Diseño de una Herramienta Computacional Basada en Inteligencia Artificial para Sistemas de Control de Empuje Vectorial

Sergio Andrés Gómez Valencia
Jaider Johany Alzate López

Trabajo de grado para optar al título de:
Ingeniero Aeroespacial

Asesor
Felipe Andrés Obando Vega, MSc, PhD

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería Aeroespacial
Carmen de Viboral, Colombia
2023

Dedicatoria

A mis padres, Sergio y Diana, y a mis queridos hermanos Santiago y Melisa. A ustedes dedico este logro con todo mi amor y gratitud. Su apoyo inquebrantable y sus sacrificios han sido indispensables en mi vida durante este viaje académico. A mis amigos más cercanos, gracias por ser fuente constante de motivación y por siempre esperar lo mejor de mí.

- S.A.G.V.

Con gratitud y aprecio, dedico este trabajo a quienes han sido parte fundamental de mi viaje. Agradezco sinceramente su constante aliento, comprensión y colaboración, elementos que han sido fundamentales para hacer posible esta locura.

- J.J.A.L.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Siglas, acrónimos y abreviaturas

CG	Centro de Gravedad
CP	Centro de Presión
T	Fuerza de Empuje (Thrust)
D	Fuerza de Arrastre (Drag)
L	Fuerza de Elevación (Lift)
C_D	Coeficiente de Arrastre
C_L	Coeficiente de Sustentación
TVC	Control de Empuje Vectorial
PID	Controlador Proporcional Integral Derivativo
GA	Algoritmos Genéticos
PSO	Optimización por Enjambre de Partículas
NNBP	Redes Neuronales Usando Retro-Propagación
NNNE	Redes Neuronales Usando Neuro-Evolución
GUI	Interfaz Gráfica de Usuario
M	Sobreimpulso
T_p	Tiempo Pico
T_s	Tiempo de Estabilización
S.M.	Margen de Estabilidad Estática
ITAE	Integral of time multiplied Absolute Error Criterion
cm	Centímetros
m	Metros
kg	Kilogramos
s	Segundos
psi	Libras por Pulgada Cuadrada
UdeA	Universidad de Antioquia

RESUMEN

En esta investigación, se profundiza en la exploración y comparación de las oportunidades que la inteligencia artificial ofrece en la sintonización de sistemas de control. Se propone la implementación de algoritmos de inteligencia artificial en una interfaz gráfica de usuario para obtener parámetros de sintonización de un sistema de control de empuje vectorial. El objetivo es lograr una sintonización rápida y eficiente utilizando datos inherentes al diseño y funcionamiento del sistema. Posteriormente, se llevan a cabo simulaciones para validar y contrastar la eficacia de estos parámetros mediante índices de desempeño, concluyendo con pruebas experimentales en un banco de pruebas. Se anticipa que los resultados contribuirán a reducir costos, simplificar procesos y mejorar la estabilidad de los sistemas de control, marcando un avance significativo en esta área de investigación. Con este enfoque integral, se busca ofrecer una contribución valiosa al desarrollo y aplicaciones prácticas de la inteligencia artificial en el campo de la ingeniería aeroespacial.

Palabras clave **— Control, estabilidad, empuje vectorial, aprendizaje automático, sintonización.**

ABSTRACT

In this study, we delve into the exploration and comparison of the opportunities that artificial intelligence offers in the tuning of control systems. The implementation of artificial intelligence algorithms within a user interface is proposed to obtain tuning values for a vector thrust control system. The main objective is to achieve a fast and efficient tuning of these systems using data inherent to their design and operation. Subsequently, simulations are conducted to validate and contrast the effectiveness of these values using performance parameters, culminating in experimental tests on a test bench. It is anticipated that the results will contribute to reducing costs, simplifying processes, and improving the stability of control systems, marking a significant advancement in this research field. With this comprehensive approach, we aim to provide a valuable contribution to the development and practical applications of artificial intelligence in aerospace engineering fields.

Keywords **— Control, stability, thrust vectoring, machine learning, tuning**

I. INTRODUCCIÓN

Los sistemas de control son una parte fundamental de la ingeniería y la tecnología, ya que permiten automatizar procesos, regular el comportamiento de dispositivos y sistemas, y mejorar la eficiencia en diversos campos, como la manufactura, la electrónica, la robótica y la aviación, entre otros. En un sistema de control un conjunto de sensores mide ciertas variables de entrada, un controlador procesa estas y genera otras señales con las que se rige un actuador que controla las variables de salida del sistema.

La sintonización de controladores es un tema fundamental en el diseño de sistemas de control, ya que bien ajustados puede mejorar significativamente el rendimiento y la estabilidad del sistema. Sin embargo, la sintonización de controladores puede ser un proceso difícil y costoso, ya que requiere una comprensión detallada del modelo del sistema. En este contexto, la inteligencia artificial se presenta como una herramienta prometedora para la sintonización de controladores, ya que puede automatizar y optimizar este proceso de ajuste. [1]

Los sistemas de empuje vectorial son un tipo de sistema utilizado en la industria aeroespacial para controlar el empuje y la dirección de un cohete, su sintonización es una tarea compleja debido a la gran cantidad de variables involucradas y a la necesidad de mantener la estabilidad y seguridad del cohete.

El objetivo de esta tesis es desarrollar una herramienta computacional que aplique técnicas de inteligencia artificial, utilizando parámetros de construcción y datos de funcionamiento de cohetes, para eficientemente determinar las constantes de sintonización de un algoritmo de control *PID* (Proporcional, Integral y Derivativo) en un sistema de empuje vectorial. Posteriormente, se realizarán simulaciones y se llevará a cabo una implementación en un banco de pruebas como parte del proceso de validación de esta herramienta.

Se espera que este enfoque permita obtener un controlador eficiente y estable sin la necesidad de un conocimiento detallado del modelo del cohete y del sistema de control y que pueda ser escalable a sistemas de empuje vectorial diseñados para cualquier cohete. Además, el sistema propuesto puede reducir el costo y la complejidad del proceso de ajuste de controladores.

II. OBJETIVOS

A. Objetivo General

Desarrollar una herramienta computacional que permita ajustar de forma automática un algoritmo de control de empuje vectorial de un cohete en función de sus características constructivas y operacionales, usando técnicas basadas en la inteligencia artificial.

B. Objetivos Específicos

- Identificar las características constructivas y operacionales que afectan el control de actitud de un cohete dotado con un sistema de control de empuje vectorial.
- Desarrollar una metodología para el ajuste automático de un algoritmo de control de empuje vectorial de un cohete en función de las características constructivas y operativas del mismo, involucrando técnicas de la inteligencia artificial.
- Diseñar una interfaz de simulación de un cohete con control de empuje vectorial auto ajustable en la cual se puedan definir las características constructivas y operacionales del mismo.
- Validar el desempeño del algoritmo de control de empuje vectorial auto ajustable mediante pruebas en un banco experimental.

III. MARCO TEÓRICO

A. Actitud de un Cohete

Inicialmente, se debe definir y conocer el concepto de centro de gravedad (CG). Este lugar geométrico se define como el punto de aplicación de la resultante de las fuerzas gravitatorias que actúan sobre un cuerpo [2]. Es de interés conocerlo debido a que cualquier rotación de un cuerpo sobre sí mismo se da alrededor de este punto. La codificación de estas rotaciones es conocida como actitud.

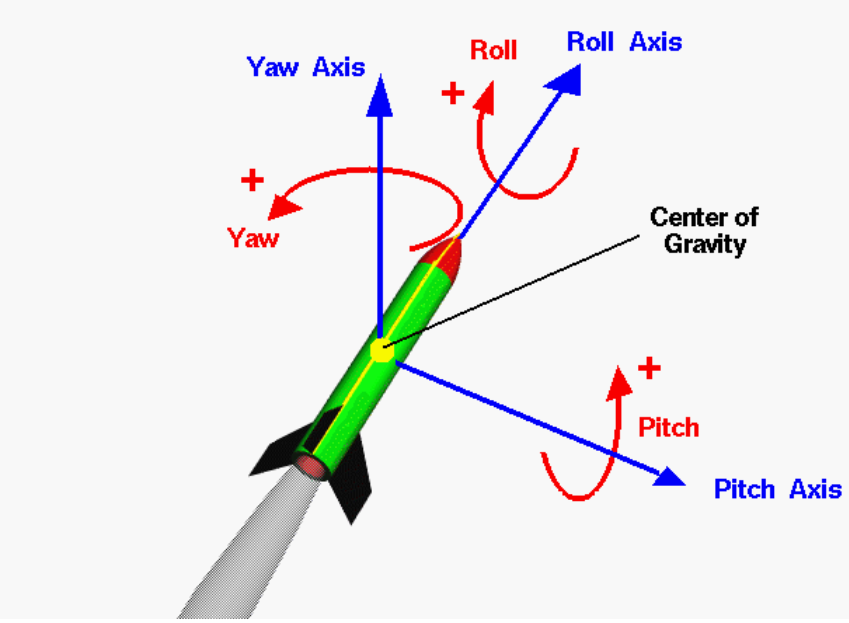

Fig. 1. Rotaciones de un Cohete sobre los Ejes del Cuerpo [3]

En la Figura 1 se ilustra los movimientos alrededor de 3 ejes ortogonales cuyo origen es el centro de gravedad del cuerpo representado como un punto amarillo. El eje de roll es el eje longitudinal y el movimiento alrededor de este eje es llamado alabeo; el eje de pitch es el eje lateral, el movimiento alrededor de este eje es llamado cabeceo; por último, está el eje de yaw el cual corresponde al eje vertical del cuerpo, el movimiento alrededor de este eje es llamado guiñada.

Existen otras formas para representar la actitud como lo es la utilización de los ángulos que forman los ejes vistos en la Figura 1 con respecto a un marco de referencia inercial, en muchos casos puesto en la Tierra, ver Figura 2. Para definir la orientación del cuerpo se utiliza una serie de rotaciones puras, estas rotaciones asociadas se denominan ángulos de Euler [4].

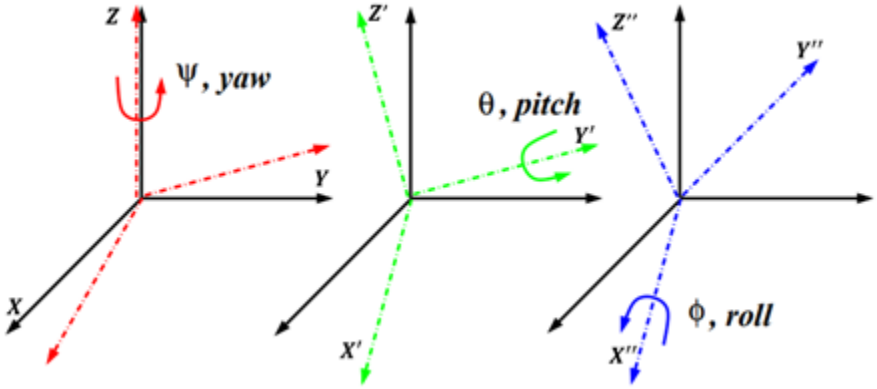

Fig. 2. Ángulos de Euler [5]

En la Figura 2 se puede observar la secuencia de rotaciones que, a partir del marco de referencia inercial de la tierra, se debe realizar para obtener la orientación de un cuerpo. Estar rotaciones llevan a ángulos de rotación denominados ángulos de Euler y estos pueden enumerados como:

- La rotación alrededor del eje Z, un ángulo ψ que corresponde al ángulo de yaw.
- La rotación alrededor del eje Y', correspondiente al eje Y después de la primera rotación, un ángulo θ que representa al ángulo de pitch.
- La rotación alrededor del eje X", correspondiente al eje X después de la segunda rotación, un ángulo ϕ que representa al ángulo de roll.

B. Estabilidad de Cohetes

Es importante, antes de definir la estabilidad en cohetes, conocer el concepto del centro de presión (CP). El centro de presión es el punto donde convergen todas las fuerzas

aerodinámicas de un cuerpo que se mueve a través del aire; este está directamente ligado al área transversal del cohete [6]. El CP se puede ver en la Figura 3 representado por un punto amarillo con centro negro bajo el centro de gravedad.

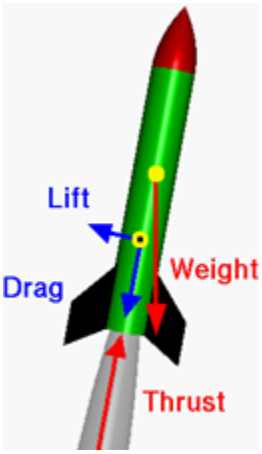

Fig. 3. Fuerzas en un cohete [7]

Las fuerzas externas que comúnmente actúan sobre los cohetes en la atmósfera terrestre e influyen en su trayectoria consisten en empuje (thrust), fuerzas aerodinámicas de arrastre y sustentación (drag y lift, respectivamente) y atracciones gravitatorias (weight).

La fuerza de empuje, F, es la fuerza producida por el sistema de propulsión de los cohetes, la mayoría de las veces actúa a lo largo del eje de la tobera. Esta fuerza se debe a la salida de flujo de masa constante y es función de la velocidad de escape efectiva c y la tasa de flujo másico del propulsor $\dot{m}$. Se relacionan como se muestra en la ecuación (1).

La fuerza de arrastre D actúa en una dirección opuesta a la trayectoria de vuelo y se debe a la resistencia al movimiento del cuerpo por parte del fluido circundante. Se expresa en los términos de la ecuación (2), donde C_D es el coeficiente de arrastre, u representa la velocidad de vuelo del vehículo y ρ representa la densidad del fluido que rodea el cohete. Para aviones y misiles con alas, el área A es el área del ala, y para misiles sin alas o vehículos de lanzamiento espacial, A es el área transversal máxima normal al eje del misil [8].

La fuerza de sustentación L es la fuerza aerodinámica que actúa en una dirección

normal a la trayectoria de vuelo. Esta se da debido al diseño del fuselaje del cohete y sus alas en caso de tenerlas, y se expresa mediante la ecuación (3), donde C_L es el coeficiente de sustentación.

$$F_{Thrust} = c\dot{m} \tag{1}$$

$$D = C_D \frac{1}{2} \rho A u^2 \tag{2}$$

$$L = C_L \frac{1}{2} \rho A u^2 \tag{3}$$

Finalmente, una de las fuerzas que siempre actúa sobre los cuerpos con inercia es la fuerza de gravedad, la cual apunta hacia el centro del cuerpo celeste más cercano, en este caso, la Tierra. Se puede considerar que esta fuerza actúa directamente en el centro de gravedad del cohete, punto amarillo de la Figura 3.

Durante el vuelo de un cohete es posible despreciar fuerzas como la presión de la radiación solar que suele ser pequeña, pero las fuerzas generadas por las ráfagas de viento o inestabilidades en el motor provocan un cambio en la actitud durante la trayectoria. Es por esto que se hace necesario aplicar métodos de estabilización en el cohete para asegurar la integridad de este y la misión.

La estabilidad de un vehículo se logra cuando este no gira ni oscila aleatoriamente durante el vuelo. Los vuelos inestables son indeseables porque las oscilaciones de cabeceo o guiñada aumentan la resistencia y causan problemas con los instrumentos y sensores. La inestabilidad a menudo conduce a la caída de los vehículos [8].

A continuación, se abordan varios métodos usados comúnmente para la estabilización de cohetes.

1) Estabilización Pasiva Uno de los métodos más comunes de estabilización es la utilización de alas en la parte inferior del fuselaje del cohete. El objetivo de las alas es hacer que el CP esté por debajo del CG aumentando el área del cohete. Así, cuando se produce una perturbación, la fuerza de sustentación genera un momento sobre el CG del cohete, haciendo

que la actitud regrese al estado inicial. A esta fuerza se le suele llamar fuerza regenerativa. Dicho fenómeno, al no tener ningún sistema que requiera energía adicional, se considera un método de estabilización pasivo. En la Figura 4a se muestra un cohete que tuvo una perturbación hacia la derecha, dicha inclinación generará sustentación en ese mismo sentido y en colaboración con la inclinación de la fuerza de arrastre generarán un torque en contra de las manecillas del reloj que tenderá a alinear el eje de simetría del cohete con la dirección de vuelo como se ve en la Figura 4b, por último en la Figura 4c se muestra el caso de una perturbación hacia la izquierda, contrario al descrito anteriormente, generando el mismo efecto restaurador, pero en sentido de las manecillas del reloj que también tenderá alinear el eje de simetría del cohete con la dirección de vuelo.

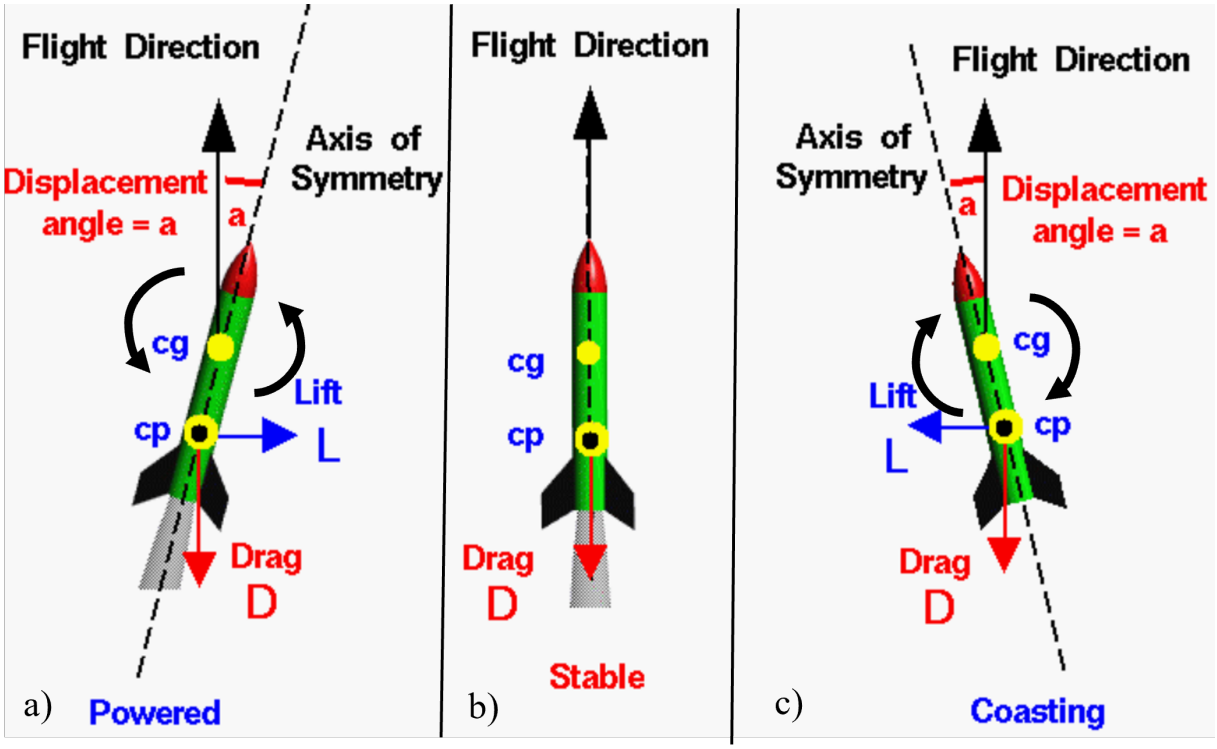

Fig. 4. Estabilidad en Cohetes [9]

Sin embargo, con las alas no se puede controlar la dirección y actitud del cohete en su vuelo,

mucho menos en ambientes donde no se dan fenómenos aerodinámicos, como lo es el espacio. Para que haya un control sobre el cohete, es necesario incluir un sistema activo de control de estabilidad, como lo es el Sistema de Empuje Vectorial (Thrust Vector Control, TVC).

2) Control de Estabilidad con Sistema de Empuje Vectorial (TVC) El TVC es uno de los métodos más comunes para controlar la actitud de un cohete. El principio de funcionamiento se basa en la idea de controlar la dirección del vector de empuje. El TVC proporciona pares de control para dirigir y estabilizar el cohete a lo largo de los tres ejes principales (balanceo, cabeceo y guiñada), superando perturbaciones de diversa naturaleza [10]. Dicho funcionamiento se aprecia en la Figura 5; en la parte a) de esta figura se muestra lo que ocurre cuando el vector de empuje es desviado hacia la izquierda del plano vertical del cohete, se genera un torque en sentido contrario de las manecillas del reloj alrededor del centro de gravedad del cohete; en la parte b) se muestra el caso de una alineación del vector de empuje con el plano vertical por lo que no se genera ningún torque. Por último, en la parte c) se muestra el caso contrario a la parte a) cuando el vector de empuje se desvía hacia la derecha del plano vertical.

En general, los métodos de empuje vectorial se pueden agrupar en 3 categorías diferentes:

1. **Inyección de fluido reactivo:** Este método consiste en inyectar un fluido en el lado del flujo que sale de una boquilla, por medio de inyectores particulares montados en los lados del motor. Cuando el fluido reactivo se inyecta en un solo lado de los gases de escape, cambia las propiedades del flujo, lo que resulta en un valor de empuje diferente en ese lado.

2. **Desviación del flujo de escape:** El método de desviación del flujo de escape consiste en colocar superficies particulares a la salida de la boquilla que se pueden orientar para dirigir el flujo y, en consecuencia, el vector de empuje, en la dirección deseada. Las más comunes de esas superficies son paletas de chorro externas y toberas adicionales externas.

3. **Manipulación mecánica del motor:** Los vehículos que hacen uso de este sistema en particular están equipados con motores móviles o boquillas que cambian de forma, que pueden orientarse y/o modificarse para alcanzar el movimiento deseado. Al inclinar el motor

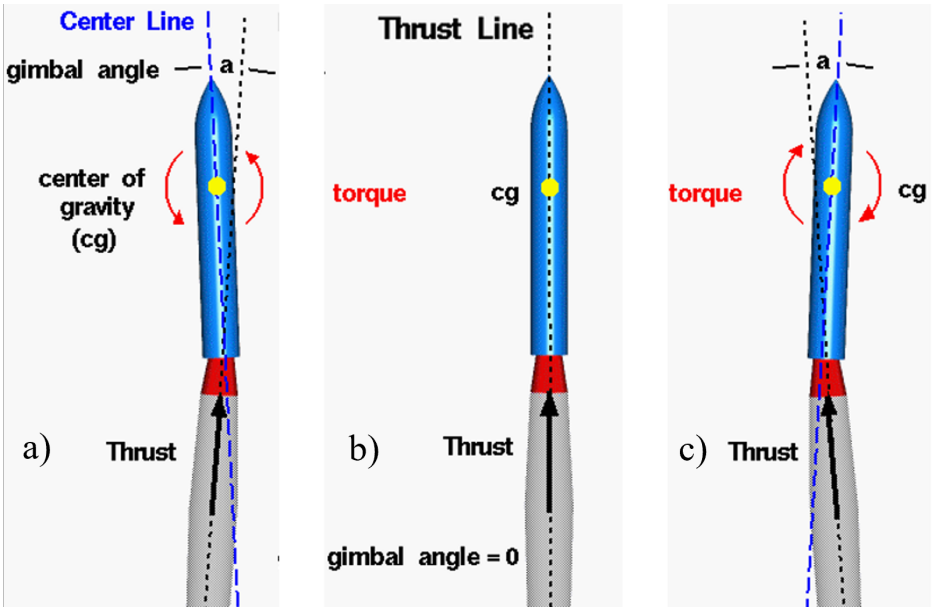

Fig. 5. Empuje Vectorial [11]

o la tobera, se generará un desfase entre el vector de empuje y el centro de masas, dando lugar a pares de control alrededor del propio centro de masas. Esos sistemas particulares generalmente también se denominan como "gimbaled thrust systems" o sistemas de empuje con cardanes y el desplazamiento del motor hace que el vector de empuje forme un ángulo con respecto al plano vertical del cohete, este ángulo es llamado gimbal angle o ángulo de cardán.

Cuando el vehículo está equipado con un solo motor, la vectorización de empuje permite el control solo de los ángulos de pitch y yaw y no tiene efecto sobre el movimiento de roll. La única forma de tener un control total sobre la dinámica de rotación del sistema a través de la manipulación mecánica es combinar la vectorización de empuje en, al menos, 2 motores diferentes [12].

Teniendo en cuenta lo anteriormente mencionado, en este punto se hace indispensable mencionar los métodos utilizados para realizar un adecuado control de los diferentes movimientos de estos sistemas vectoriales.

C. Sistemas de Control Automático

Un sistema de control automático es un conjunto de componentes interconectados que trabajan juntos con un único objetivo, el obtener el control de una determinada variable de un sistema físico, como puede ser la temperatura, velocidad, altura, posición, presión, etc. A partir de modelos matemáticos. Estos controladores son usados en una gran cantidad de aplicaciones, desde refrigeradores hasta controlar la dirección de empuje de un cohete como se ha mencionado.

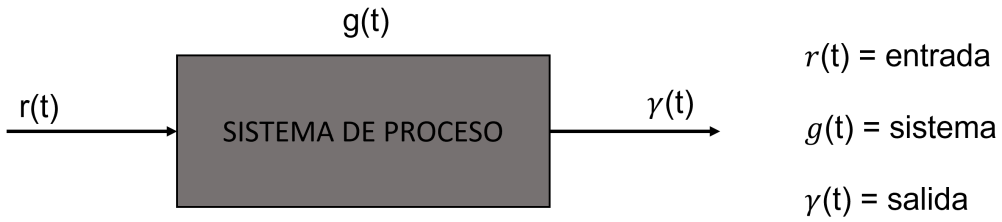

Fig. 6. Esquema de un Proceso o Planta [13]

En la Figura 6 se aprecia de manera simplificada por medio de bloques la representación de un proceso o planta cualquiera, teniendo en cuenta su entrada, salida y proceso o sistema.

Los sistemas de control pueden clasificarse en sistemas de lazo abierto y de lazo cerrado.

• **Lazo abierto:** Son sistemas en los que no se obtiene información de la salida para realizar el control, un ejemplo de esto podría ser el funcionamiento una tostadora, ya que esta sólo aplica una cantidad fija de calor durante un tiempo determinado, esperando como resultado que el pan se tueste a un nivel deseado.

• **Lazo cerrado:** Este tipo de control también es llamado "automático" en este la información de la salida es comparada con un comportamiento deseado; con el resultado de esta comparación se reingresa al sistema y se obtiene un control más preciso. Un ejemplo básico para este sistema son los termostatos en el aire acondicionado de una casa, ya que estos miden

la temperatura ambiente y enciende el dispositivo en función de la temperatura deseada por el usuario.

Existen varios tipos de sistemas de control, entre los más populares están el proporcional (P), integral (I), derivativo (D) y sus respectivas combinaciones PI, PD, PID.

1) Controlador Proporcional (P): Este es el tipo más simple, se habla de un controlador proporcional cuando la salida de este es proporcional a la diferencia entre la salida obtenida y la deseada, de aquí en adelante llamaremos a esto el "error". Matemáticamente, podemos expresarlo como se muestra en la ecuación (4).

$$u(t) = k_p e(t) \tag{4}$$

Donde $u(t)$ es la función de salida del controlador, $e(t)$ es la función del error y k_p es la constante de proporcionalidad o ganancia. Cabe resaltar que este tipo de controladores son fáciles de implementar, pero debido a su alta simplicidad puede no ser útil para la mayoría de las aplicaciones, por lo que este tipo de controladores no son muy usados.

2) Controlador Integral (I): Se habla de un controlador integral cuando la salida del controlador es proporcional a la integral del error o lo que es lo mismo, a la suma acumulada del error. Matemáticamente hablando podemos expresarlo como:

$$u(t) = k_i \int_{t_0}^{t_1} e(\tau) d\tau \tag{5}$$

Donde k_i es la ganancia del control integral y la integral del error es tomada desde un tiempo inicial t_0 hasta un tiempo t_1. El control integral suele reducir al máximo el valor de error cuando no se obtienen cambios significativos en éste con el tiempo, de aquí en adelante llamaremos a esta situación "estado estable", pero este comportamiento suele hacer que el controlador tienda a sobre-corregir y se obtenga un comportamiento no deseado.

3) Controlador Derivativo (D): Se habla de un controlador derivativo cuando su salida es proporcional a la derivada del error. Matemáticamente hablando:

$$u(t) = k_d \frac{d(e(t))}{dt} \tag{6}$$

Donde k_d es la ganancia del control derivativo. El objetivo de este control es brindar correcciones relacionadas con la velocidad a la que el error crezca o decrezca, es importante tener en cuenta que este controlador siempre debe ser combinado con otro tipo, ya que no respondería en el caso de que se tengan errores de estado estable.

4) Controlador Proporcional Integral Derivativo (PID): Este tipo de controlador es la combinación de las 3 técnicas de control anteriormente vistas, heredando y equilibrando sus ventajas y desventajas. Se habla de un controlador PID cuando la función de salida es proporcional al error sumado a proporciones de la integral y derivada de este error, matemáticamente hablando:

$$u(t) = k_p e(t) + k_i \int_{t_0}^{t_1} e(\tau) d\tau + k_d \frac{d(e(t))}{dt} \tag{7}$$

Se ve en la ecuación (7) la suma de los tres controladores vistos con sus respectivas ganancias. Uno de los desafíos para este controlador es la escogencia de estos valores de ganancia, los cuales son cruciales para su buen funcionamiento. Encontrar los valores adecuados para k_p, k_i y k_d es comúnmente llamado "sintonización" y se suelen usar estrategias matemáticas y numéricas para esto. Este tipo de controlador es muy utilizado en la industria debido a que puede controlar una amplia gama de sistemas dinámicos y a su vez adaptarse a los cambios de condiciones del sistema.

D. Inteligencia Artificial Aplicada a Sistemas de Control

En este apartado, exploraremos diversas técnicas avanzadas en el campo de la inteligencia artificial, específicamente en el ámbito del control y la optimización, como los son el control basado en redes neuronales, algoritmos genéticos, optimización por enjambre de partículas, Algoritmo de retropropagación (Back propagation) y neuro evolución. Cada uno

de los temas abordados representa un enfoque único para resolver problemas complejos, desde el control de sistemas hasta la optimización de funciones. A continuación, presentamos una descripción de cada uno de ellos:

1) Control Basado en Redes Neuronales: Las redes neuronales artificiales son artilugios matemáticos basados en el comportamiento de las neuronas biológicas y el cerebro. La unidad de procesamiento para una red neuronal es llamada igual que para el cerebro, neurona. Estas generan una única respuesta dadas unas condiciones específicas de entrada. Una representación gráfica de esto se puede ver en la Figura 7 y matemáticamente se expresa mediante la ecuación (8).

$$A = f(\sum_{i=0}^{n} X_i * W_i) \tag{8}$$

Donde:

- A: Es la salida de la neurona.
- n Es la cantidad total de datos de entrada.
- $X_1, X_2, ..., X_n$ Son los datos de entrada de la neurona.
- X_0^*, W_0^* Es la unidad de sesgo y su peso respectivamente.
- $W_1, W_2, ..., W_n$ Son los pesos relativos de cada entrada.
- f: Es llamada la función de activación de la neurona.

Una red neuronal consta de tres niveles o capas, Figura 8.

- **Capa de entrada:** Es el componente que recibe la información.
- **Capas intermedias u ocultas:** Es el componente encargado del procesamiento de los datos que se incorporaron por la capa de entrada.
- **Capa de salida:** Es el componente que se encarga de transferir los datos procesados.

Cada una de las neuronas en una red neuronal incide más o menos dependiendo de los resultados que obtenga de sus interconexiones. Lo más importante de estos sistemas es que aprenden y se forman a sí mismos [14].

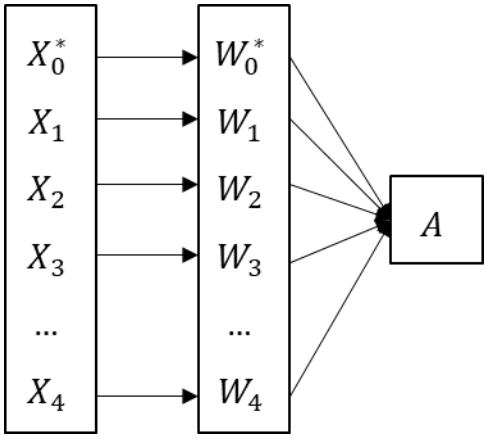

Fig. 7. Diagrama de una Neurona Artificial

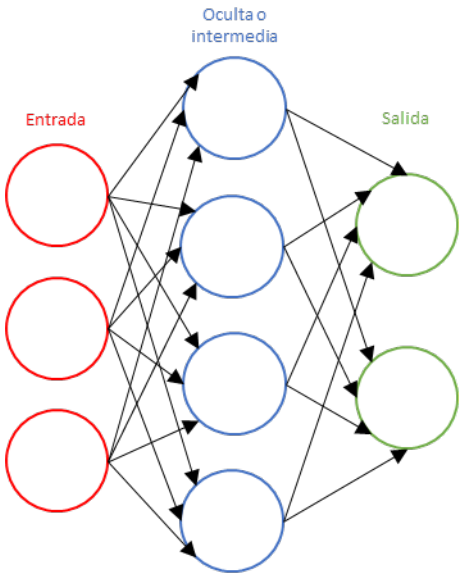

Fig. 8. Esquema de una Red Neuronal Artificial

En el ámbito de las redes neuronales, nos encontramos con una diversidad de enfoques para el entrenamiento y la optimización. Entre ellos, destacan dos métodos ampliamente utilizados: la retropropagación y la neuro evolución.

a) Algoritmo de Retropropagación (Back propagation) Es crucial comprender que una función de coste evalúa la calidad del modelo final [15]. Este representa la versión óptima del modelo, lograda a través de ajustes durante el proceso de entrenamiento para minimizar la disparidad entre las predicciones y los valores reales. Lo ideal es buscar que dicha función de coste posea o alcance los valores más pequeños posibles, valores mínimos. Para esto, se usa generalmente procesos infinitesimales que requerirían un gran costo computacional si no fueran por algoritmos como el de descenso de gradiente, también conocido como Gradient Descent o Batch Gradient Descent que implica el cálculo del gradiente de la función de coste en un punto concreto, lo que a su vez implica el cálculo de la derivada parcial de la función de coste con respecto a todos y cada uno de los parámetros que van a determinar el funcionamiento de la red. Y es aquí donde el algoritmo de backpropagation llega para resolver el problema. En la Figura 9 se muestra la conexión de una salida de la neurona k en la capa $l-2$ a la neurona i en la capa $l-1$ cuya salida pasa por la neurona j en la capa l para finalmente llegar a $y_j^{(l)}$.

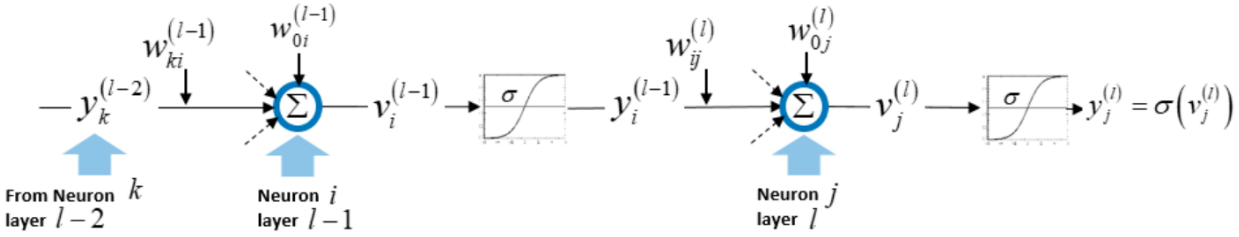

Fig. 9. Parte de una Red Neuronal de Retroalimentación

$y_k^{(l-2)}$ es la salida de la neurona k en la capa $l-2$, $y_i^{(l-1)}$ es la salida de la neurona i en la capa $l-1$, y $y_j^{(l)}$ es la salida de la neurona j en la capa l. Teniendo en cuenta la Figura 9 y lo ya planteado se puede mencionar que [16]:

Suponga que hay M salidas finales con errores cuadráticos $\mathscr{E}_m = (y_m - c_m)^2$, $1 <= m <= M$, cuya suma dividida por M es el error de salida promedio. Diferenciar $\mathscr{E}_m$ con respecto a los pesos en la capa $l-1$ da

$$\frac{\partial \mathscr{E}_m}{\partial w_{ki}^{(l-1)}} = \frac{\partial \mathscr{E}_m}{\partial v_i^{(l-1)}} \frac{\partial v_i^{(l-1)}}{\partial w_{ki}^{(l-1)}} = \frac{\partial \mathscr{E}_m}{\partial v_i^{(l-1)}} y_k^{(l-2)} \tag{9}$$

Ahora, se puede definir la última parte de la derivada así:

$$\frac{\partial \mathscr{E}_m}{\partial v_i^{(l-1)}} = \sum_{j=1}^{N_l} \frac{\partial \mathscr{E}_m}{\partial v_j^{(l)}} \frac{\partial v_j^{(l)}}{\partial v_i^{(l-1)}} = \sum_{j=1}^{N_l} \frac{\partial \mathscr{E}_m}{\partial v_j^{(l)}} w_{ij}^{(l)} \sigma'(v_i^{(l-i)}) \tag{10}$$

Sabiendo que:

- w: Valor de peso.
- $v = w^T x$
- $\sigma(v)$: Función sigmoid o logística.
- $\sigma(v) = \frac{1}{(1+e^{-v})}$: Función de activación de la neurona.

La ecuación (10) es la fórmula de retropropagación para calcular todos los gradientes hasta la primera capa oculta, comenzando con la capa de salida l

b) Neuro Evolución La neuro evolución es una técnica de inteligencia artificial que está motivada por la evolución de los sistemas nerviosos biológicos y aplica abstracciones de la evolución natural, es decir, algoritmos evolutivos, para generar redes neuronales artificiales.

El algoritmo básico de neuro evolución funciona de la siguiente manera. Se evoluciona una población de genotipos que codifican redes neuronales artificiales para encontrar una que pueda resolver un problema computacional. Luego la red neuronal se prueba en una tarea específica durante un período de tiempo determinado. A continuación se registra el rendimiento o aptitud de esta red y una vez que se determinan los mejores valores para los genotipos en la población actual, se genera una nueva población, cambiando ligeramente los genotipos, mutación, o combinando múltiples genotipos, cruzamiento. En general, los genotipos con mayor aptitud tienen mayores posibilidades de ser seleccionados para la reproducción y su descendencia reemplaza a los genotipos con valores de aptitud más bajos, formando así

una nueva generación. Este bucle generacional suele repetirse cientos o miles de veces, con la esperanza de encontrar redes mejores y con mejor rendimiento [17].

Los algoritmos de neuro evolución se dividen en tres categorías según la tarea: búsqueda de valores de pesos en la red neuronal con estructura fija; establecer la estructura de la red neuronal; establecer funciones de activación de neuronas; y varias combinaciones de las tareas anteriores [18].

La búsqueda de valores de pesos es la mutación más simple. Dentro de este tipo existen dos subtipos de mutación. La más común es una mutación suave, en la que se suma o resta un número pequeño a una conexión. Con menos probabilidad, se puede realizar una mutación brusca, que vuelve a generar un valor aleatorio para la conexión. Si se quiere hacer una mutación de pesos, se recorre la red y para cada conexión se realiza uno de los dos subtipos, según la probabilidad asignada.

Para la estructura de la red neuronal se puede realizar dos tipos de mutaciones. La primera es una mutación de nodos que añade uno nuevo a la red. Para añadirlo se elige una conexión existente, se desactiva y se coloca el nodo en ese lugar. Para conectarlo a la red, se crea una conexión que sale del nodo del cual salía la conexión desactivada hacia el nuevo nodo, y otra que salga de dicho nodo al cual estaba conectada la conexión antigua, Figura 10. Como se han añadido 2 conexiones nuevas y se ha deshabilitado otra, para no afectar a los valores de la red, la conexión entrante al nodo nuevo tiene un peso de 1, y la saliente tiene el peso que tenía la conexión antigua.

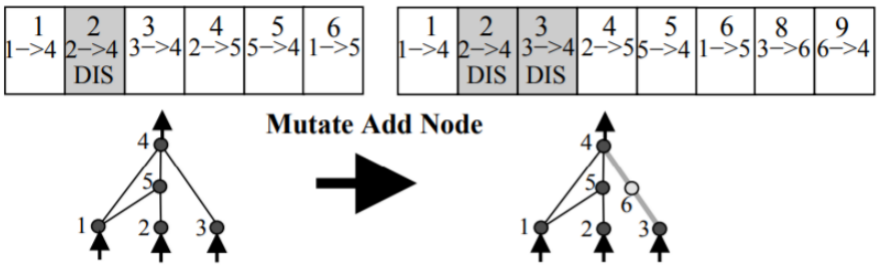

Fig. 10. Mutación de Nodos

La segunda posible mutación es la de conexión, es la más simple, ya que simplemente se escogen dos nodos y se crea una conexión con un peso aleatorio entre ellos, Figura 11. El problema obvio que hay que tener en cuenta es que esos nodos no sean del mismo nivel, esto es, por ejemplo, dos nodos de entrada [19], [20].

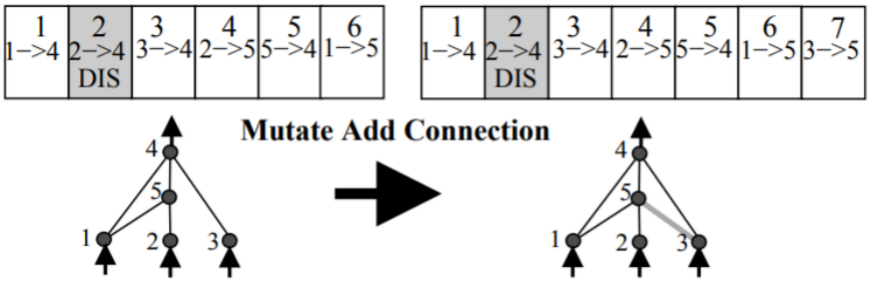

Fig. 11. Mutación de Conexión

2) Algoritmos Genéticos Los algoritmos genéticos (GA) son técnicas de optimización basadas en los conceptos biológicos de selección natural y genética. Estos algoritmos son un subconjunto de una rama mucho más profunda de la computación conocida como Computación Evolutiva.

Algunos términos básicos tener en cuenta son:

- **Población**: es un subconjunto del espacio muestral de soluciones posibles, codificadas, para un problema dado. En este conjunto cada individuo es una solución al problema que queremos resolver.
- **Cromosoma:** es una solución al problema planteado, es un elemento del espacio muestral de soluciones.
- **Genotipo**: Representación de los cromosomas hecha con cadenas de dígitos binarios.
- **Gen:** es la posición de un elemento de un cromosoma. Los individuos de una población se caracterizan por un conjunto de parámetros, variables y son particulares, denominados genes. Básicamente, los genes se unen en una cadena para formar un cromosoma, solución.
- **Alelo:** es el valor que toma un gen para un cromosoma en particular.

No existe una definición rigurosa del GA que sea aceptada por todos, sin embargo, se puede decir que la mayoría de los métodos tienen al menos los siguientes elementos en común: poblaciones de cromosomas, selección de acuerdo con la aptitud, cruzamiento para producir nueva descendencia y mutación aleatoria de nueva descendencia [21]. En la Figura 12 se muestra las características anteriormente mencionadas.

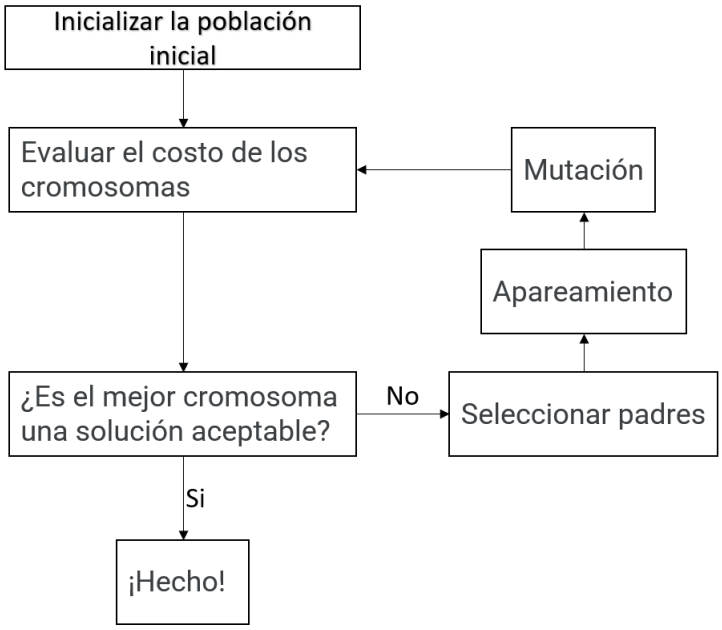

Fig. 12. Diagrama de un Algoritmo Genético Básico

La búsqueda de la solución, mejor individuo de esta población, se hace aplicando los operadores genéticos que emulan procesos naturales. Estos operadores se enmarcan en tres grupos: operadores de selección, operadores de cruce y operadores de mutación. Sin embargo, para poder aplicar estos operadores genéticos sobre la población, es necesario codificar a cada individuo, convirtiéndolo en una aproximación al valor real que facilita los mecanismos de búsqueda.

Un criterio de parada finaliza el proceso generacional. Dos criterios comunes son: número de generaciones, donde se especifica que el código se ejecutará hasta alcanzar una

determinada sucesión generacional. El otro es verificando la información genética de la población, la cual detiene la evolución cuando se verifica que hay una proliferación de un mismo individuo. Al finalizar el algoritmo, escoge el mejor habitante de la población final y se espera que un material genético proporcione la mejor respuesta al problema [22].

3) Optimización por Enjambre de Partículas (Particle Swarm Optimization) La optimización de enjambres de partículas (PSO) está basada en el comportamiento social de animales como las bandadas de pájaros o los bancos de peces. PSO tiene una matriz de población aleatoria como el algoritmo genético, pero las filas de la matriz se denominan partículas en lugar de cromosomas. Las partículas contienen valores variables y tienen una velocidad a medida que se mueven sobre la superficie del costo. Las partículas actualizan sus velocidades y posiciones en función de las mejores soluciones locales y globales. Una forma de expresar lo anterior se muestra en la ecuación (11) [23].

$$v_{m,n}^{new} = v_{m,n}^{old} + \ell_1 \times r_1 \times (p_{m,n}^{localbest} - p_{m,n}^{old}) + \ell_2 \times r_2 \times (p_{m,n}^{globalbest} - p_{m,n}^{old}) \tag{11}$$

Donde $v_{m,n}$ representa la velocidad de la partícula; $p_{m,n}$ las variables de la partícula, r_1 y r_2 números aleatorios uniformes independientes; $\ell_1 = \ell_2$: factores de aprendizaje $= 2$; $p_{m,n}^{localbest}$: mejor solución local y $p_{m,n}^{globalbest}$: la mejor solución global.

El algoritmo PSO encuentra nuevas partículas actualizando el vector de velocidad de cada partícula y agregándolo a la posición anterior. Las actualizaciones de velocidad se basan en la mejor solución global asociada con el costo más bajo encontrada para una partícula, así como en la mejor solución local asociada con el costo más bajo en la población actual. Cuando la mejor solución local tiene un costo menor que el costo de la solución global actual, reemplaza a la mejor solución global. La constante ℓ_1 es el parámetro cognitivo y la constante ℓ_2 es el parámetro social [24].

IV. METODOLOGÍA

El desarrollo de este trabajo de grado se realizó en tres etapas:

A. Etapa 1

Se hizo uso del repositorio institucional, recursos de información digital y electrónica de la universidad, además de las fuentes de información libre que se pueden consultar en internet para fundamentar el análisis de diferentes modelos de cohetes e identificar las características constructivas y operacionales que afectan el control de su actitud. A partir de este proceso se genera un modelo matemático que describe la actitud de un cohete en función de un sistema de control de empuje vectorial, como se muestra a continuación.

Para el presente trabajo se toma el marco de referencia inercial, denominado de ahora en adelante como marco de la tierra, con el eje x apuntando hacia el sur, el eje y apuntando hacia el este y por consiguiente el eje z apuntará hacia arriba.

Se tiene para el marco del cuerpo, en nuestro caso para el cohete, el eje x_b y el eje z_b saliendo perpendicular al eje longitudinal, el eje y_b se toma apuntando hacia la ojiva.

Para lograr obtener la orientación del cohete se toman 3 matrices R_1, R_2 y R_3, ecuaciones (12), (13) y (14), respectivamente, que representan estructuras de transformación luego de las rotaciones asociadas a ψ, θ y ϕ, respectivamente, en términos del marco de referencia anterior, como ya se mencionó para la Figura 2.

$$R_1 = \begin{bmatrix} cos(\psi) & sen(\psi) & 0 \\ -sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sen(\theta) \\ 0 & -sen(\theta) & cos(\theta) \end{bmatrix} \tag{13}$$

$$R_3 = \begin{bmatrix} cos(\phi) & 0 & -sen(\phi) \\ 0 & 1 & 0 \\ sen(\phi) & 0 & cos(\phi) \end{bmatrix} \tag{14}$$

Teniendo en cuenta esto, denominaremos $C_{b/e}$ la matriz de rotación que transforma vectores desde el marco de la tierra al marco del cuerpo, la cual está dada por la ecuación (15) o (16). Por simplicidad representaremos *seno* como s y *coseno* como c.

$$C_{b/e} = R_3 * R_2 * R_1 \tag{15}$$

$$C_{b/e} = \begin{bmatrix} c(\phi)c(\psi) - s(\phi)s(\theta)s(\psi) & c(\theta)s(\psi) + s(\phi)s(\theta)c(\psi) & -s(\phi)c(\theta) \\ -c(\theta)s(\psi) & c(\theta)c(\psi) & s(\theta) \\ s(\phi)c(\psi) + c(\phi)s(\theta)s(\psi) & s(\theta)s(\psi) - c(\phi)s(\theta)c(\psi) & c(\theta)c(\phi) \end{bmatrix} \tag{16}$$

El vector de peso en el marco del cuerpo $\vec{W}_b$ se representa de acuerdo a lo anterior como:

$$\vec{W}_b = C_{b/e} * m * g \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{17}$$

$$\vec{W}_b = m * g * \begin{bmatrix} s(\phi)c(\theta) \\ -s(\theta) \\ -c(\theta)c(\phi) \end{bmatrix} \tag{18}$$

Desde el marco del cuerpo es posible que se realicen dos rotaciones, β al rededor de z_b generando x_w, y_w y z_w y desde aquí la rotación α al rededor de x_w para obtener el marco del viento, ver Figura 13. Esta rotación se realiza para alinear el cohete con la dirección del flujo de aire y simplificar el análisis de las fuerzas aerodinámicas, Lift en dirección perpendicular al flujo y Drag en dirección paralela a dicho flujo.

Denominaremos $C_{w/b}$ a la matriz de rotación que transforma vectores desde el marco del cuerpo al marco del viento, la cual está expresada por la ecuación (19) o (20) .

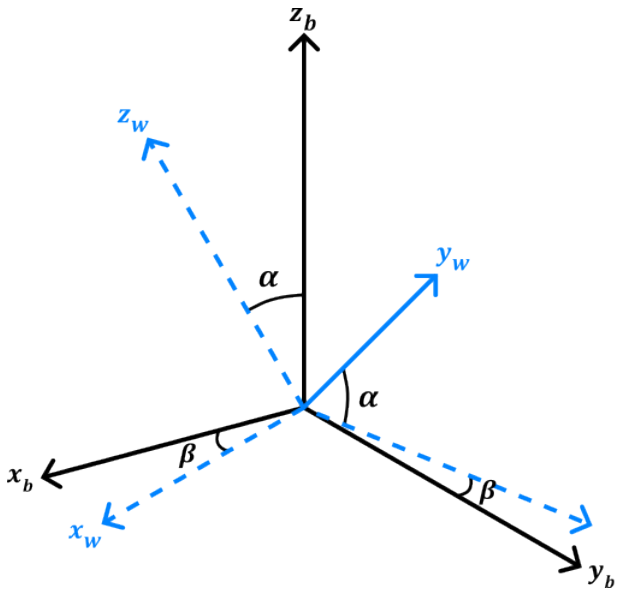

Fig. 13. Marco del Viento

$$C_{w/b} = R_2 * R_1 \tag{19}$$

$$C_{w/b} = \begin{bmatrix} c(\beta) & s(\beta) & 0 \\ -c(\alpha)s(\beta) & c(\alpha)c(\beta) & s(\alpha) \\ s(\alpha)s(\beta) & -s(\alpha)c(\beta) & c(\alpha) \end{bmatrix} \tag{20}$$

Ahora expresamos la matriz $C_{b/w}$ que transforma vectores del marco del viento al marco del cuerpo como la matriz transpuesta de $C_{w/b}$

$$C_{b/w} = (C_{w/b})^T \tag{21}$$

$$C_{b/w} = \begin{bmatrix} c(\beta) & -c(\alpha)s(\beta) & s(\alpha)s(\beta) \\ s(\beta) & c(\alpha)c(\beta) & -s(\alpha)c(\beta) \\ 0 & s(\alpha) & c(\alpha) \end{bmatrix} \tag{22}$$

Con esto ya es posible determinar el vector de Drag, que se encuentra en el marco del viento, en términos del marco del cuerpo como:

$$\vec{D_b} = C_{b/w} * D_w \tag{23}$$

$$\vec{D_b} = C_{b/w} * \begin{bmatrix} 0 \\ -D \\ 0 \end{bmatrix} \tag{24}$$

$$\vec{D_b} = D * \begin{bmatrix} c(\alpha)s(\beta) \\ -c(\alpha)c(\beta) \\ -s(\alpha) \end{bmatrix} \tag{25}$$

Para poder darle una buena interpretación al vector que representa el Thrust generado por la tobera, se hace indispensable definir los ángulos δ_{xy} y δ_{yz} como los ángulos de inclinación de la tobera con respecto al plano $x_b y_b$ y al plano $y_b z_b$, respectivamente, como se observa en la Figura 14.

Debido a esto, el vector de empuje se puede representar como:

$$\vec{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{26}$$

$$\vec{T} = T * \begin{bmatrix} -c(\delta_{xy})s(\delta_{yz}) \\ c(\delta_{xy}c(\delta_{yz}) \\ s(\delta_{xy}) \end{bmatrix} \tag{27}$$

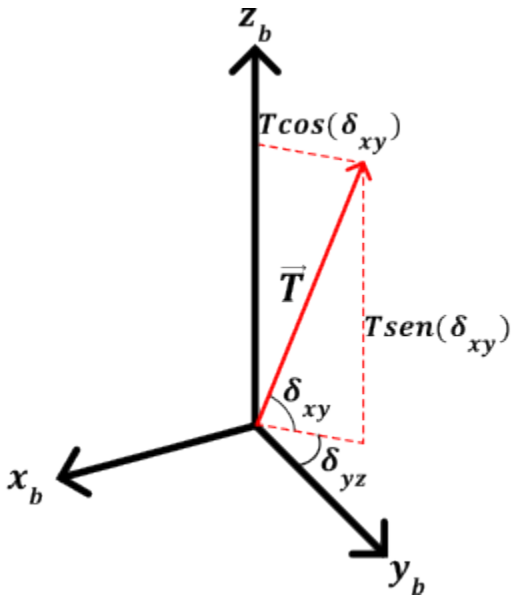

Fig. 14. Ángulos de Control para la Tobera Vectorial

En este punto ahora es posible plantear de manera vectorial un modelo de fuerzas y de momentos que corresponda con el comportamiento del cohete.

Se emplea la segunda ley de Newton, ver ecuación (28), para describir la fuerza neta del cohete.

$$\sum F = m * a \tag{28}$$

Se reescribe la fuerza neta en términos del vector de empuje, vector de peso y el vector de drag como se muestra en la ecuación (29).

$$\vec{T} + \vec{W} + \vec{D} = m * a \tag{29}$$

Reemplazando las ecuaciones (18), (25) y (27) obtenemos un modelo de fuerzas en términos de los ángulos de inclinación de la tobera y las rotaciones β y α como se muestra en la ecuación (30).

$$T * \begin{bmatrix} -c(\delta_{xy})s(\delta_{yz}) \\ c(\delta_{xy})c(\delta_{yz}) \\ s(\delta_{xy}) \end{bmatrix} + m * g * \begin{bmatrix} s(\phi)c(\theta) \\ -s(\theta) \\ -c(\theta)c(\phi) \end{bmatrix} + D * \begin{bmatrix} c(\alpha)s(\beta) \\ -c(\alpha)c(\beta) \\ -s(\alpha) \end{bmatrix} = m * \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{30}$$

Para el modelo de momentos se emplea la segunda ley de Newton para la rotación mostrada en la ecuación (31).

$$\sum \tau_{CG} = I * \vec{\alpha} \tag{31}$$

Multiplicando el momento de inercia por la aceleración angular en cada eje coordenado x, y y z se obtiene:

$$\sum \tau_{CG} = \begin{bmatrix} I_x\ddot{\theta} \\ I_y\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix} \tag{32}$$

Ahora, expandiendo la sumatoria de torques en términos de las fuerzas ya conocidas que actúan en el cohete, se obtiene la ecuación (33).

$$\begin{bmatrix} I_x\ddot{\theta} \\ I_y\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix} = \vec{\tau_T} + \vec{\tau_W} + \vec{\tau_D} \tag{33}$$

Dado que el vector de peso actúa en el punto de pivote del cohete, el CG, el vector de posición $\vec{r_W}$ será nulo y, por tanto, el producto vectorial $\vec{r_W} \times \vec{W}$ también será nulo, ver ecuación (34).

$$\begin{bmatrix} I_x\ddot{\theta} \\ I_y\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix} = \vec{r_T} \times \vec{T} + \cancelto{0}{\vec{r_W} \times \vec{W}} + \vec{r_D} \times \vec{D} \tag{34}$$

Expresando los vectores $\vec{r_T}$, $\vec{r_D}$, $\vec{T}$ y $\vec{D}$ en sus componentes se llega a:

$$\begin{bmatrix} I_x\ddot{\theta} \\ I_y\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ -l_{o,CG} \\ 0 \end{bmatrix} \times T * \begin{bmatrix} -c(\delta_{xy})s(\delta_{yz}) \\ c(\delta_{xy})c(\delta_{yz}) \\ s(\delta_{xy}) \end{bmatrix} + \begin{bmatrix} 0 \\ -l_{CP,CG} \\ 0 \end{bmatrix} \times D * \begin{bmatrix} c(\alpha)s(\beta) \\ -c(\alpha)c(\beta) \\ -s(\alpha) \end{bmatrix} \tag{35}$$

Realizando las operaciones planteadas de la ecuación (35) se obtiene:

$$\begin{bmatrix} I_x\ddot{\theta} \\ I_y\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix} = T \cdot \begin{bmatrix} -s(\delta_{xy}) \\ 0 \\ -c(\delta_{xy})s(\delta_{yz}) \end{bmatrix} \cdot l_{0,CG} + D \cdot \begin{bmatrix} s(\alpha) \\ 0 \\ c(\alpha)s(\beta) \end{bmatrix} \cdot l_{CP,CG} \tag{36}$$

Finalmente, de la ecuación (36) se pueden extraer las ecuaciones (37) y (38), que corresponden al modelo de momentos para los ejes x y z. En este punto se ha obtenido que el momento en el eje y termina siendo nulo.

$$I_x\ddot{\theta} = -T * s(\delta_{xy}) * l_{0,CG} + D * s(\alpha) * l_{CP,CG} \tag{37}$$

$$I_z\ddot{\psi} = -T * c(\delta_{xy}) * s(\delta_{yz}) * l_{0,CG} + D * c(\alpha) * s(\beta) * l_{CP,CG} \tag{38}$$

Para extraer la función de transferencia en este punto se necesita linealizar las expresiones que se han obtenido para el modelo, para esto se debe tener en cuenta que:

- La nariz del cohete siempre apunta en la dirección el viento, no se asume la existencia de vientos cruzados, entonces es posible asumir que $\alpha = \theta$ y $\psi = \beta$ [25].
- Debido a que el rango angular de la tobera es muy pequeño, de -5 a 5 grados, se supondrá que $sen(\delta_{xy})$ y $sen(\delta_{yz})$ son equivalentes a δ_{xy} y δ_{yz}, respectivamente.

Por lo anterior, y a partir de la ecuación (37) se tiene que:

$$I_x\ddot{\theta} = -T * \delta_{xy} * l_{0,CG} + D * \theta * l_{CP,CG} \tag{39}$$

Luego de linealizar es necesario utilizar la función de Laplace para describir la función de transferencia suponiendo condiciones iniciales iguales a cero.

$$I_x * (s^2\theta(s) - \cancelto{0}{s\theta(0)} - \cancelto{0}{\dot{\theta}(0)}) = D * \theta(s) * l_{CP,CG} - T * \delta(s) * l_{0,CG} \tag{40}$$

$$\frac{\theta(s)}{\delta(s)} = \frac{-T * l_{0,CG}}{I_x * s^2 - D * l_{CP,CG}} \tag{41}$$

Siguiendo el mismo análisis para la ecuación (38) podemos describir el comportamiento para el eje Z.

$$I_z\ddot{\psi} = -T * \delta_{xy} * \delta_{yz} * l_{0,CG} + D * \psi * l_{CP,CG} \tag{42}$$

$$I_x * (s^2\psi(s) - \cancelto{0}{s\psi(0)} - \cancelto{0}{\dot{\psi}(0)}) = D * \psi(s) * l_{CP,CG} - T * \delta(s) * l_{0,CG} \tag{43}$$

$$\frac{\psi(s)}{\delta(s)} = \frac{-T * l_{0,CG}}{I_z * s^2 - D * l_{CP,CG}} \tag{44}$$

De esta forma se obtienen las funciones de transferencia para la rotación alrededor de ambos ejes, ecuaciones (41) y (44).

B. Etapa 2

Se desarrolla una herramienta digital que permite la sintonización de sistemas de control de empuje para cohetes por medio de inteligencia artificial, haciendo uso del modelo matemático encontrado en la etapa 1. Esta herramienta se desarrolla usando MATLAB como lenguaje de programación y los algoritmos de inteligencia artificial escogidos para el proceso de sintonización. Una vez obtenido el aplicativo, se realizan diseños CAD en aras de una primera validación por medio de simulaciones en el entorno de programación visual Simulink, el cual está incorporado a MATLAB.

Se utiliza la Integral del Valor Absoluto del Error Multiplicada por el Tiempo, ITAE, definida en la ecuación (45), para representar la magnitud del error entre la respuesta deseada

y la respuesta real de un sistema. Lo anterior para tener un criterio de comparación entre los resultados de las simulaciones.

$$ITAE = \int_0^T t \cdot |e(t)| \cdot dt \tag{45}$$

La elección del $ITAE$ como parámetro de desempeño se basa en la relevancia de tener en cuenta las oscilaciones del controlador y su duración, tanto cuando superan el valor de referencia (SP) como cuando están por debajo de este, la inclusión del valor absoluto del error permite abordar esta situación de manera adecuada. Además, es crucial que el controlador logre una estabilización rápida, por lo que incluir el tiempo como factor en la ecuación permite incluir esta situación. En este contexto, el $ITAE$ resulta ser un criterio idóneo para diferenciar cuantitativamente los distintos controladores obtenidos.

Es importante mencionar en este punto que los algoritmos evolutivos prefieren individuos más aptos, lo cual es la base de la optimización y convergencia de los algoritmos. Cada solución en una población se denomina individuo. Cada individuo tiene su representación genética, llamada código, y su evaluación de rendimiento, llamada valor de aptitud o *fitness* [26]. En este trabajo se utiliza este parámetro para medir la eficiencia del algoritmo durante cada generación.

Se eligen cuatro cohetes y su información es obtenida a través de recursos en línea [27] ("Comercial", "ejemplo" y Prueba") y un diseño propio ("Martin"). Para cada uno de estos cohetes, se dispone de archivos de diseño en software, específicamente en el programa *OpenRocket* [28]. Este software permite la creación y simulación de diseños para cohetes amateur, incluyendo formas, dimensiones, elementos, clasificaciones de motores, materiales, entre otros aspectos. De esta manera, se pueden extraer las características constructivas y operativas necesarias para la prueba inicial de la interfaz, usándose el cohete *ejemplo*; y la sintonización del controlador PID mediante la interfaz gráfica desarrollada, en este caso usando los datos de los cohetes *Comercial*, *Martín* y *Prueba*. Las características de los cohetes seleccionados para esta sección se encuentran detallados en las Tablas I y II.

Variable	Valor	Unidades
I_x	0,03232	kg*m^2
L_{0CG}	0,373	m
D	0,0805	m
L_{CPCG}	0,106	m
T	76,71	N
v	77,78	m/s
C_D	0,3379	NA

TABLA I

DATOS DE COHETE EJEMPLO.

Cohete	$I_x[kg \cdot m^2]$	$L_{0CG}[cm]$	$D[cm]$	$L_{CPCG}[cm]$	$v_{prom}[\frac{m}{s}]$	C_D	$T[N]$
Comercial	0.00292	15.6	3.37	3.1	81.1	0.83972	11.5319
Martín	0.03368	39.2	5.48	11	80.9031	0.3111	80.8761
Prueba	0.03232	37.3	8.05	10.6	77.78	0.338	76.71

TABLA II

DATOS DE LOS COHETES COMERCIAL, MARTÍN Y PRUEBA.

Los tres cohetes fueron seleccionados teniendo en cuenta el *margen de estabilidad estática* ($S.M.$), según la ecuación (46). Se considera que un cohete es inherentemente estable cuando este margen se sitúa entre 1.5 y 2 [29]. Para el cohete "Comercial", se observa un cohete relativamente pequeño con un margen de estabilidad de 0.92, lo que indica una inestabilidad significativa. En el caso del cohete "Martin", se presenta un margen de aproximadamente 2, sugiriendo una condición de estabilidad. Por último, para el cohete "Prueba", el margen de estabilidad es de 1.32, indicando nuevamente una configuración inestable, aunque menos que el cohete "Comercial".

$$S.M. = \frac{L_{CPCG}}{D} \tag{46}$$

C. Etapa 3

Se realizan pruebas reales para la validación del desarrollo propuesto. Para ello se implementará el controlador sintonizado en un banco de pruebas experimental y se definen criterios para la evaluación del desempeño de este controlador. Para esto se hizo uso de un banco de pruebas ya diseñado y construido, que fue el resultado de una investigación del Semillero de Ingeniería Aeroespacial Delta-V junto con el cumplimiento del proyecto semestral para el curso de Control de Vuelo del pregrado de Ingeniería Aeroespacial de la Universidad de Antioquia [30], ver Anexos B y C. El banco posee un cohete de pruebas, al cual se le realizaron modificaciones en búsqueda de un mejor funcionamiento y, por tanto, unos resultados más consistentes. Se redistribuye los componentes electrónicos y la fuente de energía, adicionalmente se modifica la entrada de la manguera de aire para que ahora lo haga de manera lateral a la estructura del cohete, Figura 15, dando lugar a reorganizar de una mejor manera los elementos internos.

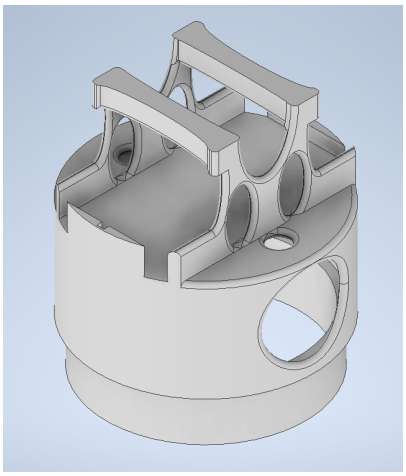

Fig. 15. Base para Electrónica.

Como con lo anterior se modifica la distribución de masas del cohete, se cambia el acople inferior del mismo para contrarrestar esto. Se le agregan divisiones que alojan contrapesos de estaño para mantener el cohete en la posición deseada, Figura 16, en una posición de estabilidad en ausencias de perturbaciones.

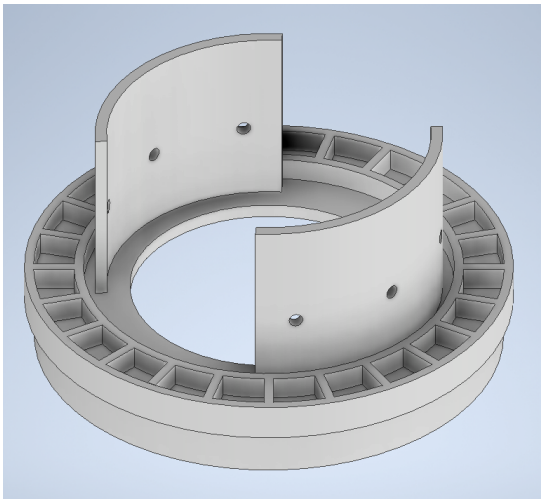

Fig. 16. Base Inferior con Contrapesos.

Adicionalmente, se rediseña el soporte que sostiene el cohete, dándole firmeza a la manguera en su entrada y un rango de movimiento más amplio a la estructura central, como se ve en la Figura 17.

Al cohete utilizado durante las pruebas se le pueden hacer varias mediciones directas para obtener algunas de las características constructivas y operacionales, pero para obtener el valor del momento de inercia debe hacerse de manera experimental, como se plantea a continuación:

- Se obtiene el valor de la masa del cohete.
- Se mide la distancia del centro de masa a la tobera vectorial.
- Con ayuda de dos cuerdas colocadas a $5cm$ del centro de masa del cohete, se cuelga como se ve en la Figura 18.
- Se miden la longitud de dichas cuerdas.
- Se pone a oscilar el cohete al rededor de su centro de masa y se graba para con ayuda del programa *Olive* calcular el valor del periodo de oscilación.
- Se calcula el momento de inercia con la ecuación (47).

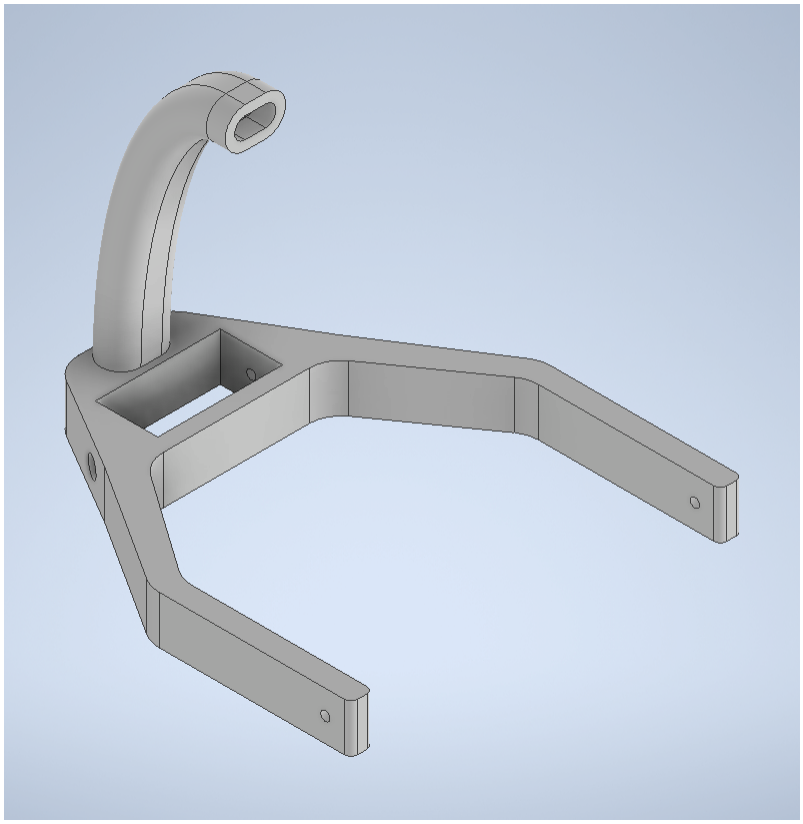

Fig. 17. Soporte General

$$I = \frac{m * g * T^2 * l_{cm-cuerda}^2}{4\pi^2 * l_{cuerda}} \tag{47}$$

Siendo m la masa del cohete, g el valor de la gravedad, T el periodo de oscilación, $l_{cm-cuerda}$ la longitud de la cuerda al centro de masa y l_{cuerda} la longitud de la cuerda.

Es importante tener en cuenta que este proceso se realiza dos veces, uno para el eje x y y del cohete, obteniéndose $I_x = 0,0063242$ y $I_y = 0,006438071$.

Por último, el valor del empuje se calcula con la ecuación (48) que corresponde a la ecuación de empuje de un cohete.

$$F = \dot{m} * v_e + (P_e - P_0) * A_e \tag{48}$$

Fig. 18. Proceso del Cálculo del Periodo de Oscilación del Cohete

Siendo $\dot{m}$ la variación de masa con respecto al tiempo, v_e el valor de la velocidad del fluido en la salida, P_e el valor dela presión del fluido en la salida, P_0 la presión atmosférica y A_e el área de la salida.

Para nuestro caso, el flujo de aire es constante y, por tanto $\dot{m} = 0$, esto nos deja la expresión del empuje como la ecuación (49).

$$F = (P_e - P_0) * A_e \tag{49}$$

Tenemos que la manguera utilizada es de 1/4 de pulgada, además se usaron valores de presión de salida de 40, 60 y 80 *psi*. Así se manejan valores de empuje de $5,525N$, $9,892N$ y $14,259N$ respectivamente. En la tabla III se resumen las características del cohete.

Eje	$I[kg \cdot m^2]$	$L_{0CG}[cm]$	D	L_{CPCG}	v_{prom}	C_D	$T_1[N]$	$T_2[N]$	$T_3[N]$
Eje 1	0.0063242	21.75	N.A	N.A	0	N.A	5.525	9.892	14.259
Eje 2	0.0064381	21.75	N.A	N.A	0	N.A	5.525	9.892	14.259

TABLA III
INFORMACIÓN DEL COHETE DEL BANCO DE PRUEBAS.

Una vez obtenidos los datos presentados en la Tabla III, estos se utilizan para la sintonización del controlador. Dado que la diferencia en el momento de inercia de ambos ejes a controlar del cohete es de 0.0001139, una variación que tiene un impacto limitado en los resultados, se considerará el cohete como masivamente simétrico. Por lo tanto, será necesario sintonizar únicamente uno de los ejes y este control podrá ser aplicado en ambos. Se ha optado por utilizar los datos del *Eje 2*, cuyo momento de inercia es $I = 0{,}0064381, \mathrm{kg} \cdot \mathrm{m}^2$, y además variar el empuje con los tres valores también descritos en la Tabla III. Dado que se lleva a cabo el análisis con cuatro algoritmos diferentes, en total se obtendrán 12 controladores.

Utilizando los datos obtenidos durante la etapa de sintonización, se lleva a cabo la evaluación práctica en un sistema real, correspondiente al banco de pruebas mostrado en la Figura 19. Con base en los índices ITAE, donde se busca el menor valor, y en el análisis cualitativo de los comportamientos observados en las gráficas de las simulaciones, se seleccionaron cuatro de ellas para ser sometidas a pruebas en el banco:

- **Algoritmo NNBP con** $T = 9{,}892$ **N o presión de aire a** 60 **psi.**
- **Algoritmo NNNE con** $T = 9{,}892$ **N o presión de aire a** 60 **psi.**
- **Algoritmo NNNE con** $T = 5{,}525$ **N o presión de aire a** 40 **psi.**
- **Algoritmo PSO con** $T = 5{,}525$ **N o presión de aire a** 40 **psi.**

Se considera la configuración de un cohete en ascenso recto, imponiendo los ángulos $\theta = 0$ y $\psi = 0$ como *Set Point* para cada controlador. Una vez seleccionadas las simulaciones, los valores de k_p, k_i, y k_d son ingresados al programa del microcontrolador del banco de pruebas que se encuentra en el Anexo A y puesto en marcha. Mientras el cohete del banco trata de estabilizarse, se registran todos los datos de actitud para su posterior análisis.

Fig. 19. Montaje Banco de Pruebas.

V. RESULTADOS

A. Etapa 1

En esta fase, se utilizan las características constructivas y operacionales del cohete de prueba, detalladas en la Tabla I, para alimentar el modelo matemático deducido y obtener así información detallada sobre la dinámica del cohete en la ausencia de un controlador, es decir, su respuesta en lazo abierto. La gráfica 20 presenta de manera visual la respuesta obtenida, donde se observa un incremento constante del ángulo θ debido a una perturbación.

B. Etapa 2

Para esta etapa, se empleó la información del modelo matemático desarrollado en la etapa 1. Además, se incorporó una función de transferencia de primer orden con una constante de tiempo de 0.01 segundos, con el propósito de simular el comportamiento del servomotor encargado de generar la acción de control y que ésta no se dé de manera instantánea, ecuación (50). El diagrama de bloques en simulink que representa el sistema de control se puede apreciar en la Figura 21.

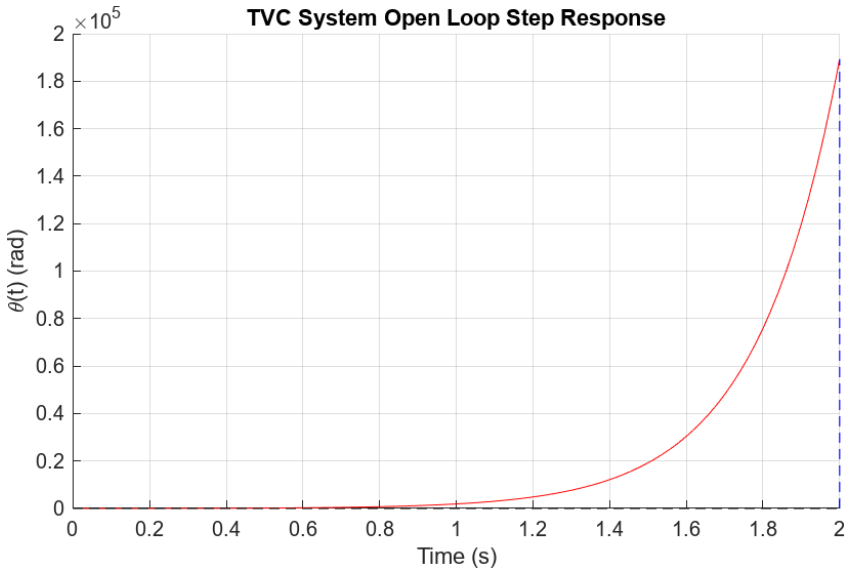

Fig. 20. Respuesta en Lazo Abierto del Sistema.

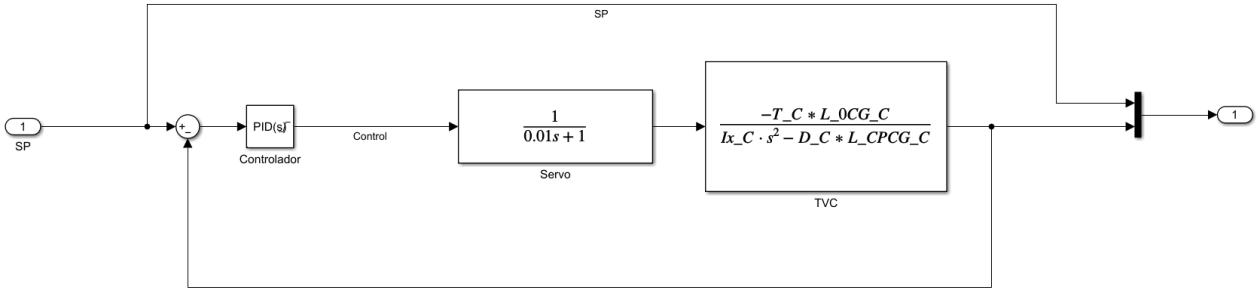

Fig. 21. Esquema del Sistema de Control en Simulink para cada uno de los Ejes de Rotación.

$$\frac{\theta(s)}{V(s)} = \frac{1}{0{,}01s + 1} \tag{50}$$

El bloque *Controlador* representa el tipo de controlador utilizado en el sistema, que en este caso es un **PID**. Dado el diseño de un TVC tipo *gimbal*, es necesario que la acción de

control generada por el controlador esté acotada. Para lograr esto, se configura una saturación en el bloque, limitando su movimiento al rango de -10° a 10° para cada eje. Asimismo, con el fin de evitar la acumulación de error en la parte integral del controlador debido a la saturación impuesta, se implementa un *antiwindup*. Este término se refiere a procedimientos que previenen este tipo de error y, en este caso, se utiliza el método de *clamping* [31]. Este bloque controlador está conectado en serie con la función de transferencia del servomotor, como se muestra en la ecuación (50). A continuación, se encuentra conectado a otro bloque que incorpora la función de transferencia del Thrust Vectoring Control System, derivada del modelo matemático establecido en las ecuaciones (41) y (44).

La sintonización del controlador se llevó a cabo mediante diversos algoritmos de inteligencia artificial, detallados en la sección de Marco Teórico. Estos incluyen **Algoritmos Genéticos** (Genetic Algorithms - GA), **Redes Neuronales con Retropropagación** (Neural Networks - Back Propagation), **Redes Neuronales con Neuro-Evolución** (Neural Networks - Neuroevolution) y **Optimización por Enjambre de Partículas** (Particle Swarm Optimization - PSO). Todos estos algoritmos se encontraron en repositorios de código abierto [32] y [33], fueron implementados en lenguaje M (MATLAB) y adaptados para la sintonización específica del Thrust Vectoring Control System (TVC).

Tras la reestructuración de los algoritmos, se desarrolló una interfaz gráfica que simplifica la introducción de datos y la sintonización de los controladores. A continuación, se presentan una serie de imágenes que ilustran el uso de esta interfaz gráfica. Para este ejemplo, se emplean los datos detallados en la Tabla I que corresponde a un cohete ejemplo.

En la Figura 22, se observan los campos habilitados para el ingreso de los datos constructivos y operativos del cohete de ejemplo. Además, en la parte derecha de la interfaz se muestra la respuesta en lazo abierto del sistema, que incluye el modelo del servomotor, como también se muestra en la Figura 20. Para la mayoría de los conjuntos de datos introducidos, se observa un comportamiento similar al que se ve en dicha Figura.

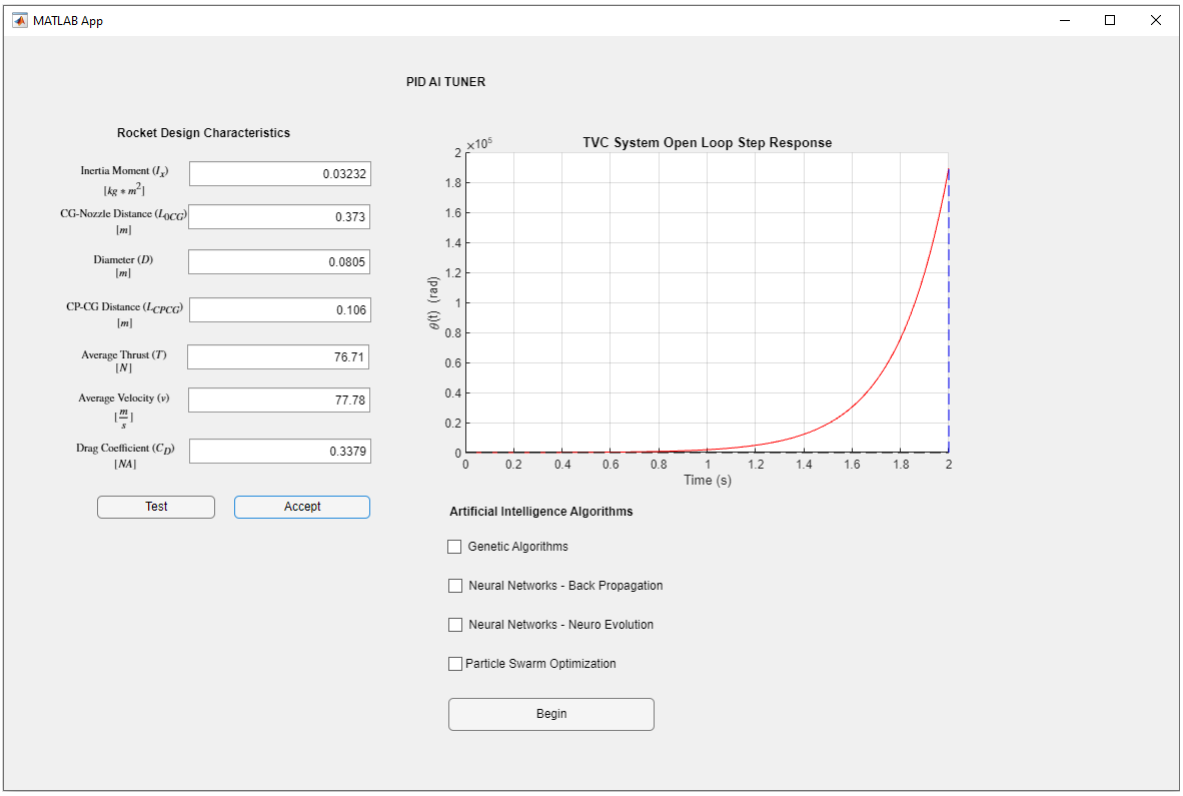

Fig. 22. Ventana Principal de la Interfaz Gráfica.

1) Algoritmos Genéticos Una vez ingresados los datos y obtenida la gráfica de lazo abierto, se puede escoger cuál o cuáles métodos se desean implementar para obtener la sintonización del control. Si se marca la casilla de "*Genetic Algorithms*" y al presionar el botón "*Begin*" se abrirá la ventana que se muestra en la Figura 23, en esta se pueden configurar los parámetros del algoritmo genético y a su vez se obtienen las gráficas del modelo de control sintonizado junto con la evolución de *fitness* en cada iteración. A continuación se muestra el listado de los parámetros a configurar para esta sintonización y su respectiva definición.

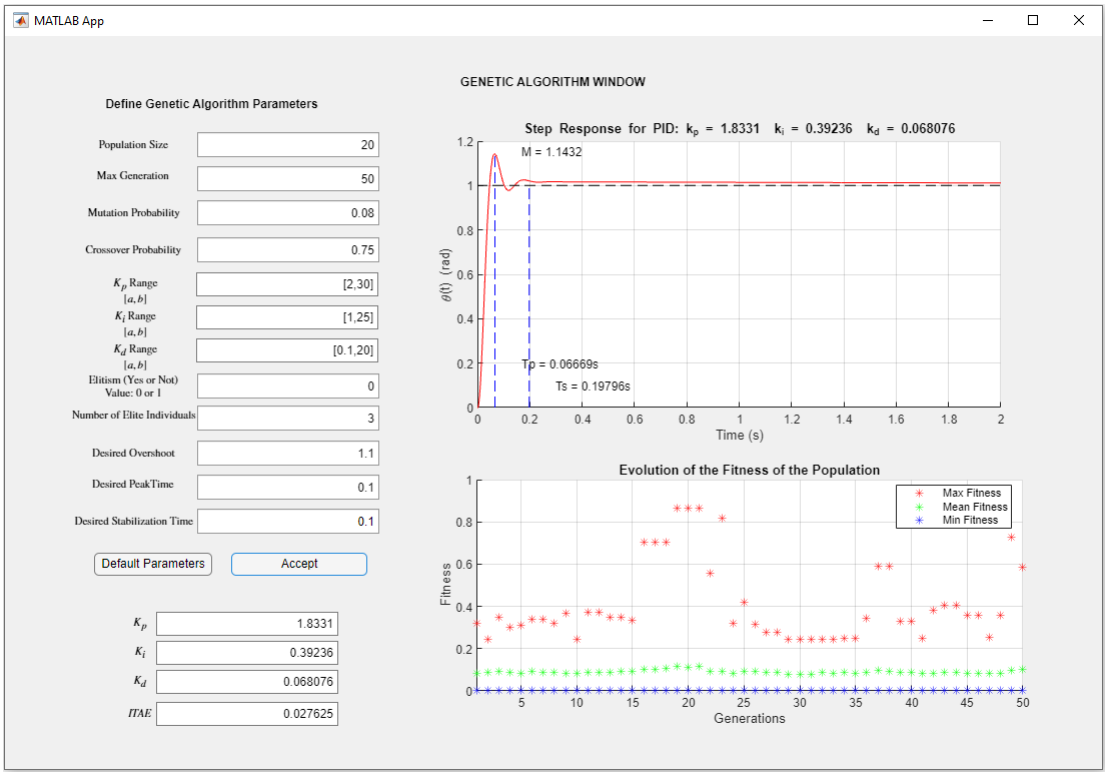

Fig. 23. Ventana de Algoritmos Genéticos en la Interfaz Gráfica.

- **Population Size:** Tamaño de la población en una generación dada.
- **Max Generation:** El número de generación que serán procesadas en el algoritmo.
- **Mutation Probability:** La probabilidad de mutación de un alelo.
- **Crossover Probability:** Probabilidad de que ocurra una operación cruzada.
- **kX Range:** El rango donde haya posibles opciones para codificar un determinado cromosoma ganador donde ”X”podría ser ”p, i o d”.
- **Elitism:** El elitismo se refiere a proteger a los individuos más aptos y a una población de operaciones de cruce y mutación a medida que avanza cada generación. Este parámetro se configura en 1 o 0 dependiendo de si se quiere ejecutar el algoritmo con este enfoque o no respectivamente.
- **Number of Elite Individuals:** El número de individuos en una generación que desea proteger de operaciones de cruce y mutación.

- **Desired Overshoot (M):** Se trata del sobreimpulso deseado para el control, se refiere al exceso inicial de la acción de control en comparación con el valor deseado o *set point*. En la gráfica de la Figura 23, este sobreimpulso se visualiza como el primer pico en la curva de color rojo. En el algoritmo, este valor está preconfigurado por defecto en 1.1, lo que representa un sobreimpulso deseado del 10 %. Sin embargo, en la gráfica observamos que el valor obtenido fue de 1.1432, correspondiente a un sobreimpulso del 14.32 %.
- **Desired Peak Time (T_p):** Se trata del tiempo al que se desea que se dé el *overshoot*, en el algoritmo está configurado por defecto como 0.1 segundos, debido a que para un TVC se necesita una respuesta rápida del sistema. Sin embargo, el valor obtenido fue de 0.0667 segundos, un poco más de la mitad del valor deseado.
- **Desired Stabilization Time (T_s):** Se trata del tiempo de estabilización, este es el tiempo en el que se desea que el sistema de control consiga alcanzar el *set-point* de manera estable. En el algoritmo está configurado por defecto como 0.1 segundos y el valor obtenido por GA fue de 0.198 segundos, casi el doble del valor deseado.

Finalmente, se ubican tres recuadros en la parte inferior izquierda. Estos recuadros contienen las constantes de sintonización k_p, k_i, y k_d correspondientes al modelo ingresado. De esta manera, estas constantes pueden ser copiadas fácilmente para su aplicación y prueba con otras herramientas, ya sean físicos o virtuales.

2) Redes Neuronales - Retro-propagación En la Figura 24 se muestra la ventana de la interfaz gráfica para la sintonización del controlador por medio de redes neuronales con el método de retro-propagación o *Back-Propagation* (BP). Durante la implementación de este método, se generan datos de sintonizaciones por medio del método GA previamente descrito, con el objetivo de entrenar una red neuronal. Por ende, los datos ingresados en la columna izquierda de la Figura 24 son los mismos usados para la implementación del método GA. La configuración para este método se encuentra en la columna derecha de la Figura 24, allí podemos encontrar los siguientes parámetros:

- **Total Data:** Tamaño total de los datos para entrenamiento.

- **Overshoot Range:** Rango para el sobreimpulso deseado, este rango se divide entre el tamaño total de los datos, en este caso 20. Cada uno de los valores en este rango se usa para realizar la sintonización y recolectarla, el algoritmo solamente almacena las sintonizaciones con un *fitness* mayor al 60 % para usarlos como datos de entrenamiento.
- **Peak Time Range:** Rango para el tiempo del sobreimpulso.
- **Stabilization Time Range:** Rango para el tiempo de estabilización.
- **Overshoot Test Data:** Datos de sobreimpulso que se usarán para testear el funcionamiento de la red neuronal obtenida.
- **PeakTime Test Data:** Datos de tiempo pico que se usarán para testear el funcionamiento de la red neuronal obtenida.
- **Stabilization Test Time:** Datos de tiempo de estabilización que se usarán para testear el funcionamiento de la red neuronal obtenida.

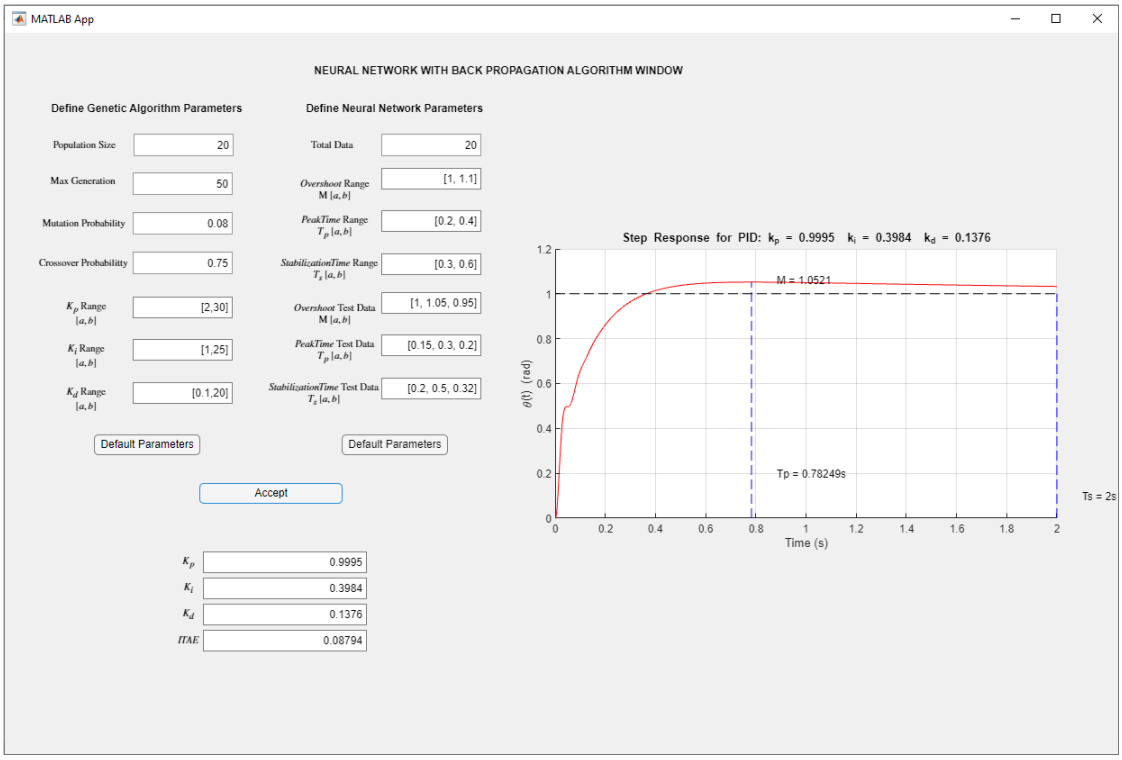

Fig. 24. Ventana de Redes Neuronales con Retro-Propagación en la Interfaz Gráfica

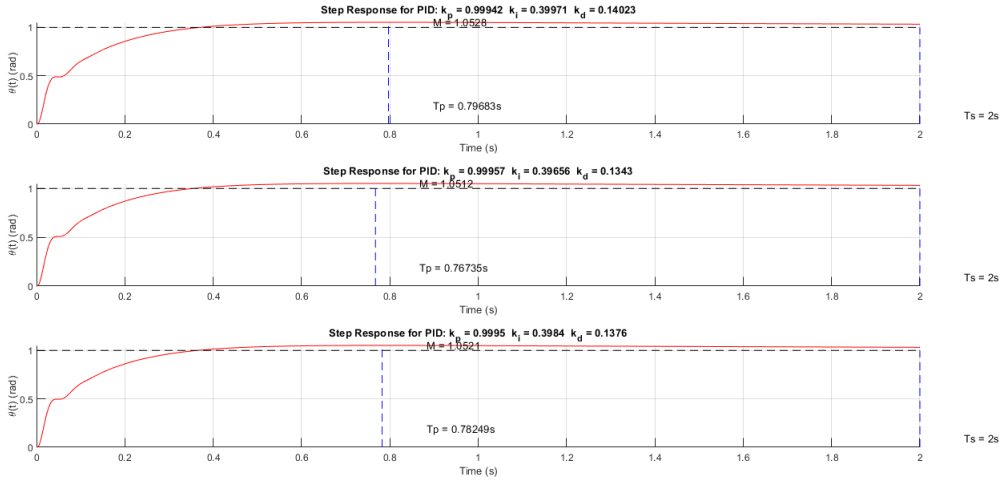

Fig. 25. Resultado de Sintonización para Test Data.

En la Figura 25, se presentan los resultados de los controladores obtenidos mediante la red neuronal, considerando como parámetros de prueba los valores $[M_i, T_{pi}, T_{si}]$ con $i = 1, 2, 3$, ingresados en los recuadros de la interfaz gráfica. Cabe destacar que este método, en comparación con todos los métodos implementados, es el que requiere más tiempo para producir un resultado. Esta demora se atribuye a las múltiples sintonizaciones necesarias para obtener datos suficientes para el entrenamiento de la red, pero una vez entrenada, ésta encuentra controladores adecuados rápidamente.

3) Redes Neuronales - Neuro-Evolución En esta sección se presentan los resultados de sintonización de una red neuronal, entrenada por el método de neuro-evolución. Como ya se mencionó en la sección del Marco Teórico, la neuro-evolución es un enfoque de optimización que combina principios de algoritmos evolutivos y aprendizaje de máquinas, específicamente para la sintonización y evolución de arquitecturas de redes neuronales. Los parámetros configurables para esta sección se pueden ver en la columna izquierda de la ventana de la interfaz.

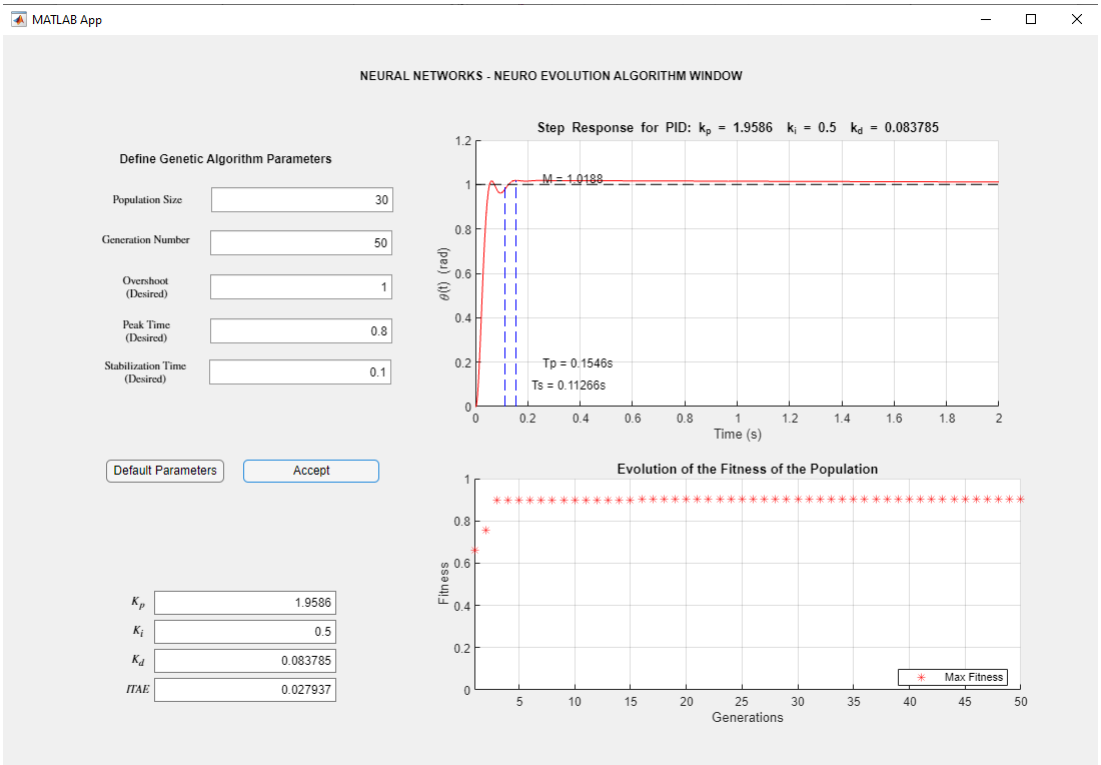

Fig. 26. Ventana de Redes Neuronales con Neuro-Evolución en la Interfaz Gráfica.

- **Population Size:** Población de datos para evolucionar.
- **Generations Number:** Número de generaciones.
- **Overshoot (Desired):** Sobreimpulso deseado.
- **Peak Time (Desired):** Tiempo pico deseado.
- **Stabilization Time (Desired):** Tiempo de estabilización deseado.

En la parte derecha de la interfaz gráfica, representada en la Figura 26, se exhibe el controlador resultante junto con sus características obtenidas durante el proceso de neuroevolución. Este método no solo busca ajustar los pesos de las conexiones neuronales, sino también la arquitectura misma de la red neuronal, permitiendo así una adaptación más completa y flexible a las complejidades del sistema de control.

La representación visual del controlador proporciona una comprensión intuitiva de cómo la red neuronal ha evolucionado para cumplir con los requisitos específicos de la tarea

de control. Además, la gráfica de evolución del *fitness*, visible en la misma ventana, traza la mejora progresiva del rendimiento del controlador a lo largo de las iteraciones.

4) Optimización por Enjambre de Partículas - PSO Por último, se presenta en la Figura 27 la interfaz de la implementación del método de optimización por enjambre de partículas (Particle Swarm Optimization - PSO). En este método, se adopta la función de costo que se muestra en la ecuación (51), determinante en la convergencia del algoritmo.

Nuevamente, en la columna izquierda se encuentran todos los parámetros que se pueden configurar para iniciar el algoritmo, más adelante se muestra una breve descripción de cada uno. En la parte superior derecha de la ventana, se muestra el comportamiento de la sintonización obtenida para el control, mientras que en la parte inferior se visualiza la evolución de la *Función de Costo* con respecto a cada iteración.

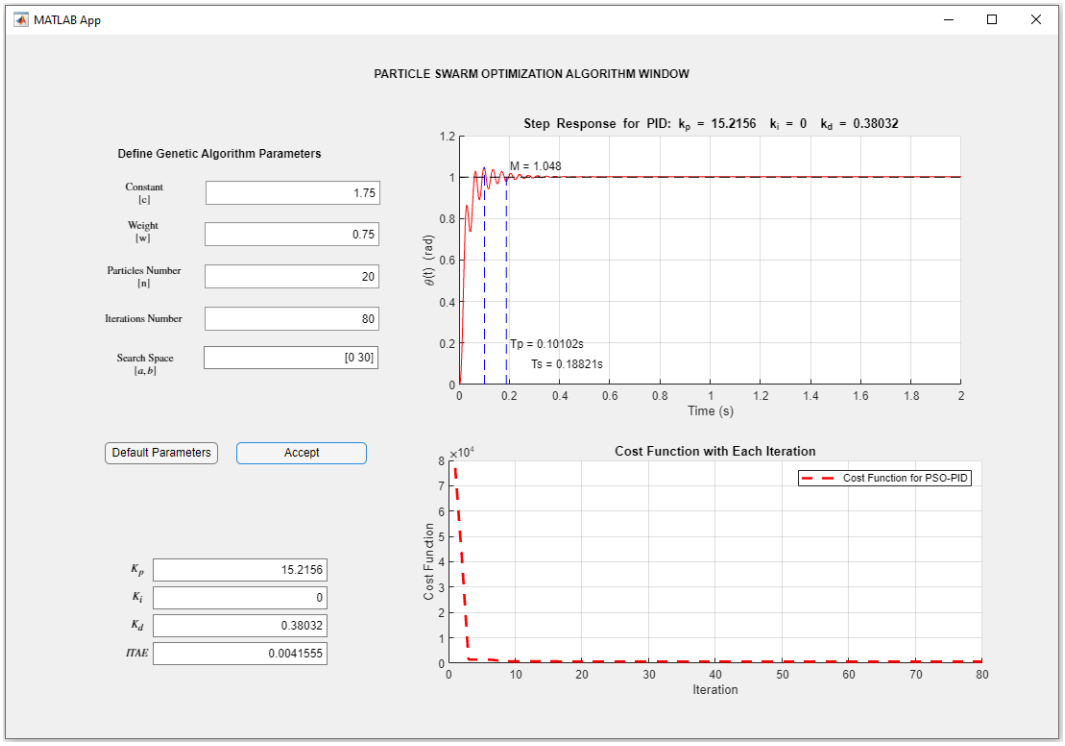

Fig. 27. Ventana de Optimización por Enjambre de Partículas en la Interfaz Gráfica.

- **Constant:** Coeficiente de aceleración cognitiva (influencia de la mejor posición local).

- **Weight:** Peso inercial (influencia de la velocidad actual).
- **Particles Number:** Número de partículas del enjambre.
- **Iterations Number:** Número de iteraciones.
- **Search Space:** Intervalo de búsqueda donde se esperan encontrar los valores de k_p, k_i, k_d adecuados para el control.

$$C.F = \sum_{i=1}^{N} |y_i - 1| * i \tag{51}$$

- **C.F:** Función de costo.
- **y:** Variable a controlar.

5) Comparación entre Varios Diseños de Cohete En esta sección, se procede a realizar nuevamente pruebas de sintonización utilizando los distintos métodos, pero en esta ocasión se emplean las características de diseño de los cohetes "Comercial", "Martin" y "Prueba".

Para llevar a cabo esta comparación, se creó el diagrama en *Simulink*, el cual se puede observar en la Figura 28. En este diagrama, se aplicó a cada sintonización a un valor de referencia (SP) de 1 grado de actitud para el cohete y se simuló a partir de las gráficas, esta simulación puede hacerse de 5 segundos con una perturbación a los 2 segundos. Se destaca un subsistema específico con un recuadro rojo, este se amplía en la Figura 29.

En este subsistema, se integran los bloques del controlador, el servomotor y el modelo. Además, se agregaron un par de bloques escalón que simulan una perturbación del 10 % en el segundo 2, con una duración de medio segundo. Esta perturbación se ha colocado en la salida del controlador para simular un cambio inesperado en el ángulo de la tobera, generando a su vez una variación en la actitud del cohete. El propósito de esta configuración es simular el comportamiento del controlador durante situaciones de cambio brusco. Es importante destacar que esta configuración específica se aplicó de manera consistente a todos los modelos.

A continuación, se exhibirán varias imágenes con el propósito de realizar un análisis cualitativo del rendimiento de los controladores sintonizados mediante los algoritmos de inteligencia artificial utilizados, abordando cada modelo de cohete: Figura 30, Figura 32 y Figura

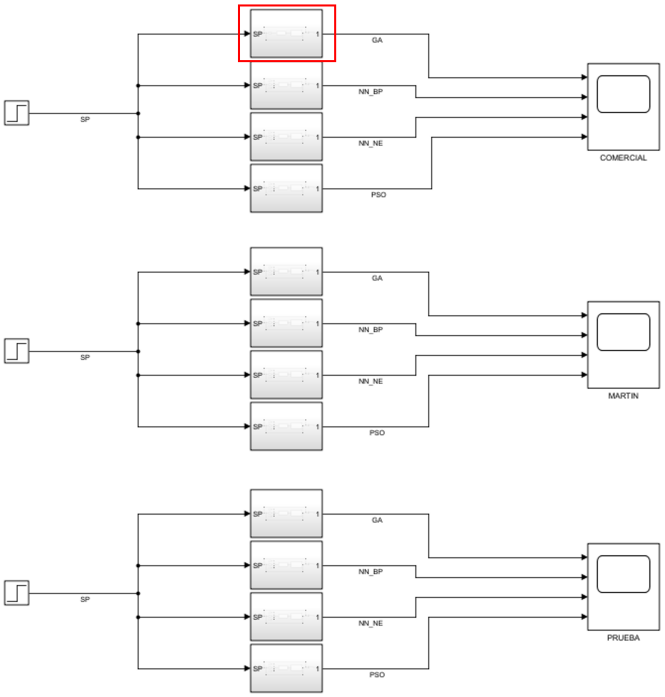

Fig. 28. Esquema en Simulink para Cohetes de Prueba.

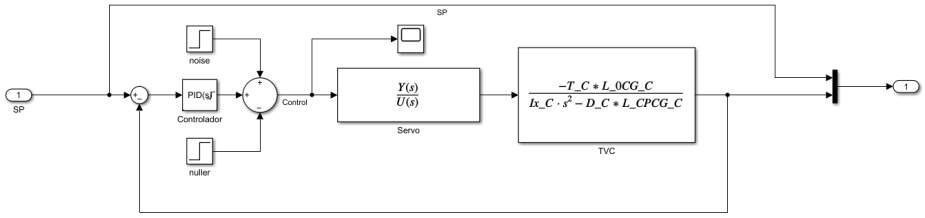

Fig. 29. Esquema en Simulink para Controlador con Perturbación.

34. También se proporcionan las imágenes de las acciones de control de cada controlador para posteriormente revisar su comportamiento: Figura 31, Figura 33 y Figura 35. Por último, en las Tablas IV, V y VI se presenta las constantes de sintonización para cada uno de los cohetes, se agrega una columna que corresponde al valor del *ITAE*.

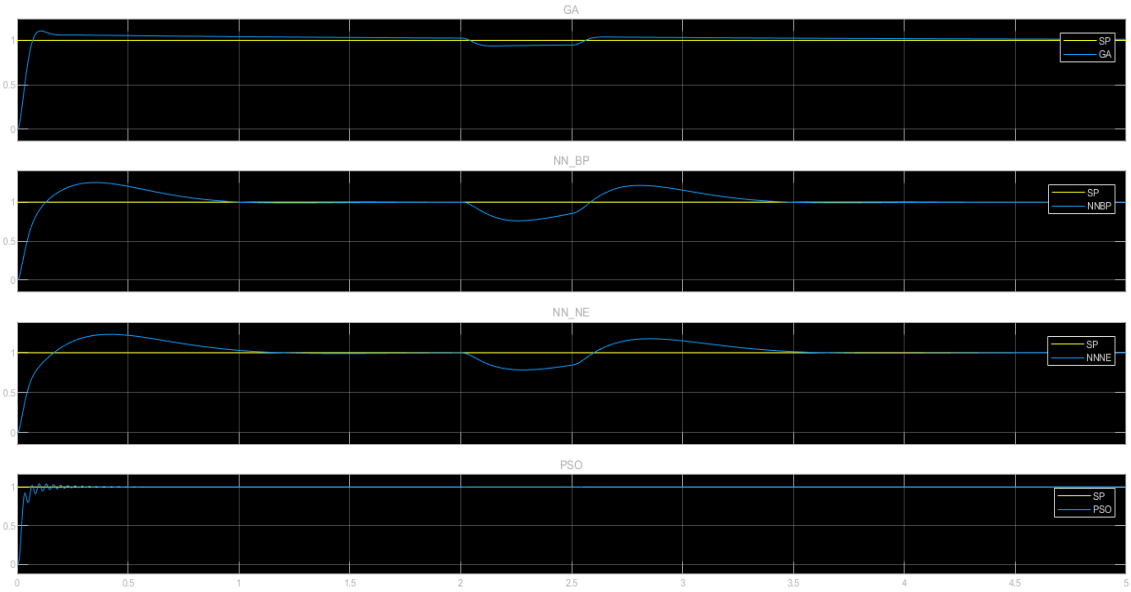

Fig. 30. Controladores con Cohete "Comercial".

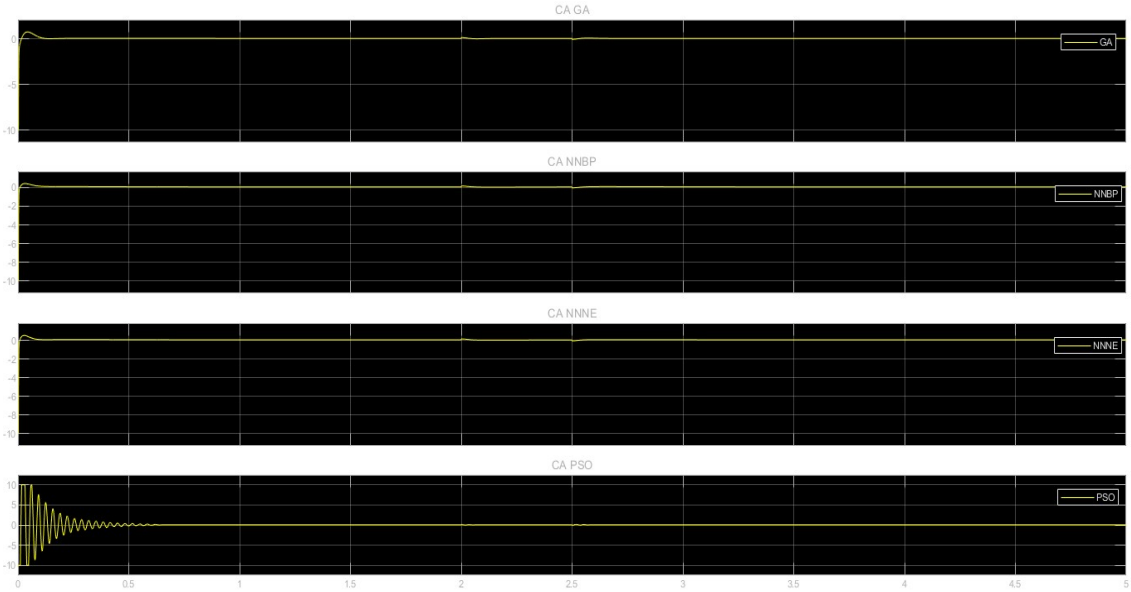

Fig. 31. Acción de Control para Controladores en Cohete "Comercial".

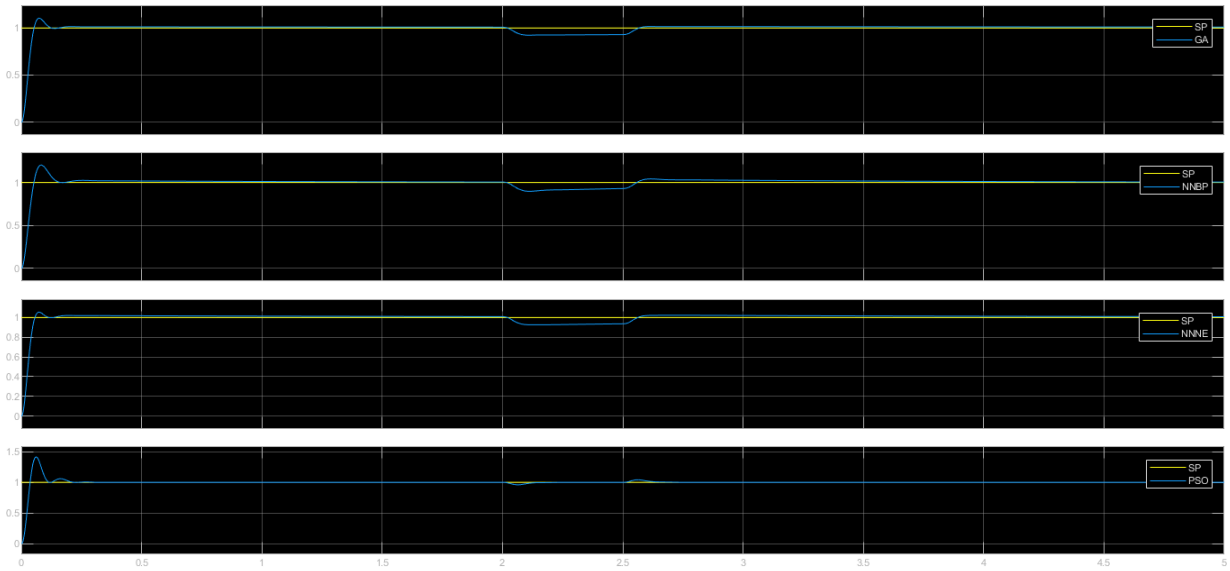

Fig. 32. Controladores con Cohete "Martin".

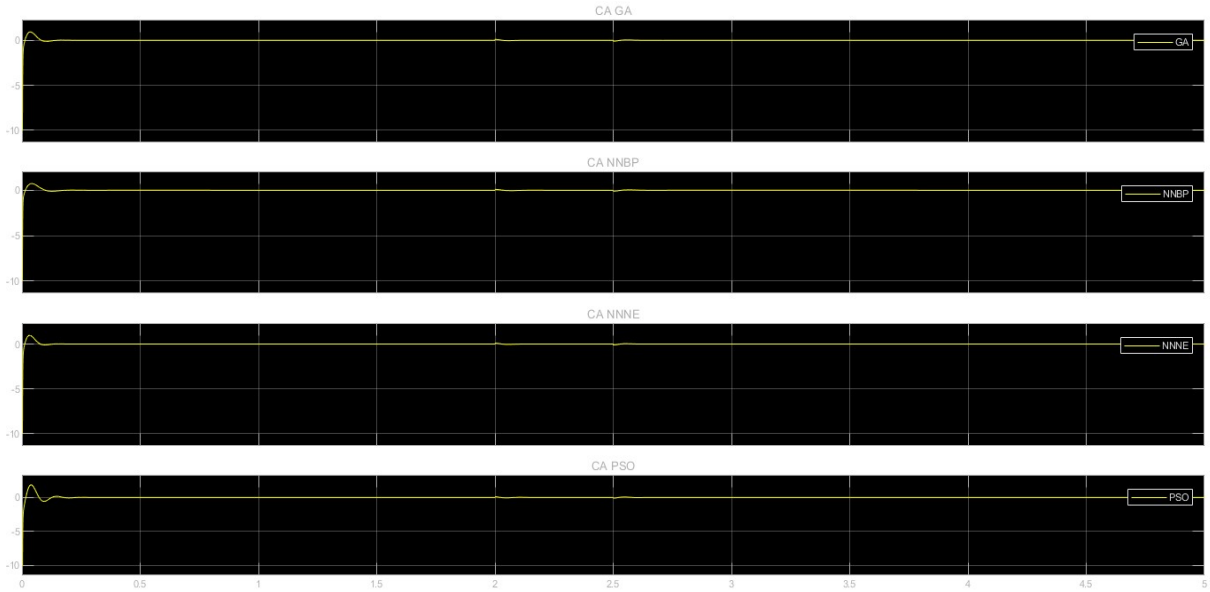

Fig. 33. Acción de Control para Controladores en Cohete "Martin".

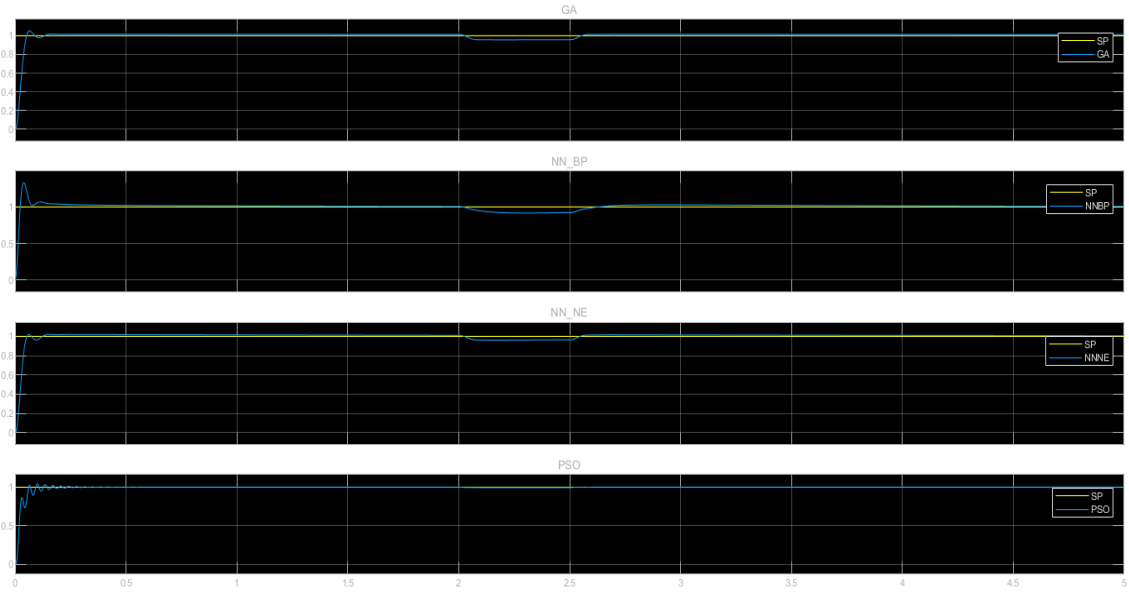

Fig. 34. Controladores con Cohete "Prueba".

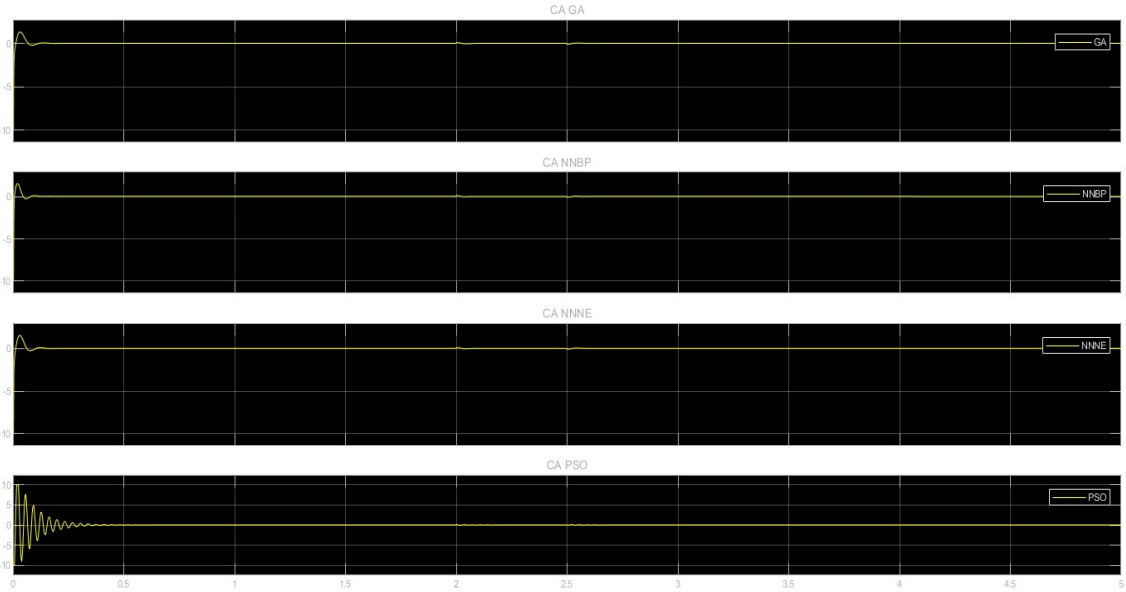

Fig. 35. Acción de Control para Controladores en Cohete "Prueba".

Algoritmo	k_p	k_i	k_p	$ITAE$
Genetic Algorithms	1.2058	0.5798	0.0667	0.3227
NN Back Propagation	0.3551	0.9543	0.0490	0.5793
NN Neuro Evolution	0.3810	0.9280	0.0583	0.5714
Particle Swarm Optimization	26.585	0.0000	0.6563	0.0265

TABLA IV

ITAE Y CONSTANTES DE CADA MÉTODO EN EL COHETE "COMERCIAL".

Algoritmo	k_p	k_i	k_p	$ITAE$
Genetic Algorithms	1.1930	0.2100	0.0552	0.2157
NN Back Propagation	0.9995	0.7374	0.0443	0.2519
NN Neuro Evolution	1.1867	0.5384	0.0595	0.2283
Particle Swarm Optimization	2.4762	29.899	0.0725	0.0177

TABLA V

ITAE Y CONSTANTES DE CADA MÉTODO EN EL COHETE "MARTIN".

Algoritmo	k_p	k_i	k_p	$ITAE$
Genetic Algorithms	1.7346	0.1922	0.0745	0.2037
NN Back Propagation	0.9995	0.6580	0.1043	0.2519
NN Neuro Evolution	1.9969	0.6811	0.0846	0.1704
Particle Swarm Optimization	15.235	0.0000	0.3817	0.0240

TABLA VI

ITAE Y CONSTANTES DE CADA MÉTODO COHETE "PRUEBA".

C. Etapa 3

1) Sintonización A continuación se presentan los resultados obtenidos para el cohete en el banco de pruebas, en las Figuras 36, 37, 38 y 39 y en las Tablas VII, VIII, IX y X.

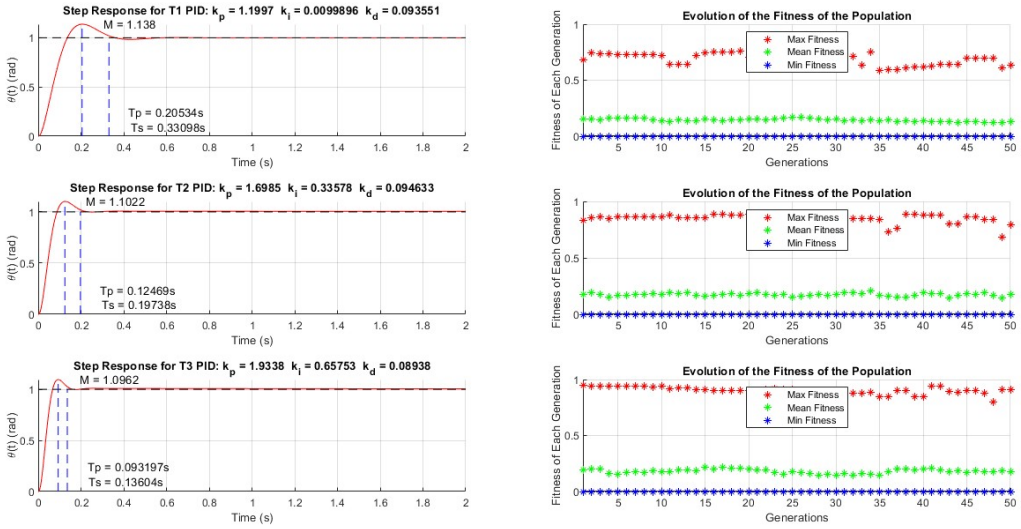

Fig. 36. Resultados para Cohete Banco - Algoritmos Genéticos.

Empuje	k_p	k_i	k_p	$ITAE$
T_1	1.1997	0.0100	0.0936	0.0088
T_2	1.6985	0.3358	0.0946	0.0131
T_3	1.9338	0.6575	0.0894	0.0140

TABLA VII

ITAE Y CONSTANTES PARA CADA EMPUJE USANDO GA.

Empuje	k_p	k_i	k_p	$ITAE$
T_1	0.9962	0.6041	0.0998	0.0467
T_2	0.9961	0.8508	0.0841	0.0367
T_3	0.7513	0.5134	0.0606	0.0287

TABLA VIII

ITAE Y CONSTANTES PARA CADA EMPUJE USANDO RETROPROPAGACIÓN.

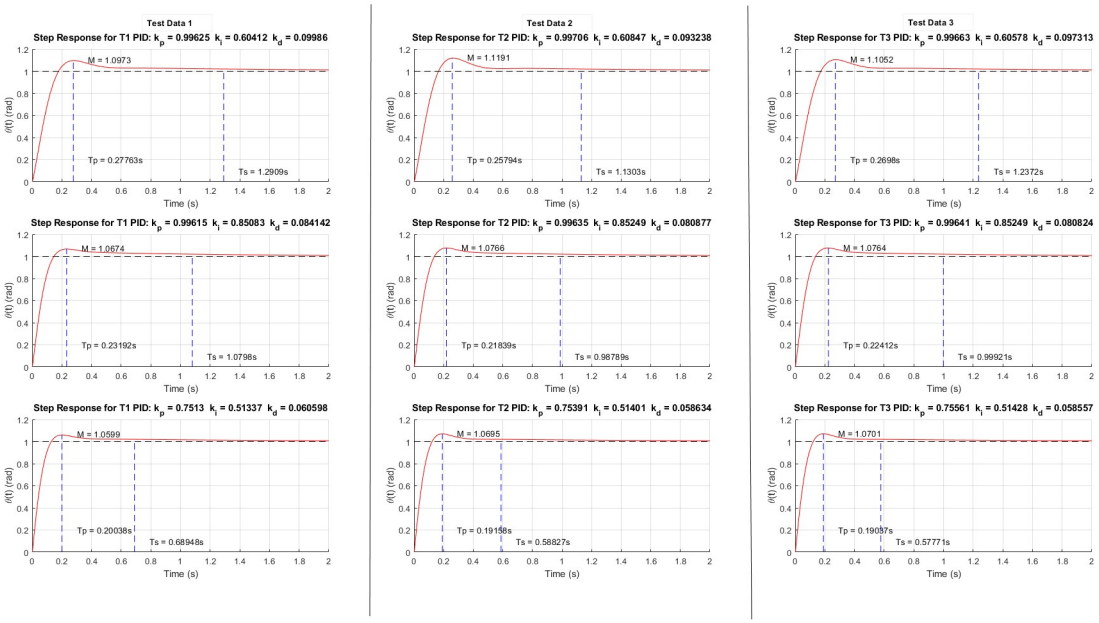

Fig. 37. Resultados para Cohete Banco - Retropropagación.

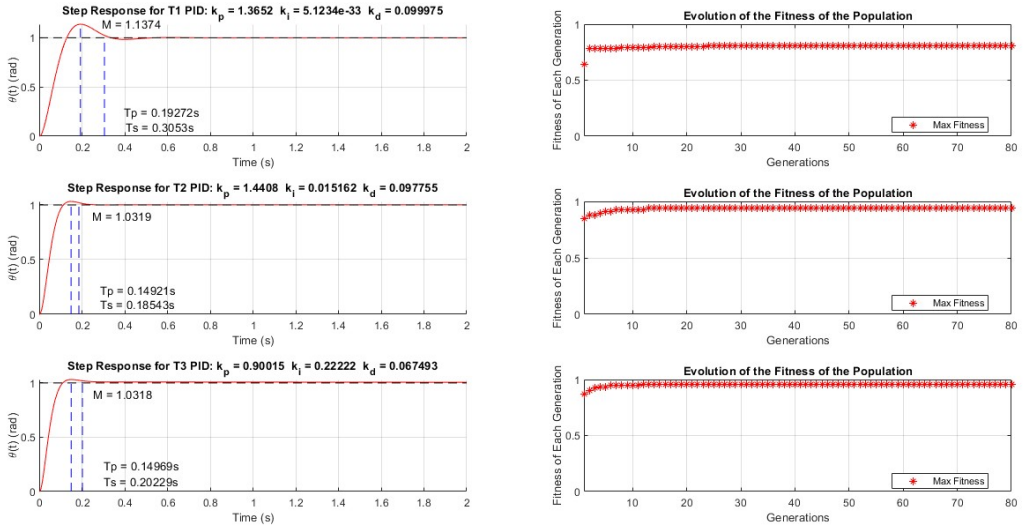

Fig. 38. Resultados para Cohete Banco - Neuroevolución.

Empuje	k_p	k_i	k_p	$ITAE$
T_1	1.3652	0.0000	0.1000	0.0072
T_2	1.4408	0.0152	0.0977	0.0029
T_3	0.9001	0.2222	0.0675	0.0171

TABLA IX

ITAE Y CONSTANTES PARA CADA EMPUJE USANDO NEURO EVOLUCIÓN.

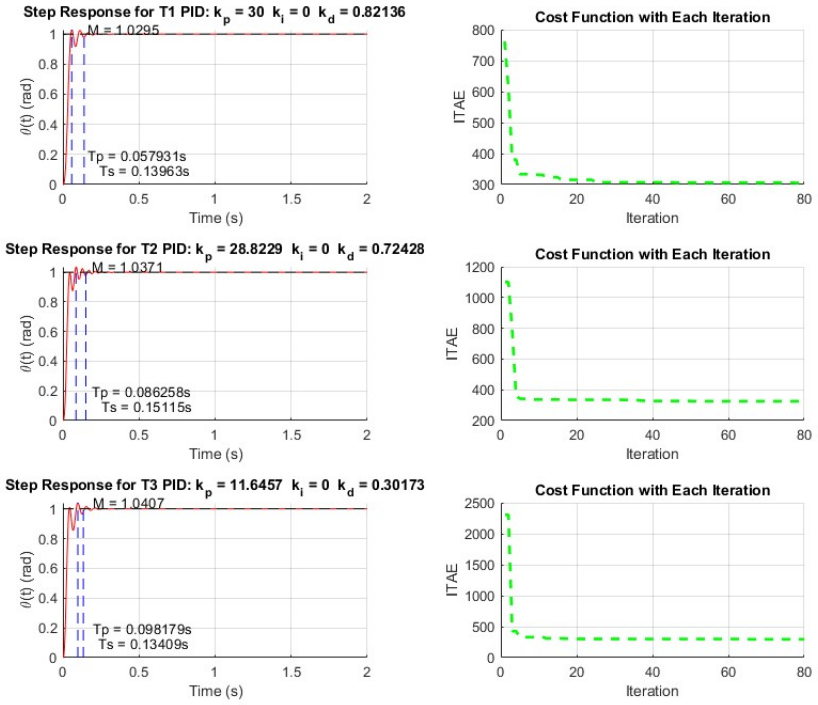

Fig. 39. Resultados para Cohete Banco - Enjambre de Partículas.

Empuje	k_p	k_i	k_p	$ITAE \times 10^{-4}$
T_1	30.000	0.0000	0.8214	8.9087
T_2	28.823	0.0000	0.7243	8.5633
T_3	11.646	0.0000	0.3017	8.2047

TABLA X
ITAE Y CONSTANTES PARA CADA EMPUJE USANDO ENJAMBRE DE PARTÍCULAS.

2) Implementación Conforme a lo detallado en la sección C de la metodología, a continuación, se presenta una breve descripción y justificación para cada una de las sintonizaciones utilizadas en el control del sistema en el banco de pruebas. Posteriormente, se proporcionan las imágenes que registran los datos recopilados por los sensores durante la ejecución de las pruebas.

- **PSO a 40 psi de presión:** La selección de este controlador se basa en su destacado rendimiento. Al revisar las tablas de resultados mostradas anteriormente en el apartado 1, se observa que el método PSO obtuvo los valores más bajos de ITAE en comparación al resto de los métodos, como se muestra en la Tabla X. En la Figura 40 se muestra el resultado del comportamiento de este controlador en un sistema real, gracias a los datos proporcionados por el banco de pruebas.
- **NNNE a 40 psi de presión:** En la misma línea de pensamiento, este controlador no solo exhibe el siguiente ITAE más bajo en comparación con los demás métodos, Tabla IX, sino que también presenta una gráfica de control con oscilaciones prácticamente nulas, ver Figura 38. Este factor adicional resalta su posible idoneidad, proporcionando una probabilidad de respuesta más estable y consistente en comparación con las demás alternativas consideradas. En la Figura 41 se muestra cualitativamente el desempeño de este controlador en el banco de pruebas.
- **NNNE a 60 psi de presión:** Este controlador es tenido en cuenta dado que cumple las dos condiciones mencionadas en los items anteriores, pero esta vez con una presión de 60 psi, el menor ITAE para este empuje, ver Tabla IX, y una gráfica con un comportamiento

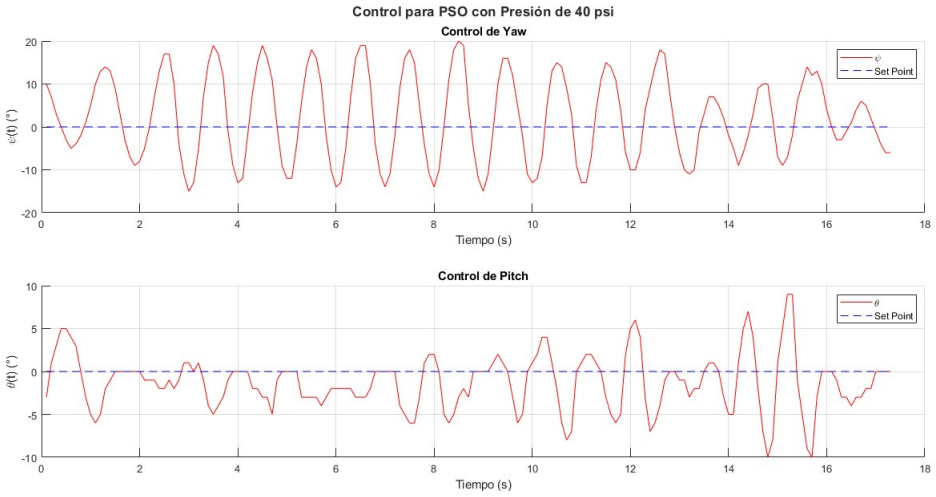

Fig. 40. Resultado de Implementación para Controlador Sintonizado con PSO 40 psi.

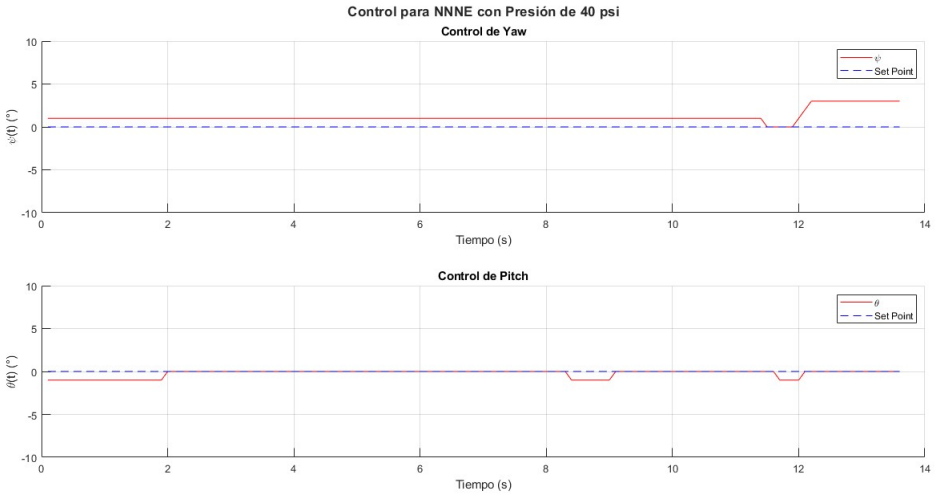

Fig. 41. Resultado de Implementación para Controlador Sintonizado con NNNE 40 psi.

muy estable como se vio anteriormente en la Figura 38. Nuevamente, en la Figura 42 se muestra el desempeño cualitativo de este controlador durante su uso en el banco de pruebas.

- **NNBP a 60 psi de presión:** Por último, el controlador obtenido por el método NNBP

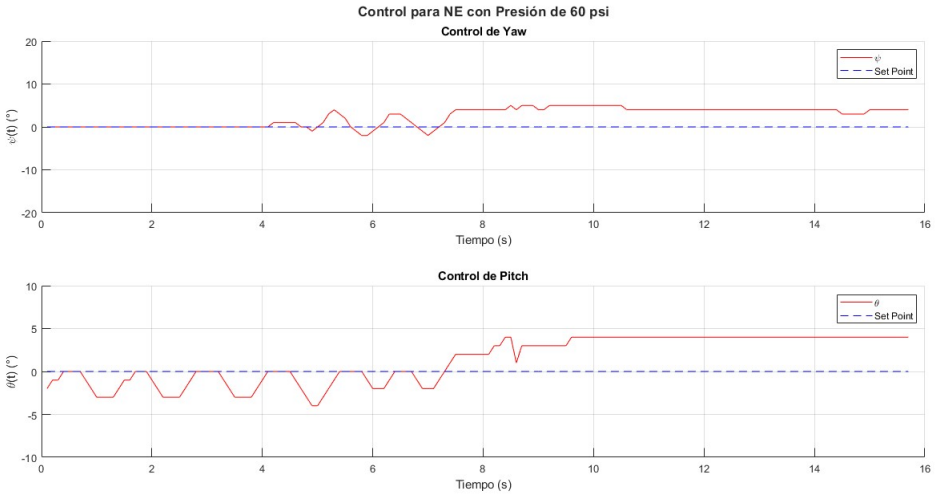

Fig. 42. Resultado de Implementación para Controlador Sintonizado con NNNE 60 psi.

es tenido en cuenta por encima del GA que presenta un ITAE menor, ya que el NNBP lo utiliza para entrenar su red neuronal, lográndose así una mayor variedad en la implantación y posibilitando resultados que se pueden estudiar desde más puntos de vista. En la Figura 43 se muestra el desempeño cualitativo de este controlador durante su uso en el banco de pruebas.

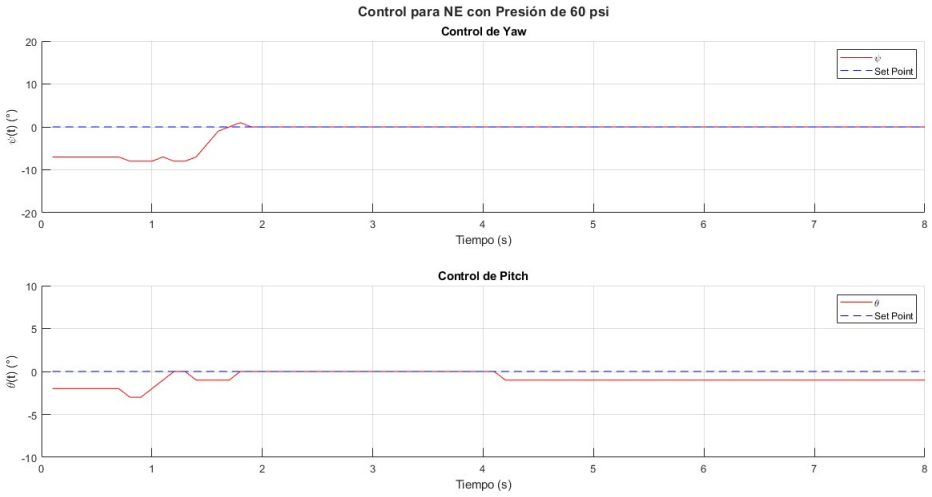

Fig. 43. Resultado de Implementación para Controlador Sintonizado con NNBP 60 psi.

VI. DISCUSIÓN

En esta sección se ofrece un análisis de los resultados obtenidos a lo largo de la investigación, brindando una oportunidad para interpretar, comparar y reflexionar sobre las implicaciones de los hallazgos. La implementación de algoritmos de inteligencia artificial en la sintonización de sistemas de control ha arrojado resultados significativos, y esta discusión se propone desentrañar las complejidades inherentes a dichos resultados. Se examinarán tanto los aspectos cualitativos como cuantitativos de los controladores sintonizados, considerando la eficiencia, estabilidad y adaptabilidad en diversas condiciones de operación. Además, se abordarán las limitaciones identificadas durante el estudio, proporcionando una visión equilibrada de los logros y desafíos encontrados. A medida que exploramos los detalles de la investigación, se busca ofrecer una comprensión más profunda de la aplicabilidad práctica y las posibles direcciones futuras en la integración de inteligencia artificial en el diseño y optimización de sistemas de control.

A. Etapa 1

Durante el desarrollo de esta etapa, se consideró el cohete como un objeto tridimensional con tres grados de libertad, representando la forma más general de modelar su actitud. Se tuvieron en cuenta únicamente dos fenómenos: los momentos generados por el sistema TVC y los momentos producidos por la fricción del cohete con el aire. Debido a las suposiciones realizadas al obtener el modelo, se descubrió que los movimientos tanto alrededor del eje "x" como del eje "y" están gobernados por funciones de transferencia idénticas, Ecuaciones 41 y 44. Por lo tanto, considerando todas las suposiciones descritas en esta etapa, se puede generalizar la Ecuación 52 para obtener la función de transferencia del control de la actitud alrededor de cualquier eje, teniendo en cuenta únicamente el momento de inercia correspondiente.

$$\frac{\alpha(s)}{\delta(s)} = \frac{-T * l_{0,CG}}{I_w * s^2 - D * l_{CP,CG}} \tag{52}$$

Donde:

- α*:* Ángulo de giro alrededor del eje w.
- δ**:** Ángulo de giro de la tobera alrededor del eje w.
- I_w**:** Momento de inercia alrededor del eje w.

B. Etapa 2

Esta etapa contiene todo lo referente a la interfaz gráfica desarrollada y a las simulaciones realizadas con diferentes diseños de cohete junto con sus comparaciones.

1) Cohete Ejemplo:

- **Algoritmos Genéticos:** En primer lugar, es relevante analizar los resultados obtenidos por el cohete de ejemplo, cuyas características se presentan en la Tabla I. En la sección 1 de los resultados se muestra la sintonización del controlador realizada mediante el método de Algoritmos Genéticos, como se observa en la Figura 23. Se aprecia que para el cohete

de prueba, este método logra una sintonización adecuada, con un tiempo de estabilización $T_s = 0{,}19796$ s, un sobre-impulso $M = 1{,}1432$ grados, un tiempo pico $T_p = 0{,}06669$ s y un valor de $ITAE = 0{,}0276$. Además, al analizar la evolución del *fitness* o *"adecuación"*, se observa que, generación tras generación, hay bastante variabilidad en este parámetro, y algunos de elementos lograron superar el 75 % de *fitness*.

- **Retro-Propagación:** Para este método, los resultados de sintonización también son adecuados, a pesar de que tiene un tiempo de estabilización $T_s = 2$ s, un sobreimpulso $M = 1{,}0521$ grados, un tiempo pico $T_p = 0{,}78249$ s y un valor de $ITAE = 0{,}08794$. Es importante señalar que (T_s) es bastante alto para esta aplicación. Además, según la gráfica de control en la Figura 24, incluso en ese tiempo no se ha alcanzado completamente el ángulo deseado (SP) pero se encuentra cerca de él. También es necesario mencionar que, dado que este método utiliza GA para crear el conjunto de datos con el que se entrenará, es el que más tiempo tarda en comparación con los demás.
- **Neuro-Evolución:** Para este método, los resultados de sintonización son adecuados, con un tiempo de estabilización $T_s = 0{,}11266$ s, un sobreimpulso $M = 1{,}01188$ grados, un tiempo pico $T_p = 0{,}37201$ s y un valor de $ITAE = 0{,}02793$. Como se observa en la Figura 26, un detalle que favorece a los resultados de *NNNE* sobre los anteriores es que durante la simulación no se presentaron oscilaciones muy pronunciadas en el control, aparte del sobreimpulso el cual es el mínimo hasta el momento. Además, la gráfica de fitness muestra resultados bastante buenos, la mayoría superiores al 70 % durante cada generación. Este método también presenta la ventaja de ser relativamente rápido en su convergencia, muy similar al tiempo que tarda GA.
- **Optimización por Enjambre de partículas:** Los resultados de sintonización para este método son adecuados, con un tiempo de estabilización $T_s = 0{,}188221$ s, un sobreimpulso $M = 1{,}048$ grados, un tiempo pico $T_p = 0{,}10102$ s y un valor de $ITAE = 0{,}0041555$. De todos los métodos evaluados, este presentó los tiempos de respuesta e $ITAE$ más bajos, aunque también fue el método con mayor cantidad de oscilaciones para alcanzar el *SP*. La Figura 27 muestra la gráfica de control y la evolución del parámetro utilizado para la convergencia de este método, la función de costo; en este caso, la convergencia se da debido

a la falta de cambio en este parámetro.

2) Comparación Entre los Métodos - Cohete Ejemplo: En esta subsección, se busca comparar los distintos métodos implementados anteriormente utilizando el mismo cohete de prueba, cuyas características se encuentran consignadas en la Tabla I. En la Figura 44, se pueden apreciar las gráficas de los controladores superpuestas durante el primer intervalo de tiempo de 2 segundos. En la gráfica, cada controlador se discrimina con un color diferente para facilitar la diferenciación y lograr una apreciación cualitativa. El comportamiento visualmente más destacado corresponde al controlador sintonizado mediante el método *NNBP*. Este muestra una imprecisión notable en comparación con los demás, ya que se aleja más del valor de referencia *SP* durante los dos segundos de simulación. Otro comportamiento notable es el generado por el método *PSO*, caracterizado por oscilaciones significativas hasta lograr estabilizarse en el *SP* de manera estable. Por otro lado, el método *GA* logra aproximarse rápidamente al *SP*, aunque con un sobreimpulso relativamente alto en comparación con los demás métodos. El método *NNNE*, a pesar de tener el menor sobreimpulso, no logra acercarse lo suficiente al *SP* como para considerarse superior en comparación con los otros métodos.

En la Tabla XII, se lleva a cabo una comparación del rendimiento obtenido por cada controlador. Para este análisis, se utilizan los parámetros previamente mencionados: T_s, T_p y M, a los cuales se les agrega el parámetro de desempeño *ITAE*. Al examinar la Tabla XII, se observa que, a excepción del sobreimpulso, el algoritmo *PSO* demostró ser superior en los demás aspectos para este modelo de cohete. Al analizar los valores de las constantes en la Tabla XI, notamos que los métodos *GA*, *NNBP* y *NNNE* tienen valores en el intervalo [0,2], mientras que *PSO* presenta valores en [0,16]. Esto implica que el control de *PSO* tiene una influencia más significativa en el sistema.

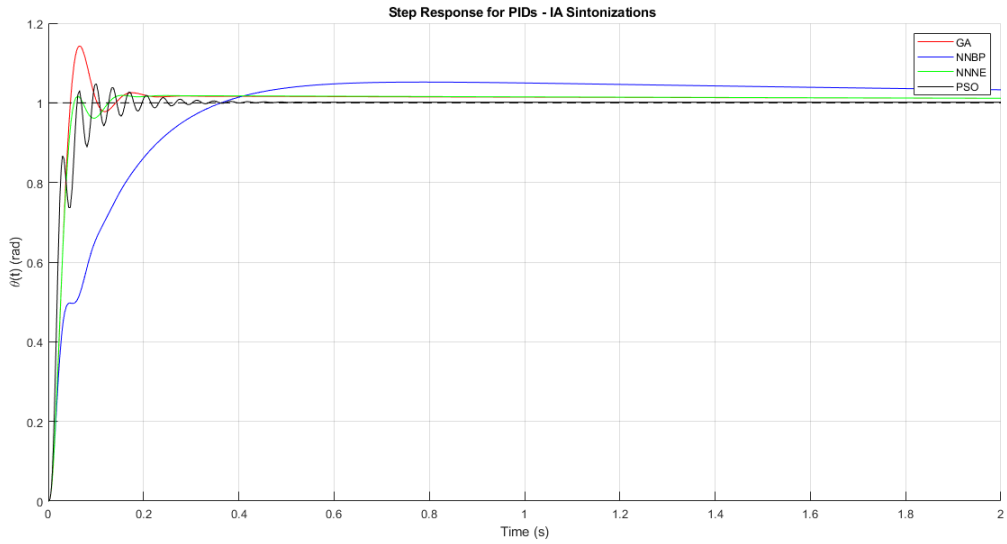

Fig. 44. Comparación del Control de Todos los Métodos.

Algorithm	k_p	k_i	k_d
Genetic Algorithms	1.8331	0.3924	0.0681
NN Back Propagation	0.9995	0.3984	0.1376
NN Neuro Evolution	1.9586	0.5000	0.0838
Particle Swarm Optimization	15.216	0.0000	0.3803

TABLA XI
CONSTANTES OBTENIDAS CON CADA MÉTODO.

3) Comparación entre Varios Diseños En esta etapa, se emplea Simulink para simular el comportamiento de los controladores sintonizados en tres diseños de cohetes distintos, como se detalla en la Tabla II. Es importante destacar que, en estas simulaciones, el valor del SP se mantiene en 1 grado, y además, se introduce una perturbación del 10 % en $t = 2$ s, con una duración de 0,5 s.

- **Cohete Comercial:** Al analizar los resultados de cada controlador en la Figura 30, se destaca, en primer lugar, la dificultad que tienen los controladores para estabilizar este

Algorithms	M [°]	T_p[s]	T_s [s]	ITAE
Genetic Algorithms	1.1432	0.0667	0.1979	0.0276
NN Back Propagation	1.0521	0.7824	2.0000	0.0879
NN Neuro Evolution	1.0188	0.1546	0.1126	0.0279
Particle Swarm Optimization	1.0480	0.1010	0.1882	0.0041

TABLA XII
COMPARACIÓN ENTRE LOS DISTINTOS MÉTODOS.

diseño, aparte del algoritmo *PSO* que logra estabilizarlo pero con un comportamiento mayormente oscilatorio. Este fenómeno podría atribuirse al hecho de que este modelo de cohete exhibe una mayor inestabilidad, considerando su margen de estabilidad y su momento de inercia relativamente pequeño. A pesar de la inestabilidad inherente del cohete, los algoritmos han logrado encontrar controladores capaces de mantenerlo equilibrado, incluso en presencia de la perturbación inducida. Al examinar la Tabla IV, se observa nuevamente que los tres primeros algoritmos convergen a sintonizaciones relativamente similares, mientras que *PSO* difiere significativamente de los demás. Esta discrepancia puede apreciarse también en la Figura 31, donde se presentan las acciones de control de cada controlador. Los métodos iniciales generan acciones de control relativamente parecidas, mientras que el algoritmo *PSO* requiere acciones de control más complejas y oscilantes para alcanzar el *SP*. No obstante, es interesante destacar que, a pesar de estas diferencias en la convergencia y las acciones de control, *PSO* demuestra una capacidad comparable para corregir la perturbación, comportándose de manera similar a los demás métodos en este aspecto.

- **Cohete Martín:** En este caso, nos encontramos con un diseño considerablemente más estable que el anterior, como se refleja en las gráficas de control en la Figura 32, ya que no muestran demasiadas oscilaciones. Nuevamente, todos los controladores logran mantener estable el cohete, pero la característica más interesante de este conjunto de gráficas se manifiesta en el momento de la perturbación. Tanto *GA* como *NNNE* como *NNBP* consiguen

corregir la perturbación con relativa rapidez y de manera similar; por último, *PSO* logra el mejor control en esta situación. Al revisar la Tabla V, observamos un comportamiento similar en las constantes, siendo relativamente similares para los tres primeros métodos, pero notoriamente diferentes para el algoritmo *PSO*, especialmente en la constante proporcional. También es importante analizar los resultados de la Figura 33, donde se presentan las acciones de control de los controladores configurados para el diseño de cohete "Martin". Predominantemente, se observa una similitud en los comportamientos de estas acciones de control. Incluso el algoritmo *PSO*, que en el diseño de cohete "Comercial" mostró una naturaleza oscilante, presenta un comportamiento notablemente diferente en este caso.

- **Cohete Prueba:** Por último, evaluamos el cohete "Prueba". Al analizar las gráficas presentadas en la Figura 34, se observa un comportamiento bastante similar al del cohete "Martin", aunque con un mejor control durante la perturbación por parte de los tres primeros métodos. En particular, para el controlador *PSO*, esta perturbación es casi imperceptible, aunque con la generación de oscilaciones muy pronunciadas para lograr el control. Las constantes resultantes exhiben un comportamiento similar al de los diseños anteriores, como se detalla en la Tabla VI. Al analizar las acciones de control de los controladores de cada método en la Figura 35, se evidencia un comportamiento similar al obtenido para el diseño "Comercial". Los tres primeros métodos exhiben un comportamiento parecido, mientras que para *PSO* se observa un comportamiento más complejo caracterizado por una gran cantidad de oscilaciones.

Para concluir los análisis realizados en esta subsección, observamos algunos hechos comunes en los resultados de los tres diseños de cohete: la similitud en el orden de las constantes para los tres primeros algoritmos (*GA*, *NNBP* y *NNNE*), el desempeño de cada controlador en el momento de la perturbación y los resultados del parámetro de desempeño *ITAE*.

La similitud entre los resultados de los métodos *GA* y *NNBP* es evidente, ya que la construcción de los datos para entrenar la red neuronal del segundo método se realizó utilizando el método *GA*. Sin embargo, la similitud con el método *NNNE* no es tan evidente. Esta similitud se debe a que la neuroevolución, en sus bases, utiliza elementos parecidos a

los de los algoritmos genéticos para entrenar su red neuronal, como se mostró en el apartado del marco teórico.

A su vez, el desempeño de cada método en el momento de la perturbación en $t = 2$ segundos es relativamente similar, en el caso método *PSO* es bastante diferente. Esto se debe a que el valor de las constantes obtenidas por este método, especialmente para la constante proporcional, tiene un valor más elevado.

Para concluir, al analizar el parámetro de desempeño *ITAE*, detallado en las tablas mostradas anteriormente, destaca que en todos los cohetes, el método PSO exhibió un *ITAE* considerablemente menor en comparación con los demás algoritmos. Esta disparidad notable se atribuye al hecho de que el método PSO tiende a converger hacia sintonizaciones con constantes de valores más altos. Aunque este comportamiento permite que el controlador tenga una mayor incidencia en el desempeño del sistema, se observa en las gráficas de las acciones de control un comportamiento oscilante que podría ser perjudicial en un montaje real. En consecuencia, para obtener un rendimiento mejorado, se sugiere explorar valores de constantes relativamente altos, o al menos ligeramente superiores a los obtenidos por los tres primeros métodos, pero evitando alcanzar magnitudes tan elevadas como las del algoritmo *PSO*. Este aspecto resalta la influencia significativa que tiene la magnitud de las constantes en la rapidez de la respuesta del controlador, siendo un factor clave para el rendimiento superior.

C. Etapa 3

1) Análisis de Controladores Encontrados en Simulación. Para llevar a cabo el análisis de esta etapa, se hace referencia a las secciones C de la fase de resultados y C de metodología. En estas se detalla el proceso de construcción y caracterización del cohete en el banco de pruebas para la implementación de los controladores. Una vez obtenidas las características detalladas en la Tabla III, se utilizan estos datos en la aplicación de algoritmos de inteligencia artificial para la sintonización de los controladores.

- **Algoritmos Genéticos:** Al revisar la Figura 36, en la parte izquierda se observa el des-

empeño de cada controlador de manera cualitativa. Cada controlador logra resultados adecuados, y conforme se aumenta el empuje, las características mostradas (M, T_p, y T_s) mejoran. Un patrón similar se refleja en las gráficas de error en la parte derecha de la imagen. Aunque el método logró mantener el índice de *fitness* en valores mayores al 60 % con todos los empujes, se observa que, al aumentar el empuje, el *fitness* es mayor y más estable a lo largo de las iteraciones, esto último cobra aún más peso al observar la disminución del parámetro ITAE conforme al aumento del empuje en la Tabla VII.

- **Retropropagación:** En la Figura 37, se presentan las gráficas de control generadas para cada empuje y conjunto de datos de prueba, utilizando los mismos conjuntos que se muestran en la Figura 24. En todas las gráficas, predomina el comportamiento de que los cambios en las características de diseño empleadas generan variaciones bastante pequeñas en los controladores obtenidos para cada conjunto. Al analizar los resultados plasmados en la Tabla VIII, se observa nuevamente que, a medida que crece el empuje, el ITAE disminuye. Sin embargo, es importante destacar que este ITAE es mayor que el obtenido para el caso de *GA*, lo que sugiere que, por el momento, este método se muestra superior. Además, es relevante señalar que este método requiere más tiempo debido a la necesidad de crear datos de entrenamiento, a diferencia del entrenamiento de la red y su prueba, que se realizan rápidamente.
- **Neuroevolución:** Al revisar la Figura 38, en la parte izquierda se presentan nuevamente las gráficas de control. A primera vista, se denota un comportamiento bastante similar al generado por *GA*, incluyendo la mejora en las características conforme aumenta el empuje. Sin embargo, se puede apreciar una diferencia menor en estas características deseadas para cada valor de empuje en comparación con los métodos anteriores. A pesar de que se presenta una mejora, no es demasiado significativa en comparación con las obtenidas para los demás valores de empuje. Este hecho se refleja cualitativamente en las gráficas de *fitness* del lado derecho de la Figura 38, ya que todas presentan una adecuación bastante alta, mayor al 75 % desde la primera generación. Cuantitativamente, este comportamiento se confirma en la Tabla IX con el análisis del parámetro *ITAE*.
- **Enjambre de Partículas:** Finalmente, se presentan los resultados del algoritmo de *PSO*.

Al examinar la Figura 39, en el lado izquierdo se aprecian las gráficas de los controladores obtenidos mediante este método, abarcando todos los valores de empuje. Se observa un comportamiento adecuado, especialmente en el aspecto de los tiempos de estabilización, aunque se evidencia una cantidad considerable de oscilaciones iniciales en comparación con los métodos anteriores. Al analizar las gráficas en la parte derecha de la figura, se nota una tendencia similar en la función de costo para cada sintonización de empuje. Asimismo, al revisar la Tabla X, se destaca que los valores de *ITAE* obtenidos son del orden de 10^{-4} y que la diferencia entre ellos es del orden de 5×10^{-5} en el parámetro con respecto a cada controlador.

2) Análisis de Datos Encontrados Durante la Implementación: En este apartado, se analizarán los resultados recogidos mediante los sensores del banco de pruebas destinado para la implementación. Para llevar a cabo este análisis, es crucial referirse especialmente a las subsecciones 1 y 2 descritas en el apartado de resultados.

- **PSO 40 psi:** Con base en los resultados presentados en la Figura 40, se evidencia que el controlador obtenido mediante el método PSO exhibe un rendimiento insatisfactorio tanto en el control del ángulo de *pitch* como en el ángulo de *yaw*. En el caso del control de pitch, se observa que el controlador genera oscilaciones de gran amplitud de manera continua, sin lograr alcanzar de manera estable el set point. Se experimenta una situación similar, aunque con amplitudes menores, en el control de yaw. Considerando estos hallazgos, a pesar de que los controladores hallados mediante el método PSO presentan los valores más bajos de ITAE, su implementación en una aplicación real puede verse limitada por la naturaleza predominantemente oscilatoria en un principio, como se ilustra en la Figura 39. Este comportamiento oscilatorio puede impedir un control satisfactorio en condiciones prácticas. Esta observación adquiere aún más relevancia al analizar nuevamente las gráficas de la acción de control, como se detalla en la sección 5 del apartado de resultados. Tanto para el cohete 'Comercial'como para el cohete 'Prueba', el controlador generado por el algoritmo PSO muestra acciones de control con numerosas oscilaciones, lo cual podría afectar la ejecución práctica del controlador. Por último, en la Tabla XIII se muestra cuan-

titativamente esta inestabilidad, evidenciando diferencias de varios órdenes de magnitud entre los controladores en simulación y en el caso real.

- **NNNE 40 psi:** Al examinar la Figura 41, que contiene los datos recopilados por los sensores del banco durante las pruebas para el controlador generado por el método *NNNE*, se destaca un rendimiento significativamente mejor en comparación con el método anterior, *PSO*. En el control del ángulo de pitch, a pesar de encontrarse cercano al set point la mayor parte del tiempo y lograr alcanzarlo en algunos momentos, es notable que logra mantenerse estable durante la mayoría de la prueba. Similarmente, el control del ángulo de yaw exhibe un comportamiento análogo, aunque consigue permanecer en el set point durante un periodo más prolongado en comparación con el control de yaw. Este comportamiento también se refleja en la Tabla XIII, donde se observa que el ITAE para el control de pitch es aproximadamente 10 veces menor que el del control de yaw. Este fenómeno podría atribuirse al diseño restrictivo del banco, que limita más el movimiento en pitch que en yaw.
- **NNNE 60 psi:** Nuevamente, se hace referencia al apartado de resultados para analizar el desempeño del controlador generado por el método *NNNE*. En la Figura 42, se observa cómo se desenvuelve en el banco de pruebas. El control para el movimiento de yaw se mantiene inicialmente bastante cercano o alineado con el set point, pero a medida que transcurre el tiempo, se aleja progresivamente. Por otro lado, en el caso del movimiento en pitch, comienza de manera inestable, acercándose al set point en un momento dado, para luego estabilizarse en un punto lejano al set point. Este comportamiento puede analizarse cuantitativamente en la Tabla XIII, donde los valores de ITAE para este caso son notablemente altos y relativamente similares. A pesar de haber logrado un control relativamente efectivo en el apartado anterior, en este caso, este método claramente no pudo obtener un controlador adecuado. Esta dificultad podría atribuirse al aumento de empuje, ya que las correcciones necesarias para las perturbaciones ambientales presentan un desafío más considerable para el controlador con una velocidad de reacción como la del obtenido. Lamentablemente, en este apartado, el único método que destaca sería el *PSO*, pero como se vio anteriormente, este no consigue un control adecuado debido a sus amplias oscilaciones

iniciales.

- **NNBP 60 psi:** Finalmente, se procede al análisis de un controlador obtenido mediante el método de retro-propagación para el empuje generado a esta presión. En la Figura 43 se presentan los resultados de los datos recolectados durante la ejecución de este método. Se observa que el controlador se desempeñó de manera similar en ambos ejes, inicialmente ligeramente alejado del set point y posteriormente mucho más cerca del mismo. En términos generales, este controlador logró un desempeño adecuado durante el tiempo que estuvo en el banco de pruebas. Este rendimiento se respalda aún más al revisar la Tabla XIII. Para el controlador del movimiento en yaw, se obtuvo el valor más bajo de *ITAE* en todas las pruebas de implementación, y para el movimiento en pitch también se logró un valor adecuado en comparación con sus predecesores.

Algoritmo	Presión [psi]	Ángulo	ITAE
PSO	40	ψ	1235.7
PSO	40	θ	428.70
NNNE	40	ψ	125.17
NNNE	40	θ	12.720
NNNE	60	ψ	419.14
NNNE	60	θ	392.85
NNBP	60	ψ	8.7750
NNBP	60	θ	25.380

TABLA XIII
ITAE PARA RESULTADOS DE IMPLEMENTACIÓN.

Se observan discrepancias significativas entre los resultados obtenidos durante la simulación y la implementación práctica para un mismo controlador. Estas diferencias abarcan varios órdenes de magnitud, particularmente en el ITAE y en las representaciones gráficas del comportamiento de los controladores. Estas disparidades se atribuyen a diversos factores, como los momentos generados por cables o conexiones externas, los momentos inducidos por el

propio movimiento de la tobera y los momentos producidos por la rigidez de la manguera que conduce el aire al sistema. Estos elementos, en conjunto, dieron lugar a una planta de control que difiere significativamente de la presentada teóricamente. Además, es crucial señalar la presencia de un error acumulativo en el programa del microcontrolador, el cual resultaba en el mal funcionamiento cuando se dejaba operar durante largos periodos de tiempo.

VII. CONCLUSIONES

En la presente sección de conclusiones, se efectúa una síntesis y reflexión, sobre los hallazgos más relevantes y los resultados alcanzados a lo largo de la investigación. Se sigue una estructura lógica, consonante con el orden establecido en las secciones precedentes de *Metodología*, *Resultados* y *Discusión*, las cuales se han subdividido en etapas numeradas (*Etapa 1*, *Etapa 2* y *Etapa 3*).

1. La formulación del modelo matemático utilizado en la ejecución de este proyecto, tanto en la etapa de simulaciones como en la etapa de implementación, carece de una estructura única. La configuración del modelo está fuertemente influenciada por la elección de la orientación de los ejes inerciales, tanto en relación con el marco del cuerpo como con la tierra y el viento.

2. En el proceso de búsqueda de un modelo matemático, resulta válido inicialmente presentar todas las suposiciones que se tomarán en cuenta, con el objetivo de evaluar la simplicidad o complejidad del modelo. En este caso, se partió de un modelo con seis grados de libertad; sin embargo, al examinar las ecuaciones, se llegó finalmente a simplificar el modelo a solo dos grados de libertad. En retrospectiva, podría haber sido más beneficioso comenzar con un modelo inicialmente más simple.

3. Los algoritmos de inteligencia artificial poseen el potencial para lograr un control adaptativo, ajustando el sistema durante su operación. Para un sistema de TVC, un requisito fundamental sería la velocidad de convergencia, especialmente en el contexto de la cohetería amateur. Entre los algoritmos presentados en este documento, el único que podría aproximarse a las velocidades de convergencia requeridas es el *NNBP*. Aunque su proceso de entrenamiento es prolongado, hacerlo de manera efectiva puede resultar en una sintonización rápida de controladores. En contraste, los demás métodos requirieron aproximadamente 15 minutos para alcanzar la convergencia, un tiempo que imposibilita un control adaptativo con las condiciones ya mencionadas.

4. De las sintonizaciones obtenidas mediante diversos métodos, se puede inferir que la magnitud de las constantes de sintonización incide directamente en el tiempo de estabilización del sistema de control, así como en el tipo de respuesta del mismo. En este escenario, se observó que constantes elevadas generaban respuestas oscilatorias durante ciertos intervalos de tiempo, lo cual resulta indeseable para un sistema con una estabilidad tan sensible como la de un cohete en vuelo, esto se corroboró con la implementación física del sistema. Además de lo anterior, es importante señalar que la utilización de constantes elevadas en un sistema presenta otra ventaja aparte de un tiempo de estabilización bajo: la corrección rápida de las perturbaciones. No obstante, se debe buscar un equilibrio en la magnitud de estas constantes. Este equilibrio permitirá que el sistema no exhiba un comportamiento oscilatorio y, al mismo tiempo, pueda reaccionar de manera ágil ante perturbaciones e ingreso de set points.

5. La velocidad de respuesta de los servomotores que controlan el TVC es un requisito fundamental en la implementación práctica. La velocidad de funcionamiento utilizada para las simulaciones en este documento es de 0.01 s, lo que ha permitido obtener resultados altamente satisfactorios en este apartado. Sin embargo, es crucial señalar que, en caso de que este tiempo de respuesta sea demasiado alto, los algoritmos de inteligencia artificial encuentran dificultades para converger, y el comportamiento del sistema físico difiere considerablemente de lo mostrado en las simulaciones.

6. El diseño de sistemas mecánicos complejos es un desafío significativo que va más allá de la mera aplicación de principios teóricos. La interacción de diversas fuerzas y condiciones físicas introduce una capa adicional de complejidad, dificultando la predicción precisa del comportamiento del sistema. La representación de estas complejidades en bancos de pruebas se convierte en una tarea ardua, ya que capturar cada variable y condición real se vuelve intrincado. Este proceso revela la necesidad imperativa de enfoques de diseño más avanzados y herramientas de simulación que puedan abordar la complejidad inherente de estos sistemas, permitiendo una comprensión más profunda y una mejora continua en su rendimiento.

VIII. RECOMENDACIONES

1. Dada la amplia gama de herramientas que Python ofrece para el manejo de datos y la creación de modelos de inteligencia artificial, sumado a los innumerables beneficios inherentes al software libre, se recomienda priorizar el uso de esta plataforma para futuros desarrollos. La versatilidad y comunidad activa respaldan la eficiencia y flexibilidad necesarias en el ámbito de la ciencia de datos y la inteligencia artificial, convirtiendo a Python en una elección estratégica para la implementación de proyectos innovadores.

2. Dado que se ha deducido un modelo tridimensional con seis grados de libertad, se abre la posibilidad de futuras investigaciones y proyectos de complejidad mucho más alta, así validar y fortalecer dicho modelo.

3. La integración de una unidad de almacenamiento en el banco de pruebas es indispensable para contrarrestar los momentos generados por cables y conexiones. Este elemento se vuelve esencial para mitigar las fuerzas inducidas por dichos elementos externos, mejorando así la estabilidad y precisión del sistema en su conjunto.

4. Un cohete diseñado para el banco, con una longitud mayor, podría generar momentos de magnitud superior a los generados por la manguera utilizada. Esto podría resultar en la desestimación de los momentos aportados por la manguera al banco de pruebas, al hacerlos prácticamente insignificantes en comparación.

5. A pesar de que el programa que operó el banco ya contaba con un filtro, mejorar este buscando suavizar los movimientos del sistema TVC, quizás mediante un filtro de Kalman y un antiwindup, podría potenciar tanto el rendimiento del banco como la validación de los datos proporcionados por la simulación.

6. Se recomienda integrar en el modelo trabajado los momentos generados internamente por el sistema. Esto no solo permitirá una representación más precisa y completa de las interacciones del sistema, sino que también contribuirá significativamente a la mejora

de la fiabilidad y realismo de los resultados obtenidos, proporcionando una base más sólida para la toma de decisiones y el análisis del rendimiento del sistema.

7. Se recomienda considerar la adopción de servomotores con un tiempo de reacción máximo de 0.03 segundos o menos para el control del TVC. En el caso específico de los servomotores *SG90* utilizados, se ha observado que su tiempo de reacción considerablemente alto ha tenido un impacto negativo en el rendimiento del sistema.

Bibliografía

[1] A. Salem, M. Moustafa, and M.E.Ammar, "Tuning PID Controllers Using Artificial Intelligence Techniques," *Asian Journal of Engineering and Technology*, vol. 2, no. 2, pp. 1–13, 2014, doi: http://dx.doi.org/10.21608/iceeng.2014.30365.

[2] M. R. O. Girón, *Lecciones de Física Mecánica 2*. Córdoba, España: Universidad de Cordoba, Departamento de Fisica Aplicada, 2006, vol. 2.

[3] NASA, "Rocket Rotations," `https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/rocket-rotations/`, Mayo 2021, accedido en abril de 2023.

[4] M. H. Kaplan, *Modern Spacecraft Dynamics Control*. Garden City, NY: Dover Publications, Inc., 2020.

[5] T. J. Ardakani, H. Alemi; Bridges, "Review of the 3-2-1 euler angles: a yaw-pitch-roll," Department of Mathematics, University of Surre, Guildford GU2 7XH UK, Abril 2010.

[6] C. Carpenter, *Aircraft Stability and Control*. Shrewsbury: Airlife, 1997.

[7] NASA, "Forces on a Rocket," `https://www.grc.nasa.gov/www/k-12/rocket/rotations.html`, Mayo 2021, accedido en abril de 2023.

[8] O. B. George P. Sutton, *Rocket Propulsion Elements*. Hoboken, New Jersey: John Wiley Sons, Inc, 2017.

[9] NASA, "Rocket Stability," `https://www.grc.nasa.gov/www/k-12/rocket/rktstab.html`, Mayo 2021, accedido en abril de 2023.

[10] L. Sopegno, P. Livreri, M. Stefanovic, and S. Valavanis, "Thrust Vector Controller Comparison for a Finless Rocket," *Machines 2023*, vol. 11, no. 3, p. 394, 2023.

[11] NASA, "Gimbaled thrust," `https://www.grc.nasa.gov/WWW/k-12/rocket/gimbaled.html`, Mayo 2021, accedido en abril de 2023.

[12] D. Bernacchia, "*Design of Thrust Vectoring attitude*," [tesis de maestría], Bolonia (Italia), UNIVERSITA' DI BOLOGNA, 2020.

[13] R. H. Gaviño, *Introducción a los Sistemas de Control*. México: Pearson, 2010.

[14] R. E. L. Briega, "Libro online de IAAR," https://iaarbook.github.io/, 2018.

[15] "Minimización de la función de coste — Interactive Chaos." [Online]. Available: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/minimizacion-de-la-funcion-de-coste

[16] T. Moon and A.-H. Najmi, *Advanced Signal Processing: A Concise Guide*. McGraw-Hill Education, 8 2020.

[17] S. Risi and J. Togelius, "Neuroevolution in games: state of the art and open challenges," *IEEE transactions on computational intelligence and AI in games*, vol. 9, no. 1, pp. 25–41, 3 2017. [Online]. Available: https://doi.org/10.1109/tciaig.2015.2494596

[18] S. Rodzin, O. Rodzina, and L. Rodzina, "Neuroevolution: Problems, algorithms, and experiments," in *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, 2016, pp. 1–4.

[19] F. Coto, "Evolución de redes neuronales mediante topologías aumentadas," Universidad Autónoma de Madrid, 2019. [Online]. Available: https://repositorio.uam.es/bitstream/handle/10486/688999/coto_guardo_felix_tfg.pdf?sequence=1&isAllowed=y

[20] S. Sohangir, S. Rahimi, and B. Gupta, "Optimized feature selection using NeuroEvolution of Augmenting Topologies (NEAT)," pp. 80–85, 2013.

[21] M. Melanie, *An Introduction to Genetic Algorithms*. Cambridge, Massachusetts: MIT, 1198.

[22] E. Veslin, "Aplicación de algoritmos genéticos en problemas de Ingeniería," *I3+*, vol. 1, p. 10, 2014.

[23] F. Gross, *Frontiers in Antennas: Next generation Design engineering*. Mcgraw-hill, 1 2011.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," vol. 4, pp. 1942–1948 vol.4, 1995.

[25] R. Huang, Y. Liu, and J. J. Zhu, "Guidance, navigation, and control System design for Tripropeller Vertical-Take-Off-and-Landing Unmanned Air Vehicle," *Journal of Aircraft*, vol. 46, no. 6, pp. 1837–1856, 11 2009. [Online]. Available: https://doi.org/10.2514/1.34476

[26] X. Yu, *Introduction to Evolutionary Algorithms*. Springer, 2010. [Online]. Available: https://www.springer.com/gp/book/9783642127296

[27] OpenRocket, "Sitio de Distribución de Open Rocket," `http://openrocket.distrib.free.fr/index.php`, 2023, accedido en Noviembre de 2023.

[28] ——, "OpenRocket Simulator," `https://openrocket.info/index.html`, 2023, accedido en Noviembre de 2023.

[29] N. S. Launch, "NASA Student Launch ARW Rocketry by the Numbers," `https://www.nasa.gov/wp-content/uploads/2023/09/nasa-sl-2024-arw-rocketry-by-the-numbers-508.pdf`, Julio 2023, accedido en Noviembre de 2023.

[30] S. León Serna, S. Gómez Valencia, J. Alzate López, and F. Obando Vega, "Design and Construction of a Low-Cost Test Bench for Thrust Vector Control Systems in Experimental Rockets," *Engineering for Transformation*, pp. 503–510, 2022.

[31] I. The MathWorks, "Control anti-windup usando el bloque PID Controller," https://la.mathworks.com/help/simulink/slref/anti-windup-control-using-a-pid-controller.html, 2023, accedido en enero de 2024.

[32] AL-Khazraji, "Tuning PID Using PSO," https://github.com/elatik/Tuning_PID_Using_PSO, Septiembre 2021, accedido en Noviembre de 2023.

[33] Divadnoslo, "Neural Networks for PID Controllers," https://github.com/divadnoslo/Neural_Networks_for_PID_Controllers, Diciembre 2020, accedido en Noviembre de 2023.

ANEXOS

Se presenta un conjunto de anexos que complementan y respaldan la información expuesta en los capítulos anteriores. Estos anexos contienen detalles adicionales, códigos, gráficos. Los anexos no solo amplían la perspectiva sobre el enfoque y la metodología utilizados, sino que también sirven como recursos valiosos para aquellos lectores interesados en una exploración más detallada del contenido. A continuación, se presenta una breve descripción de cada anexo y su contribución específica al desarrollo y comprensión del trabajo.

A. Código Usado por el Microcontrolador

```
#include <Wire.h>          // Liberia para comunicacion I2C
#include <LSM303.h>        // Libreria para controlar Acelerometro
#include <PID_v1_bc.h>     // Liberia para controlador PID
#include <ESP32Servo.h>    // Liberia para controlar servomotores

///////////////DEFINICION DE VARIABLES Y OBJETOS///////////////

//IMU
LSM303 compass;
float fgamma = (0.061/1000);
float ax, ay, az;
double psi = 0.0, theta = 0.0;
double psiF, thetaF;
double alpha = 0.38;

// Servo
Servo psiControl;
Servo thetaControl;

// Rele
char relayMode;

// PID
double psiSetpoint, psiOutput;
double thetaSetpoint, thetaOutput;
double psiServoPos, thetaServoPos;
double Kp = 0.3131, Ki= 0.2623, Kd= 0.0852; // Constantes de Sintonizacion



PID psiPID(&psiF, &psiOutput, &psiSetpoint, Kp, Ki, Kd, DIRECT);          //
    Controlador para YAW
```

```
PID thetaPID(&thetaF, &thetaOutput, &thetaSetpoint, Kp, Ki, Kd, DIRECT);     //
    Controlador para PITCH

// Definicion de Pines
#define psiControlPin 19
#define thetaControlPin  18
#define relay 15

void setup()
{
  Serial.begin(9600);      // Inicia comunicacion serial
  Wire.begin();            // Inicia comunicacion I2C
  compass.init();          // Inicio de acelerometro
  compass.enableDefault();

  psiSetpoint = 0;                    // Set Point para movimiento en yaw
  psiPID.SetMode(AUTOMATIC);
  psiPID.SetOutputLimits(-11,16);     // Asignacion de limites para la salida (
      Maximas inclinaciones de la tobera en el eje yaw)

  thetaSetpoint = 0;                  // Set Point para movimiento en pitch
  thetaPID.SetMode(AUTOMATIC);
  thetaPID.SetOutputLimits(-13, 12); // Asignacion de limites para la salida (
      Maximas inclinaciones de la tobera en el eje pitch)

  pinMode(relay, OUTPUT);              // Modo de pin Rele

  psiControl.attach(psiControlPin);  // Asignacion de pines
  thetaControl.attach(thetaControlPin);

  digitalWrite(relay, 1);              // Encendido del rele para flujo
      constante de aire
}

void loop()
```

```
{
  compass.read(); // Lectura de acelerometro

  ax = compass.a.x*fgamma; // calculo de aceleracion en eje X
  ay = compass.a.y*fgamma; // calculo de aceleracion en eje Y
  az = compass.a.z*fgamma; // calculo de aceleracion en eje Z

  psi = double(long(atan(ax/az)*180/PI)); // Calculo aplanado de
      angulo de yaw
  psiF = double(long(alpha*psi+(1-alpha)*psiF)); // Aplicacion de filtro
      pasabajos

  theta = double(long(atan(ay/az)*180/PI)); // Calculo aplanado de
      angulo de pitch
  thetaF = double(long(alpha*theta+(1-alpha)*thetaF)); // Aplicacion de filtro
       pasabajos

  //Yaw control
  psiPID.Compute(); // Se computa el controlador
       de yaw
  psiServoPos = map(psiOutput,-13,12,0,170); // Se mapea la salida del
      controlador a los valores maximo y minimo de movimiento del servomotor de
       yaw
  psiControl.write(psiServoPos); // Se pone en servomotor de
      yaw en posicion


  //Pitch control
  thetaPID.Compute(); // Se computa el
      controlador de pitch
  thetaServoPos = map(thetaOutput,-11,16,30,154); // Se mapea la salida del
      controlador a los valores maximo y minimo de movimiento del servomotor de
       pitch
  thetaControl.write(thetaServoPos); // Se pone en servomotor
      de pitch en posicion
```

```
//Datos a enviar por el puerto serial
Serial.print(psi);Serial.print(",");
Serial.print(psiF);Serial.print(",");
Serial.print(theta);Serial.print(",");
Serial.print(thetaF);Serial.print(",");
Serial.print(psiServoPos); Serial.print(",");
Serial.print(thetaServoPos); Serial.print(",");
Serial.print(psiOutput); Serial.print(",");
Serial.println(thetaOutput);
delay(100);
}
```

B. Diagrama P&ID del Banco de Pruebas

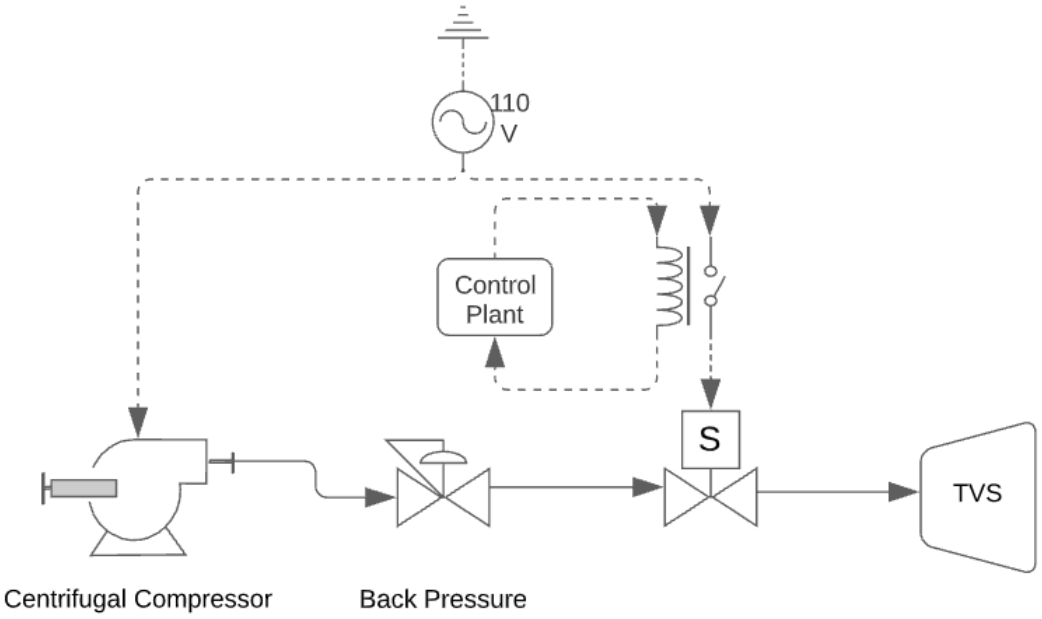

Fig. 45. Diagrama P&ID del Banco de Pruebas.

C. Presupuesto

Implementación						
Madera (3x2)	0	$ -	2	$ 4.700,00	$ -	$ 9.400,00
Tornillos de ensamble 1'' (100u)	0	$ -	1	$ 4.700,00	$ -	$ 4.700,00
PLA para impresión de piezas	0	$ -	2	$ 100.000,00	$ -	$ 200.000,00
Energía Impresora 3D	96	$ 680,63	0	$ 0,00	$ -	$ 65.340,48
Tornillos M3 (20u)	0	$ -	1	$ 7.000,00	$ -	$ 7.000,00
Servomotor (SG90)	0	$ -	2	$ 12.000,00	$ -	$ 24.000,00
Alambre rígido	0	$ -	1	$ 15.000,00	$ -	$ 15.000,00
Electroválvula (110V-[0,116]psi - 1/4'' NPT)	0	$ -	1	$ 60.000,00	$ -	$ 60.000,00
Relay (5V - 250VAC - 30VDC)	0	$ -	1	$ 10.000,00	$ -	$ 10.000,00
Manguera de aire (1m - 1/4'')	0	$ -	3	$ 7.000,00	$ -	$ 21.000,00
Acople rápido (1/4'' NPT)	0	$ -	5	$ 6.000,00	$ -	$ 30.000,00
PCB	0	$ -	1	$ 10.000,00	$ -	$ 10.000,00
ESP32 DevKit V21	0	$ -	1	$ 30.000,00	$ -	$ 30.000,00
IMU GY89	0	$ -	1	$ 30.000,00	$ -	$ 30.000,00
Batería 3,7V 18650 (3000mAh)	0	$ -	2	$ 17.000,00	$ -	$ 34.000,00
						$ 550.440,48
TOTAL						$550.440,48

Fig. 46. Presupuesto Implementación