activación de las neuronas de la capa intermedia.

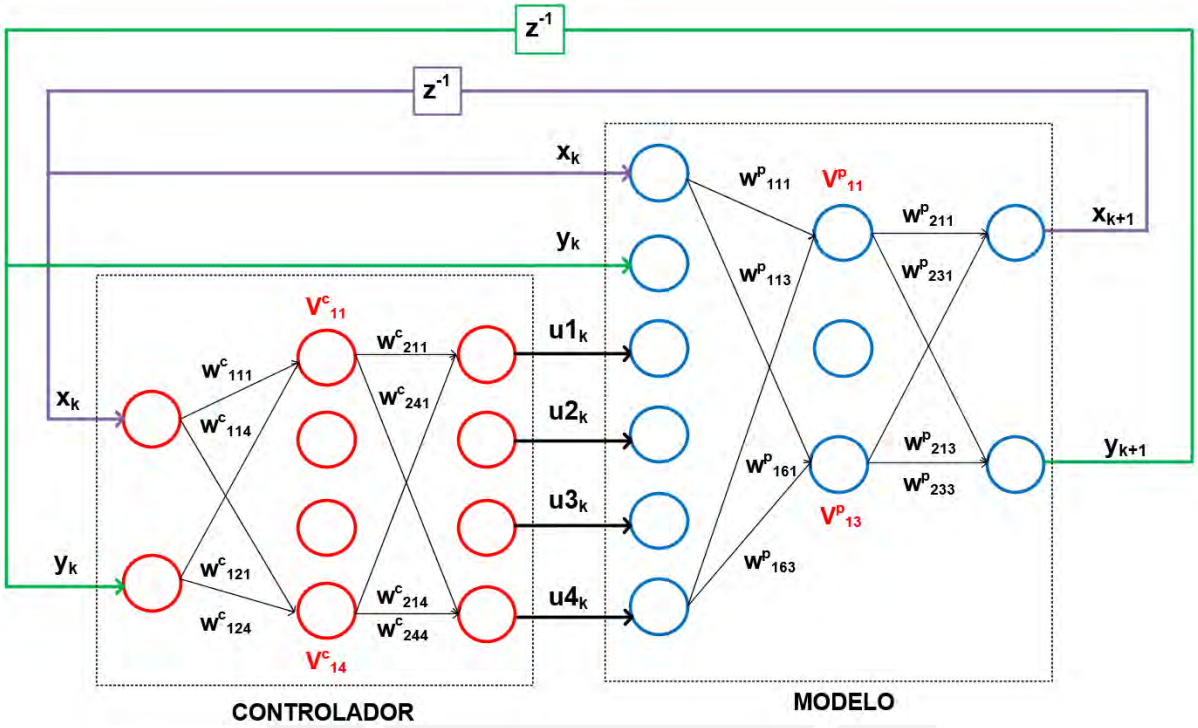


Figura 4.3. Estructura del sistema en lazo cerrado para el análisis de estabilidad

$$v_{1i}^p = w_{11i}^p x_k + w_{12i}^p y_k + w_{13i}^p u1_k + w_{14i}^p u2_k + w_{15i}^p u3_k + w_{16i}^p u4_k \ldots (4.26)$$

$$x_{k+1} = w_{211}^p f_{11}^p(v_{11}^p) + w_{212}^p f_{12}^p(v_{12}^p) + w_{213}^p f_{13}^p(v_{13}^p) \ldots (4.27)$$

$$y_{k+1} = w_{221}^p f_{11}^p(v_{11}^p) + w_{222}^p f_{12}^p(v_{12}^p) + w_{223}^p f_{13}^p(v_{13}^p) \ldots (4.28)$$

La representación LDI para las funciones de activación se muestra en la ecuación (4.29).

$$f_{1i}^p(v_{1i}^p) = \left(h_{1i1}^p(k) g_{1i1}^p + h_{1i2}^p(k) g_{1i2}^p\right) v_{1i}^p \ \forall i = 1,2,3 \ldots (4.29)$$

Donde:

$$f_{1i}^p(v_{1i}^p) = \frac{2}{1 + e^{-v_{1i}^p}} - 1$$

$$g_{1i1}^p = \min_{v_{1i}^p} f_{1i}'^p(v_{1i}^p) = 0, g_{1i2}^p = \max_{v_{1i}^p} f_{1i}'^p(v_{1i}^p) = 0.5$$

La coordenada x_{k+1} definida por la ecuación (4.27) puede ser expresada por la ecuación (4.30). Usando la representación LDI de las funciones de activación f_{1i}^{p} se obtiene la ecuación (4.31), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.32). Luego, para simplificar esta última expresión se define los coeficientes a_{abc}^{11}, a_{abc}^{12}, b_{abc}^{11}, b_{abc}^{12}, b_{abc}^{13} y b_{abc}^{14}.

$$x_{k+1} = \sum_{j=1}^{3} w_{21j}^{p} f_{1j}^{p}(v_{1j}^{p}) \dots (4.30)$$

$$x_{k+1} = \sum_{j=1}^{3} w_{21j}^{p} \left\{ h_{1j1}^{p}(k) g_{1j1}^{p} + h_{1j2}^{p}(k) g_{1j2}^{p} \right\} v_{1j}^{p} \dots (4.31)$$

$$\begin{aligned} x_{k+1} = \sum_{a=1}^{2} \sum_{b=1}^{2} \sum_{c=1}^{2} h_{11a}^{p} h_{12b}^{p} h_{13c}^{p} [& \{ g_{11a}^{p} w_{211}^{p} w_{111}^{p} + g_{12b}^{p} w_{212}^{p} w_{112}^{p} + g_{13c}^{p} w_{213}^{p} w_{113}^{p} \} x_k + \\ & \{ g_{11a}^{p} w_{211}^{p} w_{121}^{p} + g_{12b}^{p} w_{212}^{p} w_{122}^{p} + g_{13c}^{p} w_{213}^{p} w_{123}^{p} \} y_k + \\ & \{ g_{11a}^{p} w_{211}^{p} w_{131}^{p} + g_{12b}^{p} w_{212}^{p} w_{132}^{p} + g_{13c}^{p} w_{213}^{p} w_{133}^{p} \} u1_k + \\ & \{ g_{11a}^{p} w_{211}^{p} w_{141}^{p} + g_{12b}^{p} w_{212}^{p} w_{142}^{p} + g_{13c}^{p} w_{213}^{p} w_{143}^{p} \} u2_k + \\ & \{ g_{11a}^{p} w_{211}^{p} w_{151}^{p} + g_{12b}^{p} w_{212}^{p} w_{152}^{p} + g_{13c}^{p} w_{213}^{p} w_{153}^{p} \} u3_k + \\ & \{ g_{11a}^{p} w_{211}^{p} w_{161}^{p} + g_{12b}^{p} w_{212}^{p} w_{162}^{p} + g_{13c}^{p} w_{213}^{p} w_{163}^{p} \} u4_k \dots (4.32) \end{aligned}$$

Donde:

$$a_{abc}^{11} = g_{11a}^{p} w_{211}^{p} w_{111}^{p} + g_{12b}^{p} w_{212}^{p} w_{112}^{p} + g_{13c}^{p} w_{213}^{p} w_{113}^{p}$$
$$a_{abc}^{12} = g_{11a}^{p} w_{211}^{p} w_{121}^{p} + g_{12b}^{p} w_{212}^{p} w_{122}^{p} + g_{13c}^{p} w_{213}^{p} w_{123}^{p}$$
$$b_{abc}^{11} = g_{11a}^{p} w_{211}^{p} w_{131}^{p} + g_{12b}^{p} w_{212}^{p} w_{132}^{p} + g_{13c}^{p} w_{213}^{p} w_{133}^{p}$$
$$b_{abc}^{12} = g_{11a}^{p} w_{211}^{p} w_{141}^{p} + g_{12b}^{p} w_{212}^{p} w_{142}^{p} + g_{13c}^{p} w_{213}^{p} w_{143}^{p}$$
$$b_{abc}^{13} = g_{11a}^{p} w_{211}^{p} w_{151}^{p} + g_{12b}^{p} w_{212}^{p} w_{152}^{p} + g_{13c}^{p} w_{213}^{p} w_{153}^{p}$$
$$b_{abc}^{14} = g_{11a}^{p} w_{211}^{p} w_{161}^{p} + g_{12b}^{p} w_{212}^{p} w_{162}^{p} + g_{13c}^{p} w_{213}^{p} w_{163}^{p}$$

La coordenada y_{k+1} definida por la ecuación (4.28) puede ser expresada por la ecuación (4.33). Usando la representación LDI de las funciones de activación f_{1i}^{p} se obtiene la ecuación (4.34), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.35). Luego, para simplificar la última expresión se define los coeficientes a_{abc}^{21}, a_{abc}^{22}, b_{abc}^{21}, b_{abc}^{22}, b_{abc}^{23} y b_{abc}^{24}.

$$y_{k+1} = \sum_{j=1}^{3} w_{22j}^{p} f_{1j}^{p}(v_{1j}^{p}) \dots (4.33)$$

$$y_{k+1} = \sum_{j=1}^{3} w_{22j}^{p} \left\{ h_{1j1}^{p}(k) g_{1j1}^{p} + h_{1j2}^{p}(k) g_{1j2}^{p} \right\} v_{1j}^{p} \dots (4.34)$$

$$y_{k+1} = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2} h_{11a}^{p} h_{12b}^{p} h_{13c}^{p} [\{g_{11a}^{p} w_{221}^{p} w_{111}^{p} + g_{12b}^{p} w_{222}^{p} w_{112}^{p} + g_{13c}^{p} w_{223}^{p} w_{113}^{p}\} x_k +$$
$$\{g_{11a}^{p} w_{221}^{p} w_{121}^{p} + g_{12b}^{p} w_{222}^{p} w_{122}^{p} + g_{13c}^{p} w_{223}^{p} w_{123}^{p}\} y_k +$$
$$\{g_{11a}^{p} w_{221}^{p} w_{131}^{p} + g_{12b}^{p} w_{222}^{p} w_{132}^{p} + g_{13c}^{p} w_{223}^{p} w_{133}^{p}\} u1_k +$$
$$\{g_{11a}^{p} w_{221}^{p} w_{141}^{p} + g_{12b}^{p} w_{222}^{p} w_{142}^{p} + g_{13c}^{p} w_{223}^{p} w_{143}^{p}\} u2_k +$$
$$\{g_{11a}^{p} w_{221}^{p} w_{151}^{p} + g_{12b}^{p} w_{222}^{p} w_{152}^{p} + g_{13c}^{p} w_{223}^{p} w_{153}^{p}\} u3_k +$$
$$\{g_{11a}^{p} w_{221}^{p} w_{161}^{p} + g_{12b}^{p} w_{222}^{p} w_{162}^{p} + g_{13c}^{p} w_{223}^{p} w_{163}^{p}\} u4_k \ldots (4.35)$$

Donde:

$$a_{abc}^{21} = g_{11a}^{p} w_{221}^{p} w_{111}^{p} + g_{12b}^{p} w_{222}^{p} w_{112}^{p} + g_{13c}^{p} w_{223}^{p} w_{113}^{p}$$
$$a_{abc}^{22} = g_{11a}^{p} w_{221}^{p} w_{121}^{p} + g_{12b}^{p} w_{222}^{p} w_{122}^{p} + g_{13c}^{p} w_{223}^{p} w_{123}^{p}$$
$$b_{abc}^{21} = g_{11a}^{p} w_{221}^{p} w_{131}^{p} + g_{12b}^{p} w_{222}^{p} w_{132}^{p} + g_{13c}^{p} w_{223}^{p} w_{133}^{p}$$
$$b_{abc}^{22} = g_{11a}^{p} w_{221}^{p} w_{141}^{p} + g_{12b}^{p} w_{222}^{p} w_{142}^{p} + g_{13c}^{p} w_{223}^{p} w_{143}^{p}$$
$$b_{abc}^{23} = g_{11a}^{p} w_{221}^{p} w_{151}^{p} + g_{12b}^{p} w_{222}^{p} w_{152}^{p} + g_{13c}^{p} w_{223}^{p} w_{153}^{p}$$
$$b_{abc}^{24} = g_{11a}^{p} w_{221}^{p} w_{161}^{p} + g_{12b}^{p} w_{222}^{p} w_{162}^{p} + g_{13c}^{p} w_{223}^{p} w_{163}^{p}$$

Las ecuaciones (4.32) y (4.35) correspondientes a las posiciones x_{k+1} e y_{k+1} respectivamente se representa forma matricial en las ecuaciones (4.36) y (4.37) donde todos los coeficientes de las matrices A_{abc}, B_{abc} y A_i, b_i son conocidos.

$$r_{(k+1)} = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2} h_{11a}^{p}(k) h_{12b}^{p}(k) h_{13c}^{p}(k) \times \{A_{abc} r_{(k)} + b_{abc} u_{(k)}\} \ldots (4.36)$$

Donde:

$$A_{abc} = \begin{bmatrix} a_{abc}^{11} & a_{abc}^{12} \\ a_{abc}^{21} & a_{abc}^{22} \end{bmatrix}, b_{abc} = \begin{bmatrix} b_{abc}^{11} & b_{abc}^{12} & b_{abc}^{13} & b_{abc}^{14} \\ b_{abc}^{21} & b_{abc}^{22} & b_{abc}^{23} & b_{abc}^{24} \end{bmatrix}$$

$$r_{(k+1)=} \sum_{i=1}^{8} h_i^{p}(k)\{A_i r_{(k)} + b_i u_{(k)}\} \ldots (4.37)$$

El proceso continúa obteniendo la representación LDI del controlador. Las relaciones de entrada y salida del controlador son definidas desde la ecuación (4.38) hasta la ecuación (4.42), donde x_k e y_k son las entradas, u_{1k}, u_{2k}, u_{3k} y u_{4k} son las salidas, w_{1ij}^{c}, w_{2ij}^{c} son los coeficientes de conexión y v_{1i}^{c}, f_{1i}^{c} son las entradas y las funciones de activación de las neuronas de la capa intermedia respectivamente.

$$v_{1i}^{c} = w_{11i}^{c} x(k) + w_{12i}^{c} y(k) \ldots (4.38)$$
$$u_1(k) = w_{211}^{c} f_{11}^{c}(v_{11}^{c}) + w_{212}^{c} f_{12}^{c}(v_{12}^{c}) + w_{213}^{c} f_{13}^{c}(v_{13}^{c}) + w_{214}^{c} f_{14}^{c}(v_{14}^{c}) \ldots (4.39)$$

$$u_2(k) = w_{221}^c f_{11}^c(v_{11}^c) + w_{222}^c f_{12}^c(v_{12}^c) + w_{223}^c f_{13}^c(v_{13}^c) + w_{224}^c f_{14}^c(v_{14}^c) \dots (4.40)$$
$$u_3(k) = w_{231}^c f_{11}^c(v_{11}^c) + w_{232}^c f_{12}^c(v_{12}^c) + w_{233}^c f_{13}^c(v_{13}^c) + w_{234}^c f_{14}^c(v_{14}^c) \dots (4.41)$$
$$u_4(k) = w_{241}^c f_{11}^c(v_{11}^c) + w_{242}^c f_{12}^c(v_{12}^c) + w_{243}^c f_{13}^c(v_{13}^c) + w_{244}^c f_{14}^c(v_{14}^c) \dots (4.42)$$

La representación LDI para las funciones de activación puede ser representada de acuerdo a la ecuación (4.43).

$$f_{1i}^c(v_{1i}^c) = (h_{1i1}^c(k) g_{1i1}^c + h_{1i2}^c(k) g_{1i2}^c) v_{1i}^c \forall i = 1, 2, 3, 4 \dots (4.43)$$

Donde:

$$f_{1i}^c(v_{1i}^c) = \frac{2}{1 + e^{-v_{1i}^c}} - 1$$

$$g_{1i1}^c = \min_{v_{1i}^c} f_{1i}'^c(v_{1i}^c) = 0,\ g_{1i2}^c = \max_{v_{1i}^c} f_{1i}'^c(v_{1i}^c) = 0.5$$

La variable de control $u_1(k)$ definida por la ecuación (4.39) puede ser expresada por la ecuación (4.44). Usando la representación LDI de las funciones de activación f_{1i}^c se obtiene la ecuación (4.45), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.46). Luego, para simplificar la última expresión se define los coeficientes f_{abc}^{11} y f_{abc}^{12}.

$$u_1(k) = \sum_{j=1}^{4} w_{21j}^c f_{1j}^c(v_{1j}^c) \dots (4.44)$$

$$u_1(k) = \sum_{j=1}^{4} w_{21j}^c \{h_{1j1}^c(k) g_{1j1}^c + h_{1j2}^c(k) g_{1j2}^c\} v_{1j}^c \dots (4.45)$$

$$u_1(k) = \sum_{a=1}^{2} \sum_{b=1}^{2} \sum_{c=1}^{2} \sum_{d=1}^{2} h_{11a}^c h_{12b}^c h_{13c}^c h_{14d}^c$$

$$\{g_{11a}^c w_{211}^c w_{111}^c + g_{12b}^c w_{212}^c w_{112}^c + g_{13c}^c w_{213}^c w_{113}^c + g_{14d}^c w_{214}^c w_{114}^c\} x(k)$$

$$\{g_{11a}^c w_{211}^c w_{121}^c + g_{12b}^c w_{212}^c w_{122}^c + g_{13c}^c w_{213}^c w_{123}^c + g_{14d}^c w_{214}^c w_{124}^c\} y(k) \dots (4.46)$$

Donde:

$$f_{abcd}^{11} = g_{11a}^c w_{211}^c w_{111}^c + g_{12b}^c w_{212}^c w_{112}^c + g_{13c}^c w_{213}^c w_{113}^c + g_{14d}^c w_{214}^c w_{114}^c$$
$$f_{abcd}^{12} = g_{11a}^c w_{211}^c w_{121}^c + g_{12b}^c w_{212}^c w_{122}^c + g_{13c}^c w_{213}^c w_{123}^c + g_{14d}^c w_{214}^c w_{124}^c$$

La variable de control $u_2(k)$ definida por la ecuación (4.40) puede ser expresada por la ecuación (4.47). Usando la representación LDI de las funciones de activación f_{1i}^c se obtiene la ecuación (4.48), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.49). Luego, para simplificar la última expresión se define los coeficientes f_{abc}^{21} y f_{abc}^{22}.

$$u_2(k) = \sum_{j=1}^{4} w_{22j}^c f_{1j}^c(v_{1j}^c) \dots (4.47)$$

$$u_2(k) = \sum_{j=1}^{4} w_{22j}^c \{h_{1j1}^c(k) g_{1j1}^c + h_{1j2}^c(k) g_{1j2}^c\} v_{ij}^c \dots (4.48)$$

$$u_2(k) = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2}\sum_{d=1}^{2} h_{11a}^c h_{12b}^c h_{13c}^c h_{14d}^c$$

$$\{g_{11a}^c w_{221}^c w_{111}^c + g_{12b}^c w_{222}^c w_{112}^c + g_{13c}^c w_{223}^c w_{113}^c + g_{14d}^c w_{224}^c w_{114}^c\} x(k)$$

$$\{g_{11a}^c w_{221}^c w_{121}^c + g_{12b}^c w_{222}^c w_{122}^c + g_{13c}^c w_{223}^c w_{123}^c + g_{14d}^c w_{224}^c w_{124}^c\} y(k) \dots (4.49)$$

Donde:

$$f_{abcd}^{21} = g_{11a}^c w_{221}^c w_{111}^c + g_{12b}^c w_{222}^c w_{112}^c + g_{13c}^c w_{223}^c w_{113}^c + g_{14d}^c w_{224}^c w_{114}^c$$

$$f_{abcd}^{22} = g_{11a}^c w_{221}^c w_{121}^c + g_{12b}^c w_{222}^c w_{122}^c + g_{13c}^c w_{223}^c w_{123}^c + g_{14d}^c w_{224}^c w_{124}^c$$

La variable de control $u_3(k)$ definida por la ecuación (4.41) puede ser expresada por la ecuación (4.50). Usando la representación LDI de las funciones de activación f_{1i}^c se obtiene la ecuación (4.51), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.52). Luego, para simplificar la última expresión se define los coeficientes f_{abc}^{31} y f_{abc}^{32}.

$$u_3(k) = \sum_{j=1}^{4} w_{23j}^c f_{1j}^c(v_{1j}^c) \dots (4.50)$$

$$u_3(k) = \sum_{j=1}^{4} w_{23j}^c \{h_{1j1}^c(k) g_{1j1}^c + h_{1j2}^c(k) g_{1j2}^c\} v_{1j}^c \dots (4.51)$$

$$u_3(k) = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2}\sum_{d=1}^{2} h_{11a}^c h_{12b}^c h_{13c}^c h_{14d}^c$$

$$\{g_{11a}^c w_{231}^c w_{111}^c + g_{12b}^c w_{232}^c w_{112}^c + g_{13c}^c w_{233}^c w_{113}^c + g_{14d}^c w_{234}^c w_{114}^c\} x(k)$$

$$\{g_{11a}^c w_{231}^c w_{121}^c + g_{12b}^c w_{232}^c w_{122}^c + g_{13c}^c w_{233}^c w_{123}^c + g_{14d}^c w_{234}^c w_{124}^c\} y(k) \dots (4.52)$$

Donde:

$$f_{abcd}^{31} = g_{11a}^c w_{231}^c w_{111}^c + g_{12b}^c w_{232}^c w_{112}^c + g_{13c}^c w_{233}^c w_{113}^c + g_{14d}^c w_{234}^c w_{114}^c$$

$$f_{abcd}^{32} = g_{11a}^c w_{231}^c w_{121}^c + g_{12b}^c w_{232}^c w_{122}^c + g_{13c}^c w_{233}^c w_{123}^c + g_{14d}^c w_{234}^c w_{124}^c$$

La variable de control $u_4(k)$ definida por la ecuación (4.42) puede ser expresada por la ecuación (4.53). Usando la representación LDI de las funciones de activación f_{1i}^c se obtiene la ecuación (4.54), la cual puede ser reordenada a la expresión mostrada en la ecuación (4.55). Luego, para simplificar la última expresión se define los coeficientes f_{abc}^{41} y f_{abc}^{42}.

$$u_4(k) = \sum_{j=1}^{4} w_{24j}^c f_{1j}^c(v_{1j}^c) \dots (4.53)$$

$$u_4(k) = \sum_{j=1}^{4} w_{24j}^c \{h_{1j1}^c(k) g_{1j1}^c + h_{1j2}^c(k) g_{1j2}^c\} v_{1j}^c \dots (4.54)$$

$$u_4(k) = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2}\sum_{d=1}^{2} h_{11a}^{c} h_{12b}^{c} h_{13c}^{c} h_{14d}^{c}$$

$$\{g_{11a}^{c} w_{241}^{c} w_{111}^{c} + g_{12b}^{c} w_{242}^{c} w_{112}^{c} + g_{13c}^{c} w_{243}^{c} w_{113}^{c} + g_{14d}^{c} w_{244}^{c} w_{114}^{c}\} x(k)$$

$$+\{g_{11a}^{c} w_{241}^{c} w_{121}^{c} + g_{12b}^{c} w_{242}^{c} w_{122}^{c} + g_{13c}^{c} w_{243}^{c} w_{123}^{c} + g_{14d}^{c} w_{244}^{c} w_{124}^{c}\} y(k) \ldots (4.55)$$

Donde:

$$f_{abcd}^{41} = g_{11a}^{c} w_{241}^{c} w_{111}^{c} + g_{12b}^{c} w_{242}^{c} w_{112}^{c} + g_{13c}^{c} w_{243}^{c} w_{113}^{c} + g_{14d}^{c} w_{244}^{c} w_{114}^{c}$$

$$f_{abcd}^{42} = g_{11a}^{c} w_{241}^{c} w_{121}^{c} + g_{12b}^{c} w_{242}^{c} w_{122}^{c} + g_{13c}^{c} w_{243}^{c} w_{123}^{c} + g_{14d}^{c} w_{244}^{c} w_{124}^{c}$$

Las ecuaciones (4.46), (4.49), (4.52) y (4.55) correspondientes a las variables de control $u_1(k), u_2(k), u_3(k)$ y $u_4(k)$ respectivamente se representa forma matricial en las ecuaciones (4.56) y (4.57) donde todos los coeficientes de la matriz f_{abcd} son conocidos.

$$u_{(k)} = \sum_{a=1}^{2}\sum_{b=1}^{2}\sum_{c=1}^{2}\sum_{d=1}^{2} h_{11a}^{c}(k) h_{12b}^{c}(k) h_{13c}^{c}(k) h_{14d}^{c}(k)\, f_{abcd} r_{(k)} \ldots (4.56)$$

Donde:

$$f_{abcd} = \begin{bmatrix} f_{abcd}^{11} & f_{abcd}^{12} \\ f_{abcd}^{21} & f_{abcd}^{22} \\ f_{abcd}^{31} & f_{abcd}^{32} \\ f_{abcd}^{41} & f_{abcd}^{42} \end{bmatrix}$$

$$u_{(k)} = \sum_{i=1}^{16} h_i^{c}(k) f_i r_{(k)} \ldots (4.57)$$

Integrando la ecuación (4.57) correspondiente a las variables de control u_k en la ecuación (4.37) correspondiente a las variables de estado r_k del sistema obtenemos las ecuaciones (4.58) y (4.59) que representa al sistema en lazo cerrado y su correspondiente matriz H_{ij} donde todos sus coeficientes son conocidos. El número de matrices en este caso sería $i_{max} \times j_{max} = 128$.

$$r_{(k+1)} = \sum_{i=1}^{16}\sum_{j=1}^{8} h_i^{p}(k) h_j^{c}(k)\{A_i + b_i f_j\} r_{(k)} \ldots (4.58)$$

$$H_{ij} = A_i + b_i f_j \ldots (4.59)$$

El procedimiento para encontrar una matriz P común positiva definida que permita analizar la estabilidad del sistema en lazo cerrado se describe en las siguientes líneas. El primer paso consiste en verificar que se cumpla las condiciones necesarias para la existencia de P para ello se debe cumplir que las matrices H_k y $H_k H_r$ sean estables para $k, r = 1, 2, \ldots, 128$; caso contrario, significa que no existe una matriz común P positiva definida y termina el proceso.

Si se cumple las dos condiciones necesarias, se continúa con el tercer paso el cual consiste en calcular P_k a partir de la ecuación (4.60) o (4.61), las cuales son equivalentes. Luego, se debe verificar si existe una matriz común positiva definida P_r en el conjunto $P_k \mid k = 1,2,\ldots 128$ que cumpla la condición mostrada en la ecuación (4.62). En caso no se pueda obtener la matriz P_r se continua con el cuarto paso el cual consiste en calcular las matrices Q'_k a partir de la ecuación (4.63) eligiendo una matriz Q positiva definida y posteriormente verificar que todas las matrices Q'_k sean positivas definidas. En caso se cumpla esta última condición se asegura la existencia de una matriz común P; caso contrario, se debe volver a repetir el último paso eligiendo una nueva matriz Q.

$$P_k = \theta^{-1}(-(\eta(H_k))^{-1}\theta(Q_k))\ \forall k = 1,2,\ldots,128\ldots(4.60)$$

$$H_k^T P_k H_k - P_k = -Q_k\ldots(4.61)$$

$$H_k^T P_r H_k - P_r < 0\ldots(4.63)\ \forall k = 1,2,\ldots,k\ldots(4.62)$$

$$Q'_k = \theta^{-1}(G_k\theta(Q))\ \forall k = 2,3,\ldots,128\ldots(4.63)$$

Donde:

$$G_k = \eta(H_k)(\eta(H_1))^{-1}$$

Con el resultado del análisis de estabilidad, se puede conocer si el sistema en lazo cerrado; es decir, el controlador y el modelo del sistema posterior a su entrenamiento pueden alcanzar la posición de equilibrio deseada $r_k^* = 0$ a partir de cualquier posición inicial r_o. Además, lo anterior permitirá justificar si es necesario tener más de un controlador para alcanzar todas las posiciones deseadas r_k^*.

4.4. Diagrama de flujo y aspectos relevantes en la programación

El entrenamiento del controlador tiene como objetivo minimizar el error cuadrático existente entre la posición del sistema r_k y la posición deseada r_k^* respecto al sistema en lazo cerrado a partir de la obtención de las matrices óptimas v^c y w^c pertenecientes al controlador. El diagrama de flujo del proceso de entrenamiento se muestra en la Figura 4.4, el cual se puede dividir en los siguientes grupos de tareas: inicialización, ejecución y validación. Luego, de iniciar el entrenamiento, este no culmina siempre que se supere el número máximo de iteraciones o alcance un porcentaje de error relativo mínimo deseado (condiciones de convergencia). Asimismo, el entrenamiento, en cada iteración, se realiza considerando las posiciones iniciales, las posiciones deseadas y la cantidad de datos de entrenamiento. Finalmente, al terminar cada iteración, se procede a evaluar el entrenamiento realizado para verificar si se alcanzaron las condiciones de convergencia.

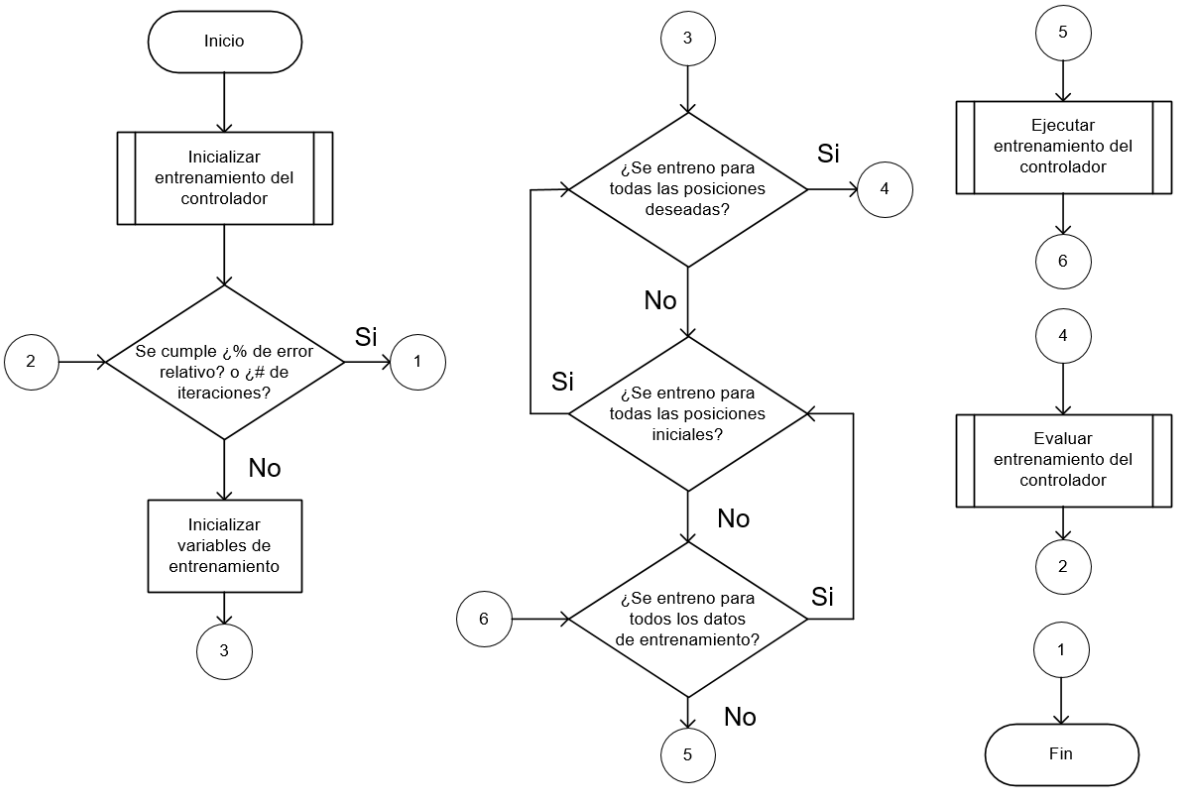


Figura 4.4. Diagrama de flujo del entrenamiento del controlador

La Figura 4.5 muestra el diagrama de flujo correspondiente a cada uno de los tres procesos. En el proceso de inicialización, se carga la información del modelo del sistema como la cantidad de neuronas de entradas, intermedias y salidas; así mismo, las matrices de coeficientes de conexión v^p y w^p. Luego, se definen las posiciones iniciales y deseadas que se utilizaran en el entrenamiento del controlador. También, se inicializa la estructura del controlador, donde se define el número de neuronas de entradas, intermedias, salida e inicializan las matrices v^c y w^c de manera aleatoria. Finalmente, se totalizan los valores de las posiciones deseadas que se usarán en el entrenamiento, las cuales servirán en la etapa de validación para el cálculo del porcentaje de error relativo.

El proceso de ejecución corresponde a una iteración completa de entrenamiento, donde se calcula la posición estimada por el sistema en lazo cerrado r_k y las derivadas parciales totales $\frac{\partial \bar{r}_k}{\partial \bar{w}_{ij}^c}$ y $\frac{\partial \bar{r}_k}{\partial \bar{v}_{ij}^c}$ para cada uno de los instantes de tiempo t_k. Con ambos resultados, se calculan las derivadas parciales totales $\frac{\partial \bar{J}}{\partial \bar{v}_{ij}^c}$, $\frac{\partial \bar{J}}{\partial \bar{w}_{ij}^c}$ usando las ecuaciones (4.10) y (4.11). Finalmente, se obtiene el error cuadrático total correspondiente a la iteración del entrenamiento.

En el proceso de evaluación, se actualizan las matrices v^c y w^c usando el método de la gradiente descendente. Luego, se calcula el porcentaje de error relativo y se actualiza el contador de iteraciones.

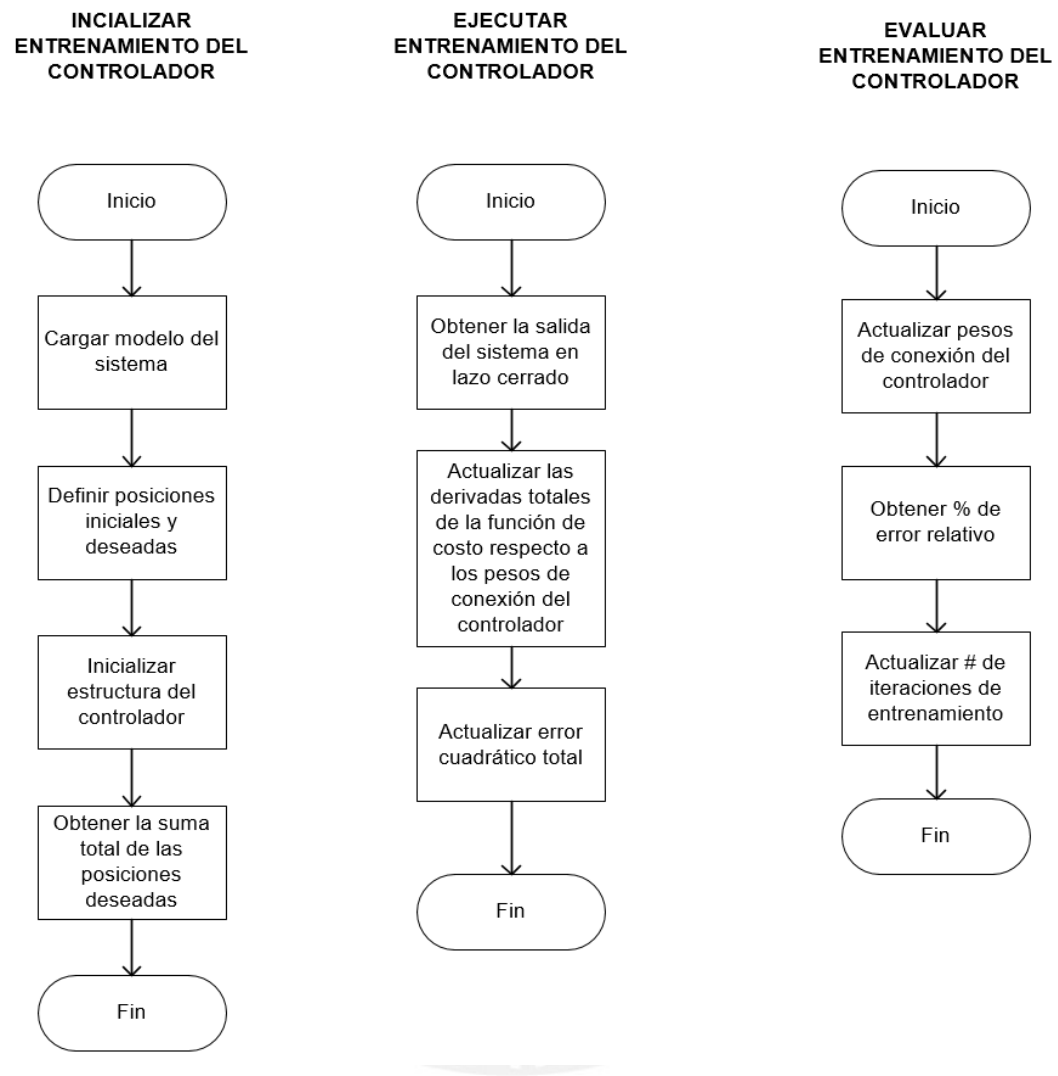


Figura 4.5. Diagrama de flujo de los procesos para el entrenamiento del controlador

Capítulo V: RESULTADOS E IMPLEMENTACIÓN MEDIANTE SIMULACIÓN

En el presente capítulo, se presentan los resultados obtenidos del entrenamiento del controlador y del análisis de estabilidad del sistema en lazo cerrado. Luego, se describe y desarrolla la metodología propuesta para la integración de controladores. Finalmente, se realiza la implementación del controlador mediante simulación.

5.1. Resultados del entrenamiento y análisis de estabilidad

En el diseño del controlador se obtiene buenos resultados definiendo para las coordenadas de posición x e y un total de siete particiones ($M = N = 7$) y sus respectivas posiciones representativas para cada una de las 49 subregiones a_{ij} generadas como se muestra en la Figura 5.1.

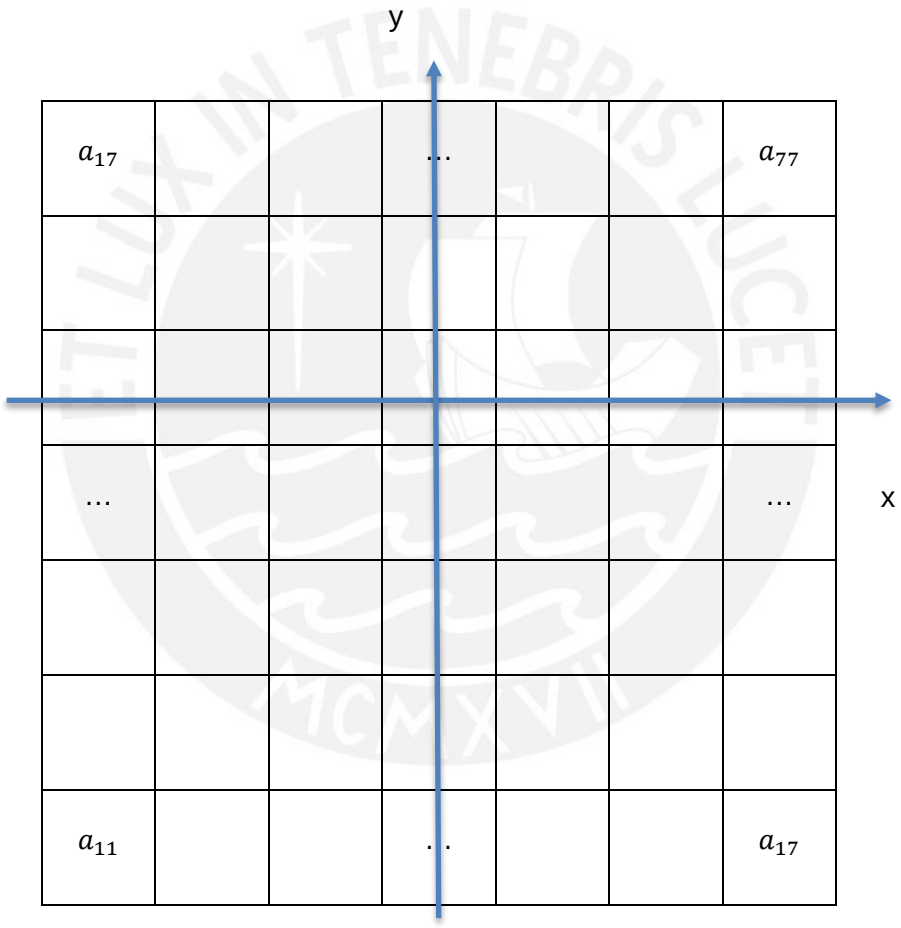


Figura 5.1. Particiones y subregiones en el plano XY

A partir de los valores de las posiciones representativas correspondiente a cada subregión a_{ij}, en el entrenamiento del controlador se elige una posición deseada (r^*) y cuatro posiciones iniciales $(r_0^1, r_0^2, r_0^3 \, y \, r_0^4)$ cuyos valores se muestran en la Tabla 5.1 a manera de ejemplo. La implementación del algoritmo de entrenamiento en Matlab se muestra en el Anexo 04 del presente documento. Para el entrenamiento se ha considerado una tasa de

aprendizaje (η) de 0.01 y como criterios de convergencia un número máximo de iteraciones de 8000 y un porcentaje de error relativo mínimo de 0.2%.

Tabla 5.1. Posiciones iniciales y deseada para entrenamiento del controlador

Descripción	Valor
Posición inicial 1 (r_0^1)	$(-0.6092, -0.5946)$
Posición inicial 2 (r_0^2)	$(-0.2145, -0.2139)$
Posición inicial 3 (r_0^3)	$(0.2284, 0.2301)$
Posición inicial 4 (r_0^4)	$(0.6451, 0.6247)$
Posición deseada (r^*)	$(-0.4176, 0.448)$

Los resultados obtenidos bajo las condiciones descritas anteriormente se muestran, a manera de ejemplo, en la Figura 5.2. Se observa la respuesta del sistema en lazo cerrado partiendo de la posición inicial 1 (r_0^1) hasta alcanzar la posición deseada (r^*), la cual cumple con las especificaciones de diseño deseadas definidas en la Tabla 4.1 del capítulo anterior. Ambas repuestas presentan un comportamiento sobre amortiguado, un tiempo de establecimiento de 5 ms, un error en estado estacionario y un sobre impulso igual a cero.

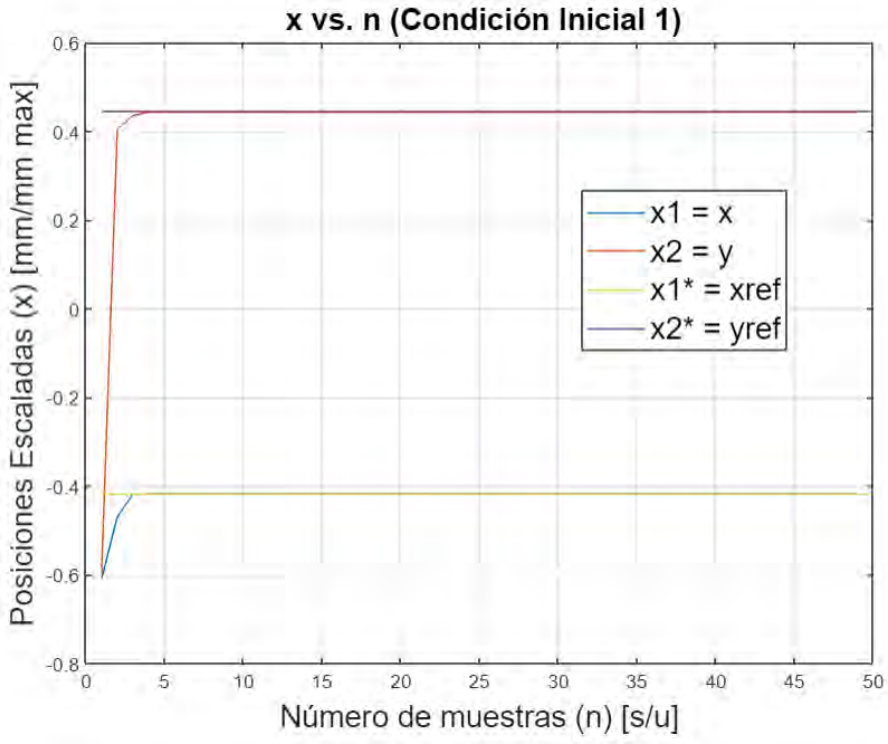


Figura 5.2. Respuesta del sistema en lazo cerrado

Además, las señales de control del sistema, obtenidas por el controlador, que corresponden a las presiones de las cámaras internas del endoscopio se muestran en la Figura 5.3, donde

se puede observar que sus valores se encuentran por debajo del valor máximo de 1 y se activan solamente las cámaras internas $px+$ y $py-$ para alcanzar la posición deseada.

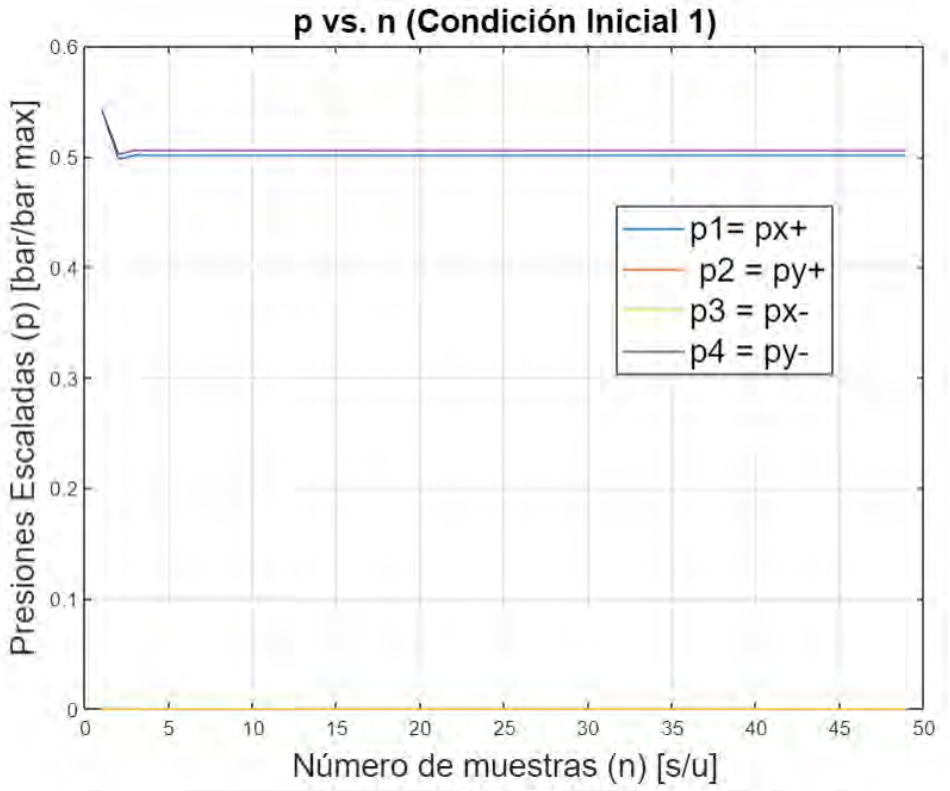


Figura 5.3. Señales de control del sistema en lazo cerrado

Los valores óptimos de las matrices de conexión v^c y w^c obtenidos posterior al entrenamiento se muestran en las ecuaciones (5.1) y (5.2) respectivamente. Cabe resaltar que también se alcanza buenos resultados para otras posiciones deseadas bajo el mismo algoritmo de entrenamiento del controlador.

$$v^c = \begin{bmatrix} -0.2343 & 0.0744 & -1.0550 & -0.9364 \\ -0.0938 & 0.0212 & -0.3380 & -0.1732 \\ 0.1686 & -0.2422 & 0.1933 & 0.0997 \end{bmatrix} \dots (5.1)$$

$$w^c = \begin{bmatrix} -0.0211 & 0.3084 & 0.2194 & -0.0453 \\ -0.0306 & -0.1779 & -0.1843 & -0.0745 \\ -0.5966 & 0.5340 & 0.6708 & -0.5422 \\ -0.4833 & 0.3803 & 0.5444 & -0.5508 \end{bmatrix} \dots (5.2)$$

De acuerdo al análisis de estabilidad del sistema en lazo cerrado desarrollado en el capítulo anterior, a partir de las matrices v^c y w^c obtenidas se procede a calcular las matrices A_i, b_i y f_j para $i = 1 \dots 16$ y $j = 1 \dots 8$; con ello, se obtiene las matrices H_{ij} del sistema en lazo cerrado. El código para el cálculo de las 128 matrices H_{ij} se muestra en el Anexo 06 del presente documento. Por ejemplo, las matrices H_{116} y H_{117} se muestran en la ecuación (5.3) y (5.4),

de las cuales solo la matriz H_{117} es positiva definida (estable). El código para la evaluación de las condiciones de estabilidad de las matrices H_{ij} también se encuentra en el Anexo 06.

$$H_{116} = \begin{bmatrix} -0.1986 & 0.1523 \\ 0.2784 & -0.0717 \end{bmatrix} \dots (5.3) \; H_{117} = \begin{bmatrix} 0.0537 & 0.0230 \\ 0.0526 & 0.0482 \end{bmatrix} \dots (5.4)$$

Debido a que no se cumplen las condiciones necesarias para garantizar la estabilidad del sistema en lazo cerrado con el controlador diseñado, se concluye que no es posible controlar el sistema solamente usando un controlador para todas las posiciones deseadas r^* de su rango de operación.

5.2. Integración de controladores

El diseño del controlador propuesto es una solución local, debido a que solo permite controlar al cuerpo del endoscopio dentro de un rango de movimiento deseado r^*. Por ello, la solución propuesta es integrar un conjunto de controladores locales y obtener un controlador global que permita controlar el cuerpo del endoscopio en todo su rango de movimiento deseado r^*. El control difuso Takagi – Sugeno es la metodología que se aplica para la integración de dichos controladores. Primero, se identifica los rangos de operación de cada controlador local en función de las coordenadas normalizadas deseadas x^* e y^*, las cuales se muestran en la Tabla 5.1. En total se identifican siete particiones para cada una de las coordenadas x^* e y^*. Por ejemplo, la partición (rango) 2 de la coordenada deseada x^* es $[-0.6307, 0.2151]$ y de la coordenada deseada y^* es $[-0.6162, -0.2105]$.

Tabla 5.1. Partición de las coordenadas normalizadas deseadas x^* e y^*

	Rango 1	Rango 2	Rango 3	Rango 4	Rango 5	Rango 6	Rango 7
x^*	$\begin{bmatrix} -1.0000 \\ -0.4265 \end{bmatrix}$	$\begin{bmatrix} -0.6307 \\ -0.2151 \end{bmatrix}$	$\begin{bmatrix} -0.4265 \\ 0.0000 \end{bmatrix}$	$\begin{bmatrix} -0.2151 \\ -0.2299 \end{bmatrix}$	$\begin{bmatrix} 0.0000 \\ 0.4551 \end{bmatrix}$	$\begin{bmatrix} 0.2299 \\ 0.6716 \end{bmatrix}$	$\begin{bmatrix} 0.4551 \\ 1.0000 \end{bmatrix}$
y^*	$\begin{bmatrix} -1.000 \\ -0.4171 \end{bmatrix}$	$\begin{bmatrix} -0.6162 \\ -0.2105 \end{bmatrix}$	$\begin{bmatrix} -0.4171 \\ -0.0000 \end{bmatrix}$	$\begin{bmatrix} -0.2105 \\ 0.2306 \end{bmatrix}$	$\begin{bmatrix} 0.0000 \\ 0.4556 \end{bmatrix}$	$\begin{bmatrix} 0.2306 \\ 0.6701 \end{bmatrix}$	$\begin{bmatrix} 0.4556 \\ 1.0000 \end{bmatrix}$

Segundo, se define para cada una de las particiones de x^* e y^* su respectiva función de membresía F_{x^*} y F_{y^*}, las cuáles se muestran en la Figura 5.4. Las funciones de membresía representan el grado pertenencia de cada valor deseado x^* e y^* a cada uno de los siete rangos identificados. Para la partición 1 y 7 se han definido funciones de pertenencia trapezoidales y para las particiones 2, 3, 4, 5 y 6 funciones de pertenencia triangulares. Por ejemplo, como caso 1, para la posición deseada r^* igual a $(-0.7, 0.8)$, la coordenada deseada

x^* pertenece a la partición 1 y la coordenada deseada y^* a la partición 7. Por otro lado, como caso 2, la posición deseada r^* igual a $(-0.5, 0.5)$, la coordenada deseada x^* pertenece a la partición 1 y 2 mientras que la coordenada deseada y^* pertenece a la partición 6 y 7. De lo anteriormente descrito, se puede resaltar que dependiendo de los valores de las coordenadas deseadas x^* e y^*, estas pueden pertenecer solamente a una partición como en el primer caso o pueden pertenecer a dos particiones como en el segundo caso.

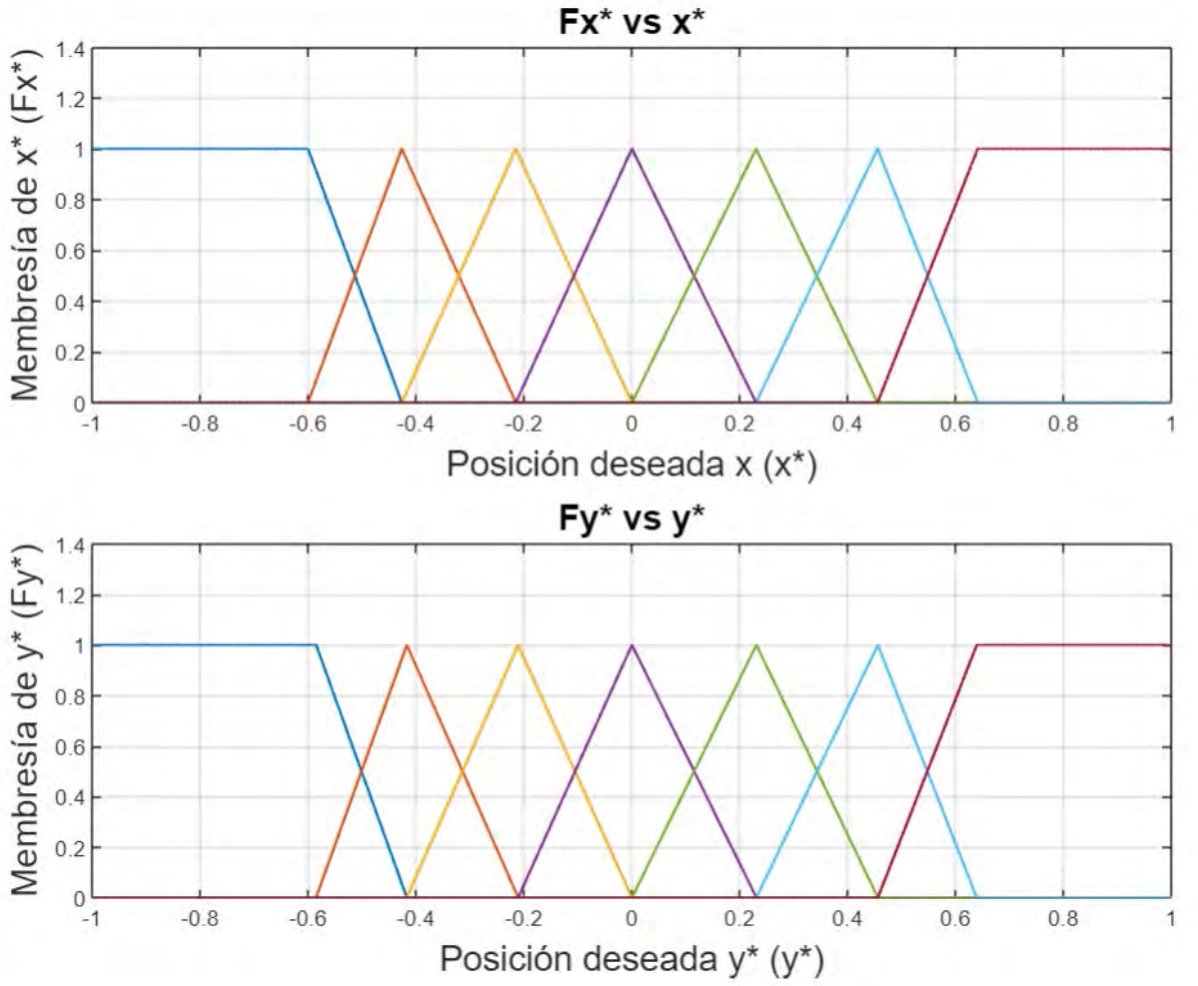


Figura 5.4. Funciones de membresía de las coordenadas deseadas x^* e y^*

Tercero, se define la base de reglas, donde se indica el controlador que asigna a cada región de posiciones deseadas r^* tomando en cuenta las particiones de las coordenadas deseadas x^* e y^*. La representación gráfica de la base de reglas se muestra en la Figura 5.5; por ejemplo, para la partición cinco de la coordenada deseada x^* $(x5)$ y la partición tres de la coordenada deseada y^* $(y3)$ corresponde el controlador u_{53}, el cual se obtiene en el proceso de entrenamiento del controlador considerando como posición deseada r^*_{53}. De forma similar, se obtiene el resto de controladores de la base de reglas dando un total de 49 controladores posibles en todo el rango de movimiento del cuerpo del endoscopio.

	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$
$y1$	u_{11}	u_{21}	u_{31}	u_{41}	u_{51}	u_{61}	u_{71}
$y2$	u_{12}	u_{22}	u_{32}	u_{42}	u_{52}	u_{62}	u_{72}
$y3$	u_{13}	u_{23}	u_{33}	u_{43}	u_{53}	u_{63}	u_{73}
$y4$	u_{14}	u_{24}	u_{34}	u_{44}	u_{54}	u_{64}	u_{74}
$y5$	u_{15}	u_{25}	u_{35}	u_{45}	u_{55}	u_{65}	u_{75}
$y6$	u_{16}	u_{26}	u_{36}	u_{46}	u_{56}	u_{66}	u_{76}
$y7$	u_{17}	u_{27}	u_{37}	u_{47}	u_{57}	u_{67}	u_{77}

Figura 5.5. Base de reglas el controlador

Cuarto, como paso final se aplica el método de inferencia que permite determinar la acción de control a aplicar en tiempo real de acuerdo a la ecuación (5.5), donde $F_{x_i^*}$ y $F_{y_j^*}$ representan las funciones de pertenencias de las coordenadas deseadas x_i^* e y_j^* para cada una de sus respectivas particiones y u_{ij} la ley de control para cada región de movimiento del cuerpo del endoscopio.

$$u = \frac{\sum_{i=1}^{7}\sum_{j=1}^{7} min\left(F_{x_i^*}, F_{y_j^*}\right) u_{ij}}{\sum_{i=}^{7}\sum_{j=1}^{7} min\left(F_{x^*}, F_{y^*}\right)} \dots (5.5)$$

5.3. Implementación del controlador mediante simulación

Considerando el control neuronal para la obtención de los controladores locales y el control difuso Takagi – Sugeno para la obtención del controlador global se procede a implementar el controlador en Matlab y simular el comportamiento del sistema en lazo cerrado considerando un tren de escalones como posiciones deseadas r_k^*, cuyos valores se muestran en la Tabla 5.2, de manera que se pueda verificar el desempeño del controlador global en todo el rango de movimiento del cuerpo del endoscopio.

Tabla 5.2. Posiciones deseadas r_k^*

	r_1^*	r_2^*	r_3^*	r_4^*	r_5^*	r_6^*	r_7^*
x_k^*	-0.6092	-0.4199	-0.2145	0.0000	0.2284	0.4460	0.6451
y_k^*	-0.5946	-0.4153	-0.2139	0.0000	0.2301	0.4425	0.6247

El código que implementa el controlador se comparte en el Anexo 05 correspondiente al presente documento. La implementación del controlador inicia con la carga de las matrices de conexión v^c y w^c de los controladores locales y de las funciones de pertenencia F_{x^*} y F_{y^*} correspondientes a las particiones de las coordenadas deseadas x e y. Para poder simular el comportamiento del sistema en lazo cerrado, se define las posiciones deseadas r^* a través de un tren de escalones y la posición inicial r_o del sistema; asimismo, se realiza la carga de las matrices de conexión v^p y w^p del modelo del sistema. Respecto a la ejecución del algoritmo de control en tiempo real, la secuencia se inicia con el cálculo del error entre la posición deseada r^* y la posición actual r del endoscopio. Luego, se realiza el cálculo de las variables de control u para cada controlador local utilizando la ecuación de la red neuronal y sus respectivas matrices de conexión v^c y w^c. Las variables de control resultantes son restringidas a valores positivos por debajo de su valor máximo. Después, para cada una de las coordenadas deseadas x^* e y^* se calcula el valor de la función de pertenencia F_{x^*} y F_{y^*} respectivamente correspondiente a cada una sus particiones. Finalmente, para obtener la variable de control global se realiza la suma ponderada las variables de control locales utilizando la ecuación (5.5), donde los respectivos pesos para cada acción de control se obtienen a partir del mínimo de cada par de funciones de pertenencia $F_{x_i^*}$ y $F_{y_j^*}$ correspondiente a cada partición.

En las Figuras 5.6 y 5.7 se muestran la comparación entre las respuestas de las coordenadas de posición del modelo (x_k, y_k) con las coordenadas de posición deseadas (x_k^*, y_k^*) respectivamente del sistema en lazo cerrado. Evaluando las especificaciones en el dominio del tiempo, como el máximo sobreimpulso, el error en estado estacionario y el tiempo de establecimiento se puede observar que existe una variación de acuerdo al valor de la posición deseada $r_k^* = x_k^*, y_k^*$, debido a que después del entrenamiento de los coeficientes v^c y w^c de los controladores para las diferentes particiones de posición deseadas r_k^* se alcanzan diferentes porcentajes de error relativo total $(\%e_r)$.

En la Figura 5.6, se compara la respuesta entre la coordenada deseada x^*, en línea de color roja, con la respuesta de la coordenada x, en línea de color azul. De la misma manera en la Figura 5.7, se compara la respuesta entre la coordenada deseada y^*, en línea de color roja, con la respuesta de la coordenada y, en línea de color azul. Como se puede observar, el sistema en lazo cerrado es capaz de seguir la trayectoria deseada en todo su rango de movimiento con una respuesta sobre amortiguada que cumple con las especificaciones de diseño en el tiempo del controlador mostrados en la Tabla 4.1. del capítulo anterior.

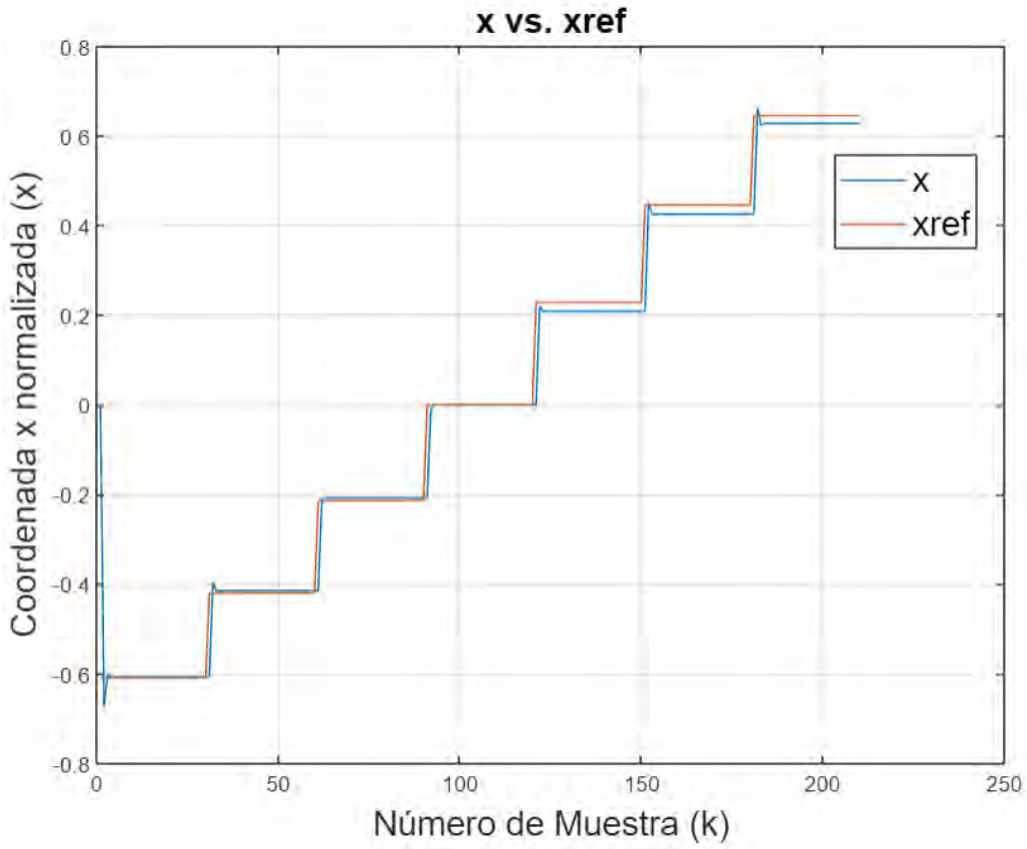


Figura 5.6. Comparación entre la coordenada x_k y la coordenada deseada x_k^*

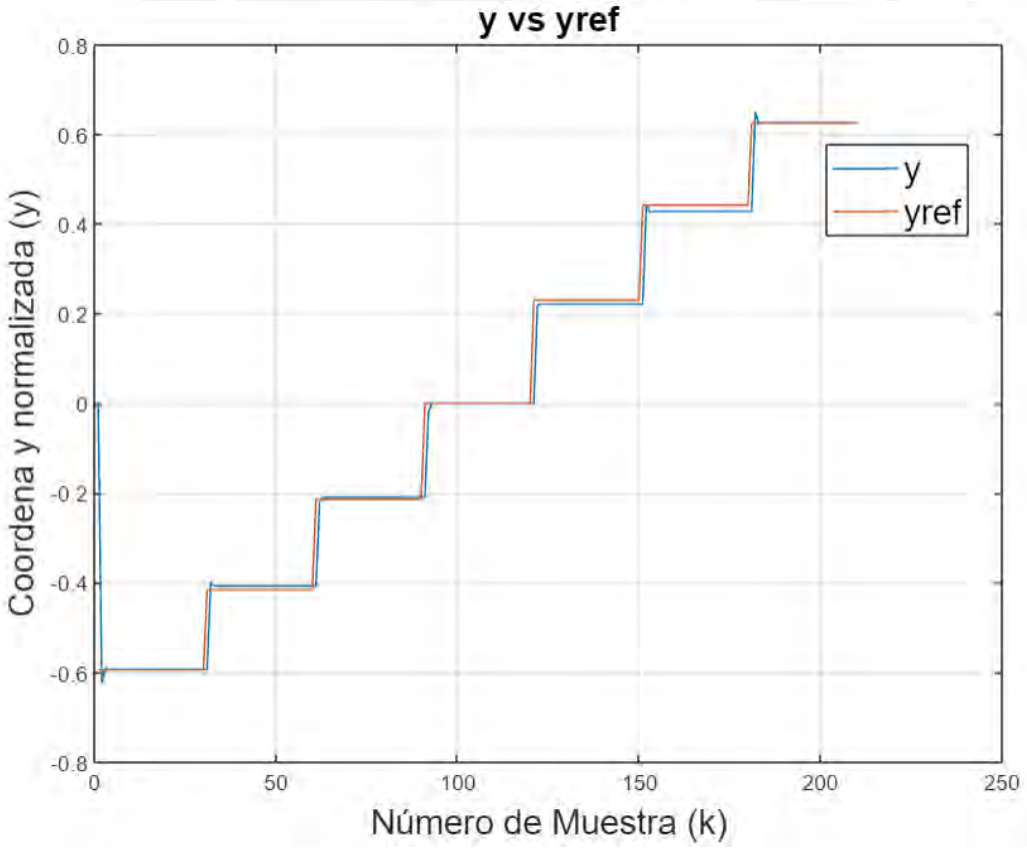


Figura 5.7. Comparación entre la coordenada y_k y la coordenada deseada y_k^*

Las variables de control, que se muestran en la Figura 5.8, son calculadas por el controlador en cada instante de tiempo, lo cual permite aplicar las presiones correctas $px+$, $py+$, $px-$ y $py-$ en las cámaras internas del endoscopio para alcanzar la trayectoria de las posiciones deseada r^*. Además, se puede observar que los valores de las presiones son siempre positivos y por debajo de su valor máximo permitido igual a 1.

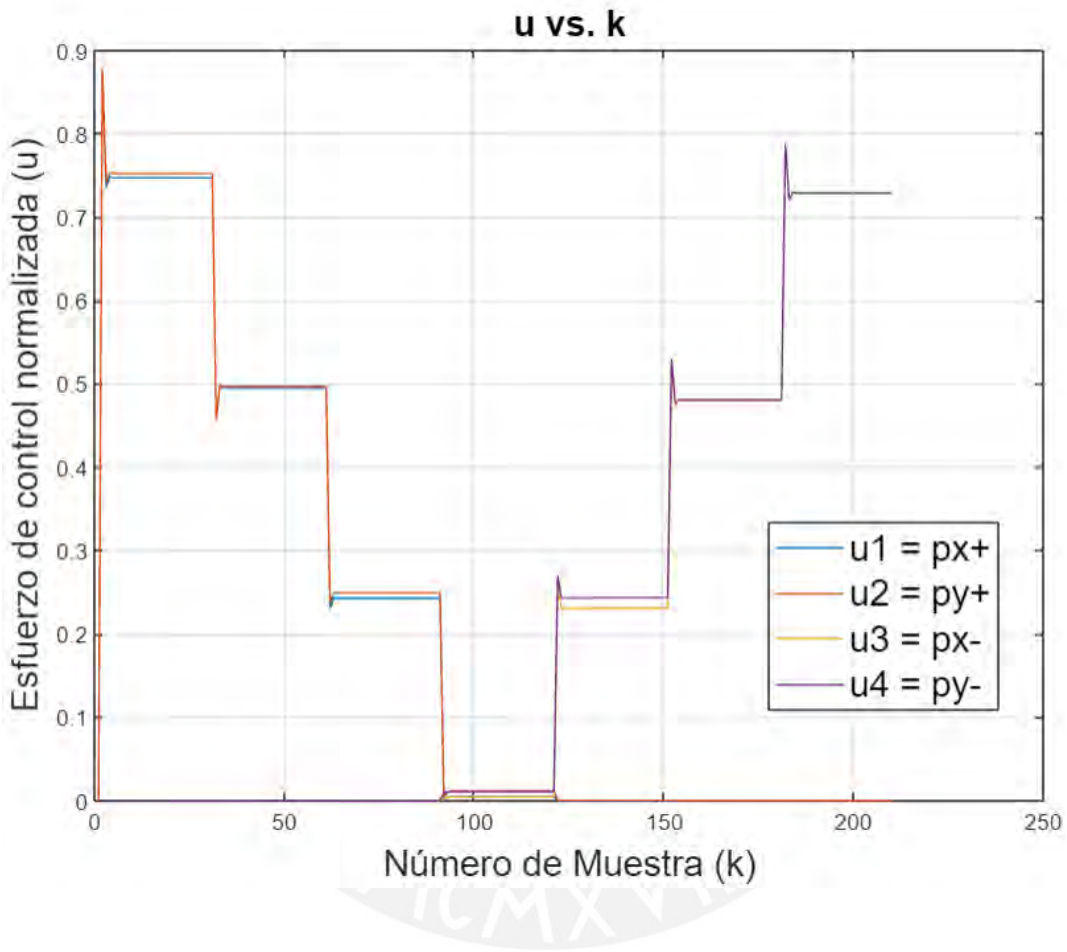


Figura 5.8. Respuesta de las variables de control u_k

Finalmente, se realizan las pruebas de desempeño del controlador antes perturbaciones en la variable de control u (presión de las cámaras internas del endoscopio) y ruido de medición en la variable a controlar r (posición del efector final del endoscopio). En la Figura 5.9, se muestra la respuesta del controlador considerando una perturbación del 1% respecto a la presión máxima, ya que para valores mayores no se presenta un buen desempeño del sistema en lazo cerrado. La gráfica a muestra la respuesta de la coordenada de posición x, la gráfica b muestra la coordenada de posición y y la gráfica c muestra las variables de control $px+$, $py+$, $px-$ y $py-$.

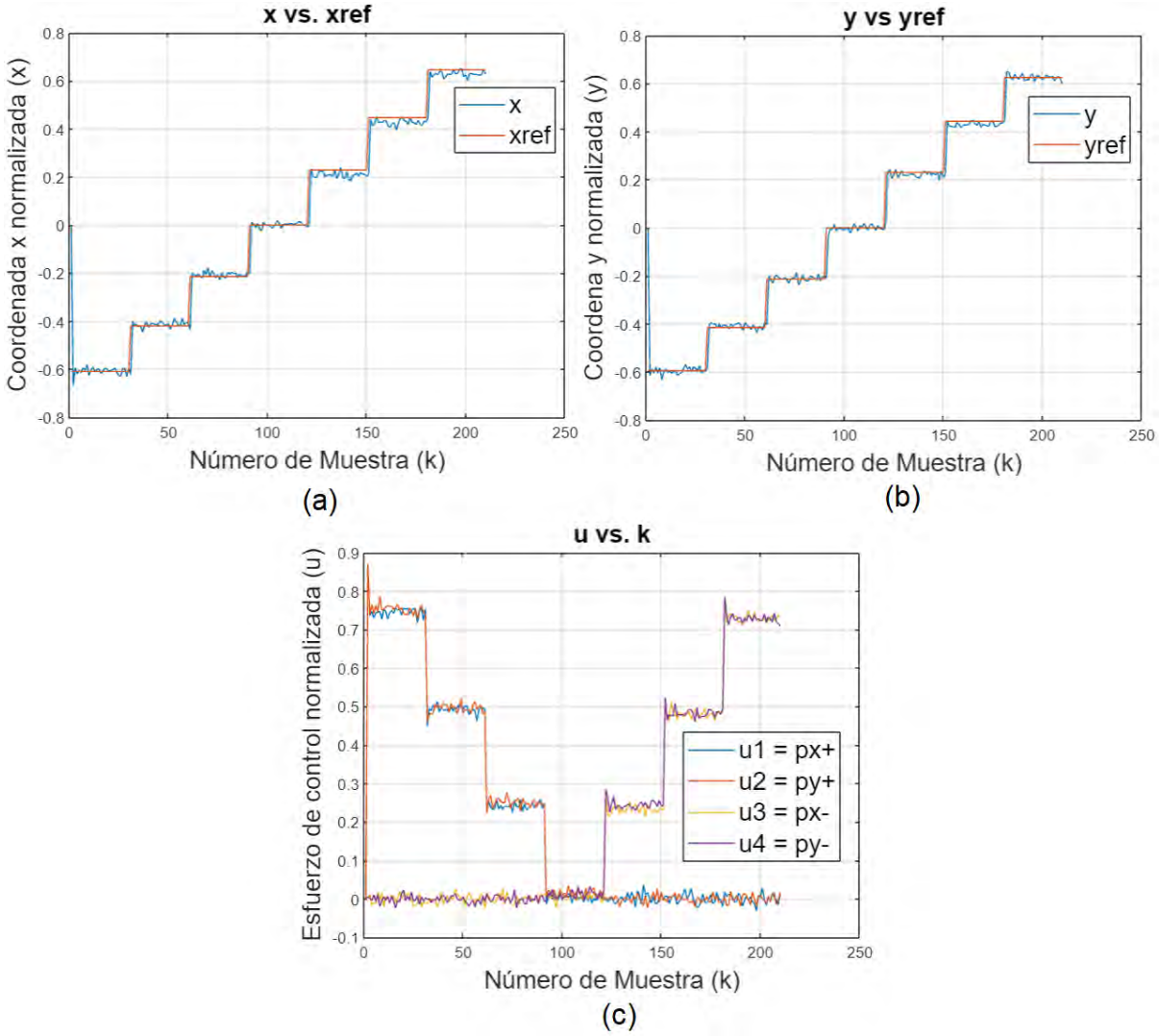


Figura 5.9. Respuesta del sistema ante perturbaciones en la variable de control

En la Figura 5.10, se muestra la respuesta del controlador considerando un ruido de medición del 10% respecto a la posición máxima, ya que para valores mayores no se presenta un buen desempeño del sistema en lazo cerrado. La gráfica a muestra la respuesta de la coordenada de posición x, la gráfica b muestra la coordenada de posición y y la gráfica c muestra las variables de control $px+$, $py+$, $px-$ y $py-$.

Con los resultados anteriormente explicados, se valida el correcto desempeño del controlador de posición bajo las condiciones de operación descritas.

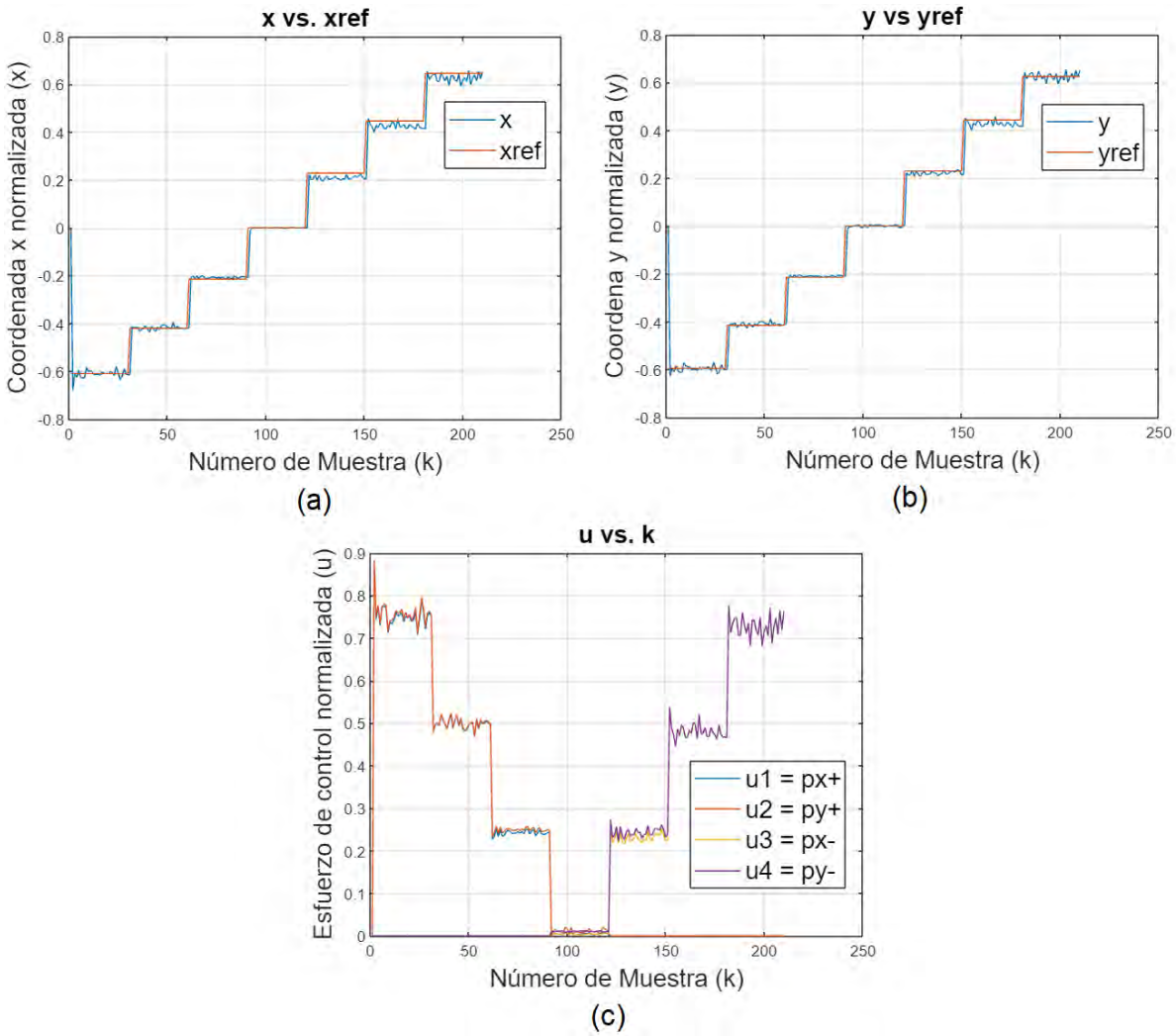


Figura 5.10. Respuesta del sistema ante ruido de medición en la variable a controlar

CONCLUSIONES

Las conclusiones que se alcanzaron con el desarrollo del presente trabajo son las siguientes:

Se realizó la revisión de las tecnologías utilizadas en la cirugía laparoscópica asistida por endoscopio blando, donde se revisa las técnicas actuales de modelamiento y control de los manipuladores o actuadores blandos aplicados al control de movimiento de endoscopios.

La dinámica del sistema se representó a través de un modelo basado en una red neuronal recurrente con realimentación a la salida. El entrenamiento se realizó utilizando como optimizador el método de la gradiente descendente y el algoritmo DBP (Dynamic Back Propagation en inglés). Los datos de entrenamiento se generaron a partir del software de simulación Abaqus CAE.

La validación del modelo del sistema se realizó evaluando el porcentaje de error relativo entre las posiciones estimadas por el modelo r_k y las posiciones deseadas r_k^* alcanzando un valor del 30% Los datos de validación se generaron a partir del software de simulación Abaqus CAE.

El diseño del controlador de posición consistió en el entrenamiento de una red neuronal tipo FeedForward (en inglés), donde se realizó un entrenamiento dinámico utilizando como optimizador el método de la gradiente descendente y el algoritmo DBP (Dynamic Back Propagation en inglés). El algoritmo de entrenamiento del controlador se implementó en Matlab.

El análisis de estabilidad del sistema en lazo cerrado permitió concluir que se requieren diferentes controladores locales para el control de diferentes posiciones deseadas del endoscopio.

El control difuso Takagi – Sugeno se utilizó para la integración de los controladores locales y la obtención de un controlador global para todo el rango de movimiento del endoscopio. El algoritmo del control se implementó en Matlab.

La validación del desempeño del controlador se realizó usando un tren de escalones de posiciones deseadas r_k^* en todo el rango de movimiento del endoscopio. La respuesta del sistema en lazo cerrado del sistema cumplió con las especificaciones deseadas en el dominio del tiempo.

BIBLIOGRAFÌA

Braganza D., Dawson D., Walker I, Nath N. (2007). ***A neural Networks Controller for Continuum Robots***. IEEE Transactions on Robotics

Chen C., Tang W., Hu Y., et al. (2020). ***Fiber-reinforced soft bending actuator control utilizing on/off valves.*** IEEE Robotics and Automation Letters.

Cianchetti M., Laschi C., Menciassi A., Dario P. (2018). ***Biomedical applications of soft robotics***. Nature Review Materials.

Centurelli A., Arleo L., Rizzo A. (2022). ***Closed Loop Dynamic Control of a Soft Manipulator Using Deep Reinforcement Learning***. IEEE Robotics and Automation Letters.

Connolly F., Walsh C., Bertoldi K. (2017). ***Automatic design of fiber-reinforced soft actuators for trajectory matching***. The Proceedings of the National Academy of Sciences.

Elsayed Y., Vicensi A., Lekakou C., Geng T. (2014). ***Finite Element Analysis and Design Optimization of a Pneumatically Actuating Silicone Module for Robotic Surgery Applications.*** SoRo Soft Robotics.

Falkenhahn, V., Mahl, T., Hildebrant, A., Neumann, R., Sawodny O. (2017). ***Dynamic Control of the Bionic Handling Assistant***. IEEE/ASME Transaction on Mechatronics.

Falkenhahn, V., Mahl, T., Hildebrant, A., Neumann, R., Sawodny O. (2015). ***Dynamic modeling of bellows-actuated continuum robots using Euler-Lagrange formalism***. IEEE Transaction on Robotics.

Franco E., Garriga-Casanovas A., Tang J., Rodríguez F., Astolfi A. (2022). ***Adaptive Energy Shaping Control of a Class of Nonlinear Soft Continuum Manipulators***. IEEE/ASME TRANSACTION ON MECHATRONICS

Franco E., Garriga-Casanovas A., Ayatullah T., Sugiharto Arif (2021). ***Nonlinear energy-based control of soft continuum pneumatic manipulators***. Springer Verlag.

Franco E., Garriga A., Rizzo A., Donaire A. (2021). ***Energy shaping control with integral action for soft continuum manipulators***. Mechanism and Machine Therory.

Frás J, Czarnowski J., Maciás M., Glókwa J., Cianchetti M., Menciassi A. (2015). ***New STIFF-FLOP module construction idea for improved actuation and sensing***. IEEE International Conference on Robotics and Automation (ICRA)

Gerboni G., Ranzani T. Diodato A., Ciuti G., Cianchetti M., Menciassi A. (2015). ***Modular soft mechatronic manipulator for minimally invasive surgery (MIS): overall architecture and development of a fully integrated soft module***. Springer Verlag.

Gerboni G., Ranzani T. Diodato A., Ciuti G., Cianchetti M., Menciassi A. (2017). ***Feedback Control of Soft Robot Actuators via Commercial Flex Bend Sensors***. IEEE/ASME Transactions on Mechatronics.

George T., Ansari Y., Falotico E., Laschi C. (2018). ***Control Strategies for Soft Robotic Manipulators: A Survey.*** Soft Robotics.

Hunt K., Sbarbaro D., Zbikowski R., Gawthrop P. (1992). ***Neural Networks for Control Systems – A Survey.*** Automatica Vol. 28, No.6, pp. 1083 – 1112.

Hwang J., Hsiao F. (2003). ***Stability Analysis of Neural Network Interconnected Systems.*** IEE Transactions on Neural Networks Vol.14, No. 1

Jin Y., Pipe A., Winfield A. (1993). ***Dynamic DBP learning algorithm for real time applications.*** Third International Conference on Artificial Neural Networks.

Khan A., Shao Z., Li, S., Wang, Q., Guan, N. (2020). ***Which is the best PID variant for pneumatic soft robors? An experimental study.*** IEEE/CCA Journal of Automatica Sinica.

Khan AH., Li, S. (2020). ***Sliding mode control with PID sliding surface for active vibration damping of pneumatically actuated soft robots.*** IEEE Access.

Kim D., Kim S., Kim T., Kang B., Lee M., Park W., Ku S. (2021). ***Review of machine learning methods in soft robotics***. PLOS ONE Journal

Lenssen J., Naghini H., Abayazid M. (2019). ***Evaluation of design aspects of modular pneumatic soft robotic endoscopes***. IEEE International Conference on Soft Robotics (RoboSoft).

Ma H., Zhou J., Zhang J., Zhang L. (2021). ***Research of the Inverse Kinematics Prediction of a Soft Biomimetic Actuator vía BP Neural Network.*** IEE Access.

Narendra K., Parthasarathy K., (1990). ***Identification and Control of Dynamical Systems Using Neural Networks***. IEEE Transactions on Neural Networks Vol1, No1.

Skorina EH., Luo, M., Ozel S., et al., (2015). ***Feedforward augmented sliding mode motion control of antagonistic soft pneumatic actuators.*** Proceedings of the IEEE International Conference on Robotics and Automation.

Tanaka, K. (1996). ***An Approach to Stability Criteria of Neural-Network Control Systems.*** IEE Transactions of Neural Networks Vol7, No3

Tanaka, K., Sano M. (1993). ***Fuzzy stability of a class of nonlinear systems.*** Information Sciences 71, 3-26

Thuruthel T., Ansari Y., Falotico E., Laschi C. (2018). ***Control Strategies for Soft Robotic Manipulators: A Survey***. SoRo Soft Robotics

Tovah W., Sang-Eun S. (2022). **Robotic Surgery Techniques to Improve Traditional Laparoscopy**. Society of Laparoscopic & Robotic Surgeons Vol 26, Issue 2

Wang, T., Zhang, YC., Chen, Z. (2018). ***Design and verification of model-based nonlinear controller for fluidic soft actuators.*** Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics.

Wang X., Li Y., Kwok K. (2021). ***A Survey for Machine Learning - Based Control of Continuum Robots***. Frontiers in Robotics and AI.

Webster R., Jones B., (2010). ***Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review***. The International Journal of Robotics Research.

Werbos P. (1974). ***Beyond Regression: New tools for prediction and analysis in the behavioral sciences.*** Harvard University.

ANEXOS

ANEXO 01: Código para la generación de datos de entrenamiento y validación

```
%DESCRIPTION: Set input output data generation for training and validation
%for a import mat file with matrix data
%INPUT:  vector u / u1 = px+, u2 = py+, u3 = px-, u4 = py-
%OUTPUT: vector y / y1 = x, y2 = y, y3 = z


clc;
clear all;
close all;

ts=0.001;
%tf=27.2;
tf=4;
t=0:ts:tf;

%{
%Training data generation
load('dataTraining021022.mat');
tRaise=[0.1 0.15 0.2 0.25];
tSet=[0.4 0.3 0.2 0.1];

pattern1=[linspace(0,1,tRaise(1)/ts) ones(1,tSet(1)/ts) linspace(1,0,tRaise(1)/ts)↙
zeros(1,tSet(1)/ts)];
pattern1=[pattern1 pattern1];
pattern2=[linspace(0,1,tRaise(2)/ts) ones(1,tSet(2)/ts) linspace(1,0,tRaise(2)/ts)↙
zeros(1,tSet(2)/ts)];
pattern2=[pattern2 pattern2];
pattern3=[linspace(0,1,tRaise(3)/ts) ones(1,tSet(3)/ts) linspace(1,0,tRaise(3)/ts)↙
zeros(1,tSet(3)/ts)];
pattern3=[pattern3 pattern3];
pattern4=[linspace(0,1,tRaise(4)/ts) ones(1,tSet(4)/ts) linspace(1,0,tRaise(4)/ts)↙
zeros(1,tSet(4)/ts)];
pattern4=[pattern4 pattern4];

um(1,:)=0.3*[pattern1 pattern2 pattern3 pattern4 pattern1 pattern2 pattern3 pattern4↙
zeros(1,13.6/ts)];
um(2,:)=0.3*[pattern4 pattern3 pattern2 pattern1 zeros(1,6.8/ts) pattern4 pattern3↙
pattern2 pattern1 zeros(1,6.8/ts)];
um(3,:)=0.3*[zeros(1,13.6/ts) pattern1 pattern2 pattern3 pattern4 pattern1 pattern2↙
pattern3 pattern4];
um(4,:)=0.3*[zeros(1,6.8/ts) pattern4 pattern3 pattern2 pattern1 zeros(1,6.8/ts)↙
pattern4 pattern3 pattern2 pattern1];

%}

%Validation data generation
load('dataValidation031022.mat');
per=2;
wang=2*pi/per;
tPattern=0:ts:(per/2);
pattern=sin(wang*tPattern);
n=length(pattern);
um(1,:)=0.3*[pattern pattern(1,2:n) zeros(1,per/ts)];
um(2,:)=0.3*[pattern zeros(1,per/2/ts) pattern(1,2:n) zeros(1,per/2/ts)];
um(3,:)=0.3*[zeros(1,per/ts) pattern pattern(1,2:n)];
```

```
um(4,:)=0.3*[zeros(1,per/2/ts) pattern zeros(1,per/2/ts) pattern(1,2:n)];
ym(:,1)=dataValidation031022(:,2);
ym(:,2)=dataValidation031022(:,3);
ym(:,3)=dataValidation031022(:,4);

um=um';
t=t';

%save dataTraining021022 t um ym;
save dataValidation031022 t um ym;

figure(1);
plot(t,um(:,1));
figure(2);
plot(t,ym);
```

ANEXO 02: Código para el entrenamiento del modelo del sistema

```
% Descripción: Código para entrenar el modelo dinámico del endoscopio
% Algoritmo de entrenamiento: Dynamic Back Propagation (DBP)
% Número de entradas: 2/3/4
% Número de salidas: 3
% Tiempo de muestreo: 1 ms
% Estructura del modelo no lineal: yk = f(uk, yk-1)

clear;
clc;
close all;

load('dataTraining021022.mat');

%px+ / py+
%col=[1 2];
%fil=[1 6801];

%px+ / py-
% col=[1 4];
% fil=[6802 13601];

%px- / py+
%col=[2 3];
%fil=[13602 20401];

%px- / py-
%col=[3 4];
%fil=[20402 27200];

%um=[um(:,col(1)) um(:,col(2))];
%um=um(fil(1):round(fil(1)+(fil(2)-fil(1))/2),:);
%um=um(fil(1):fil(2),:);
%ym=ym(fil(1):round(fil(1)+(fil(2)-fil(1))/2),1:2);
%ym=ym(fil(1):fil(2),1:2);
ym=ym(:,1:2);
%t=t(fil(1):round(fil(1)+(fil(2)-fil(1))/2));
%t=t(fil(1):fil(2));

nu = length(t);

figure(1);
plot(t,um);
grid on;
xlabel('Tiempo(t) [s]','FontSize',16);
ylabel('Presion(um) [bar]','FontSize',16);
title ('um vs. t','FontSize',16);
%legend('um1','um2','FontSize',16);
legend('um1 = px+','um2 = py+','um3 = px-','um4 = py-','FontSize',16);

figure(2);
plot(t,ym);
grid on;
```

```
xlabel('Tiempo(t) [s]','FontSize',16);
ylabel('Posición(ym) [mm]','FontSize',16);
title ('ym vs. t','FontSize',16);
legend('ym1 = x','ym2 = y','FontSize',16);
%legend('ym1 = x','ym2 = y','ym3 = z','FontSize',16);

% Escalamiento
factx = max(abs(um));
facty = max(abs(ym));

xesc(:,1) = um(:,1)./factx(1,1);
xesc(:,2) = um(:,2)./factx(1,2);
xesc(:,3) = um(:,3)./factx(1,3);
xesc(:,4) = um(:,4)./factx(1,4);

yesc(:,1) = ym(:,1)./facty(1,1);
yesc(:,2) = ym(:,2)./facty(1,2);
%yesc(:,3) = ym(:,3)./facty(1,3);

figure(3);
%plot(t,yesc(:,1),t,yesc(:,2),t,yesc(:,3));
plot(t,yesc(:,1),t,yesc(:,2));
grid on;
xlabel('Tiempo (t) [s]','FontSize',16);
ylabel('Posición escalada(yesc) [mm/mm]','FontSize',16);
title ('yesc vs. t','FontSize',16);
legend('yesc1 = y1/y1max','yesc2 = y2/y2max','FontSize',16);
%legend('yesc1 = y1/y1max','yesc2 = y2/y2max','yesc3 = y3/
y3max','FontSize',16);
ndata = nu;
dataoutesc = yesc;
u=xesc;


% Number of neurons (input, hidden and output layers)
%ne = 5;    % No bias
%ne = 4;
ne = 6;
nm = 3;   % nm = 10
%ns = 3;
ns = 2;
% Intializing coefficients v, w, sigmoid center and slope
v = 0.1*randn(ne,nm);
w = 0.1*randn(nm,ns);
c = zeros(nm,1);
a = 1*ones(nm,1);

%load soft4modelpxmaspymas;
%load soft4model01;

% Introducing learning parameters
eta  = input('Introduce learning rate [v w]: ');
etac = input('Introduce learning rate [c: sigmoid center]: ');
etaa = input('Introduce learning rate [a: sigmoid slope]: ');
```

```
errormax = input('Introduce maximum value of error function (percentage %) :
 ');
errormax = errormax/100;
contmax = input('Introduce number of iteration steps: ');

% Training
outsum2 = sum(dataoutesc.^2);
outsum2 = outsum2';
outsum2total = sum(outsum2);
cont = 1;
erreltotal = 1;
   dw_old = 0;
   dv_old = 0;
   da_old = 0;
   dc_old = 0;

while( (erreltotal > errormax) & (cont < contmax) )
   ersum2 = zeros(ns,1);
   dJdw = 0;
   dJdv = 0;
   dJda = 0;
   dJdc = 0;
   dy1dw_t = zeros(nm,ns);
   dy2dw_t = zeros(nm,ns);
   %dy3dw_t = zeros(nm,ns);
   dy1dv_t = zeros(ne,nm);
   dy2dv_t = zeros(ne,nm);
   %dy3dv_t = zeros(ne,nm);
   dy1dc_t = zeros(nm,1);
   dy2dc_t = zeros(nm,1);
   %dy3dc_t = zeros(nm,1);
   dy1da_t = zeros(nm,1);
   dy2da_t = zeros(nm,1);
   %dy3da_t = zeros(nm,1);
   dJdw_t  = zeros(nm,ns);
   dJdv_t  = zeros(ne,nm);
   dJdc_t  = zeros(nm,1);
   dJda_t  = zeros(nm,1);

   x = yesc(1,:);   % Initial state
   %x1 = zeros(1,ns);
   x = x';
   %x1 = x1';
   %u1 = zeros(1,4);
   %u1 = u1';
   outputesc(1,:) = x;
   for k = 1:ndata-1
      in_red = [ x
                 %x1
                 u(k,1)
                 u(k,2)
                 u(k,3)
                 u(k,4)
```

```
                %ul
                ];

        m = v'*in_red;
        n = 2.0./(1 + exp(-(m-c)./a)) - 1;
%       n = m;          % Lineal
        out_red = w'*n;
        %outputesc(k,:) = out_red';
        outputesc(k+1,:) = out_red';
        dndm = diag((1 - n.*n)./(2*a));
%       dndm = diag(ones(nm,1));        % Lineal
%       dy1dw_s = [ n   zeros(nm,1)  zeros(nm,1)];
        dy1dw_s = [ n   zeros(nm,1)];
%       dy2dw_s = [ zeros(nm,1)   n  zeros(nm,1)];
        dy2dw_s = [ zeros(nm,1)   n ];
%       dy3dw_s = [ zeros(nm,1)   zeros(nm,1)   n];
        dy1dv_s = in_red*w(:,1)'*dndm;
        dy2dv_s = in_red*w(:,2)'*dndm;
%       dy3dv_s = in_red*w(:,3)'*dndm;
        dy1dc_s = w(:,1) .* ((n.*n-1)./(2.0.*a));
        dy2dc_s = w(:,2) .* ((n.*n-1)./(2.0.*a));
%       dy3dc_s = w(:,3) .* ((n.*n-1)./(2.0.*a));
        dy1da_s = w(:,1) .* ((n.*n-1).*(m-c)./(2*a.*a));
        dy2da_s = w(:,2) .* ((n.*n-1).*(m-c)./(2*a.*a));
%       dy3da_s = w(:,3) .* ((n.*n-1).*(m-c)./(2*a.*a));
%       jacob = w'*dndm*(v(1:3,:))';
        jacob = w'*dndm*(v(1:2,:))';
%       dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t +
 jacob(1,3).*dy3dw_t;
        dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t;
%       dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t +
 jacob(2,3).*dy3dw_t;
        dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t;
%       dy3dw_t = dy3dw_s + jacob(3,1).*dy1dw_t + jacob(3,2).*dy2dw_t +
 jacob(3,3).*dy3dw_t;
%       dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t +
 jacob(1,3).*dy3dv_t;
        dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t;
%       dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t +
 jacob(2,3).*dy3dv_t;
        dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t;
%       dy3dv_t = dy3dv_s + jacob(3,1).*dy1dv_t + jacob(3,2).*dy2dv_t +
 jacob(3,3).*dy3dv_t;
%       dy1dc_t = dy1dc_s + jacob(1,1).*dy1dc_t + jacob(1,2).*dy2dc_t +
 jacob(1,3).*dy3dc_t;
        dy1dc_t = dy1dc_s + jacob(1,1).*dy1dc_t + jacob(1,2).*dy2dc_t;
%       dy2dc_t = dy2dc_s + jacob(2,1).*dy1dc_t + jacob(2,2).*dy2dc_t +
 jacob(2,3).*dy3dc_t;
        dy2dc_t = dy2dc_s + jacob(2,1).*dy1dc_t + jacob(2,2).*dy2dc_t;
%       dy3dc_t = dy3dc_s + jacob(3,1).*dy1dc_t + jacob(3,2).*dy2dc_t +
 jacob(3,3).*dy3dc_t;
%       dy1da_t = dy1da_s + jacob(1,1).*dy1da_t + jacob(1,2).*dy2da_t +
 jacob(1,3).*dy3da_t;
        dy1da_t = dy1da_s + jacob(1,1).*dy1da_t + jacob(1,2).*dy2da_t;
```

```
%     dy2da_t = dy2da_s + jacob(2,1).*dy1da_t + jacob(2,2).*dy2da_t +
 jacob(2,3).*dy3da_t;
      dy2da_t = dy2da_s + jacob(2,1).*dy1da_t + jacob(2,2).*dy2da_t;
%     dy3da_t = dy3da_s + jacob(3,1).*dy1da_t + jacob(3,2).*dy2da_t +
 jacob(3,3).*dy3da_t;
      out_des = yesc(k+1,:);
      out_des = out_des';
      er = (out_red - out_des);
      erJ = (out_red - out_des).^1;
      q1 = 1;     % qq = 0 >> solo se mide la primera variable
      q2 = 1;
%     q3 = 1;
%     dJdw_t = dJdw_t + q1*erJ(1,1).*dy1dw_t + q2*erJ(2,1).*dy2dw_t +
 q3*erJ(3,1).*dy3dw_t;
      dJdw_t = dJdw_t + q1*erJ(1,1).*dy1dw_t + q2*erJ(2,1).*dy2dw_t;
%     dJdv_t = dJdv_t + q1*erJ(1,1).*dy1dv_t + q2*erJ(2,1).*dy2dv_t +
 q3*erJ(3,1).*dy3dv_t;
      dJdv_t = dJdv_t + q1*erJ(1,1).*dy1dv_t + q2*erJ(2,1).*dy2dv_t;
%     dJdc_t = dJdc_t + q1*erJ(1,1).*dy1dc_t + q2*erJ(2,1).*dy2dc_t +
 q3*erJ(3,1).*dy3dc_t;
      dJdc_t = dJdc_t + q1*erJ(1,1).*dy1dc_t + q2*erJ(2,1).*dy2dc_t;
%     dJda_t = dJda_t + q1*erJ(1,1).*dy1da_t + q2*erJ(2,1).*dy2da_t +
 q3*erJ(3,1).*dy3da_t ;
      dJda_t = dJda_t + q1*erJ(1,1).*dy1da_t + q2*erJ(2,1).*dy2da_t;
      ersum2 = ersum2 + er.^2;
      %x1 = x;
      x = out_red;     % output turns to be input for the next step
      %u1 = [u(k,1); u(k,2); u(k,3); u(k,4)];
   end
      dJdw_t = dJdw_t/ndata;
      dJdv_t = dJdv_t/ndata;
      dJdc_t = dJdc_t/ndata;
      dJda_t = dJda_t/ndata;
      dw = dJdw_t;
      dv = dJdv_t;
      dc = dJdc_t;
      da = dJda_t;

  ersum2total = sum(ersum2);
  cont = cont + 1;
  if ( rem(cont,1) == 0 )
      errorrel(cont/1,:) = sqrt(ersum2'./outsum2');
      errorreltotal(cont/1,1) = sqrt(ersum2total/outsum2total);
      erreltotal = errorreltotal(cont/1,1);
      cont;
      erreltotal
  end
      w = w - eta*dw;
      v = v - eta*dv;
      c = c - etac*dc;
      a = a - etaa*da;
      dw_old = dw;
      dv_old = dv;
end
```

```
figure(4);
plot(errorreltotal*100);
hold on;
plot(errorrel*100);
title('e vs. k','FontSize',16);
legend('errorTotal','errorX','errorY','FontSize',16);
xlabel('Número de iteraciones (k) [s/u]','FontSize',16);
ylabel('Funcion error (e) [%]','FontSize',16);
grid on;
%legend('errorX','errorY','errorZ');
figure(5);
plot(yesc(:,1),'-r');
hold on;
plot(outputesc(:,1),'-b');
title('Salida deseada (red) vs. Salida del modelo (blue) - X','FontSize',16);
legend('dataDeseadaX','dataEstimadaX','FontSize',16);
xlabel('Número de muestra (k) [s/u] ','FontSize',16);
ylabel('Posición Escalada (xk) [mm/mm]','FontSize',16);
grid on;
figure(6);
plot(yesc(:,2),'-r');
hold on;
plot(outputesc(:,2),'-b');
title('Salida deseada (red) vs. Salida del modelo (blue) - Y','FontSize',16);
legend('dataDeseadaY','dataEstimadaY','FontSize',16);
xlabel('Número de muestra (k) [s/u] ','FontSize',16);
ylabel('Posición Escalada (yk) [mm/mm]','FontSize',16);
grid on;
%figure(8);
%plot(yesc(:,3),'-r');
%hold on;
%plot(outputesc(:,3),'-b');
%title('Salida deseada (red) vs. Salida del modelo (blue)- Z','FontSize',16);
%legend('dataDeseadaZ','dataEstimadaZ');
%grid on;

ana=input("Save = 1");
if ana==1
 %    save soft4modelpxmaspymas ne nm ns v w c a factx facty;
      save soft4model01 ne nm ns v w c a factx facty;
end
```

ANEXO 03: Código para la validación del modelo del sistema

```
% Descripción: Código para valida el modelo del endoscopio blando

clear;
clc;
close all;

uast1 = [0.75 0.75 0.75 0.75 0.75 0.75 0.75;
         0.50 0.50 0.50 0.50 0.50 0.50 0.50;
         0.25 0.25 0.25 0.25 0.25 0.25 0.25;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00];

uast2 = [0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00;
         0.75 0.50 0.25 0.00 0.00 0.00 0.00];

uast3 = [0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.00 0.00 0.00 0.00 0.00 0.00 0.00;
         0.25 0.25 0.25 0.25 0.25 0.25 0.25;
         0.50 0.50 0.50 0.50 0.50 0.50 0.50;
         0.75 0.75 0.75 0.75 0.75 0.75 0.75];

uast4 = [0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75;
         0.00 0.00 0.00 0.00 0.25 0.50 0.75];

%xast = [];

%yast = [];

% uast1=[ 0.1000  0.3000  0.5000  0.7000  0.9000;
%         0.1000  0.3000  0.5000  0.7000  0.9000];
% xast1=[-0.0941 -0.2915 -0.4836 -0.6671 -0.8395
%        -0.0832 -0.2804 -0.4734 -0.6585 -0.8329
%        -0.0724 -0.2686 -0.4614 -0.6473 -0.8232
%        -0.0621 -0.2563 -0.4481 -0.6340 -0.8108
%        -0.0527 -0.2441 -0.4341 -0.6191 -0.7961];
% yast1=[-0.0801 -0.0509 -0.0220  0.0061  0.0331
%        -0.2680 -0.2391 -0.2096 -0.1800 -0.1508
%        -0.4514 -0.4240 -0.3951 -0.3653 -0.3350
```

```
%        -0.6270 -0.6022 -0.5751 -0.5461 -0.5160
%        -0.7925 -0.7710 -0.7465 -0.7195 -0.6906];
% zast1=[-0.0869 -0.1717 -0.2541 -0.3326 -0.4062
%        -0.1744 -0.2592 -0.3415 -0.4199 -0.4931
%        -0.2597 -0.3447 -0.4270 -0.5054 -0.5787
%        -0.3416 -0.4267 -0.5092 -0.5878 -0.6613
%        -0.4190 -0.5042 -0.5870 -0.6659 -0.7397];

%load('dataTraining021022.mat');
load('dataValidation031022.mat');

%px+ / py+
%col=[1 2];
%fil=[1 6801];
%fil=[1 1000];

%px+ / py-
% col=[1 4];
% fil=[6802 13601];

%px- / py+
%col=[2 3];
%fil=[13602 20401];

%px- / py-
%col=[3 4];
%fil=[20402 27200];

%um=[um(:,col(1)) um(:,col(2))];
%um=um(fil(1):round(fil(1)+(fil(2)-fil(1))/2),:);
%um=um(fil(1):fil(2),:);
%ym=ym(fil(1):round(fil(1)+(fil(2)-fil(1))/2),1:2);
%ym=ym(fil(1):fil(2),1:2);
ym  = ym(:,1:2);
%t=t(fil(1):round(fil(1)+(fil(2)-fil(1))/2));
%t=t(fil(1):fil(2));

nu = length(t);

figure(1);
plot(t,um);
grid on;
xlabel('Tiempo(t) [s]','FontSize',16);
ylabel('Presion (um) [bar]','FontSize',16);
title ('um vs. t','FontSize',16);
legend('um1 = px+','um2 = py+','um3 = px-','um4 = py-','FontSize',16);
%legend('um1 = px+','um2 = py+','FontSize',16);

figure(2);
plot(t,ym);
grid on;
xlabel('Tiempo(t) [s]','FontSize',16);
ylabel('Posición(ym) [mm]','FontSize',16);
title ('ym vs. t','FontSize',16);
```

```
%legend('ym1 = x','ym2 = y','ym3 = z','FontSize',16);
legend('ym1 = x','ym2 = y','FontSize',16);

% Escalamiento
factx = max(abs(um));
facty = max(abs(ym));

xesc(:,1) = um(:,1)./factx(1,1);
xesc(:,2) = um(:,2)./factx(1,2);
xesc(:,3) = um(:,3)./factx(1,3);
xesc(:,4) = um(:,4)./factx(1,4);

yesc(:,1) = ym(:,1)./facty(1,1);
yesc(:,2) = ym(:,2)./facty(1,2);
%yesc(:,3) = ym(:,3)./facty(1,3);

figure(3);
%plot(t,yesc(:,1),t,yesc(:,2),t,yesc(:,3));
plot(t,yesc(:,1),t,yesc(:,2));
grid on;
xlabel('Tiempo (t) [s]','FontSize',16);
ylabel('Posición escalada(yesc) [mm/mm max]','FontSize',16);
title ('yesc vs. t','FontSize',16);
legend('yesc1 = y1/y1max','yesc2 = y2/y2max','FontSize',16);
%legend('yesc1 = y1/y1max','yesc2 = y2/y2max','yesc3 = y3/
y3max','FontSize',16);

ndata = nu;
%ndata = 10;

dataoutesc = yesc;
u = xesc;

%load soft4modelpxmaspymas;
load soft4model01;

%Validation
outsum2 = sum(dataoutesc.^2);
outsum2 = outsum2';
outsum2total = sum(outsum2);

ersum2 = zeros(ns,1);

x = yesc(1,:);
x = x';
%x=zeros(2,1);
outputesc(1,:) = x;
erSum = 0;
erSumTotal = 0;
%filfil = 7;
%colcol = 7;
for k = 1:ndata-1
  in_red = [ x
             u(k,1)
```

```
            u(k,2)
            u(k,3)
            u(k,4)
           ];
  % in_red = [ x
  %            uast1(filfil,colcol)
  %            uast2(filfil,colcol)
  %            uast3(filfil,colcol)
  %            uast4(filfil,colcol)
  %           ];

 m = v'*in_red;
 n = 2.0./(1 + exp(-(m-c)./a)) - 1;
 %n = m;         % Lineal
 out_red = w'*n;
 %outputesc(k,:) = out_red';
 outputesc(k+1,:) = out_red';
 out_des = yesc(k+1,:);
 out_des = out_des';
 er = (out_red - out_des);
 erSum = erSum + er.^2;
 erSumTotal = sum(erSum);
 x = out_red;     % output turns to be input for the next step
 end

 erRel = sqrt(erSum./outsum2);
 erRelTotal = sqrt(erSumTotal/outsum2total);

figure(4);
plot(yesc(:,1),'-r');
hold on;
plot(outputesc(:,1),'-b');
%title('Posición X Escalada: Del modelo (blue)','FontSize',16);
title('Posición X Escalada: Deseada (red) vs. Del modelo
 (blue)','FontSize',16);
xlabel('Número de muestras (k) [s/u]','FontSize',16);
ylabel('Posición Escalada [mm/mm max]','FontSize',16);
legend('DataDeseadaX','DataEstimadaX','FontSize',16);
grid on;

figure(5);
plot(yesc(:,2),'-r');
hold on;
plot(outputesc(:,2),'-b');
%title('Posición Y Escalada: Del modelo (blue)','FontSize',16);
title('Posición Y Escalada: Deseada (red) vs. Del modelo
 (blue)','FontSize',16);
xlabel('Número de muestras (k) [s/u]','FontSize',16);
ylabel('Posición Escalada [mm/mm max]','FontSize',16);
legend('DataDeseadaY','DataEstimadaY','FontSize',16);
grid on;
```

ANEXO 04: Código para entrenamiento del controlador

```
% Descripción: Código para el entrenamiento del controlador de posición del
% endoscopio blando
% Algoritmo de entranamiento: Dynamic back-propagation algorithm (DBP)
% Estrategia de entrenamiento: Utilizar diferentes posiciones deseadas
% en todo el rango de operación. Probar efecto con y sin neurona bias.
% Objetivo: Estabilizar el sistema para diferentes posiciones deseadas y
% posiciones iniciales.
% Datos de entrenamiento: Normalizados en el rango de -1 a 1
% Nota: Se requiere conocer x* and u* para regulación y tracking
% Modelo del sistema: Red Neuronal Recurrente

clear;
clc;
close all;

disp('Hello');

%load soft4modelxmasymas;
load soft4model01;

nem = ne;
nmm = nm;
nsm = ns;
vm = v;
wm = w;
cm = c;
am = a;

ndata = 50;

xast = [-0.6092 -0.6240 -0.6312 -0.6307 -0.6242 -0.6136 -0.6016;
        -0.4081 -0.4199 -0.4260 -0.4265 -0.4231 -0.4176 -0.4126;
        -0.2031 -0.2107 -0.2145 -0.2151 -0.2142 -0.2134 -0.2151;
         0.0024 -0.0004 -0.0002 -0.0000 -0.0007 -0.0039 -0.0117;
         0.2193  0.2240  0.2278  0.2299  0.2284  0.2221  0.2093;
         0.4298  0.4415  0.4503  0.4551  0.4541  0.4460  0.4296;
         0.6307  0.6495  0.6636  0.6716  0.6719  0.6635  0.6451];
yast = [-0.5946 -0.4106 -0.2178 -0.0185  0.2057  0.4312  0.6525;
        -0.6060 -0.4153 -0.2160 -0.0110  0.2174  0.4448  0.6654;
        -0.6136 -0.4179 -0.2139 -0.0051  0.2255  0.4527  0.6711;
        -0.6162 -0.4171 -0.2105 -0.0000  0.2306  0.4556  0.6701;
        -0.6190 -0.4153 -0.2081  0.0018  0.2301  0.4507  0.6594;
        -0.6084 -0.4080 -0.2023  0.0047  0.2283  0.4425  0.6437;
        -0.5933 -0.3946 -0.1925  0.0095  0.2264  0.4324  0.6247];
% Initial state for learning
xiniPos = [4 5 6 7];
% x_ini1=[xast1(1,1) yast1(1,1) zast1(1,1)]';
x_ini1 = [xast(xiniPos(1),xiniPos(1)) yast(xiniPos(1),xiniPos(1))]';
% x_ini2=[xast1(2,2) yast1(2,2) zast1(2,2)]';
x_ini2 = [xast(xiniPos(2),xiniPos(2)) yast(xiniPos(2),xiniPos(2))]';
% x_ini3=[xast1(3,3) yast1(3,3) zast1(3,3)]';
x_ini3 = [xast(xiniPos(3),xiniPos(3)) yast(xiniPos(3),xiniPos(3))]';
```

```
% x_ini4=[xast1(5,4) yast1(5,4) zast1(5,4)]';
x_ini4 = [xast(xiniPos(4),xiniPos(4)) yast(xiniPos(4),xiniPos(4))]';
x_ini = [x_ini1 x_ini2 x_ini3 x_ini4];

% Desired position state for learning
%xref = [xast2(:,3) xast2(:,6)];
%xref = [xast1(5,5) yast1(5,5) zast1(5,5)]';

%xref1 = [xast(1,1) yast(1,1)]';
%xref2 = [xast(2,2) yast(2,2)]';
%xref = [xref1 xref2];

xrefFil = 2;
xrefCol = 1;
xref = [xast(xrefFil,xrefCol) yast(xrefFil,xrefCol)]';

%Control Network Structure
%ne = 4  ;    % Con bias
ne = 3;     % Sin bias
nm = 4;
ns = 4;

v = 0.1*randn(ne,nm);
w = 0.1*randn(nm,ns);
c = zeros(nm,1);
a = ones(nm,1);

%load soft4controlxmasymas
%load soft4control11;

%Training parameters
eta  = input('Learning rate of v and w: ');
etaa  = input('Learning rate of sigmoid slope a : ');
%etac  = input('Learning rate of sigmoid center c : ');
etac = 0;
errormax = input('Input maximum error (%) : ');
errormax = errormax/100;
contmax = input('Input maximum number of iterations for learning: ');

cont = 1;
erreltotal = 1;

outsum2 = 0;

% Calculate from Desired position state to obtain error
aa =1;
bb =1;
for i = aa:bb
     r = xref(:,i);     % Vector de estado deseado
     %r = xref(1:2,i);  % Vector de estado deseado
     %dataoutesc = ones(ndata,3) * diag(r);
     dataoutesc = ones(ndata,2) * diag(r);
     outsum2 = sum(dataoutesc.^2) + outsum2;
     %outsum2 = outsum2';
```

```
end
outsum2total = sum(outsum2);

while( (erreltotal > errormax) & (cont < contmax) )

   %ersum2 = zeros(3,1);
   ersum2 = zeros(2,1);
   dJdw = 0;
   dJdv = 0;
   dx1dw_t = zeros(nm,ns);
   dx2dw_t = zeros(nm,ns);
   %dx3dw_t = zeros(nm,ns);
   dx1dv_t = zeros(ne,nm);
   dx2dv_t = zeros(ne,nm);
   %dx3dv_t = zeros(ne,nm);
   dx1dc_t = zeros(nm,1);
   dx2dc_t = zeros(nm,1);
   %dx3dc_t = zeros(nm,1);
   dx1da_t = zeros(nm,1);
   dx2da_t = zeros(nm,1);
   %dx3da_t = zeros(nm,1);
   dJdw_t  = zeros(nm,ns);
   dJdv_t  = zeros(ne,nm);
   dJdc_t  = zeros(nm,1);
   dJda_t  = zeros(nm,1);


%    x = x_ini;
%    clear u;
%    clear estado;

   ktot = 0;

   for i = aa:bb
        r = xref(:,i);     % Vector de estado deseado
        %r = xref(1:2,i);     % Vector de estado deseado
        %dataoutesc = ones(ndata,3) * diag(r);
        dataoutesc = ones(ndata,2) * diag(r);

    for j = 1:4

         x = x_ini(:,j);

        for k = 1:ndata-1
%          if( k <= ndata/2)
%             r = [ xlast
%                   x2ast ];
%             uast1 = uast;
%          else
%             r = [ 0
%                   0 ];
%             uast1 = 0;
%          end
```

```
in_red = x + 0*1*0.001*randn(2,1) -r;
%in_red = x(1:2,1) + 0*1*0.001*randn(2,1) -r;
%m = v'*in_red; %sin bias
m = v'*[1;in_red]; %con bias
n = 2.0./(1 + exp(-(m-c)./a)) - 1;
%     n = m;
out_red = w'*n;
if out_red(1) < 0
  out_red(1) = 0;
end

if out_red(2) < 0
    out_red(2) = 0;
end

if out_red(3) < 0
    out_red(3) = 0;
end

if out_red(4) < 0
    out_red(4) = 0;
end

%         if (i == 1||i == 5||i == 9)
%             out_red(2) = 0;
%             out_red(3) = 0;
%             out_red(4) = 0;
%         end
%
%         if (i == 2||i == 6||i == 10)
%             out_red(1) = 0;
%             out_red(3) = 0;
%             out_red(4) = 0;
%         end
%
%         if (i == 3||i == 7||i == 11)
%             out_red(1) = 0;
%             out_red(2) = 0;
%             out_red(4) = 0;
%         end
%
%         if (i == 4||i == 8||i == 12)
%             out_red(1) = 0;
%             out_red(2) = 0;
%             out_red(3) = 0;
%         end

%     u(k,j) = out_red' + uast1;
u(k,:,j) = out_red';
uEscalado(k,:,j) = out_red'*diag(factx');
[ x' out_red' ];
estado(k,:,j) = x';
estadoEscalado(k,:,j) = x'*diag(facty');
```

```
%     modelo
in_red_m = [x;out_red];
m_m = vm'*in_red_m;
n_m = 2.0./(1 + exp(-(m_m-cm)./am)) - 1;
x = wm'*n_m;

dndm_m = diag((1 - n_m.*n_m)./(2*am));
jacob = wm'*dndm_m*(vm(1:2,:))';
dxdu = wm'*dndm_m*(vm(3:6,:))';

%     neurocontrolador
dndm = diag((1 - n.*n)./(2*a));
%     dndm = diag(ones(nm,1));
du1dw_s = [ n zeros(nm,1) zeros(nm,1) zeros(nm,1)];
%du1dw_s = [ n zeros(nm,1)];
%du1dw_s = n;
du2dw_s = [ zeros(nm,1) n zeros(nm,1) zeros(nm,1)];
%du2dw_s = [ zeros(nm,1) n ];
du3dw_s = [ zeros(nm,1) zeros(nm,1) n zeros(nm,1)];
du4dw_s = [ zeros(nm,1) zeros(nm,1) zeros(nm,1) n];

du1dv_s = [1;in_red]*w(:,1)'*dndm;
du2dv_s = [1;in_red]*w(:,2)'*dndm;
du3dv_s = [1;in_red]*w(:,3)'*dndm;
du4dv_s = [1;in_red]*w(:,4)'*dndm;

%           du1dv_s = in_red*w(:,1)'*dndm;
%           du2dv_s = in_red*w(:,2)'*dndm;
%           du3dv_s = in_red*w(:,3)'*dndm;
%           du4dv_s = in_red*w(:,4)'*dndm;

du1dc_s = w(:,1) .* ((n.*n-1)./(2.0.*a));
%du1dc_s = w.* ((n.*n-1)./(2.0.*a));
du2dc_s = w(:,2) .* ((n.*n-1)./(2.0.*a));
du3dc_s = w(:,3) .* ((n.*n-1)./(2.0.*a));
du4dc_s = w(:,4) .* ((n.*n-1)./(2.0.*a));

du1da_s = w(:,1) .* ((n.*n-1).*(m-c)./(2*a.*a));
%du1da_s = w.* ((n.*n-1).*(m-c)./(2*a.*a));
du2da_s = w(:,2) .* ((n.*n-1).*(m-c)./(2*a.*a));
du3da_s = w(:,3) .* ((n.*n-1).*(m-c)./(2*a.*a));
du4da_s = w(:,4) .* ((n.*n-1).*(m-c)./(2*a.*a));

dudx = w'*dndm*v(2:3,:)';
%dudx = w'*dndm*v';

jacob_t = dxdu*dudx + jacob;

%dx1dw_t = dxdu(1,1).*du1dw_s + dxdu(1,2).*du2dw_s +
dxdu(1,3).*du3dw_s +  dxdu(1,4).*du4dw_s + jacob_t(1,1).*dx1dw_t +
jacob_t(1,2).*dx2dw_t + jacob_t(1,3).*dx3dw_t;
%dx1dw_t = dxdu(1,1).*du1dw_s + dxdu(1,2).*du2dw_s +
jacob_t(1,1).*dx1dw_t + jacob_t(1,2).*dx2dw_t + jacob_t(1,3).*dx3dw_t;
```

```
        dx1dw_t = dxdu(1,1).*du1dw_s + dxdu(1,2).*du2dw_s +
dxdu(1,3).*du3dw_s +  dxdu(1,4).*du4dw_s + jacob_t(1,1).*dx1dw_t +
jacob_t(1,2).*dx2dw_t;
        %dx2dw_t = dxdu(2,1).*du1dw_s + dxdu(2,2).*du2dw_s +
dxdu(2,3).*du3dw_s +  dxdu(2,4).*du4dw_s + jacob_t(2,1).*dx1dw_t +
jacob_t(2,2).*dx2dw_t + jacob_t(2,3).*dx3dw_t;
        %dx2dw_t = dxdu(2,1).*du1dw_s + dxdu(2,2).*du2dw_s +
jacob_t(2,1).*dx1dw_t + jacob_t(2,2).*dx2dw_t + jacob_t(2,3).*dx3dw_t;
        dx2dw_t = dxdu(2,1).*du1dw_s + dxdu(2,2).*du2dw_s +
dxdu(2,3).*du3dw_s +  dxdu(2,4).*du4dw_s + jacob_t(2,1).*dx1dw_t +
jacob_t(2,2).*dx2dw_t;
        %dx3dw_t = dxdu(3,1).*du1dw_s + dxdu(3,2).*du2dw_s +
dxdu(3,3).*du3dw_s +  dxdu(3,4).*du4dw_s + jacob_t(3,1).*dx1dw_t +
jacob_t(3,2).*dx2dw_t + jacob_t(3,3).*dx3dw_t;
        %dx3dw_t = dxdu(3,1).*du1dw_s + dxdu(3,2).*du2dw_s +
jacob_t(3,1).*dx1dw_t + jacob_t(3,2).*dx2dw_t + jacob_t(3,3).*dx3dw_t;

        %dx1dv_t = dxdu(1,1).*du1dv_s + dxdu(1,2).*du2dv_s +
dxdu(1,3).*du3dv_s +  dxdu(1,4).*du4dv_s + jacob_t(1,1).*dx1dv_t +
jacob_t(1,2).*dx2dv_t + jacob_t(1,3).*dx3dv_t;
        %dx1dv_t = dxdu(1,1).*du1dv_s + dxdu(1,2).*du2dv_s +
jacob_t(1,1).*dx1dv_t + jacob_t(1,2).*dx2dv_t + jacob_t(1,3).*dx3dv_t;
        dx1dv_t = dxdu(1,1).*du1dv_s + dxdu(1,2).*du2dv_s +
dxdu(1,3).*du3dv_s +  dxdu(1,4).*du4dv_s + jacob_t(1,1).*dx1dv_t +
jacob_t(1,2).*dx2dv_t;
        %dx2dv_t = dxdu(2,1).*du1dv_s + dxdu(2,2).*du2dv_s +
dxdu(2,3).*du3dv_s +  dxdu(2,4).*du4dv_s + jacob_t(2,1).*dx1dv_t +
jacob_t(2,2).*dx2dv_t + jacob_t(2,3).*dx3dv_t;
        %dx2dv_t = dxdu(2,1).*du1dv_s + dxdu(2,2).*du2dv_s +
jacob_t(2,1).*dx1dv_t + jacob_t(2,2).*dx2dv_t + jacob_t(2,3).*dx3dv_t;
        dx2dv_t = dxdu(2,1).*du1dv_s + dxdu(2,2).*du2dv_s +
dxdu(2,3).*du3dv_s +  dxdu(2,4).*du4dv_s + jacob_t(2,1).*dx1dv_t +
jacob_t(2,2).*dx2dv_t;
        %dx3dv_t = dxdu(3,1).*du1dv_s + dxdu(3,2).*du2dv_s +
dxdu(3,3).*du3dv_s +  dxdu(3,4).*du4dv_s + jacob_t(3,1).*dx1dv_t +
jacob_t(3,2).*dx2dv_t + jacob_t(3,3).*dx3dv_t;
        %dx3dv_t = dxdu(3,1).*du1dv_s + dxdu(3,2).*du2dv_s +
jacob_t(3,1).*dx1dv_t + jacob_t(3,2).*dx2dv_t + jacob_t(3,3).*dx3dv_t;

        %dx1dc_t = dxdu(1,1).*du1dc_s + dxdu(1,2).*du2dc_s +
dxdu(1,3).*du3dc_s +  dxdu(1,4).*du4dc_s + jacob_t(1,1).*dx1dc_t +
jacob_t(1,2).*dx2dc_t + jacob_t(1,3).*dx3dc_t;
        %dx1dc_t = dxdu(1,1).*du1dc_s + dxdu(1,2).*du2dc_s +
jacob_t(1,1).*dx1dc_t + jacob_t(1,2).*dx2dc_t + jacob_t(1,3).*dx3dc_t;
        dx1dc_t = dxdu(1,1).*du1dc_s + dxdu(1,2).*du2dc_s +
dxdu(1,3).*du3dc_s +  dxdu(1,4).*du4dc_s + jacob_t(1,1).*dx1dc_t +
jacob_t(1,2).*dx2dc_t;
        %dx2dc_t = dxdu(2,1).*du1dc_s + dxdu(2,2).*du2dc_s +
dxdu(2,3).*du3dc_s +  dxdu(2,4).*du4dc_s + jacob_t(2,1).*dx1dc_t +
jacob_t(2,2).*dx2dc_t + jacob_t(2,3).*dx3dc_t;
        %dx2dc_t = dxdu(2,1).*du1dc_s + dxdu(2,2).*du2dc_s +
jacob_t(2,1).*dx1dc_t + jacob_t(2,2).*dx2dc_t + jacob_t(2,3).*dx3dc_t;
```

```
        dx2dc_t = dxdu(2,1).*du1dc_s + dxdu(2,2).*du2dc_s +
dxdu(2,3).*du3dc_s +  dxdu(2,4).*du4dc_s + jacob_t(2,1).*dx1dc_t +
jacob_t(2,2).*dx2dc_t;
        %dx3dc_t = dxdu(3,1).*du1dc_s + dxdu(3,2).*du2dc_s +
dxdu(3,3).*du3dc_s +  dxdu(3,4).*du4dc_s + jacob_t(3,1).*dx1dc_t +
jacob_t(3,2).*dx2dc_t + jacob_t(3,3).*dx3dc_t;
        %dx3dc_t = dxdu(3,1).*du1dc_s + dxdu(3,2).*du2dc_s +
jacob_t(3,1).*dx1dc_t + jacob_t(3,2).*dx2dc_t + jacob_t(3,3).*dx3dc_t;

        %dx1da_t = dxdu(1,1).*du1da_s + dxdu(1,2).*du2da_s +
dxdu(1,3).*du3da_s +  dxdu(1,4).*du4da_s + jacob_t(1,2).*dx1da_t +
jacob_t(1,2).*dx2da_t + jacob_t(1,3).*dx3da_t;
        %dx1da_t = dxdu(1,1).*du1da_s + dxdu(1,2).*du2da_s +
jacob_t(1,2).*dx1da_t + jacob_t(1,2).*dx2da_t + jacob_t(1,3).*dx3da_t;
        dx1da_t = dxdu(1,1).*du1da_s + dxdu(1,2).*du2da_s +
dxdu(1,3).*du3da_s +  dxdu(1,4).*du4da_s + jacob_t(1,2).*dx1da_t +
jacob_t(1,2).*dx2da_t;
        %dx2da_t = dxdu(2,1).*du1da_s + dxdu(2,2).*du2da_s +
dxdu(2,3).*du3da_s +  dxdu(2,4).*du4da_s + jacob_t(2,1).*dx1da_t +
jacob_t(2,2).*dx2da_t + jacob_t(2,3).*dx3da_t;
        %dx2da_t = dxdu(2,1).*du1da_s + dxdu(2,2).*du2da_s +
jacob_t(2,1).*dx1da_t + jacob_t(2,2).*dx2da_t + jacob_t(2,3).*dx3da_t;
        dx2da_t = dxdu(2,1).*du1da_s + dxdu(2,2).*du2da_s +
dxdu(2,3).*du3da_s +  dxdu(2,4).*du4da_s + jacob_t(2,1).*dx1da_t +
jacob_t(2,2).*dx2da_t;
        %dx3da_t = dxdu(3,1).*du1da_s + dxdu(3,2).*du2da_s +
dxdu(3,3).*du3da_s +  dxdu(3,4).*du4da_s + jacob_t(3,1).*dx1da_t +
jacob_t(3,2).*dx2da_t + jacob_t(3,3).*dx3da_t;
        %dx3da_t = dxdu(3,1).*du1da_s + dxdu(3,2).*du2da_s +
jacob_t(3,1).*dx1da_t + jacob_t(3,2).*dx2da_t + jacob_t(3,3).*dx3da_t;

        out_des = dataoutesc(k+1,:);
        out_des = out_des';
        er = (x - out_des);
        %er = x(1:2,1) - out_des;
        erJ = x - out_des;
        %erJ = x(1:2,1) - out_des;

        q1 = 1;
        q2 = 1;
        %q3 = 1;

        %dJdw_t =  dJdw_t + q1*erJ(1,1).*dx1dw_t + q2*erJ(2,1).*dx2dw_t +
q3*erJ(3,1).*dx3dw_t;
        dJdw_t =  dJdw_t + q1*erJ(1,1).*dx1dw_t + q2*erJ(2,1).*dx2dw_t;
        %dJdv_t =  dJdv_t + q1*erJ(1,1).*dx1dv_t + q2*erJ(2,1).*dx2dv_t +
q3*erJ(3,1).*dx3dv_t;
        dJdv_t =  dJdv_t + q1*erJ(1,1).*dx1dv_t + q2*erJ(2,1).*dx2dv_t;
        %dJdc_t =  dJdc_t + q1*erJ(1,1).*dx1dc_t + q2*erJ(2,1).*dx2dc_t +
q3*erJ(3,1).*dx3dc_t;
        dJdc_t =  dJdc_t + q1*erJ(1,1).*dx1dc_t + q2*erJ(2,1).*dx2dc_t;
        %dJda_t =  dJda_t + q1*erJ(1,1).*dx1da_t + q2*erJ(2,1).*dx2da_t +
q3*erJ(3,1).*dx3da_t;
        dJda_t =  dJda_t + q1*erJ(1,1).*dx1da_t + q2*erJ(2,1).*dx2da_t;
```

```
            ersum2 = ersum2 + er.^2;
            %       if( (abs(x(1,1)) > 10)  | (abs(x(2,1)) > 10) )
            %           break;
            %       end

          end
          ktot = ktot + k;
      end
    end

    dJdw_t = dJdw_t/ktot;
    dJdv_t = dJdv_t/ktot;
    dJdc_t = dJdc_t/ktot;
    dJda_t = dJda_t/ktot;
    dw = dJdw_t;
    dv = dJdv_t;
    dc = dJdc_t;
    da = dJda_t;
    w = w - eta*dw;
    v = v - eta*dv;
    c = c - etac*dc;
    a = a - etaa*da;

    dw_old = dw;
    dv_old = dv;
    dc_old = dc;
    da_old = da;

    ersum2total = sum(ersum2)
    erreltotal = ersum2total/outsum2total
    cont = cont + 1;
    cont

end

figure(1);
plot(estado(:,:,1));
hold on;
plot(dataoutesc);
title(' x vs. n (Condición Inicial 1)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones Escaladas (x) [mm/mm max]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
%legend('x1 = x','x2 = y','FontSize',16);
legend('x1 = x','x2 = y','x1* = xref','x2* = yref','FontSize',16);
grid on;
figure(2);
plot(u(:,:,1));
title('p vs. n (Condición Inicial 1)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones Escaladas (p) [bar/bar max]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;
```

```
figure(3);
plot(estado(:,:,2));
hold on;
plot(dataoutesc);
title(' x vs. n (Condición Inicial 2)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones Escaladas (x) [mm/mm max]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
%legend('x1 = x','x2 = y','FontSize',16);
legend('x1 = x','x2 = y','x1* = xref','x2* = yref','FontSize',16);
grid on;
figure(4);
plot(u(:,:,2));
title('p vs. n (Condición Inicial 2)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones Escaladas (p) [bar/bar max]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;

figure(5);
plot(estado(:,:,3));
hold on;
plot(dataoutesc);
title(' x vs. n (Condición Inicial 3)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones Escaladas (x) [mm/mm max]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
%legend('x1 = x','x2 = y','FontSize',16);
legend('x1 = x','x2 = y','x1* = xref','x2* = yref','FontSize',16);
grid on;
figure(6);
plot(u(:,:,3));
title('p vs. n (Condición Inicial 3)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones Escaladas (p) [bar/bar max]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;

figure(7);
plot(estado(:,:,4));
hold on;
plot(dataoutesc);
title(' x vs. n (Condición Inicial 4)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones Escaladas (x) [mm/mm max]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
%legend('x1 = x','x2 = y','FontSize',16);
legend('x1 = x','x2 = y','x1* = xref','x2* = yref','FontSize',16);
grid on;
figure(8);
plot(u(:,:,4));
```

```
title('p vs. n (Condición Inicial 4)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones Escaladas (p) [bar/bar max]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;
%{
figure(9);
plot(estadoEscalado(:,:,1));
title(' x vs. n (Condición Inicial 1)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones(x) [mm]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
legend('x1 = x','x2 = y','FontSize',16);
grid on;
figure(10);
plot(uEscalado(:,:,1));
title('p vs. n (Condición Inicial 1)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones (p) [bar]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;

figure(11);
plot(estadoEscalado(:,:,2));
title(' x vs. n (Condición Inicial 2)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones(x) [mm]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
legend('x1 = x','x2 = y','FontSize',16);
grid on;
figure(12);
plot(uEscalado(:,:,2));
title('p vs. n (Condición Inicial 2)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones (p) [bar]','FontSize',16);
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;

figure(13);
plot(estadoEscalado(:,:,3));
title(' x vs. n (Condición Inicial 3)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones(x) [mm]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
legend('x1 = x','x2 = y','FontSize',16);
grid on;
figure(14);
plot(uEscalado(:,:,3));
title('p vs. n (Condición Inicial 3)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones (p) [bar]','FontSize',16)
```

```
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;

figure(15);
plot(estadoEscalado(:,:,4));
title(' x vs. n (Condición Inicial 4)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Posiciones(x) [mm]','FontSize',16);
%legend('x1 = x','x2 = y','x3 = z','FontSize',16);
legend('x1 = x','x2 = y','FontSize',16);
grid on;
figure(16);
plot(uEscalado(:,:,4));
title('p vs. n (Condición Inicial 4)','FontSize',16);
xlabel('Número de muestras (n) [s/u]','FontSize',16);
ylabel('Presiones (p) [bar]','FontSize',16)
legend('p1= px+',' p2 = py+','p3 = px-','p4 = py-','FontSize',16);
%legend('p1= px+',' p2 = py+','FontSize',16);
grid on;
%}
ana=input('Guardar[SI = 1]');
if ana == 1
    %save soft4controlxmasymas55 v w c a nm ne ns factx facty;
    %save soft4controlxmasymas11 v w c a nm ne ns factx facty;
    save soft4control21 v w c a nm ne ns factx facty;
end
```

ANEXO 05: Simulación del sistema en lazo cerrado

```
clear;
close all;
clc;

% xast1=[-0.0941 -0.2915 -0.4836 -0.6671 -0.8395
%        -0.0832 -0.2804 -0.4734 -0.6585 -0.8329
%        -0.0724 -0.2686 -0.4614 -0.6473 -0.8232
%        -0.0621 -0.2563 -0.4481 -0.6340 -0.8108
%        -0.0527 -0.2441 -0.4341 -0.6191 -0.7961];
% yast1=[-0.0801 -0.0509 -0.0220  0.0061  0.0331
%        -0.2680 -0.2391 -0.2096 -0.1800 -0.1508
%        -0.4514 -0.4240 -0.3951 -0.3653 -0.3350
%        -0.6270 -0.6022 -0.5751 -0.5461 -0.5160
%        -0.7925 -0.7710 -0.7465 -0.7195 -0.6906];
% zast1=[-0.0869 -0.1717 -0.2541 -0.3326 -0.4062
%        -0.1744 -0.2592 -0.3415 -0.4199 -0.4931
%        -0.2597 -0.3447 -0.4270 -0.5054 -0.5787
%        -0.3416 -0.4267 -0.5092 -0.5878 -0.6613
%        -0.4190 -0.5042 -0.5870 -0.6659 -0.7397];

xast = [-0.6092 -0.6240 -0.6312 -0.6307 -0.6242 -0.6136 -0.6016;
        -0.4081 -0.4199 -0.4260 -0.4265 -0.4231 -0.4176 -0.4126;
        -0.2031 -0.2107 -0.2145 -0.2151 -0.2142 -0.2134 -0.2151;
         0.0024 -0.0004 -0.0002 -0.0000 -0.0007 -0.0039 -0.0117;
         0.2193  0.2240  0.2278  0.2299  0.2284  0.2221  0.2093;
         0.4298  0.4415  0.4503  0.4551  0.4541  0.4460  0.4296;
         0.6307  0.6495  0.6636  0.6716  0.6719  0.6635  0.6451];
yast = [-0.5946 -0.4106 -0.2178 -0.0185  0.2057  0.4312  0.6525;
        -0.6060 -0.4153 -0.2160 -0.0110  0.2174  0.4448  0.6654;
        -0.6136 -0.4179 -0.2139 -0.0051  0.2255  0.4527  0.6711;
        -0.6162 -0.4171 -0.2105 -0.0000  0.2306  0.4556  0.6701;
        -0.6190 -0.4153 -0.2081  0.0018  0.2301  0.4507  0.6594;
        -0.6084 -0.4080 -0.2023  0.0047  0.2283  0.4425  0.6437;
        -0.5933 -0.3946 -0.1925  0.0095  0.2264  0.4324  0.6247];

%Membership function of xast (in the range -1 to 1)
nx=3200; %Number of sample point between -1 and 1
%nx0=1; nx1=round(nx/10); nx2=nx1*3; nx3=nx1*5; nx4=nx1*7; nx5=nx1*9; nx6=nx;
nx0 = 1; nx1 = 590+50; nx2 = 918; nx3 = 1256; nx4 = 1600;
nx5 = 1968; nx6 = 2328; nx7 = 2675-50; nx8 = 3200;
nx01 = nx0:nx1;      nx01=nx01';
nx12 = (nx1+1):nx2;  nx12=nx12';
nx23 = (nx2+1):nx3;  nx23=nx23';
nx34 = (nx3+1):nx4;  nx34=nx34';
nx45 = (nx4+1):nx5;  nx45=nx45';
nx56 = (nx5+1):nx6;  nx56=nx56';
nx67 = (nx6+1):nx7;  nx67=nx67';
nx78 = (nx7+1):nx8;  nx78=nx78';
fdx1(nx0:nx1,1) = 0*nx01 + 1;
fdx1(nx1+1:nx2,1) = (nx12-nx2)/(nx1-nx2);
fdx1(nx2+1:nx,1) = 0*(nx2+1:nx)';
fdx2(nx0:nx1,1) = 0*nx01;
```

```
fdx2(nx1+1:nx2,1) = (nx12-nx1)/(nx2-nx1);
fdx2(nx2+1:nx3,1) = (nx23-nx3)/(nx2-nx3);
fdx2(nx3+1:nx,1) = 0*(nx3+1:nx)';
fdx3(nx0:nx2,1)=0*(nx0:nx2)';
fdx3(nx2+1:nx3,1)=(nx23-nx2)/(nx3-nx2);
fdx3(nx3+1:nx4,1)=(nx34-nx4)/(nx3-nx4);
fdx3(nx4+1:nx,1)=0*(nx4+1:nx)';
fdx4(nx0:nx3,1) = 0*(nx0:nx3)';
fdx4(nx3+1:nx4,1) = (nx34-nx3)/(nx4-nx3);
fdx4(nx4+1:nx5,1) = (nx45-nx5)/(nx4-nx5);
fdx4(nx5+1:nx,1) = 0*(nx5+1:nx)';
fdx5(nx0:nx4,1) = 0*(nx0:nx4)';
fdx5(nx4+1:nx5,1) = (nx45-nx4)/(nx5-nx4);
fdx5(nx5+1:nx6,1) = (nx56-nx6)/(nx5-nx6);
fdx5(nx6+1:nx,1) = 0*(nx6+1:nx)';
fdx6(nx0:nx5,1) = 0*(nx0:nx5)';
fdx6(nx5+1:nx6,1) = (nx56-nx5)/(nx6-nx5);
fdx6(nx6+1:nx7,1) = (nx67-nx7)/(nx6-nx7);
fdx6(nx7+1:nx,1) = 0*(nx7+1:nx)';
fdx7(nx0:nx6,1) = 0*(nx0:nx6)';
fdx7(nx6+1:nx7,1) = (nx67-nx6)/(nx7-nx6);
fdx7(nx7+1:nx,1) = 0*nx78 + 1;
nfdx = length(fdx1);
nnx = 1:nx; nnx = nnx';
%xfuz = (nnx-2000)/(nx-1); % Vector x in the range -1 to 0
xfuz = linspace(-1,1,nx);
figure(1);
subplot(2,1,1);
plot(xfuz,fdx1,'Linewidth',1.25); hold on;
plot(xfuz,fdx2,'Linewidth',1.25);
plot(xfuz,fdx3,'Linewidth',1.25);
plot(xfuz,fdx4,'Linewidth',1.25);
plot(xfuz,fdx5,'Linewidth',1.25);
plot(xfuz,fdx6,'Linewidth',1.25);
plot(xfuz,fdx7,'Linewidth',1.25);
title('Fx* vs x*','FontSize',16);
grid on;
xlabel('Posición deseada x (x*)','FontSize',16);
ylabel('Membresía de x* (Fx*)','FontSize',16);
axis([-1 1 0 1.4]);


%Membership function of yast (in the range -1 to 1)
ny = 3200; %Number of sample point between -1 and 1
%ny0=1; ny1=round(ny/10); ny2=ny1*3; ny3=ny1*5; ny4=ny1*7; ny5=ny1*9; ny6=ny;
ny0 = 1; ny1 = 614+50; ny2 = 933; ny3 = 1263; ny4 = 1600;
ny5 = 1969; ny6 = 2329; ny7 = 2672-50; ny8 = 3200;
ny01 = ny0:ny1;       ny01=ny01';
ny12 = (ny1+1):ny2;   ny12=ny12';
ny23 = (ny2+1):ny3;   ny23=ny23';
ny34 = (ny3+1):ny4;   ny34=ny34';
ny45 = (ny4+1):ny5;   ny45=ny45';
ny56 = (ny5+1):ny6;   ny56=ny56';
ny67 = (ny6+1):ny7;   ny67=ny67';
```

```
ny78 = (ny7+1):ny8;  ny78=ny78';
fdy1(ny0:ny1,1) = 0*ny01 + 1;
fdy1(ny1+1:ny2,1) = (ny12-ny2)/(ny1-ny2);
fdy1(ny2+1:ny,1) = 0*(ny2+1:ny)';
fdy2(ny0:ny1,1) = 0*ny01;
fdy2(ny1+1:ny2,1) = (ny12-ny1)/(ny2-ny1);
fdy2(ny2+1:ny3,1) = (ny23-ny3)/(ny2-ny3);
fdy2(ny3+1:ny,1) = 0*(ny3+1:ny)';
fdy3(ny0:ny2,1)=0*(ny0:ny2)';
fdy3(ny2+1:ny3,1)=(ny23-ny2)/(ny3-ny2);
fdy3(ny3+1:ny4,1)=(ny34-ny4)/(ny3-ny4);
fdy3(ny4+1:ny,1)=0*(ny4+1:ny)';
fdy4(ny0:ny3,1) = 0*(ny0:ny3)';
fdy4(ny3+1:ny4,1) = (ny34-ny3)/(ny4-ny3);
fdy4(ny4+1:ny5,1) = (ny45-ny5)/(ny4-ny5);
fdy4(ny5+1:ny,1) = 0*(ny5+1:ny)';
fdy5(ny0:ny4,1) = 0*(ny0:ny4)';
fdy5(ny4+1:ny5,1) = (ny45-ny4)/(ny5-ny4);
fdy5(ny5+1:ny6,1) = (ny56-ny6)/(ny5-ny6);
fdy5(ny6+1:ny,1) = 0*(ny6+1:ny)';
fdy6(ny0:ny5,1) = 0*(ny0:ny5)';
fdy6(ny5+1:ny6,1) = (ny56-ny5)/(ny6-ny5);
fdy6(ny6+1:ny7,1) = (ny67-ny7)/(ny6-ny7);
fdy6(ny7+1:ny,1) = 0*(ny7+1:ny)';
fdy7(ny0:ny6,1) = 0*(ny0:ny6)';
fdy7(ny6+1:ny7,1) = (ny67-ny6)/(ny7-ny6);
fdy7(ny7+1:ny,1) = 0*ny78 + 1;
nfdy = length(fdy1);
nny = 1:ny; nny = nny';
%yfuz = (nny-2000)/(ny-1); % Vector y in the range 0 to 1
yfuz = linspace(-1,1,ny);
figure(1);
subplot(2,1,2);
plot(yfuz,fdy1,'Linewidth',1.25); hold on;
plot(yfuz,fdy2,'Linewidth',1.25);
plot(yfuz,fdy3,'Linewidth',1.25);
plot(yfuz,fdy4,'Linewidth',1.25);
plot(yfuz,fdy5,'Linewidth',1.25);
plot(yfuz,fdy6,'Linewidth',1.25);
plot(yfuz,fdy7,'Linewidth',1.25);
title('Fy* vs y*','FontSize',16);
grid on;
xlabel('Posición deseada y* (y*)','FontSize',16);
ylabel('Membresía de y* (Fy*)','FontSize',16);
axis([-1 1 0 1.4]);


%Weight of trained neurocontrollers
load soft4control1;
ac11 = a; cc11 = c; vc11 = v; wc11 = w;
nec = ne; nmc = nm; nsc = ns;
load soft4control12;
ac12 = a; cc12 = c; vc12 = v; wc12 = w;
load soft4control13;
```

```
ac13 = a; cc13 = c; vc13 = v; wc13 = w;
load soft4control14;
ac14 = a; cc14 = c; vc14 = v; wc14 = w;
load soft4control15;
ac15 = a; cc15 = c; vc15 = v; wc15 = w;
load soft4control16;
ac16 = a; cc16 = c; vc16 = v; wc16 = w;
load soft4control17;
ac17 = a; cc17 = c; vc17 = v; wc17 = w;
load soft4control21;
ac21 = a; cc21 = c; vc21 = v; wc21 = w;
load soft4control22;
ac22 = a; cc22 = c; vc22 = v; wc22 = w;
load soft4control23;
ac23 = a; cc23 = c; vc23 = v; wc23 = w;
load soft4control24;
ac24 = a; cc24 = c; vc24 = v; wc24 = w;
load soft4control25;
ac25 = a; cc25 = c; vc25 = v; wc25 = w;
load soft4control26;
ac26 = a; cc26 = c; vc26 = v; wc26 = w;
load soft4control27;
ac27 = a; cc27 = c; vc27 = v; wc27 = w;
load soft4control31;
ac31 = a; cc31 = c; vc31 = v; wc31 = w;
load soft4control32;
ac32 = a; cc32 = c; vc32 = v; wc32 = w;
load soft4control33;
ac33 = a; cc33 = c; vc33 = v; wc33 = w;
load soft4control34;
ac34 = a; cc34 = c; vc34 = v; wc34 = w;
load soft4control35;
ac35 = a; cc35 = c; vc35 = v; wc35 = w;
load soft4control36;
ac36 = a; cc36 = c; vc36 = v; wc36 = w;
load soft4control37;
ac37 = a; cc37 = c; vc37 = v; wc37 = w;
load soft4control41;
ac41 = a; cc41 = c; vc41 = v; wc41 = w;
load soft4control42;
ac42 = a; cc42 = c; vc42 = v; wc42 = w;
load soft4control43;
ac43 = a; cc43 = c; vc43 = v; wc43 = w;
load soft4control44;
ac44 = a; cc44 = c; vc44 = v; wc44 = w;
load soft4control45;
ac45 = a; cc45 = c; vc45 = v; wc45 = w;
load soft4control46;
ac46 = a; cc46 = c; vc46 = v; wc46 = w;
load soft4control47;
ac47 = a; cc47 = c; vc47 = v; wc47 = w;
load soft4control51;
ac51 = a; cc51 = c; vc51 = v; wc51 = w;
load soft4control52;
```

```
ac52 = a; cc52 = c; vc52 = v; wc52 = w;
load soft4control53;
ac53 = a; cc53 = c; vc53 = v; wc53 = w;
load soft4control54;
ac54 = a; cc54 = c; vc54 = v; wc54 = w;
load soft4control55;
ac55 = a; cc55 = c; vc55 = v; wc55 = w;
load soft4control56;
ac56 = a; cc56 = c; vc56 = v; wc56 = w;
load soft4control57;
ac57 = a; cc57 = c; vc57 = v; wc57 = w;
load soft4control61;
ac61 = a; cc61 = c; vc61 = v; wc61 = w;
load soft4control62;
ac62 = a; cc62 = c; vc62 = v; wc62 = w;
load soft4control63;
ac63 = a; cc63 = c; vc63 = v; wc63 = w;
load soft4control64;
ac64 = a; cc64 = c; vc64 = v; wc64 = w;
load soft4control65;
ac65 = a; cc65 = c; vc65 = v; wc65 = w;
load soft4control66;
ac66 = a; cc66 = c; vc66 = v; wc66 = w;
load soft4control67;
ac67 = a; cc67 = c; vc67 = v; wc67 = w;
load soft4control71;
ac71 = a; cc71 = c; vc71 = v; wc71 = w;
load soft4control72;
ac72 = a; cc72 = c; vc72 = v; wc72 = w;
load soft4control73;
ac73 = a; cc73 = c; vc73 = v; wc73 = w;
load soft4control74;
ac74 = a; cc74 = c; vc74 = v; wc74 = w;
load soft4control75;
ac75 = a; cc75 = c; vc75 = v; wc75 = w;
load soft4control76;
ac76 = a; cc76 = c; vc76 = v; wc76 = w;
load soft4control77;
ac77 = a; cc77 = c; vc77 = v; wc77 = w;
% Initial position of soft robot

umax = 1;
xini = 0; yini = 0;
pos(1,1) = xini; pos(2,1) = yini;
xdes = diag(xast);
ydes = diag(yast);
nref = 30;
ndata = nref * length(xdes);
out_red = zeros(4,1);

for i = 0:(length(xdes)-1)
    posdes(i*nref+1:(i+1)*nref,:)=ones(nref,1)*[xdes(i+1) ydes(i+1)];
end
```

```
load soft4model01;

nem = ne; nmm = nm; nsm = ns;
vm = v; wm = w; cm = c; am = a;

k = 1;

%Simulation
for n = 1:ndata

    xx(k,1) = pos(1,1);
    yy(k,1) = pos(2,1);
    nn(k,1) = n;
    uu(k,:) = out_red';
    r = posdes(k,:)';
    in_red = pos - r;

    %11
    mc11 = vc11'*[1;in_red];
    nc11 = 2.0./(1 + exp(-(mc11-cc11)./ac11)) - 1;
    uc11 = wc11'*nc11;
    if uc11(1) < 0; uc11(1) = 0; end; if uc11(2) < 0; uc11(2) = 0; end
    if uc11(3) < 0; uc11(3) = 0; end; if uc11(4) < 0; uc11(4) = 0; end
    if uc11(1) > umax; uc11(1) = umax; end; if uc11(2) > umax; uc11(2) =
 umax; end
    if uc11(3) > umax; uc11(3) = umax; end; if uc11(4) > umax; uc11(4) =
 umax; end
    %12
    mc12 = vc12'*[1;in_red];
    nc12 = 2.0./(1 + exp(-(mc12-cc12)./ac12)) - 1;
    uc12 = wc12'*nc12;
    if uc12(1) < 0; uc12(1) = 0; end; if uc12(2) < 0; uc12(2) = 0; end
    if uc12(3) < 0; uc12(3) = 0; end; if uc12(4) < 0; uc12(4) = 0; end
    if uc12(1) > umax; uc12(1) = umax; end; if uc12(2) > umax; uc12(2) =
 umax; end
    if uc12(3) > umax; uc12(3) = umax; end; if uc12(4) > umax; uc12(4) =
 umax; end
    %13
    mc13 = vc13'*[1;in_red];
    nc13 = 2.0./(1 + exp(-(mc13-cc13)./ac13)) - 1;
    uc13 = wc13'*nc13;
    if uc13(1) < 0; uc13(1) = 0; end; if uc13(2) < 0; uc13(2) = 0; end
    if uc13(3) < 0; uc13(3) = 0; end; if uc13(4) < 0; uc13(4) = 0; end
    if uc13(1) > umax; uc13(1) = umax; end; if uc13(2) > umax; uc13(2) =
 umax; end
    if uc13(3) > umax; uc13(3) = umax; end; if uc13(4) > umax; uc13(4) =
 umax; end
    %14
    mc14 = vc14'*[1;in_red];
    nc14 = 2.0./(1 + exp(-(mc14-cc14)./ac14)) - 1;
    uc14 = wc14'*nc14;
    if uc14(1) < 0; uc14(1) = 0; end; if uc14(2) < 0; uc14(2) = 0; end
    if uc14(3) < 0; uc14(3) = 0; end; if uc14(4) < 0; uc14(4) = 0; end
```

```
   if uc14(1) > umax; uc14(1) = umax; end; if uc14(2) > umax; uc14(2) =
umax; end
   if uc14(3) > umax; uc14(3) = umax; end; if uc14(4) > umax; uc14(4) =
umax; end
   %15
   mc15 = vc15'*[1;in_red];
   nc15 = 2.0./(1 + exp(-(mc15-cc15)./ac15)) - 1;
   uc15 = wc15'*nc15;
   if uc15(1) < 0; uc15(1) = 0; end; if uc15(2) < 0; uc15(2) = 0; end
   if uc15(3) < 0; uc15(3) = 0; end; if uc15(4) < 0; uc15(4) = 0; end
   if uc15(1) > umax; uc15(1) = umax; end; if uc15(2) > umax; uc15(2) =
umax; end
   if uc15(3) > umax; uc15(3) = umax; end; if uc15(4) > umax; uc15(4) =
umax; end
   %16
   mc16 = vc16'*[1;in_red];
   nc16 = 2.0./(1 + exp(-(mc16-cc16)./ac16)) - 1;
   uc16 = wc16'*nc16;
   if uc16(1) < 0; uc16(1) = 0; end; if uc16(2) < 0; uc16(2) = 0; end
   if uc16(3) < 0; uc16(3) = 0; end; if uc16(4) < 0; uc16(4) = 0; end
   if uc16(1) > umax; uc16(1) = umax; end; if uc16(2) > umax; uc16(2) =
umax; end
   if uc16(3) > umax; uc16(3) = umax; end; if uc16(4) > umax; uc16(4) =
umax; end
   %17
   mc17 = vc17'*[1;in_red];
   nc17 = 2.0./(1 + exp(-(mc17-cc17)./ac17)) - 1;
   uc17 = wc17'*nc17;
   if uc17(1) < 0; uc17(1) = 0; end; if uc17(2) < 0; uc17(2) = 0; end
   if uc17(3) < 0; uc17(1) = 0; end; if uc17(4) < 0; uc17(2) = 0; end
   if uc17(1) > umax; uc17(1) = umax; end; if uc17(2) > umax; uc17(2) =
umax; end
   if uc17(3) > umax; uc17(3) = umax; end; if uc17(4) > umax; uc17(4) =
umax; end

   %21
   mc21 = vc21'*[1;in_red];
   nc21 = 2.0./(1 + exp(-(mc21-cc21)./ac21)) - 1;
   uc21 = wc21'*nc21;
   if uc21(1) < 0; uc21(1) = 0; end; if uc21(2) < 0; uc21(2) = 0; end
   if uc21(3) < 0; uc21(3) = 0; end; if uc21(4) < 0; uc21(4) = 0; end
   if uc21(1) > umax; uc21(1) = umax; end; if uc21(2) > umax; uc21(2) =
umax; end
   if uc21(3) > umax; uc21(3) = umax; end; if uc21(4) > umax; uc21(4) =
umax; end
   %22
   mc22 = vc22'*[1;in_red];
   nc22 = 2.0./(1 + exp(-(mc22-cc22)./ac22)) - 1;
   uc22 = wc22'*nc22;
   if uc22(1) < 0; uc22(1) = 0; end; if uc22(2) < 0; uc22(2) = 0; end
   if uc22(3) < 0; uc22(3) = 0; end; if uc22(4) < 0; uc22(4) = 0; end
   if uc22(1) > umax; uc22(1) = umax; end; if uc22(2) > umax; uc22(2) =
umax; end
```

```
   if uc22(3) > umax; uc22(3) = umax; end; if uc22(4) > umax; uc22(4) =
umax; end
   %23
   mc23 = vc23'*[1;in_red];
   nc23 = 2.0./(1 + exp(-(mc23-cc23)./ac23)) - 1;
   uc23 = wc23'*nc23;
   if uc23(1) < 0; uc23(1) = 0; end; if uc23(2) < 0; uc23(2) = 0; end
   if uc23(3) < 0; uc23(3) = 0; end; if uc23(4) < 0; uc23(4) = 0; end
   if uc23(1) > umax; uc23(1) = umax; end; if uc23(2) > umax; uc23(2) =
umax; end
   if uc23(3) > umax; uc23(3) = umax; end; if uc23(4) > umax; uc23(4) =
umax; end
   %24
   mc24 = vc24'*[1;in_red];
   nc24 = 2.0./(1 + exp(-(mc24-cc24)./ac24)) - 1;
   uc24 = wc24'*nc24;
   if uc24(1) < 0; uc24(1) = 0; end; if uc24(2) < 0; uc24(2) = 0; end
   if uc24(3) < 0; uc24(3) = 0; end; if uc24(4) < 0; uc24(4) = 0; end
   if uc24(1) > umax; uc24(1) = umax; end; if uc24(2) > umax; uc24(2) =
umax; end
   if uc24(3) > umax; uc24(3) = umax; end; if uc24(4) > umax; uc24(4) =
umax; end
   %25
   mc25 = vc25'*[1;in_red];
   nc25 = 2.0./(1 + exp(-(mc25-cc25)./ac25)) - 1;
   uc25 = wc25'*nc25;
   if uc25(1) < 0; uc25(1) = 0; end; if uc25(2) < 0; uc25(2) = 0; end
   if uc25(3) < 0; uc25(3) = 0; end; if uc25(4) < 0; uc25(4) = 0; end
   if uc25(1) > umax; uc25(1) = umax; end; if uc25(2) > umax; uc25(2) =
umax; end
   if uc25(3) > umax; uc25(3) = umax; end; if uc25(4) > umax; uc25(4) =
umax; end
   %26
   mc26 = vc26'*[1;in_red];
   nc26 = 2.0./(1 + exp(-(mc26-cc26)./ac26)) - 1;
   uc26 = wc26'*nc26;
   if uc26(1) < 0; uc26(1) = 0; end; if uc26(2) < 0; uc26(2) = 0; end
   if uc26(3) < 0; uc26(3) = 0; end; if uc26(4) < 0; uc26(4) = 0; end
   if uc26(1) > umax; uc26(1) = umax; end; if uc26(2) > umax; uc26(2) =
umax; end
   if uc26(3) > umax; uc26(3) = umax; end; if uc26(4) > umax; uc26(4) =
umax; end
   %27
   mc27 = vc27'*[1;in_red];
   nc27 = 2.0./(1 + exp(-(mc27-cc27)./ac27)) - 1;
   uc27 = wc27'*nc27;
   if uc27(1) < 0; uc27(1) = 0; end; if uc27(2) < 0; uc27(2) = 0; end
   if uc27(3) < 0; uc27(3) = 0; end; if uc27(4) < 0; uc27(4) = 0; end
   if uc27(1) > umax; uc27(1) = umax; end; if uc27(2) > umax; uc27(2) =
umax; end
   if uc27(3) > umax; uc27(3) = umax; end; if uc27(4) > umax; uc27(4) =
umax; end

   %31
```

```
    mc31 = vc31'*[1;in_red];
    nc31 = 2.0./(1 + exp(-(mc31-cc31)./ac31)) - 1;
    uc31 = wc31'*nc31;
    if uc31(1) < 0; uc31(1) = 0; end; if uc31(2) < 0; uc31(2) = 0; end
    if uc31(3) < 0; uc31(3) = 0; end; if uc31(4) < 0; uc31(4) = 0; end
    if uc31(1) > umax; uc31(1) = umax; end; if uc31(2) > umax; uc31(2) =
umax; end
    if uc31(3) > umax; uc31(3) = umax; end; if uc31(4) > umax; uc31(4) =
umax; end
    %32
    mc32 = vc32'*[1;in_red];
    nc32 = 2.0./(1 + exp(-(mc32-cc32)./ac32)) - 1;
    uc32 = wc32'*nc32;
    if uc32(1) < 0; uc32(1) = 0; end; if uc32(2) < 0; uc32(2) = 0; end
    if uc32(3) < 0; uc32(3) = 0; end; if uc32(4) < 0; uc32(4) = 0; end
    if uc32(1) > umax; uc32(1) = umax; end; if uc32(2) > umax; uc32(2) =
umax; end
    if uc32(3) > umax; uc32(3) = umax; end; if uc32(4) > umax; uc32(4) =
umax; end
    %33
    mc33 = vc33'*[1;in_red];
    nc33 = 2.0./(1 + exp(-(mc33-cc33)./ac33)) - 1;
    uc33 = wc33'*nc33;
    if uc33(1) < 0; uc33(1) = 0; end; if uc33(2) < 0; uc33(2) = 0; end
    if uc33(3) < 0; uc33(3) = 0; end; if uc33(4) < 0; uc33(4) = 0; end
    if uc33(1) > umax; uc33(1) = umax; end; if uc33(2) > umax; uc33(2) =
umax; end
    if uc33(3) > umax; uc33(3) = umax; end; if uc33(4) > umax; uc33(4) =
umax; end
    %34
    mc34 = vc34'*[1;in_red];
    nc34 = 2.0./(1 + exp(-(mc34-cc34)./ac34)) - 1;
    uc34 = wc34'*nc34;
    if uc34(1) < 0; uc34(1) = 0; end; if uc34(2) < 0; uc34(2) = 0; end
    if uc34(3) < 0; uc34(3) = 0; end; if uc34(4) < 0; uc34(4) = 0; end
    if uc34(1) > umax; uc34(1) = umax; end; if uc34(2) > umax; uc34(2) =
umax; end
    if uc34(3) > umax; uc34(3) = umax; end; if uc34(4) > umax; uc34(4) =
umax; end
    %35
    mc35 = vc35'*[1;in_red];
    nc35 = 2.0./(1 + exp(-(mc35-cc35)./ac35)) - 1;
    uc35 = wc35'*nc35;
    if uc35(1) < 0; uc35(1) = 0; end; if uc35(2) < 0; uc35(2) = 0; end
    if uc35(3) < 0; uc35(3) = 0; end; if uc35(4) < 0; uc35(4) = 0; end
    if uc35(1) > umax; uc35(1) = umax; end; if uc35(2) > umax; uc35(2) =
umax; end
    if uc35(3) > umax; uc35(3) = umax; end; if uc35(4) > umax; uc35(4) =
umax; end
    %36
    mc36 = vc36'*[1;in_red];
    nc36 = 2.0./(1 + exp(-(mc36-cc36)./ac36)) - 1;
    uc36 = wc36'*nc36;
    if uc36(1) < 0; uc36(1) = 0; end; if uc36(2) < 0; uc36(2) = 0; end
```

```
    if uc36(3) < 0; uc36(3) = 0; end; if uc36(4) < 0; uc36(4) = 0; end
    if uc36(1) > umax; uc36(1) = umax; end; if uc36(2) > umax; uc36(2) =
umax; end
    if uc36(3) > umax; uc36(3) = umax; end; if uc36(4) > umax; uc36(4) =
umax; end
    %37
    mc37 = vc37'*[1;in_red];
    nc37 = 2.0./(1 + exp(-(mc37-cc37)./ac37)) - 1;
    uc37 = wc37'*nc37;
    if uc37(1) < 0; uc37(1) = 0; end; if uc37(2) < 0; uc37(2) = 0; end
    if uc37(3) < 0; uc37(3) = 0; end; if uc37(4) < 0; uc37(4) = 0; end
    if uc37(1) > umax; uc37(1) = umax; end; if uc37(2) > umax; uc37(2) =
umax; end
    if uc37(3) > umax; uc37(3) = umax; end; if uc37(4) > umax; uc37(4) =
umax; end

    %41
    mc41 = vc41'*[1;in_red];
    nc41 = 2.0./(1 + exp(-(mc41-cc41)./ac41)) - 1;
    uc41 = wc41'*nc41;
    if uc41(1) < 0; uc41(1) = 0; end; if uc41(2) < 0; uc41(2) = 0; end
    if uc41(3) < 0; uc41(3) = 0; end; if uc41(4) < 0; uc41(4) = 0; end
    if uc41(1) > umax; uc41(1) = umax; end; if uc41(2) > umax; uc41(2) =
umax; end
    if uc41(3) > umax; uc41(3) = umax; end; if uc41(4) > umax; uc41(4) =
umax; end
    %42
    mc42 = vc42'*[1;in_red];
    nc42 = 2.0./(1 + exp(-(mc42-cc42)./ac42)) - 1;
    uc42 = wc42'*nc42;
    if uc42(1) < 0; uc42(1) = 0; end; if uc42(2) < 0; uc42(2) = 0; end
    if uc42(3) < 0; uc42(3) = 0; end; if uc42(4) < 0; uc42(4) = 0; end
    if uc42(1) > umax; uc42(1) = umax; end; if uc42(2) > umax; uc42(2) =
umax; end
    if uc42(3) > umax; uc42(3) = umax; end; if uc42(4) > umax; uc42(4) =
umax; end
    %43
    mc43 = vc43'*[1;in_red];
    nc43 = 2.0./(1 + exp(-(mc43-cc43)./ac43)) - 1;
    uc43 = wc43'*nc43;
    if uc43(1) < 0; uc43(1) = 0; end; if uc43(2) < 0; uc43(2) = 0; end
    if uc43(3) < 0; uc43(3) = 0; end; if uc43(4) < 0; uc43(4) = 0; end
    if uc43(1) > umax; uc43(1) = umax; end; if uc43(2) > umax; uc43(2) =
umax; end
    if uc43(3) > umax; uc43(3) = umax; end; if uc43(4) > umax; uc43(4) =
umax; end
    %44
    mc44 = vc44'*[1;in_red];
    nc44 = 2.0./(1 + exp(-(mc44-cc44)./ac44)) - 1;
    uc44 = wc44'*nc44;
    if uc44(1) < 0; uc44(1) = 0; end; if uc44(2) < 0; uc44(2) = 0; end
    if uc44(3) < 0; uc44(3) = 0; end; if uc44(4) < 0; uc44(4) = 0; end
    if uc44(1) > umax; uc44(1) = umax; end; if uc44(2) > umax; uc44(2) =
umax; end
```

```
   if uc44(3) > umax; uc44(3) = umax; end; if uc44(4) > umax; uc44(4) =
umax; end
   %45
   mc45 = vc45'*[1;in_red];
   nc45 = 2.0./(1 + exp(-(mc45-cc45)./ac45)) - 1;
   uc45 = wc45'*nc45;
   if uc45(1) < 0; uc45(1) = 0; end; if uc45(2) < 0; uc45(2) = 0; end
   if uc45(3) < 0; uc45(3) = 0; end; if uc45(4) < 0; uc45(4) = 0; end
   if uc45(1) > umax; uc45(1) = umax; end; if uc45(2) > umax; uc45(2) =
umax; end
   if uc45(3) > umax; uc45(3) = umax; end; if uc45(4) > umax; uc45(4) =
umax; end
   %46
   mc46 = vc46'*[1;in_red];
   nc46 = 2.0./(1 + exp(-(mc46-cc46)./ac46)) - 1;
   uc46 = wc46'*nc46;
   if uc46(1) < 0; uc46(1) = 0; end; if uc46(2) < 0; uc46(2) = 0; end
   if uc46(3) < 0; uc46(3) = 0; end; if uc46(4) < 0; uc46(4) = 0; end
   if uc46(1) > umax; uc46(1) = umax; end; if uc46(2) > umax; uc46(2) =
umax; end
   if uc46(3) > umax; uc46(3) = umax; end; if uc46(4) > umax; uc46(4) =
umax; end
   %47
   mc47 = vc47'*[1;in_red];
   nc47 = 2.0./(1 + exp(-(mc47-cc47)./ac47)) - 1;
   uc47 = wc47'*nc47;
   if uc47(1) < 0; uc47(1) = 0; end; if uc47(2) < 0; uc47(2) = 0; end
   if uc47(3) < 0; uc47(3) = 0; end; if uc47(4) < 0; uc47(4) = 0; end
   if uc47(1) > umax; uc47(1) = umax; end; if uc47(2) > umax; uc47(2) =
umax; end
   if uc47(3) > umax; uc47(3) = umax; end; if uc47(4) > umax; uc47(4) =
umax; end

   %51
   mc51 = vc51'*[1;in_red];
   nc51 = 2.0./(1 + exp(-(mc51-cc51)./ac51)) - 1;
   uc51 = wc51'*nc51;
   if uc51(1) < 0; uc51(1) = 0; end; if uc51(2) < 0; uc51(2) = 0; end
   if uc51(3) < 0; uc51(3) = 0; end; if uc51(3) < 0; uc51(3) = 0; end
   if uc51(1) > umax; uc51(1) = umax; end; if uc51(2) > umax; uc51(2) =
umax; end
   if uc51(3) > umax; uc51(3) = umax; end; if uc51(4) > umax; uc51(4) =
umax; end
   %52
   mc52 = vc52'*[1;in_red];
   nc52 = 2.0./(1 + exp(-(mc52-cc52)./ac52)) - 1;
   uc52 = wc52'*nc52;
   if uc52(1) < 0; uc52(1) = 0; end; if uc52(2) < 0; uc52(2) = 0; end
   if uc52(3) < 0; uc52(3) = 0; end; if uc52(4) < 0; uc52(4) = 0; end
   if uc52(1) > umax; uc52(1) = umax; end; if uc52(2) > umax; uc52(2) =
umax; end
   if uc52(3) > umax; uc52(3) = umax; end; if uc52(4) > umax; uc52(4) =
umax; end
   %53
```

```
    mc53 = vc53'*[1;in_red];
    nc53 = 2.0./(1 + exp(-(mc53-cc53)./ac53)) - 1;
    uc53 = wc53'*nc53;
    if uc53(1) < 0; uc53(1) = 0; end; if uc53(2) < 0; uc53(2) = 0; end
    if uc53(3) < 0; uc53(3) = 0; end; if uc53(4) < 0; uc53(4) = 0; end
    if uc53(1) > umax; uc53(1) = umax; end; if uc53(2) > umax; uc53(2) =
umax; end
    if uc53(3) > umax; uc53(3) = umax; end; if uc53(4) > umax; uc53(4) =
umax; end
    %54
    mc54 = vc54'*[1;in_red];
    nc54 = 2.0./(1 + exp(-(mc54-cc54)./ac54)) - 1;
    uc54 = wc54'*nc54;
    if uc54(1) < 0; uc54(1) = 0; end; if uc54(2) < 0; uc54(2) = 0; end
    if uc54(3) < 0; uc54(3) = 0; end; if uc54(4) < 0; uc54(4) = 0; end
    if uc54(1) > umax; uc54(1) = umax; end; if uc54(2) > umax; uc54(2) =
umax; end
    if uc54(3) > umax; uc54(3) = umax; end; if uc54(4) > umax; uc54(4) =
umax; end
    %55
    mc55 = vc55'*[1;in_red];
    nc55 = 2.0./(1 + exp(-(mc55-cc55)./ac55)) - 1;
    uc55 = wc55'*nc55;
    if uc55(1) < 0; uc55(1) = 0; end; if uc55(2) < 0; uc55(2) = 0; end
    if uc55(3) < 0; uc55(3) = 0; end; if uc55(4) < 0; uc55(4) = 0; end
    if uc55(1) > umax; uc55(1) = umax; end; if uc55(2) > umax; uc55(2) =
umax; end
    if uc55(3) > umax; uc55(3) = umax; end; if uc55(4) > umax; uc55(4) =
umax; end
    %56
    mc56 = vc56'*[1;in_red];
    nc56 = 2.0./(1 + exp(-(mc56-cc56)./ac56)) - 1;
    uc56 = wc56'*nc56;
    if uc56(1) < 0; uc56(1) = 0; end; if uc56(2) < 0; uc56(2) = 0; end
    if uc56(3) < 0; uc56(3) = 0; end; if uc56(4) < 0; uc56(4) = 0; end
    if uc56(1) > umax; uc56(1) = umax; end; if uc56(2) > umax; uc56(2) =
umax; end
    if uc56(3) > umax; uc56(3) = umax; end; if uc56(4) > umax; uc56(4) =
umax; end
    %57
    mc57 = vc57'*[1;in_red];
    nc57 = 2.0./(1 + exp(-(mc57-cc57)./ac57)) - 1;
    uc57 = wc57'*nc57;
    if uc57(1) < 0; uc57(1) = 0; end; if uc57(2) < 0; uc57(2) = 0; end
    if uc57(3) < 0; uc57(3) = 0; end; if uc57(4) < 0; uc57(4) = 0; end
    if uc57(1) > umax; uc57(1) = umax; end; if uc57(2) > umax; uc57(2) =
umax; end
    if uc57(3) > umax; uc57(3) = umax; end; if uc57(4) > umax; uc57(4) =
umax; end

    %61
    mc61 = vc61'*[1;in_red];
    nc61 = 2.0./(1 + exp(-(mc61-cc61)./ac61)) - 1;
    uc61 = wc61'*nc61;
```

```
    if uc61(1) < 0; uc61(1) = 0; end; if uc61(2) < 0; uc61(2) = 0; end
    if uc61(3) < 0; uc61(3) = 0; end; if uc61(4) < 0; uc61(4) = 0; end
    if uc61(1) > umax; uc61(1) = umax; end; if uc61(2) > umax; uc61(2) =
umax; end
    if uc61(3) > umax; uc61(3) = umax; end; if uc61(4) > umax; uc61(4) =
umax; end
    %62
    mc62 = vc62'*[1;in_red];
    nc62 = 2.0./(1 + exp(-(mc62-cc62)./ac62)) - 1;
    uc62 = wc62'*nc62;
    if uc62(1) < 0; uc62(1) = 0; end; if uc62(2) < 0; uc62(2) = 0; end
    if uc62(3) < 0; uc62(3) = 0; end; if uc62(4) < 0; uc62(4) = 0; end
    if uc62(1) > umax; uc62(1) = umax; end; if uc62(2) > umax; uc62(2) =
umax; end
    if uc62(3) > umax; uc62(3) = umax; end; if uc62(4) > umax; uc62(4) =
umax; end
    %63
    mc63 = vc63'*[1;in_red];
    nc63 = 2.0./(1 + exp(-(mc63-cc63)./ac63)) - 1;
    uc63 = wc63'*nc63;
    if uc63(1) < 0; uc63(1) = 0; end; if uc63(2) < 0; uc63(2) = 0; end
    if uc63(3) < 0; uc63(3) = 0; end; if uc63(4) < 0; uc63(4) = 0; end
    if uc63(1) > umax; uc63(1) = umax; end; if uc63(2) > umax; uc63(2) =
umax; end
    if uc63(3) > umax; uc63(3) = umax; end; if uc63(4) > umax; uc63(4) =
umax; end
    %64
    mc64 = vc64'*[1;in_red];
    nc64 = 2.0./(1 + exp(-(mc64-cc64)./ac64)) - 1;
    uc64 = wc64'*nc64;
    if uc64(1) < 0; uc64(1) = 0; end; if uc64(2) < 0; uc64(2) = 0; end
    if uc64(3) < 0; uc64(3) = 0; end; if uc64(4) < 0; uc64(4) = 0; end
    if uc64(1) > umax; uc64(1) = umax; end; if uc64(2) > umax; uc64(2) =
umax; end
    if uc64(3) > umax; uc64(3) = umax; end; if uc64(4) > umax; uc64(4) =
umax; end
    %65
    mc65 = vc65'*[1;in_red];
    nc65 = 2.0./(1 + exp(-(mc65-cc65)./ac65)) - 1;
    uc65 = wc65'*nc65;
    if uc65(1) < 0; uc65(1) = 0; end; if uc65(2) < 0; uc65(2) = 0; end
    if uc65(3) < 0; uc65(3) = 0; end; if uc65(4) < 0; uc65(4) = 0; end
    if uc65(1) > umax; uc65(1) = umax; end; if uc65(2) > umax; uc65(2) =
umax; end
    if uc65(3) > umax; uc65(3) = umax; end; if uc65(4) > umax; uc65(4) =
umax; end
    %66
    mc66 = vc66'*[1;in_red];
    nc66 = 2.0./(1 + exp(-(mc66-cc66)./ac66)) - 1;
    uc66 = wc66'*nc66;
    if uc66(1) < 0; uc66(1) = 0; end; if uc66(2) < 0; uc66(2) = 0; end
    if uc66(3) < 0; uc66(3) = 0; end; if uc66(4) < 0; uc66(4) = 0; end
    if uc66(1) > umax; uc66(1) = umax; end; if uc66(2) > umax; uc66(2) =
umax; end
```

```
   if uc66(3) > umax; uc66(3) = umax; end; if uc66(4) > umax; uc66(4) =
umax; end
   %67
   mc67 = vc67'*[1;in_red];
   nc67 = 2.0./(1 + exp(-(mc67-cc67)./ac67)) - 1;
   uc67 = wc67'*nc67;
   if uc67(1) < 0; uc67(1) = 0; end; if uc67(2) < 0; uc67(2) = 0; end
   if uc67(3) < 0; uc67(3) = 0; end; if uc67(4) < 0; uc67(4) = 0; end
   if uc67(1) > umax; uc67(1) = umax; end; if uc67(2) > umax; uc67(2) =
umax; end
   if uc67(3) > umax; uc67(3) = umax; end; if uc67(4) > umax; uc67(4) =
umax; end

   %71
   mc71 = vc71'*[1;in_red];
   nc71 = 2.0./(1 + exp(-(mc71-cc71)./ac71)) - 1;
   uc71 = wc71'*nc71;
   if uc71(1) < 0; uc71(1) = 0; end; if uc71(2) < 0; uc71(2) = 0; end
   if uc71(3) < 0; uc71(3) = 0; end; if uc71(4) < 0; uc71(4) = 0; end
   if uc71(1) > umax; uc71(1) = umax; end; if uc71(2) > umax; uc71(2) =
umax; end
   if uc71(3) > umax; uc71(3) = umax; end; if uc71(4) > umax; uc71(4) =
umax; end
   %72
   mc72 = vc72'*[1;in_red];
   nc72 = 2.0./(1 + exp(-(mc72-cc72)./ac72)) - 1;
   uc72 = wc72'*nc72;
   if uc72(1) < 0; uc72(1) = 0; end; if uc72(2) < 0; uc72(2) = 0; end
   if uc72(3) < 0; uc72(3) = 0; end; if uc72(4) < 0; uc72(4) = 0; end
   if uc72(1) > umax; uc72(1) = umax; end; if uc72(2) > umax; uc72(2) =
umax; end
   if uc72(3) > umax; uc72(3) = umax; end; if uc72(4) > umax; uc72(4) =
umax; end
   %73
   mc73 = vc73'*[1;in_red];
   nc73 = 2.0./(1 + exp(-(mc73-cc73)./ac73)) - 1;
   uc73 = wc73'*nc73;
   if uc73(1) < 0; uc73(1) = 0; end; if uc73(2) < 0; uc73(2) = 0; end
   if uc73(3) < 0; uc73(3) = 0; end; if uc73(4) < 0; uc73(4) = 0; end
   if uc73(1) > umax; uc73(1) = umax; end; if uc73(2) > umax; uc73(2) =
umax; end
   if uc73(3) > umax; uc73(3) = umax; end; if uc73(4) > umax; uc73(4) =
umax; end
   %74
   mc74 = vc74'*[1;in_red];
   nc74 = 2.0./(1 + exp(-(mc74-cc74)./ac74)) - 1;
   uc74 = wc74'*nc74;
   if uc74(1) < 0; uc74(1) = 0; end; if uc74(2) < 0; uc74(2) = 0; end
   if uc74(3) < 0; uc74(3) = 0; end; if uc74(4) < 0; uc74(4) = 0; end
   if uc74(1) > umax; uc74(1) = umax; end; if uc74(2) > umax; uc74(2) =
umax; end
   if uc74(3) > umax; uc74(3) = umax; end; if uc74(4) > umax; uc74(4) =
umax; end
   %75
```

```
    mc75 = vc75'*[1;in_red];
    nc75 = 2.0./(1 + exp(-(mc75-cc75)./ac75)) - 1;
    uc75 = wc75'*nc75;
    if uc75(1) < 0; uc75(1) = 0; end; if uc75(2) < 0; uc75(2) = 0; end
    if uc75(3) < 0; uc75(3) = 0; end; if uc75(4) < 0; uc75(4) = 0; end
    if uc75(1) > umax; uc75(1) = umax; end; if uc75(2) > umax; uc75(2) =
umax; end
    if uc75(3) > umax; uc75(3) = umax; end; if uc75(4) > umax; uc75(4) =
umax; end
    %76
    mc76 = vc76'*[1;in_red];
    nc76 = 2.0./(1 + exp(-(mc76-cc76)./ac76)) - 1;
    uc76 = wc76'*nc76;
    if uc76(1) < 0; uc76(1) = 0; end; if uc76(2) < 0; uc76(2) = 0; end
    if uc76(3) < 0; uc76(3) = 0; end; if uc76(4) < 0; uc76(4) = 0; end
    if uc76(1) > umax; uc76(1) = umax; end; if uc76(2) > umax; uc76(2) =
umax; end
    if uc76(3) > umax; uc76(3) = umax; end; if uc76(4) > umax; uc76(4) =
umax; end
    %77
    mc77 = vc77'*[1;in_red];
    nc77 = 2.0./(1 + exp(-(mc77-cc77)./ac77)) - 1;
    uc77 = wc77'*nc77;
    if uc77(1) < 0; uc77(1) = 0; end; if uc77(2) < 0; uc77(2) = 0; end
    if uc77(3) < 0; uc77(3) = 0; end; if uc77(4) < 0; uc77(4) = 0; end
    if uc77(1) > umax; uc77(1) = umax; end; if uc77(2) > umax; uc77(2) =
umax; end
    if uc77(3) > umax; uc77(3) = umax; end; if uc77(4) > umax; uc77(4) =
umax; end

    %Weighting of controllers using membership functions
    kx = round(r(1,1)*nx/2 + nx/2);
    fdx(1,1) = fdx1(kx,1);
    fdx(2,1) = fdx2(kx,1);
    fdx(3,1) = fdx3(kx,1);
    fdx(4,1) = fdx4(kx,1);
    fdx(5,1) = fdx5(kx,1);
    fdx(6,1) = fdx6(kx,1);
    fdx(7,1) = fdx7(kx,1);
    ky = round(r(2,1)*ny/2 + ny/2);
    fdy(1,1) = fdy1(ky,1);
    fdy(2,1) = fdy2(ky,1);
    fdy(3,1) = fdy3(ky,1);
    fdy(4,1) = fdy4(ky,1);
    fdy(5,1) = fdy5(ky,1);
    fdy(6,1) = fdy6(ky,1);
    fdy(7,1) = fdy7(ky,1);

    %fdxx(k,:) = fdx';
    %fdyy(k,:) = fdy';
    zzz = 1;
    for m = 1:7
        for q = 1:7
            fxy(zzz,1) = min(fdy(m,1),fdx(q,1));
```

```
            zzz = zzz + 1;
        end
    end
    fxyxy(k,:) = fxy';
    ufuz = [uc11 uc12 uc13 uc14 uc15 uc16 uc17...
            uc21 uc22 uc23 uc24 uc25 uc26 uc27...
            uc31 uc32 uc33 uc34 uc35 uc36 uc37...
            uc41 uc42 uc43 uc44 uc45 uc46 uc47...
            uc51 uc52 uc53 uc54 uc55 uc56 uc57...
            uc61 uc62 uc63 uc64 uc65 uc66 uc67...
            uc71 uc72 uc73 uc74 uc75 uc76 uc77];
    out_red = (ufuz*fxy)/sum(fxy);

    %Soft Robot Model
    in_red_m = [pos;out_red];
    m_m = vm'*in_red_m;
    n_m = 2.0./(1 + exp(-(m_m-cm)./am)) - 1;
    pos = wm'*n_m;

    k = k + 1;
end
figure(2);
plot(nn,xx,nn,posdes(:,1));
legend('x','xref','FontSize',16);
xlabel('Número de Muestra (k)','FontSize',16);
ylabel('Coordenada x normalizada (x)','FontSize',16);
grid on;
title('x vs. xref','FontSize',16);
figure(3);
plot(nn,yy,nn,posdes(:,2));
legend('y','yref','FontSize',16);
grid on;
title('y vs yref','FontSize',16);
xlabel('Número de Muestra (k)','FontSize',16);
ylabel('Coordena y normalizada (y)','FontSize',16)
figure(4);
plot(uu);
legend('u1 = px+','u2 = py+','u3 = px-','u4 = py-','FontSize',16);
grid on;
title('u vs. k','FontSize',16);
xlabel('Número de Muestra (k)','FontSize',16);
ylabel('Esfuerzo de control normalizada (u)','FontSize',16)
```

ANEXO 06: Análisis de condiciones de estabilidad del sistema en lazo cerrado

```
clear all;
clc;
close all;


% A1 = [1 -0.5
%       1  0];
% A2 = [-1 -0.5
%        1   0];

% A1 = [1.503 -0.588
%       1       0];
% A2 = [1 -0.361
%       1     0];

% A1 = [0.2 -0.5
%        1  0];

% A2 = [-1 -0.5
%       1   0];
% Q = eye(2);
%
% q = Q;
% a1= A1;
% a2 = A2;
%
% bm = [q(1,1);q(1,2);q(2,2)];
%
% a1m = [a1(1,1)^2-1 2*a1(1,1)*a1(2,1) a1(2,1)^2
%       a1(1,1)*a1(1,2) a1(1,1)*a1(2,2)+a1(1,2)*a1(2,1)-1 a1(2,1)*a1(2,2)
%       a1(1,2)^2 2*a1(1,2)*a1(2,2) a1(2,2)^2-1];
%
% a2m = [a2(1,1)^2-1 2*a2(1,1)*a2(2,1) a2(2,1)^2
%       a2(1,1)*a2(1,2) a2(1,1)*a2(2,2)+a2(1,2)*a2(2,1)-1 a2(2,1)*a2(2,2)
%       a2(1,2)^2 2*a2(1,2)*a2(2,2) a2(2,2)^2-1];
%
% p1 = -inv(a1m)*bm;
% p2 = -inv(a2m)*bm;
%
% p1m = [p1(1,1) p1(2,1)
%        p1(2,1) p1(3,1)];
%
% p2m = [p2(1,1) p2(2,1)
%        p2(2,1) p2(3,1)];
%
% P1 = dlyap(A1,Q);
% P2 = dlyap(A2,Q);
%
% return;

load soft4control26;
wc1 = v(2:3,:);
```

```
wc2 = w;

load soft4model01;
wp1 = v;
wp2 = w;

%Matrix A
k = 1;
for a = 1:2
    for b = 1:2
        for c = 1:2
            for i = 1:2
                for j = 1:2
                    if a == 1
                        gp11a = 0;
                    else
                        gp11a = 0.5;
                    end
                    if b == 1
                        gp12b = 0;
                    else
                        gp12b = 0.5;
                    end
                    if c == 1
                        gp13c = 0;
                    else
                        gp13c = 0.5;
                    end
                    gp = [gp11a;gp12b;gp13c];
                    %gp = [gp11a;gp12b];
                    A(i,j,k) = wp1(j,:)*(wp2(:,i).*gp);
                end
            end
            k = k + 1;
        end
    end
end
%Matrix B
k = 1;
for a = 1:2
    for b = 1:2
        for c = 1:2
            for i = 1:2
                for j = 1:4
                    if a == 1
                        gp11a = 0;
                    else
                        gp11a = 0.5;
                    end
                    if b == 1
                        gp12b = 0;
                    else
                        gp12b = 0.5;
                    end
```

```
                    if c == 1
                        gp13c = 0;
                    else
                        gp13c = 0.5;
                    end
                    gp = [gp11a;gp12b;gp13c];
                    B(i,j,k) = wp1(j+2,:)*(wp2(:,i).*gp);
                end
            end
            k = k + 1;
        end
    end
end
%Matrix f
k = 1;
for a = 1:2
    for b = 1:2
        for c = 1:2
            for d = 1:2
                for i = 1:4
                    for j = 1:2
                        if a == 1
                            gc11a = 0;
                        else
                            gc11a = 0.5;
                        end
                        if b == 1
                            gc12b = 0;
                        else
                            gc12b = 0.5;
                        end
                        if c == 1
                            gc13c = 0;
                        else
                            gc13c = 0.5;
                        end
                        if d == 1
                            gc14d = 0;
                        else
                            gc14d = 0.5;
                        end
                        gc = [gc11a;gc12b;gc13c;gc14d];
                        f(i,j,k) = wc1(j,:)*(wc2(:,i).*gc);
                    end
                end
                k = k + 1;
            end

        end
    end
end
%Matrix H
k = 1;
for i = 1:8
```

```
    for j = 1:16
        H(:,:,k) = A(:,:,i) + B(:,:,i)*f(:,:,j);
        k = k+1;
    end
end

%Paso 1

Hn = H(:,:,16:128);

for i=1:113
 autoHn=eig(Hn(:,:,i));
 for k = 1:2
    if autoHn(k) > 0
        disp('1. Se cumple la condición de suficiencia');
    else
        disp('1. No se cumple la condición de suficiencia');
    end
 end
 disp(i);
end
return;

for i =1:113
    for j =1: 113
        Hm = Hn(:,:,i)*Hn(:,:,j);
        autoHm = eig(Hm);
        for k = 1:2
            if autoHm(k) > 0
                disp('2. Se cumple la condición de suficiencia')
            else
                disp('2. No se cumple la condición de suficiencia');
            end
        end
    end
end
```