PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

MAESTRÍA EN INGENIERÍA MECATRÓNICA

MODELAMIENTO DINÁMICO DE LOS PARAMETROS DE CONTROL DE VUELO DE UNA AERONAVE DEL TIPO ALA VOLANTE UTILIZANDO REDES NEURONALES ARTIFICIALES.

Tesis para optar al grado de Magíster en Ingeniería Mecatrónica

Presentado por:

Ing. Aeronáutico CARLOS SAITO VILLANUEVA

Asesor:

Dr. Antonio Manuel Moran Cárdenas

Abril 2018

Lima - Perú

RESUMEN

Esta tesis de investigación propone obtener el modelo aerodinámico de una aeronave del tipo ala volante utilizando redes neuronales artificiales con el fin de mejorar la performance del controlador de vuelo del sistema de navegación de un vehículo aéreo no tripulado. Actualmente la aeronave pierde altitud al momento de realizar los giros, y se entiende porque es un problema con el ángulo de cabeceo y velocidad de vuelo.

El tipo de Red Neuronal Artificial (RNA) utilizada es de Back Propagation Dinámico y tiene dos capas intermedias con 100 neuronas cada una. Se utilizó este tipo de RNA porque permite entrenar un modelo dinámico como es el caso de una aeronave. Las variables de entrada utilizadas para el entrenamiento fueron: posición del elevador, posición de los alerones, posición del throtle y aceleraciones en los tres ejes de la aeronave. Las variables de salida fueron: ángulo de cabeceo, ángulo de alabeo, cambio en el tiempo de ángulo de cabeceo y alabeo, velocidad y altitud. Asimismo, se utilizó un "bias" para tomar en consideración fuerzas o perturbaciones que no se pueden medir.

La metodología utilizada permitió realizar el modelado de manera satisfactoria del ángulo de cabeceo y velocidad. Los errores de entrenamiento fueron de 36% y 5.5% respectivamente. La validación de ambos parámetros fue de 68% y 3.38%. La metodología aplicada todavía necesita ser mejorada para obtener un error de entrenamiento satisfactorio en el ángulo de alabeo y mejorar los entrenamientos obtenidos para el ángulo de cabeceo y velocidad.

Este trabajo demuestra que el modelamiento de una aeronave del tipo ala volante es más complejo que una aeronave convencional. Son pocos los trabajos de investigación sobre modelamiento de aeronaves que han realizado el modelamiento de este tipo de aeronaves. En la mayoría de los casos utilizan técnicas diferentes a las de RNA y realizan modelamiento lineal y no dinámico como se ha realizado en esta tesis.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE GRÁFICOS

INTRODUCCIÓN

Los vehículos aéreos no tripulados (UAV, por sus siglas en inglés) son utilizados para aplicaciones civiles y militares. En ambos casos cumplen misiones complejas y son exigidos para obtener información que sea útil para la correcta toma de decisiones. Los UAVs necesitan integrar software y hardware de manera óptima. La capacidad limitada de carga útil de los UAVs pequeños impide el uso de sistemas de aviónica de alta performance, equipados con sensores inerciales y de adquisición de data durante el vuelo (Jung & Tsiotras, 2006).

Esta tesis de investigación propone obtener el modelo aerodinámico de una aeronave del tipo ala volante utilizando redes neuronales artificiales con el fin de mejorar la performance del controlador de vuelo del sistema de navegación de un vehículo aéreo no tripulado. Actualmente la aeronave pierde altitud al momento de realizar los giros, esta condición será analizada con mayor detalle durante la presente tesis.

El algoritmo de control que utiliza la aeronave es Proporcional, Integral y Derivativo (PID) y tiene como tarea principal controlar las superficies de control de vuelo de la aeronave y poder seguir la ruta planificada. Las ganancias del controlador PID son estáticas y se encontraron de manera experimental (vuelo manual), observando el comportamiento de la aeronave durante el vuelo desde tierra. Esta configuración manual no considera factores externos, como la fuerza del viento o vibraciones al momento de volar. El rendimiento de la aeronave disminuye ya que el controlador solo está configurado para volar bajo ciertas condiciones.

El objetivo principal de la tesis es obtener el modelo dinámico de la aeronave utilizando redes neuronales artificiales para mejorar el control de vuelo. Los objetivos específicos son los siguientes, estudiar la teoría de redes neuronales dinámicas para el modelamiento de aeronaves y diseñar la red neuronal artificial que se utilizará para el modelamiento de la aeronave, obtener la data suficiente para realizar el entrenamiento de la red neuronal y realizar el entrenamiento para obtener un modelo matemático de la aeronave el cual pueda ser validado con otra información de vuelo.

Los resultados esperados al finalizar esta tesis de investigación son: un modelo aerodinámico de la aeronave basado en redes neuronales artificiales que pueda servir para proponer un controlador de vuelo en un siguiente trabajo de investigación.

La metodología para el desarrollo de la tesis consta de cuatro (04) pasos:

- El primer paso es estudiar la teoría de redes neuronales artificiales que se han utilizado para el modelamiento de aeronave. Esta información servirá para diseñar la red neuronal artificial dinámica que se utilizará para el modelamiento de la aeronave propuesta en esta tesis.
- El segundo paso es obtener y adecuar la información de los sensores de la aeronave que será utilizada como variables de entrada y de salida para realizar el entrenamiento de la red neuronal artificial. Este trabajo será realizado utilizando la información de vuelo en distintos escenarios.
- El tercer paso es obtener el modelo dinámico de la aeronave por intermedio del entrenamiento de la red neuronal artificial utilizando herramientas computacionales para poder predecir las variables de cabeceo, alabeo y guiñada durante el vuelo.
- El cuarto paso es realizar las pruebas de validación del modelo dinámico encontrado utilizando otros vuelos de la misma aeronave.

Capítulo 01

EL SISTEMA MECATRÓNICO: LA AERONAVE

En este capítulo se explica la aeronave como un sistema mecatrónico, los componentes mecánicos, aeronáuticos, electrónicos y de control. Asimismo, se realiza un análisis aerodinámico de la aeronave y se definen las ecuaciones del modelo matemático de la aeronave. También, se explica la relación de las variables de las ecuaciones de movimiento y aerodinámicas para identificar las variables de control y de estado.

1.1 Aeronave como un Sistema Mecatrónico

La aeronave que se desea modelar para el presente trabajo de investigación es del tipo ala volante, desarrollada en el Grupo de Investigación de Sistemas Aéreos No Tripulados de la Pontificia Universidad Católica del Perú y tiene las siguientes características técnicas y operativas:

- Material de fabricación: foam de alta densidad
- Tiempo de vuelo: 50 minutos
- Velocidad de crucero: 12 m/s
- Velocidad mínima: 9 m/s
- Velocidad máxima: 16 m/s
- Peso máximo de despegue: 3.5 kg
- Peso operativo: 2.8 kg
- Capacidad de vuelo: programado para realizar un vuelo por si solo.
- Altitud máxima de vuelo: 6000 m
- Motor eléctrico tipo brushless
- Batería de 16.8 V y 6000 mAh
- Peso de carga útil: 1 kg

Figura 1. 1 Aeronova No Tripulada

Esta aeronave ha realizado vuelos satisfactorios y representativos para el grupo y a nivel nacional, por ejemplo, ha sido la primera aeronave no tripulada en volar sobre el volcán Ubinas en Moquegua a más de 6,000 msnm como parte de las actividades del proyecto para el monitoreo de volcanes con el Instituto Geofísico del Perú y ha sobrevolado el glaciar Suyuparina en Cuzco para colaborar con los estudios que viene realizando la Universidad de Zurich sobre el deshielo. (IGP, 2016) (Guzman, 2016).

Toda aeronave como la que se muestra en la Figura 1.1 tiene componentes mecánicos (aeronáuticos), electrónicos y de control. Esto se dividen de la siguiente manera:

- Componentes mecánicos: el fuselaje, las superficies móviles, la hélice ylos brazos que transfieren el movimiento de los servomotores a las superficies móviles.
- Componentes electrónicos: servomotores para mover las superficies de control, el sistema de propulsión y el sistema de comunicación.
- Componentes de control: el sistema de navegación.

El sistema de navegación tiene como principal componente el controlador de vuelo, el cual se muestra en la figura 1.2

Figura 1. 2 Controlador de Vuelo PIXHAWK[1]

El controlador tiene las siguientes características técnicas:

Procesador:

- 32bit STM32F427 Cortex M4 core con FPU
- 168MHz
- 256KB RAM
- 2MB Flash

[1] *http://ardupilot.org/plane/docs/common-pixhawk-wiring-and-quick-start.html*

- 32bit STM32F103 failsafe co-processor

Sensores:

- ST Micro L3GF20H 16Bit Giroscopo
- ST Micro LSM303D 14bit acelerometro / magnetometro
- Invensense MPU 6000 3-ejes acelerometro / giroscopo
- MEAS MS5611 barometro

Interfaces:

- 5x UART (puerto serial), uno capaz de alta potencia, 2x con control fluido de HW
- 2x CAN (uno con un transceiver, uno con un conector de expansión)
- Spektrum DSM/DSM2/DSM-X® Satellite input compatible
- FUTABA S-BUS® input y output compatible
- PPM input sumador de señal
- RSSI (PWM o Voltaje) input
- I2C
- SPI
- 3.3V y 6.6V ADC inputs
- Puerto microUSB interno y Puerto extendible microUSB externo

Posee tripe redundancia para la alimentación de energía como lo son:

- Input del “Power Module” (4.8V a 5.4V)
- Input del rail de Servos (4.8V a 5.4V)
- Input del puerto USB (4.8V a 5.4V)

El controlador de vuelo permite que la aeronave pueda realizar vuelos completamente autónomos siguiendo el plan de vuelo. El algoritmo de control que gobierna las respuestas del controlador de vuelo está basado en PID (Proporcional, Integral y Derivativo), la ecuación básica se muestra en la figura 1.3. La figura 1.4 muestra el diagrama de control para los alerones, tomando en consideración el ángulo de alabeo, el cambio en la velocidad de alabeo y el alabeo requerido para llegar al punto deseado.

$$u = \underbrace{K_p e}_{\text{Proportional Term}} + \underbrace{K_i \int_0^t e\,dt}_{\text{Integral Term}} + \underbrace{K_d \frac{d}{dt} e}_{\text{Differential Term}}$$

Figura 1. 3 Ecuación Controlador PID[2]

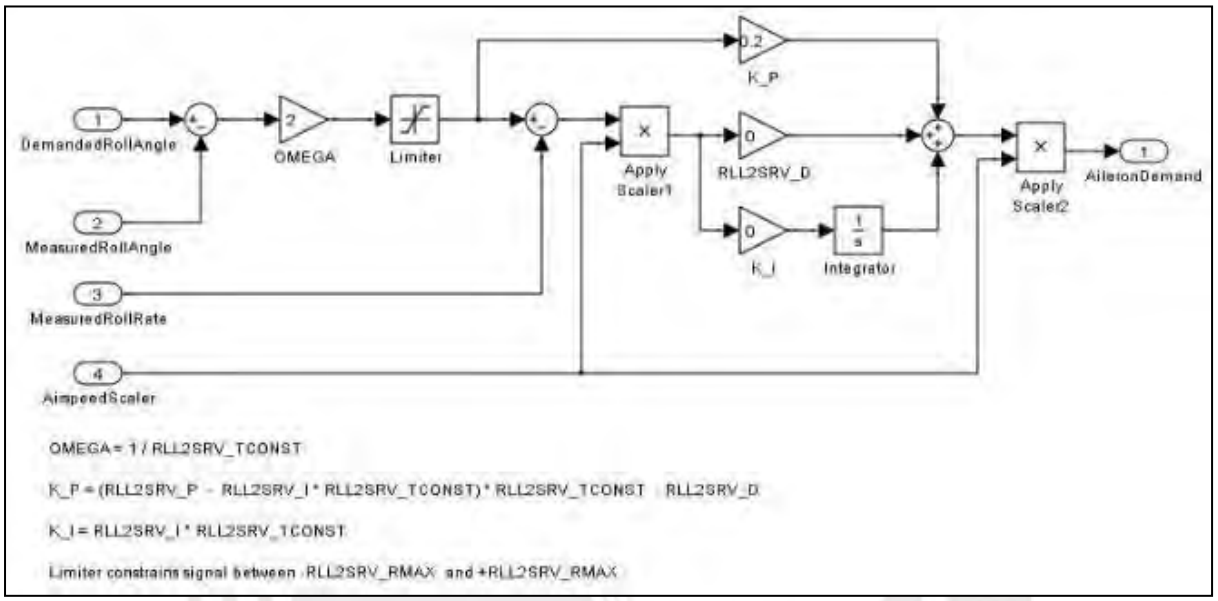


Figura 1. 4 Diagrama de Control de Alerones[3]

Entonces lo importante es poder configurar las ganancias PID (K_p, K_i, K_d) para que el vuelo de la aeronave sea lo más estable posible.

La capacidad de procesamiento del PIXHAWK permite utilizar algoritmos matemáticos de control más avanzados como el "filtro extendido de Kalman (EKF, por sus siglas en inglés)" el cual es utilizado para estimar la posición del vehículo, velocidad y orientación angular basado en las mediciones del giróscopo, acelerómetro, magnetómetro, GPS, velocímetro y presión barométrica. La ventaja de utilizar este algoritmo es que al fusionar todas las mediciones disponibles puede dejar de utilizar una medición con un error significativo para

[2] *http://wiki.theuavguide.com/wiki/PID_Controller*

[3] *http://ardupilot.org/plane/docs/roll-pitch-controller-tuning.html*

que el vehículo se vuelva menos susceptible a los errores de un sensor en particular (Ardupilot, 2017).

El algoritmo EKF implementado en el PIXHAWK estima un total de 22 estados. A continuación, se detalla una descripción simple y no matemática de cómo trabaja el filtro (Ardupilot, 2017):

1. Los ratios "rates" de los ángulos medidos por el IMU son integrados para calcular la posición angular.
2. Las aceleraciones del IMU son convertidas utilizando la posición angular de los ejes X, Y, Z a los ejes Norte, Este y Apuntando al centro de la Tierra y corregidas por gravedad. (Transformation from body coordinates to Earth coordinates).
3. Las aceleraciones son integradas para calcular la velocidad.
4. La velocidad es integrada para calcular la posición.

Esto 4 primeros pasos sirven para predecir los estados, que son las variables que se estiman como por ejemplo el banqueo, cabeceo, guiñada, altitud, velocidad, etc.

5. Estimar los ruidos del giróscopo y acelerómetro. Esta estimación es utilizada para medir el crecimiento del error en los ángulos, velocidades y cálculo de posición utilizando la data del IMU. Si no se realiza una corrección utilizando otra medición, el error estimado seguirá creciendo.

Los pasos del 1 al 5 son repetidos cada vez que se obtiene información nueva del IMU hasta que una nueva medición de otro sensor se encuentra disponible.

1.2 Los 06 Grados de Libertad de toda Aeronave

Toda aeronave o plataformas de vuelo posee 06 grados de libertad (6-DOF), los cuales pueden ser representados con ecuaciones matemáticas de movimiento. Asimismo, se necesitan superficies de control para gobernar a la aeronave en estos 6-DOF. Estas superficies de control son las siguientes (Understanding RC airplane controls, 2017):

- Elevadores: esta superficie de control se encuentra en el estabilizador horizontal de la aeronave. Controlan la actitud de cabeceo (pitch) de la aeronave. Cuando los elevadores

se deflectan para arriba, la nariz del avión asciende y sucede lo contrario cuando se deflectan para abajo.

- Alerones: esta superficie de control se encuentra en los extremos posteriores de ambas alas y controlan el alabeo (roll) de la aeronave. Operan de manera coordinada y opuesta, por ejemplo, cuando el alerón de un ala se deflecta para abajo, el alerón del ala opuesta se deflecta para arriba y viceversa.
- Timón de dirección: esta superficie también conocida como rudder, se encuentra en la parte posterios del estabilizador vertical de la aeronave. Es utilizado para el control de la guiñada (yaw) de la aeronave. Al deflectarse el timón de dirección para la derecha o izquierda, la nariz de la aeronave lo hace en ese mismo sentido.

La figura 1.5, ilustra la posición de las superficies de control y los movimientos que originan cada una de ellas en una aeronave convencional.

Sin embargo, la aeronave que se utiliza como elemento de estudio para esta tesis no cuenta con estabilizador horizontal ni vertical en la parte posterior, solo cuenta con los alerones como superficies de control en los extremos de las alas para los movimientos de cabeceo y alabeo. Esta superficie de control se denomina “elevon”.

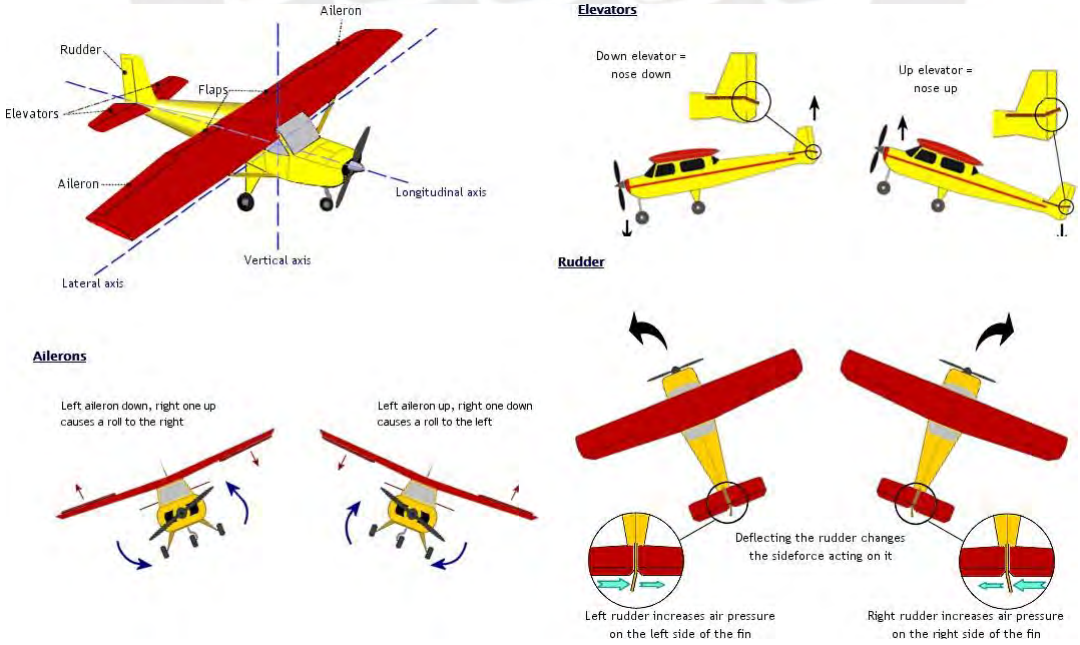


Figura 1. 5 Superficies de Control[4]

[4] *http://www.rc-airplane-world.com/rc-airplane-controls.html*

La siguiente figura ilustra los ángulos, ejes de rotación y desplazamiento de toda aeronave

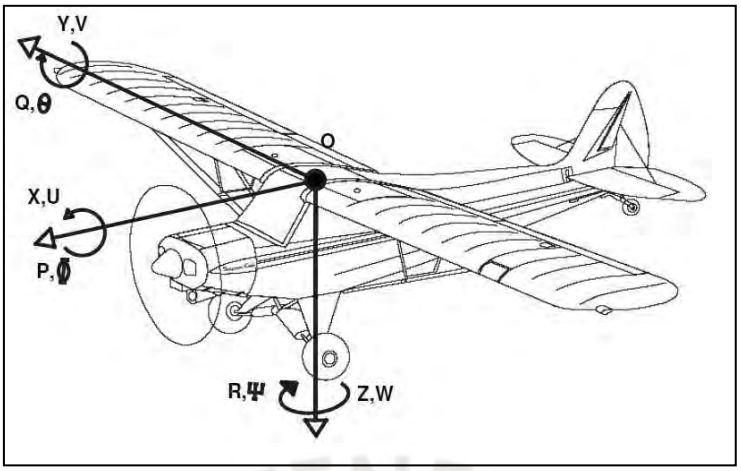


Figura 1. 6 Ángulos, velocidad de cambio (rates), ejes positivos y el vector velocidad en cada uno de los ejes de una aeronave.

Fuente: (Stevens & Lewis, 2003)

1.3 Análisis Aerodinámico

La aeronave utilizada es de tipo ala volante, la cual posee características aerodinámicas que serán descritas a continuación. Estas características aerodinámicas le otorgan a la aeronave la performance necesaria para realizar los vuelos que son analizados. La teoría de "La Linea de Sustentación" del software XLFR5 fue utilizada para realizar el análisis aerodinámico.

La figura 1.7 muestra las partes principales de la aeronave.

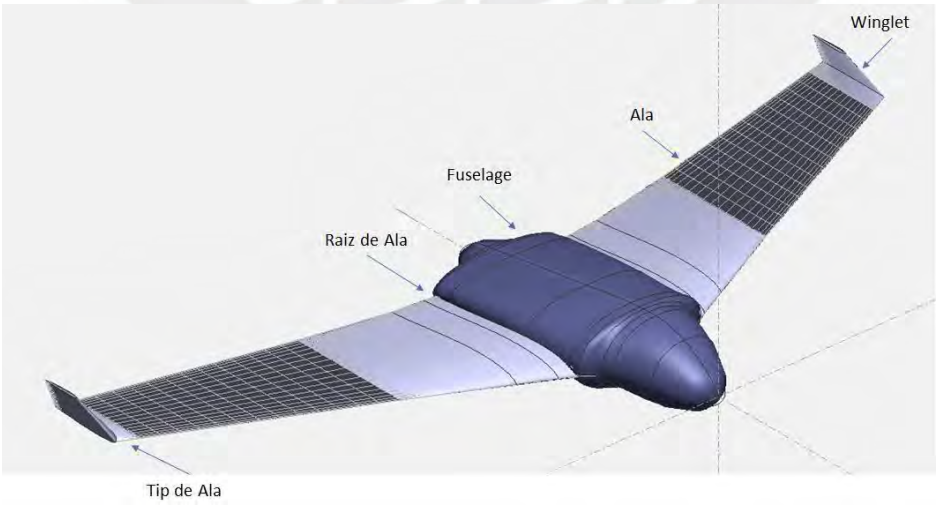


Figura 1. 7 Partes Principales de la Aeronave

Fuente: Propia

El ala tiene el perfil autosustentable HS322 y el winglet tiene un perfil simétrico NACA0010. Ambos perfiles son analizados en el rango de numero Reynolds desde 86mil en el tip de ala hasta 390mil en la raíz del ala. La figura 1.8 muestra el perfil alar HS322 y la figura 1.9 muestra el perfil NACA0010.

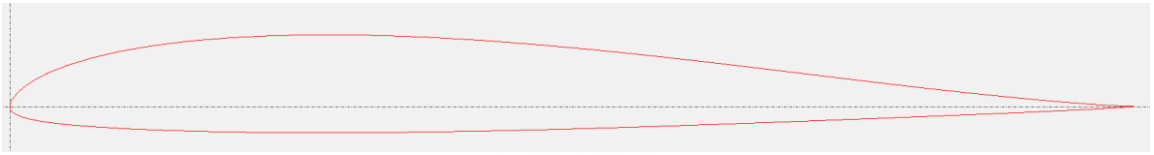


Figura 1. 8 Perfil Autosustentable HS322

Fuente: Propia

El perfil HS322 es autosustentable porque la forma que posee le permite mantener estabilidad positiva durante el vuelo sin la necesidad de que la aeronave tenga un estabilizador horizontal.

Figura 1. 9 Perfil Simétrico NACA0010

Fuente: Propia

El perfil NACA0010 es simétrico por la forma que posee. La cuerda o línea que une los extremos del perfil lo divide a este en dos partes iguales.

A continuación, se analizarán las características aerodinámicas del perfil HS322. La figura 1.10 muestra algunos conceptos importantes a tomar en consideración al momento del análisis aerodinámico. El viento relativo es opuesto a la dirección de vuelo de la aeronave y su magnitud es la que se utiliza para realizar los cálculos de sustentación y resistencia. La cuerda o cuerda alar, es la línea recta que une los extremos del perfil. El ángulo de ataque es el que se forma entre el vector del viento relativo y la cuerda del perfil alar.

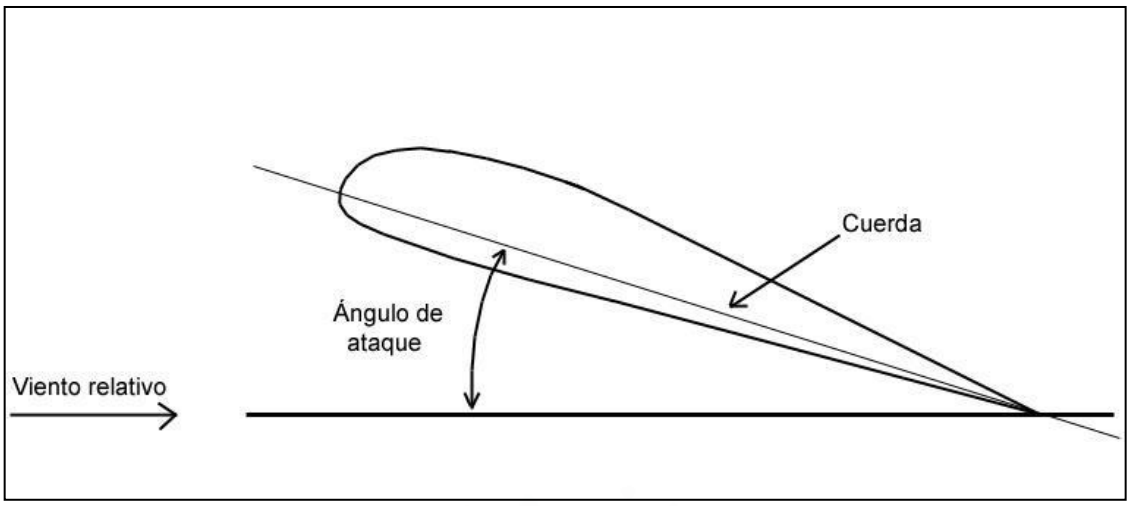


Figura 1. 10 Definiciones básicas de un perfil alar

Fuente: Propia

El coeficiente de sustentación es un valor adimensional que se utiliza para calcular la sustentación o fuerza ascendente del perfil. Este valor adimensional representa la diferencia de presiones que existe entre la parte inferior y superior del perfil. El siguiente gráfico muestra la curva de sustentación del perfil HS322.

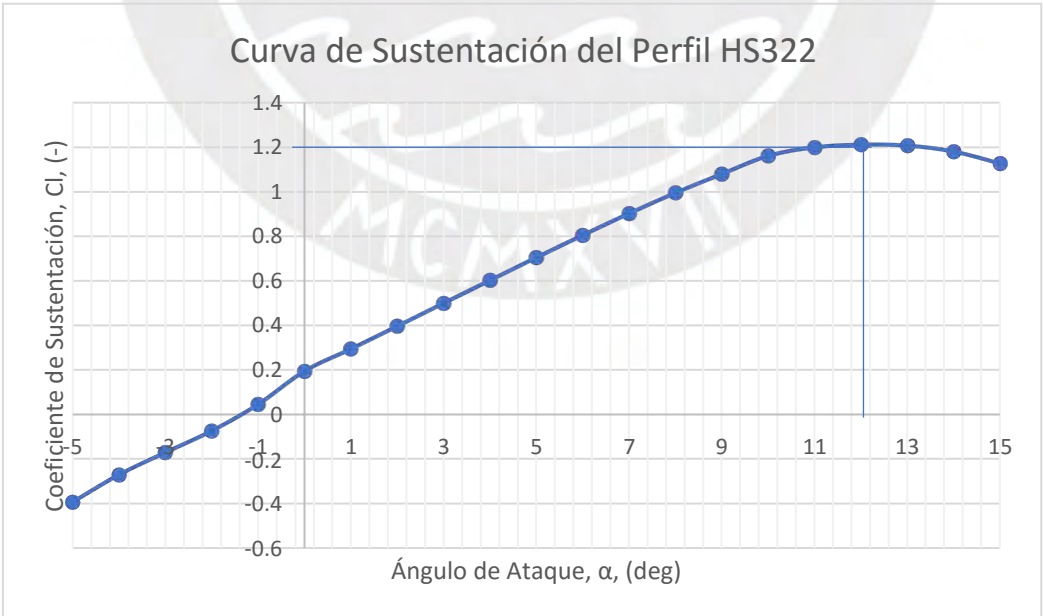


Gráfico 1. 1 Curva de Sustentación del Perfil HS322

Fuente: Propia

La parte lineal de la curva es la zona de mejor performance, este es el rango de ángulos de ataque en los cuales el perfil puede sustentarse durante el vuelo. Este perfil puede generar sustentación desde un ángulo de ataque de 0° a 10°. El ángulo de ataque máximo es de 12° y el coeficiente de sustentación máximo es de 1.2.

La polar de resistencia, es la curva que compara el coeficiente de sustentación con el coeficiente de resistencia. La siguiente grafica muestra la polar de resistencia para el perfil alar HS322.

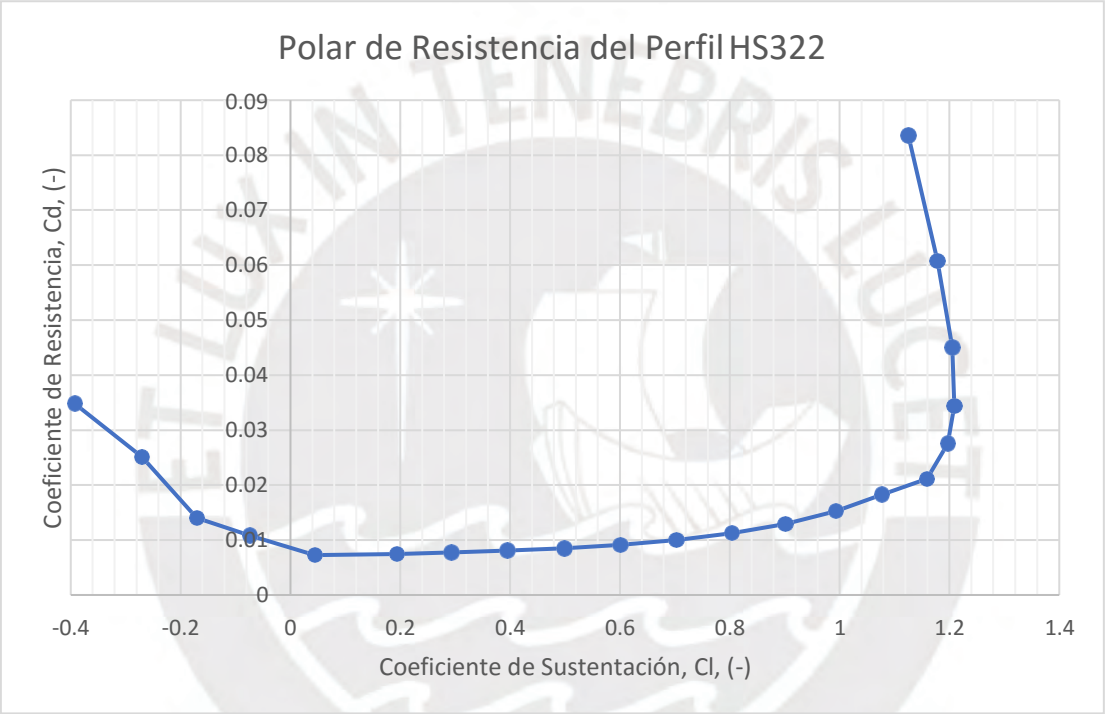


Gráfico 1. 2 Polar de Resistencia del Perfil Alar HS322

Fuente: Propia

Al conocer los coeficientes de sustentación y resistencia del perfil alar, se continua con el análisis aerodinámico de la superficie alar. La figura 1.11 muestra el diseño de la aeronave en el software XFLR5. Este diseño es el que se ha utilizado para calcular los coeficientes aerodinámicos de la aeronave.

El coeficiente de sustentación de la aeronave (CL) se puede analizar utilizando la gráfica 1.3. Se observa que la curva es continua y la zona de perdida de sustentación, entre los ángulos 11°

a 15° no es abrupta. La aeronave no perderá sustentación instantáneamente si por algún motivo el ángulo de ataque excede el valor de 13°.

La polar de resistencia (gráfica 1.4) de la aeronave compara el CL y el coeficiente de resistencia (CD) de la aeronave, muestra que la forma y aerodinámica de la aeronave no produce resistencia considerable en la zona de sustentación donde se desarrolla el vuelo, entre los ángulos -3° a 10°. Sin embargo, cuando el coeficiente de sustentación aumenta por encima de 1, la resistencia también aumenta considerablemente, por lo tanto, no se recomienda realizar vuelos por encima de este coeficiente de sustentación.

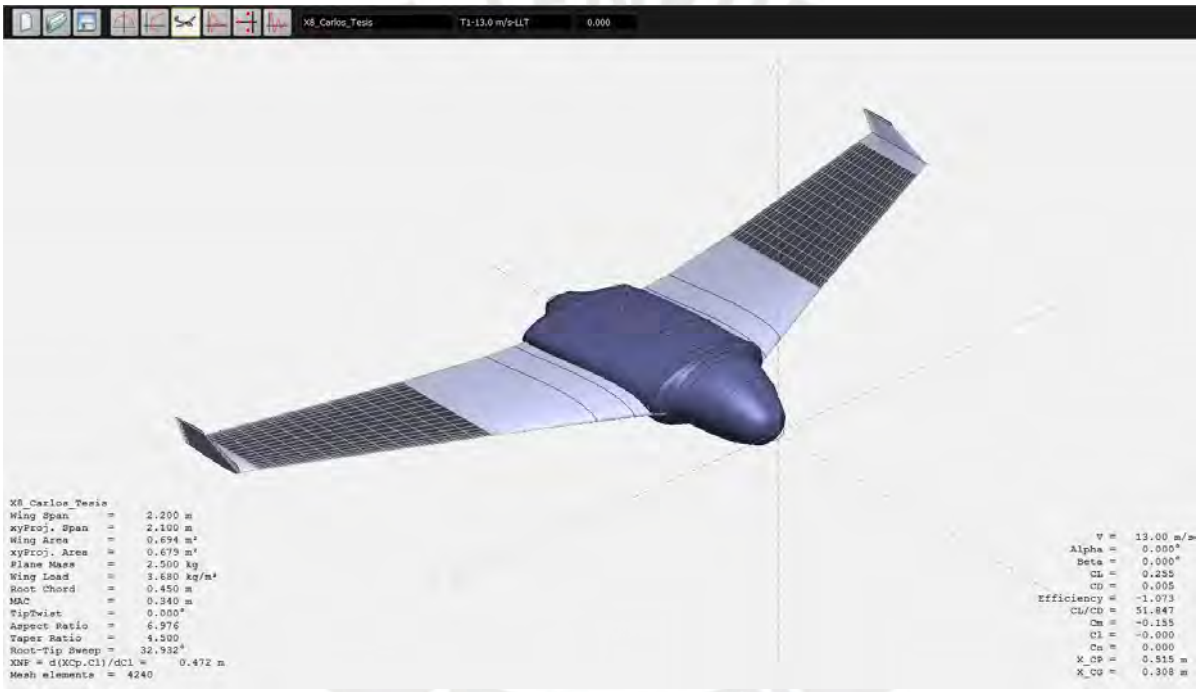


Figura 1. 11 Aeronave diseñada en Software XFLR5

Fuente: Propia

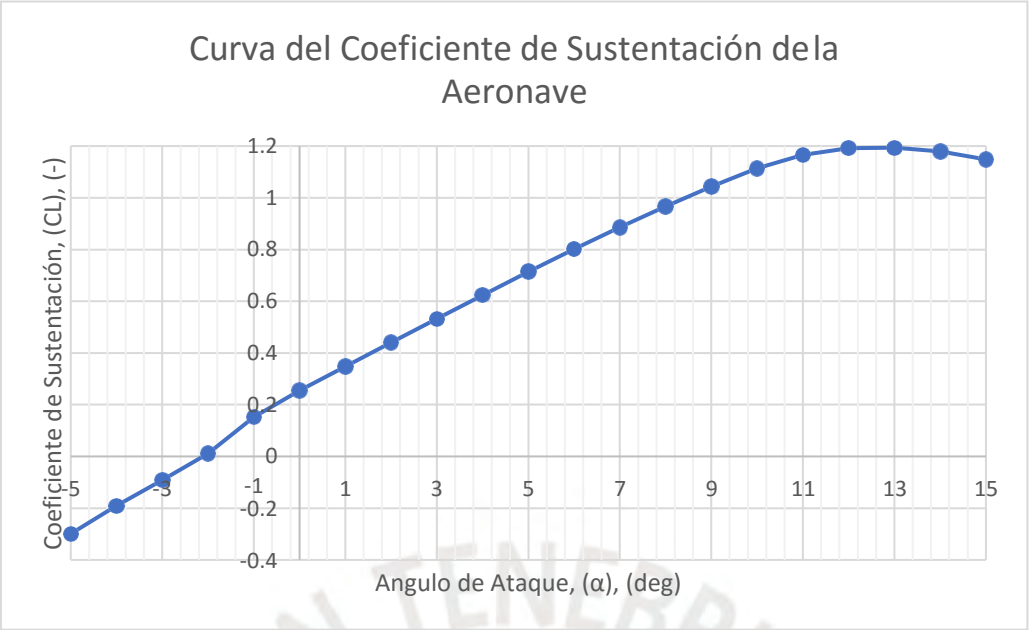


Gráfico 1. 3 Curva del Coeficiente de Resistencia de la Aeronave

Fuente: Propia

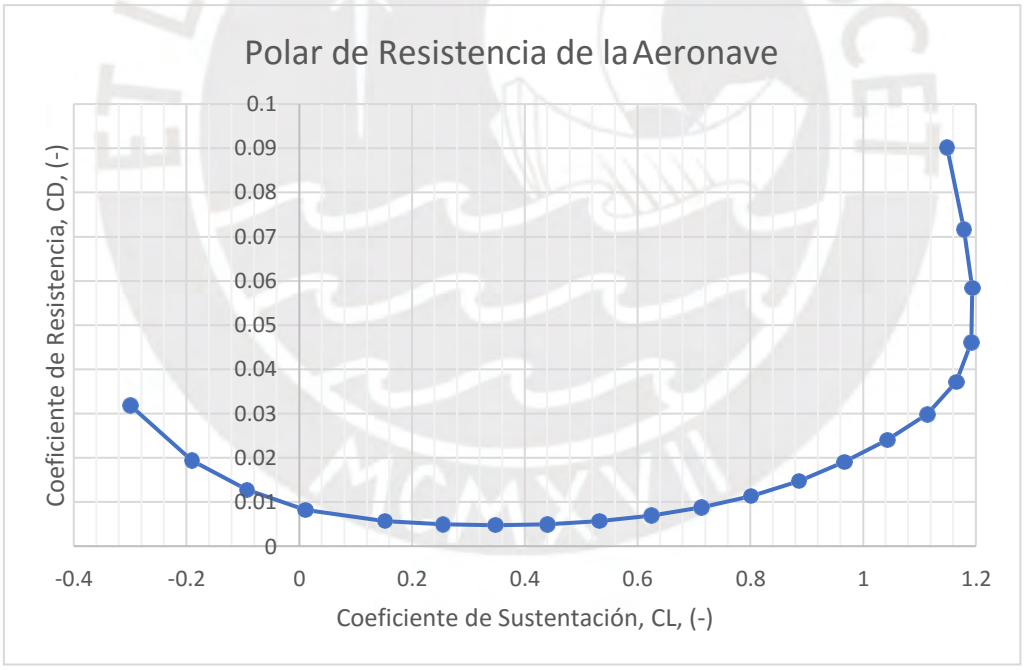


Gráfico 1. 4 Polar de Resistencia de la Aeronave

Fuente: Propia

La curva de eficiencia aerodinámica relaciona el coeficiente de sustentación y resistencia. Esta curva nos permite identificar el ángulo de ataque donde hay menor resistencia.

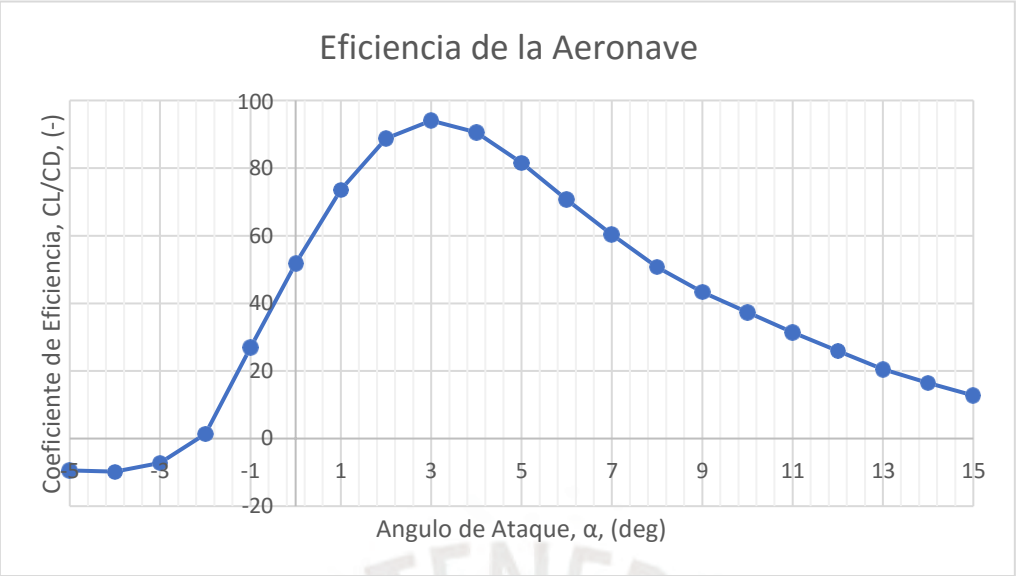


Gráfico 1. 5 Curva de Eficiencia de la Aeronave

Fuente: Propia

Utilizando las curvas aerodinámicas de la aeronave y conociendo que el ángulo de ataque para una mejor performance es de 3°, se obtiene que la sustentación de la aeronave es de 3.9 kg y la resistencia de 0.04 kg al nivel del mar. El peso de operación de la aeronave es de 2.8 kg. Se concluye que la aeronave puede realizar vuelos sin problema.

1.4 Relación de la Variables Aerodinámicas

El modelamiento matemático de la aeronave es la parte más importante en el sistema de control de vuelo. El control del UAV es alcanzado cuando se controlan los alerones, el timón de dirección, el timón de profundidad o elevador y la potencia del motor, los cuales son representados por δ_a, δ_r, δ_e, y δ_f. Los parámetros del UAV como velocidad (V), ángulo de ataque (α), ángulo de deslizamiento lateral (β), ratio de alabeo (roll rate) (p), ratio de cabeceo (pitch rate) (q), ratio de guiñada (yaw rate) (r), ángulo de alabeo (φ), ángulo de cabeceo (Θ), ángulo de guiñada (ψ) y tres parámetros de posición con respecto a la Tierra, x_e, y_e, H son considerados variables de estado.

La variable de control y la variable de estados del modelo matemático de uns aeronave convencional se pueden expresar de la siguiente manera:

$$u = [\delta_a \ \delta_r \ \delta_e \ \delta_f]^T,$$

$$x = [V \ \alpha \ \beta \ p \ q \ r \ \varphi \ \theta \ \psi \ x_e \ y_e \ H \ ]^T.$$

En el caso de la aeronave que se estudia en esta tesis, las variables de control y de estado son:

$$u = [\delta_a \ \delta_e \ \delta_f]^T,$$

$$x = [V \ \alpha \ \beta \ p \ q \ \varphi \ \theta \ x_e \ y_e \ H \ ]^T.$$

Las figuras 1.12 a 1.14 muestran los ángulos y ratios (rates) de la aeronave que es motivo de estudio en esta tesis.

Se utilizan las siguientes suposiciones para simplificar los cálculos (Lung & Geng, 2011):

- Asumir que el avión es un cuerpo rígido con masa constante. Ignorar la elasticidad de la aeronave.
- Considerar a la Tierra como un sistema inercial; ignorar la rotación de la Tierra.
- Ignorar la curvatura de la Tierra, considerar que es un plano.
- Asumir que la aceleración de la gravedad no cambia con la altitud.
- Asumir que la geometría del avión es simétrica y que la masa está bien distribuida.

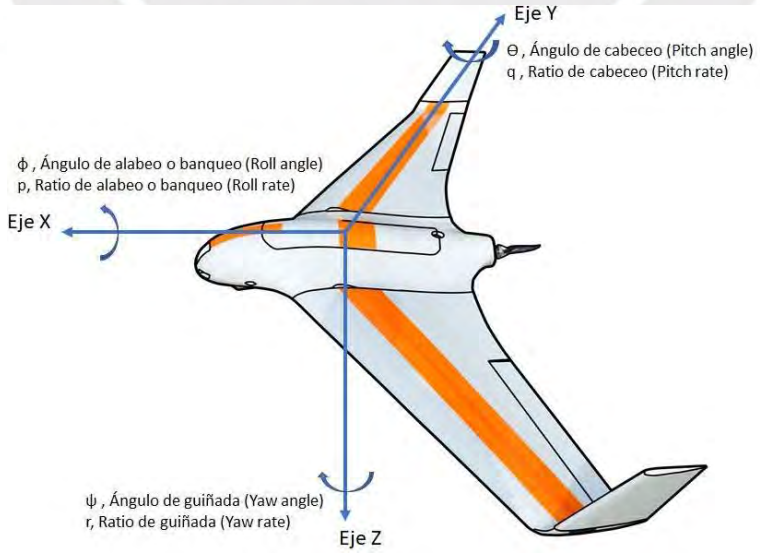


Figura 1. 12: Ejes, Ángulos y Ratios de la Aeronave

Fuente: Propia

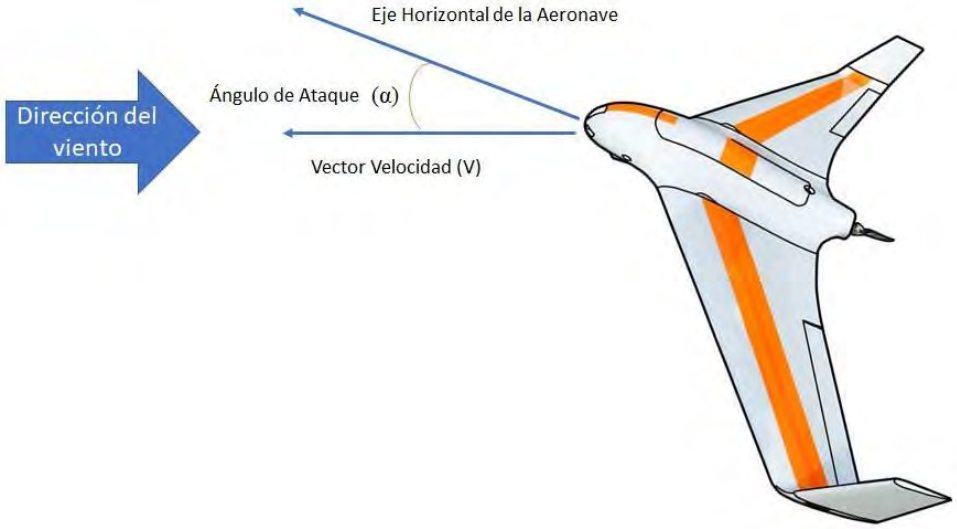


Figura 1. 13: Ángulo de Ataque y Vector de Velocidad

Fuente: Propia

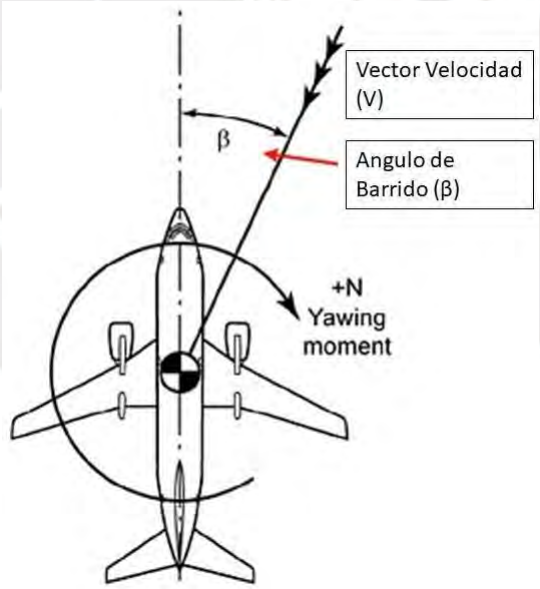


Figura 1. 14: Ángulo de Barrido y Vector Velocidad[5]

[5] https://www.theairlinepilots.com/forum/viewtopic.php?t=396

Se establecen 06 ecuaciones de movimiento en el eje de coordenadas de la aeronave para analizar las fuerzas del UAV.

$$\dot{u} = \frac{F_x}{m} - qw + rv, \quad (1)$$

$$\dot{v} = \frac{F_y}{m} + pw - ru, \quad (2)$$

$$\dot{w} = \frac{F_z}{m} - pv + qu, \quad (3)$$

$$\dot{p} = \frac{1}{I_X I_z - I_{xz}^2}[I_z L + I_{xz} N + (I_x - I_y + I_z) I_{xz} pq + (I_y I_z - I_z^2 - I_{xz}^2) qr], \quad (4)$$

$$\dot{q} = \frac{1}{I_y}[M - I_{xz}(p^2 - r^2) - (I_x - I_z) pr] \quad (5)$$

$$\dot{r} = \frac{1}{I_X I_z - I_{xz}^2}[I_{xz} L + I_x N + (I_x^2 - I_x I_y + I_{xz}^2) pq + (I_x - I_y - I_z) qr] \quad (6)$$

Donde *u, v, w* son los tres componentes de la velocidad de vuelo *V* en el eje de coordenadas del cuerpo de la aeronave x-, y-, z- respectivamente. F_x, F_y, F_z son fuerzas, incluida la gravedad, empuje del motor y fuerzas aerodinámicas en los ejes x-, y-, z- respectivamente. L, M, N son los momentos de alabeo, guiñada y cabeceo. I_x, I_y , I_z son los momentos de inercia alrededor del eje de coordenadas del cuerpo de la aeronave y el I_{xz} es el producto de inercia.

Las ecuaciones diferenciales de los ángulos de Euler de los ángulos de actitud de la aeronave son los siguientes:

$$\dot{\varphi} = p + q sin\varphi tan\theta + r cos\varphi tan\theta, \quad (7)$$

$$\dot{\theta} = q cos\varphi - r sin\varphi, \quad (8)$$

$$\dot{\psi} = q\frac{sin\varphi}{cos\theta} + r\frac{cos\varphi}{cos\theta}, \quad (9)$$

Donde φ, Θ, ψ son los ángulos de alabeo, cabeceo y guiñada en el plano de la tierra respectivamente, es por lo que se utilizan los subíndices “e”.

Las ecuaciones diferenciales para las coordenadas de la aeronave son:

$$\dot{x}_e = \{ucos\theta + (vsin\varphi + wcos\varphi)sin\theta\}cos\psi - (vcos\varphi - wsin\varphi)sin\psi, \quad (10)$$

$$\dot{y}_e = \{ucos\theta + (vsin\varphi + wcos\varphi)sin\theta\}sin\psi + (vcos\varphi - wsin\varphi)cos\psi, \quad (11)$$

$$\dot{z}_e = -sin\theta + (vsin\varphi - wcos\varphi)cos\theta. \quad (12)$$

Todas estas ecuaciones describen el modelo matemático de la aeronave.

Las ecuaciones de movimiento de la aeronave se fundamentan en las ecuaciones aerodinámicas (Lung & Geng, 2011) de fuerza y momento. Estas ecuaciones son la base para el modelo aerodinámico.

Coeficientes aerodinámicos de fuerza

- Coeficiente de sustentación:

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\alpha^3}}\alpha^3 + C_{L_q}\frac{q\bar{c}}{V} + C_{L_{\delta_{e\beta^2}}}\delta_e\beta^2 + C_{L_{\delta_f}}\delta_f + C_{L_{\alpha\delta_f}}\alpha\delta_f, \quad (13)$$

- Coeficiente de arrastre:

$$C_D = C_{D_0} + C_{D_\alpha}\alpha + C_{L_{D^2}}\alpha^2 + C_{L_{D^3}}\alpha^3 + C_{D_q}\frac{q\bar{c}}{V} + C_{D_{\delta_r}}\delta_r + C_{D_{\delta_f}}\delta_f + C_{D_{\alpha\delta_f}}\alpha\delta_f, \quad (14)$$

- Coeficiente de fuerza lateral:

$$C_Y = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{V} + C_{Y_r}\frac{rb}{V} + C_{Y_{\delta_r}}\delta_r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\alpha\delta_r}}\alpha\delta_r. \quad (15)$$

Las ecuaciones de fuerza aerodinámica son:

$$Sustentación = L = C_L\frac{1}{2}\rho V^2 S, \quad (16)$$

$$Arrastre = D = C_D\frac{1}{2}\rho V^2 S, \quad (17)$$

$$Fuerza\ lateral = Y = C_Y\frac{1}{2}\rho V^2 S. \quad (18)$$

Coeficientes aerodinámicos de momento

- Coeficiente de momento de cabeceo:

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_r}}\delta r + C_{l_{\delta_a}}\delta a + C_{D_{\alpha\delta_a}}\alpha\delta a, \quad (19)$$

- Coeficiente de momento de alabeo:

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\propto^2}}\propto^2 + C_{m_q}\frac{q\bar{c}}{V} + C_{m_{\delta_e}}\delta e + C_{m_{\beta^2}}\beta^2 + C_{L_{\delta_f}}\delta f + C_{m_r}\frac{rb}{2V}, \quad (20)$$

- Coeficiente de momento de guiñada:

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_r}}\delta r + C_{n_{\delta_a}}\delta a + C_{n_q}\frac{q\bar{c}}{V}, \quad (21)$$

Las ecuaciones de momento son:

$$Momento\ de\ Cabeceo = L = C_l \frac{1}{2}\rho V^2 S, \quad (22)$$

$$Momento\ de\ Alabeo = M = C_m \frac{1}{2}\rho V^2 S, \quad (23)$$

$$Momento\ de\ Guiñada = N = C_n \frac{1}{2}\rho V^2 S. \quad (24)$$

Donde "q" es la presión dinámica, "ρ" es la densidad del aire, "V" es la velocidad de vuelo del UAV, "S" es el área alar, "b" es la envergadura del ala y "c" es la cuerda media aerodinámica.

De las ecuaciones de movimiento y las ecuaciones aerodinámicas como se puede apreciar se encuentran relacionadas, las variables son dependientes unas de otras. Solo por citar un ejemplo, se utiliza la ecuación (25) que es el coeficiente de sustentación, para calcular la fuerza de sustentación o fuerza en el eje z (F_z). Esta ecuación necesita la retroalimentación del ángulo de ataque, velocidad de vuelo, ángulo de deslizamiento, empuje del motor y posición del elevador. Al momento de entrenamiento de la red neuronal para obtener el modelo dinámico de la aeronave se debe de tomar en consideración estas relaciones.

Capítulo 02

PREPARACIÓN DE LA INFORMACIÓN: LA RED NEURONAL ARTIFICIAL

En este capítulo se explica la importancia de utilizar Red Neuronales Artificiales (RNA) para el modelamiento de la aeronave. Asimismo, se obtiene la información de los sensores de la aeronave para realizar el diseño la RNA y se explica el problema que se está investigando. Al finalizar, se muestra el resultado del entrenamiento para encontrar el modelo de la aeronave.

2.1. Red Neuronal Artificial para el Modelamiento de una Aeronave.

Es necesario utilizar técnicas que permitan identificar el modelo de la aeronave con la información de vuelos realizados. Según lo expresado en las secciones anteriores, se puede identificar el modelo no lineal de la aeronave utilizando las ecuaciones de movimiento de 6DOF, para propósitos de esta investigación se utilizará la teoría de redes neuronales artificiales para la identificación de modelos dinámicos.

Las redes neuronales artificiales son sistemas de procesamiento de información formados por un gran número de unidades de procesamiento, idénticas o similares, altamente interconectadas

y realizan procesos locales. Una importante característica de estas redes es la naturaleza adaptativa, lo que quiere decir que el conocimiento es adquirido del medio ambiente (data experimental) a través de un proceso adaptativo llamado aprendizaje. Está compuesta de un gran número de elementos altamente interconectados (neuronas) trabajando al unísono para resolver problemas específicos (Jang, Sun, & Mizutani, 1997).

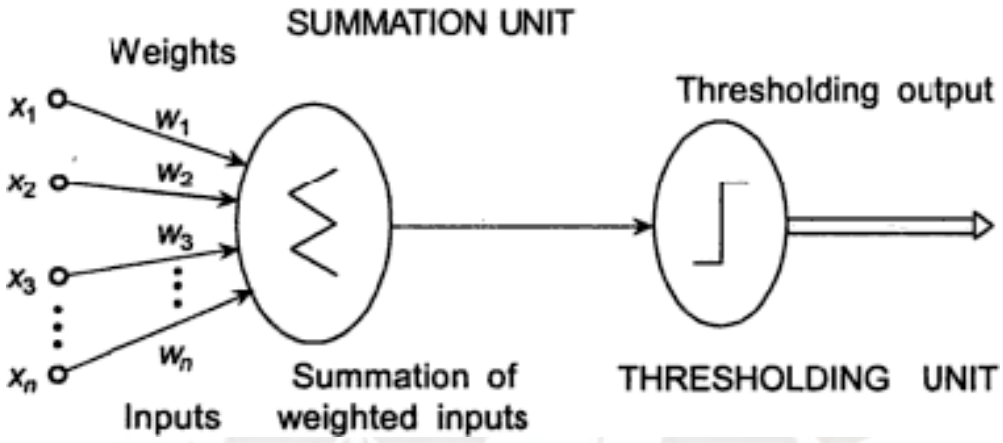


Figura 2. 1 Modelo Simple de una Neurona Artificial

Fuente: (Behera & Rana, 2014)

Las redes neuronales por lo general están organizadas en capas. Las capas están formadas por un número de "nodos" interconectados que contienen una "función de activación". El patrón de la red inicia con una "capa de entrada", la cual se comunica con una o más "capas ocultas" donde se realiza el procesamiento de la información a través de un sistema de "conexiones" ponderadas (Haykin, 2002).

Las redes neuronales artificiales son similares a las redes neuronales biológicas porque realizan funciones de manera colectiva y en paralelo, en lugar de que exista una delimitación clara de tareas a las que se asignan varias unidades. El término "Red Neuronal" usualmente se refiere a modelos empleados en estadísticas, psicología cognitiva e inteligencia artificial (Behera & Rana, 2014).

Un problema de identificación de sistemas puede ser formulado como una tarea de optimización donde el objetivo es encontrar un modelo y un conjunto de parámetros que

minimizan el error de predicción entre la data medida y la salida del modelo. Los enfoques de identificación de sistemas existentes son altamente analíticos y se basan en la derivación del modelo del sistema (Behera & Rana, 2014).

En los últimos años, diferentes algoritmos han sido propuestos para la identificación de sistemas lineales y no – lineales, y estos algoritmos utilizan diferentes formas de conocimiento sobre el sistema. La identificación de sistemas desconocidos se utiliza en temas de control, ecualización de canales, cancelación de ruido en redes de comunicación y teleconferencias, etc. La metodología propuesta de identificación del sistema se realiza al ajustar parámetros de un modelo dado hasta que la salida, para una entrada en particular, coincida tan bien como sea posible con la salida medida del sistema que está siendo identificado para esta misma entrada. Después de haber identificado el sistema, la salida puede ser anticipada para una entrada del sistema (Behera & Rana, 2014). Con lo cual se puede anticipar el comportamiento de la aeronave para una determinada posición de las superficies de control.

La identificación de sistemas de una aeronave determina un modelo matemático para la dinámica del sistema utilizando un conjunto de data de entrada y salida. Este modelo incluye las ecuaciones de movimiento de la física cinética y cinemática de toda la aeronave. El sistema lineal de la aeronave muestra las propiedades de Markov, donde futuros estados del sistema de la aeronave pueden ser determinados por los estados actuales y los estados futuros son una combinación lineal de los estados actuales. Lo recomendable es utilizar una red neuronal artificial entrenada utilizando el algoritmo de "backpropagation" para minimizar el error entre el historial en el tiempo de las salidas del sistema aprendido y las salidas del sistema de la aeronave (Kirkpatrick, May, & Valasek, Aircraft System Identification Using Artificial Neural Networks, 2013)

2.2. Información de los Sensores A bordo de la Aeronave.

La información utilizada para poder obtener el modelo dinámico de la aeronave proviene del IMU (Unidad de Medición Inercial) y sensor de velocidad de vuelo. La aeronave cuenta con un controlador de vuelo que posee estos sensores, el cual durante todo el vuelo registra información que es útil para el análisis.

El IMU que se ha utilizado posee las siguientes capacidades de sensado[6]:

- Aceleración lineal en 3D
- Velocidad angular en 3D
- Campo magnético de la Tierra en 3D
- Presión Barométrica

El análisis del movimiento longitudinal y lateral / direccional requiere la siguiente información durante un periodo de tiempo:

- Velocidad de vuelo (V) (m/s)
- Ratio de Cabeceo (Pitch Rate) (q) (deg/s)
- Ángulo de Cabeceo (Pitch Angle) (Θ) (deg)
- Posición del Elevador (Elevator Position) (δ_e) (PWM)
- Posición de la Potencia (Throttle Position) (δ_T) (PWM)
- Ratio de Alabeo (Roll Rate) (p) (deg/s)
- Ángulo de Alabeo (Bank angle) (φ) (deg)
- Posición de los Alerones (Aleron Position) (δ_a) (PWM)

Las entradas de control de los alerones, elevadores y motor son registradas como una señal PWM ("Pulse Width Modulated"). El elevador cuando esta deflectado totalmente hacia arriba tiene un valor ideal de PWM de 2000 y totalmente hacia abajo el valor PWM es 1100. En el caso de los alerones, cuando se realiza un alabeo a la derecha el valor ideal de PWM es 2000 y un alabeo a la izquierda el valor PWM es 1100. Para el caso del motor, cuando se encuentra en máxima potencia el valor ideal es 2000 y en potencia mínima es de 1100. Sin embargo, para efectos del análisis estos valores de PWM han sido convertidos a sus valore reales en ángulos al observase la deflexión de las superficies y porcentaje de empuje del motor. Los estados, que son las demás variables que se utilizaron, son registradas en las unidades antes mostradas.

A continuación, se muestra esta información, primero las indicaciones en los controles de la radio

[6] Información obtenida de la página web: https://pixhawk.ethz.ch/electronics/imu

Figura 2. 2 Canales de Control en el Radio Control para Operar la Aeronave.

En la computadora donde se monitorea el vuelo se puede observar como varían los valores de PWM previamente explicados, asimismo la señal de input cuando se mueven los canales de la radio. La figura 2.3 muestra cada uno de los canales. "Ch1in" (línea rosada) es la entrada que se origina en el radio control para realizar un alabeo, "Ch1out" (línea amarilla) es la salida en la aeronave o la deflexión física real en los alerones. "Ch2in" (línea verde) es la entrada que se origina en el radio control para realizar un cabeceo, "Ch2out" (línea naranja) es la salida en la aeronave o deflexión física real en los elevadores. "Ch3in" (línea azul) es la entrada que origina en el radio control para aumentar o disminuir la potencia del motor.

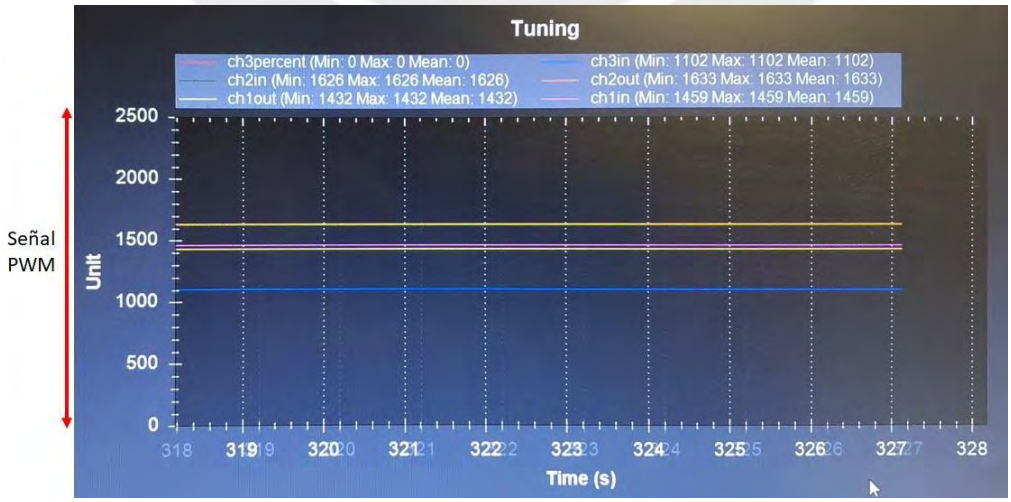


Figura 2. 3 Pantalla en el Software de Control de la Aeronave para observar los canales de control de la aeronave.

2.3. El Ángulo de Alabeo y Cabeceo

2.3.1 Ángulo de Alabeo

El ángulo de alabeo se da como consecuencia de girar a la aeronave sobre el eje longitudinal (alabeo o roll). Las superficies de control son los alerones. La figura 2.4 muestra un alabeo a la derecha y a la izquierda. La figura 2.5 muestra la trayectoria de la aeronave al momento de realizar un alabeo a la derecha.

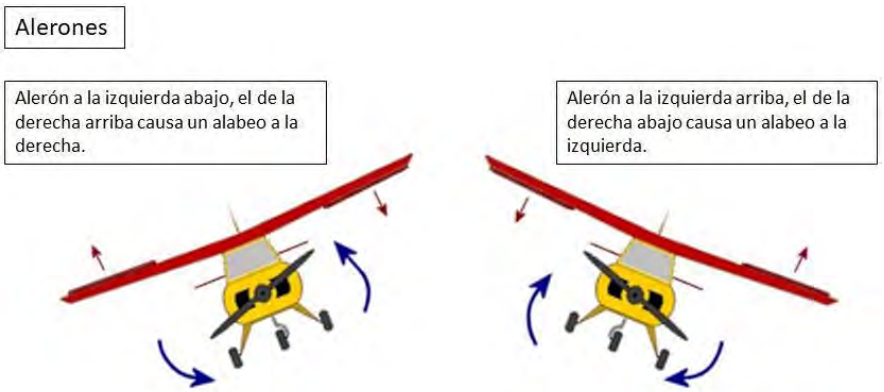


Figura 2. 4 Alabeo de la Aeronave a la Derecha e Izquierda[7]

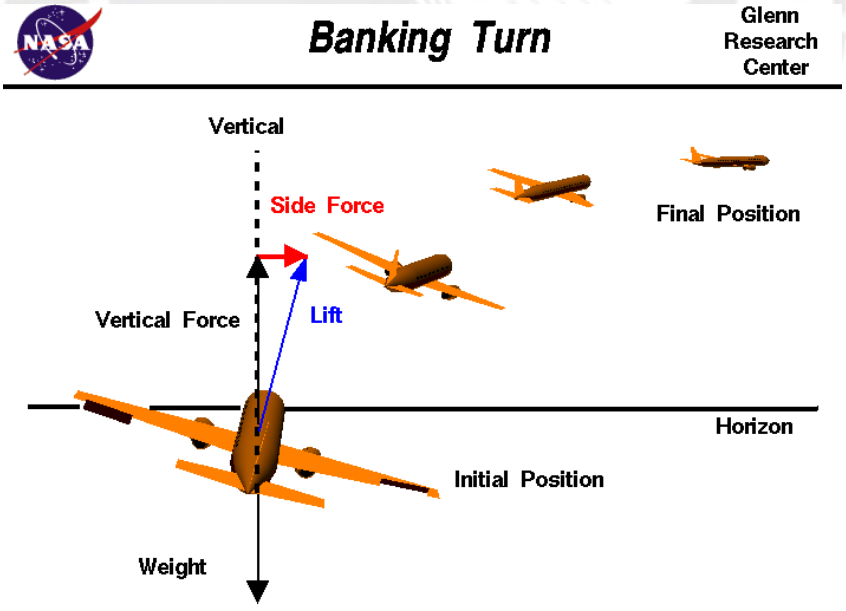


Figura 2. 5 Alabeo a la Derecha, Dirección de Desplazamiento de la Aeronave

Fuente: Glenn Research Center

[7] *http://www.rc-airplane-world.com/rc-airplane-controls.html*

En las siguientes figuras se va a mostrar la señal de entrada que se origina en el radio control y la deflexión de las superficies de control.

La figura 2.6 muestra un alabeo a la derecha. La señal de entrada “Ch1in” (línea morada) aumenta el valor de 1457 a 1878 y la señal de salida “Ch1out” (línea amarilla) disminuye de 1435 a 1082, lo que significa que el alerón del ala derecha ha subido para realizar un alabeo a la derecha. Sin embargo, como estas superficies de control también controlan el cabeceo, entonces se muestra una desviación en la señal de entrada “Ch2in” (línea verde) para que la aeronave mantenga la altitud durante el giro. La señal de salida “Ch2out” (línea naranja) disminuye el valor de 1634 a 1296 lo que significa que el alerón del ala izquierda ha bajado.

La figura 2.7 muestra un alabeo a la izquierda. En este caso, la señal de entrada “Ch1in” (línea morada) disminuye el valor de 1459 a 1039 y la señal de salida “Ch1out” (línea amarilla) aumenta de 1433 a 1778, lo que significa que el alerón del ala derecha ha bajado para realizar un alabeo a la izquierda. Así como en el caso anterior, sucede lo contrario con la señal de salida “Ch2out” (línea naranja) al aumentar el valor de 1632 a 1970 lo que significa que el alerón del ala izquierda ha subido.

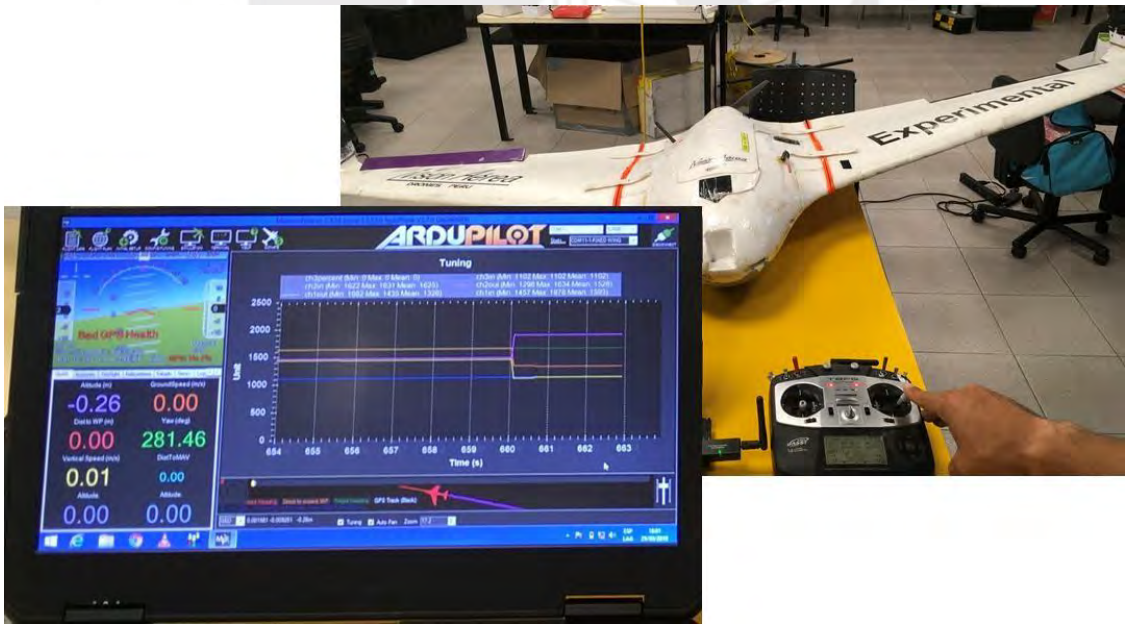


Figura 2. 6 Alabeo a la Derecha.

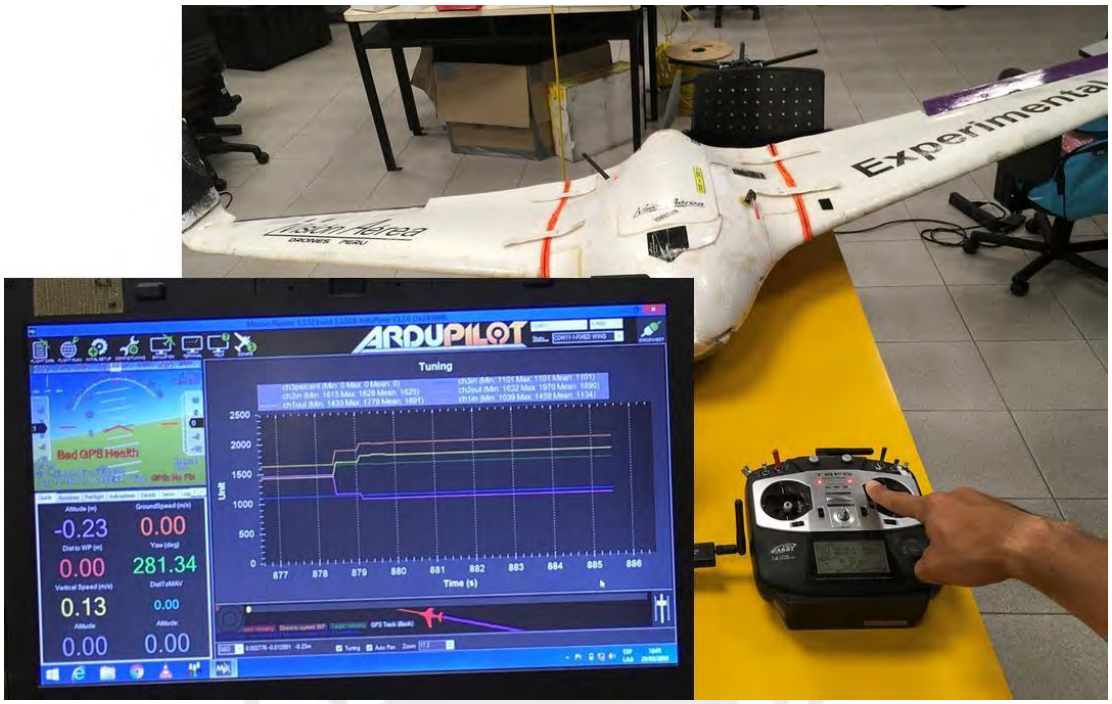


Figura 2. 7 Alabeo a la Izquierda

2.3.2 Ángulo de Cabeceo

La siguiente variable que se debe de analizar es el cabeceo, la figura 2.8 muestra cómo se controla el cabeceo con el elevador.

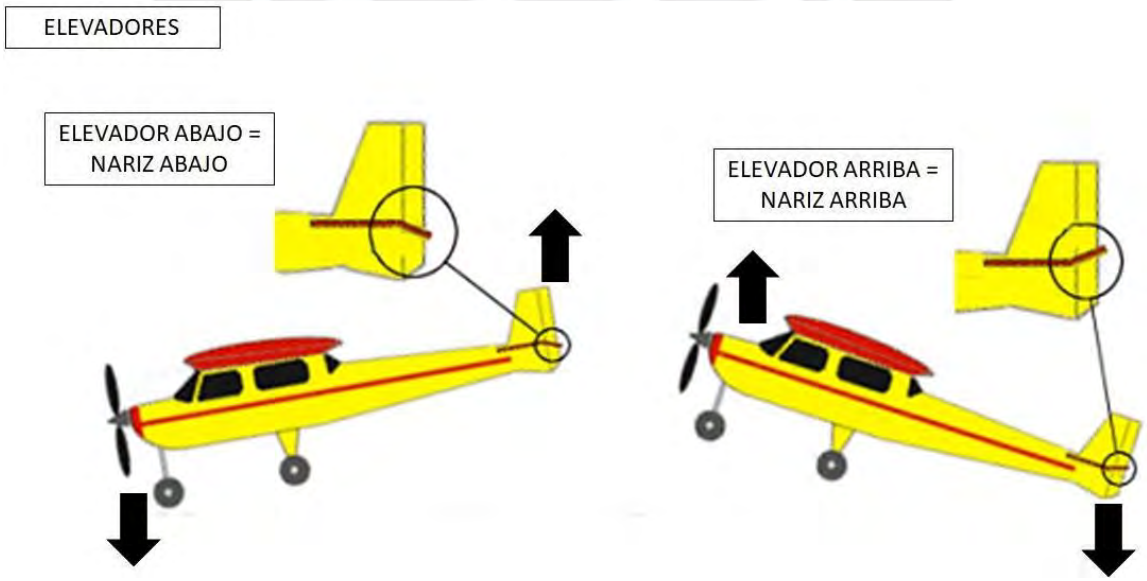


Figura 2. 8 Control del Cabeceo con el Elevador

En el software de control también se muestra como la señal de control varia cuando se gira el elevador. La figura 2.9 muestra a los elevadores deflectados abajo, lo que significa que la aeronave realizará un cabeceo abajo (pitch down). La señal del “Ch2in” (línea verde) disminuye de 1626 a 1205 y la señal de salida “Ch2out” (línea naranja) también disminuye, con lo cual se comprueba que la aeronave realizará un cabeceo abajo. Al mismo tiempo, se observa la señal de salida “Ch1out” (línea amarilla) que aumenta, lo cual significa que el alerón del ala derecha ha bajado.

La figura 2.10 muestra a los elevadores deflectados arriba, lo que significa que la aeronave realizará un cabeceo arriba (pitch up). La señal del Ch2in (línea verde) aumenta de 1626 a 2045 y la señal de salida “Ch2out” (línea naranja) también aumenta, con lo cual se comprueba que la aeronave realizará un cabeceo arriba. La señal de salida “Ch1out” (línea amarilla) disminuye, lo cual significa que el alerón del ala derecha ha subido.

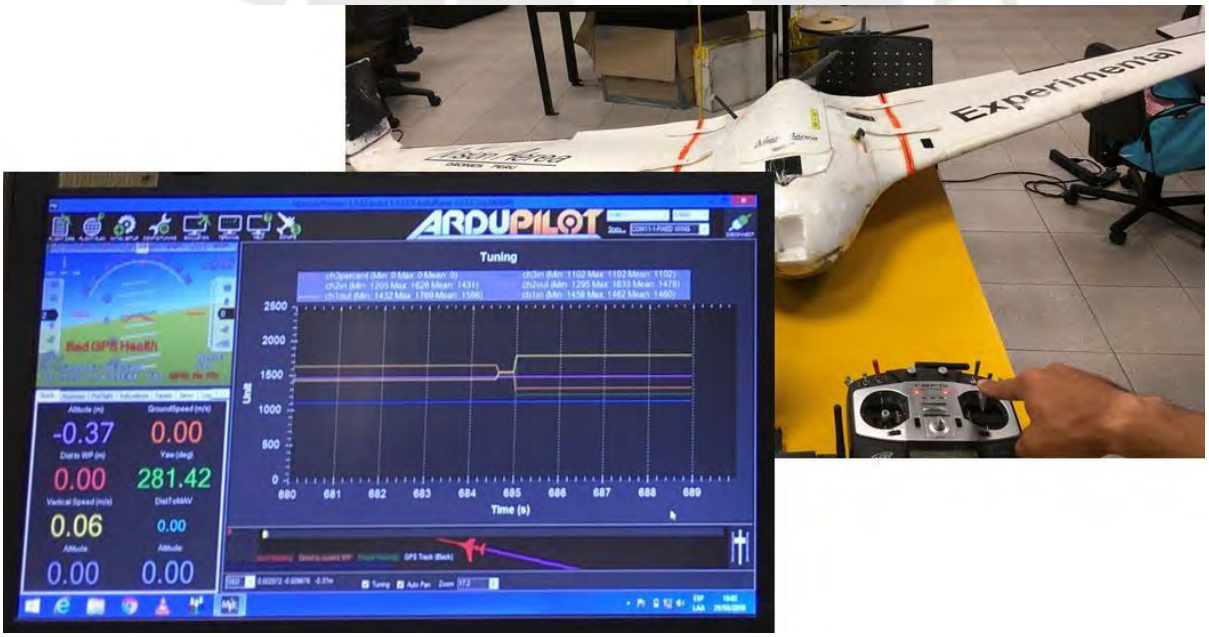


Figura 2. 9 Cabeceo Abajo – Pitch Down

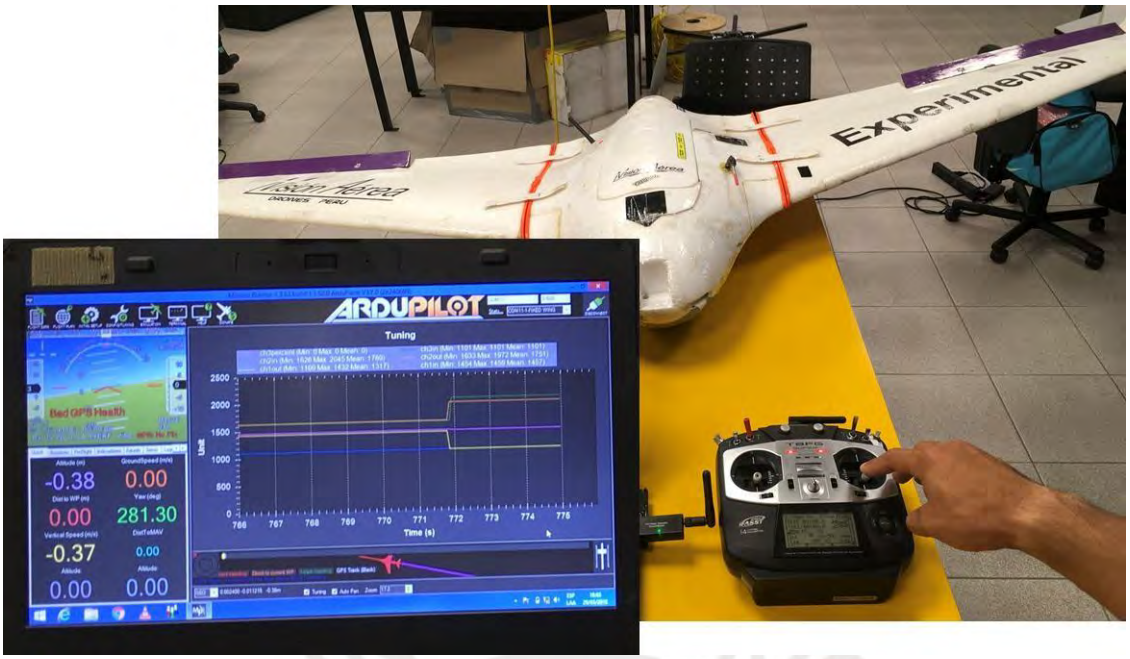


Figura 2. 10 Cabeceo Arriba – Pitch Up

2.4. Identificación del Problema Durante Vuelo.

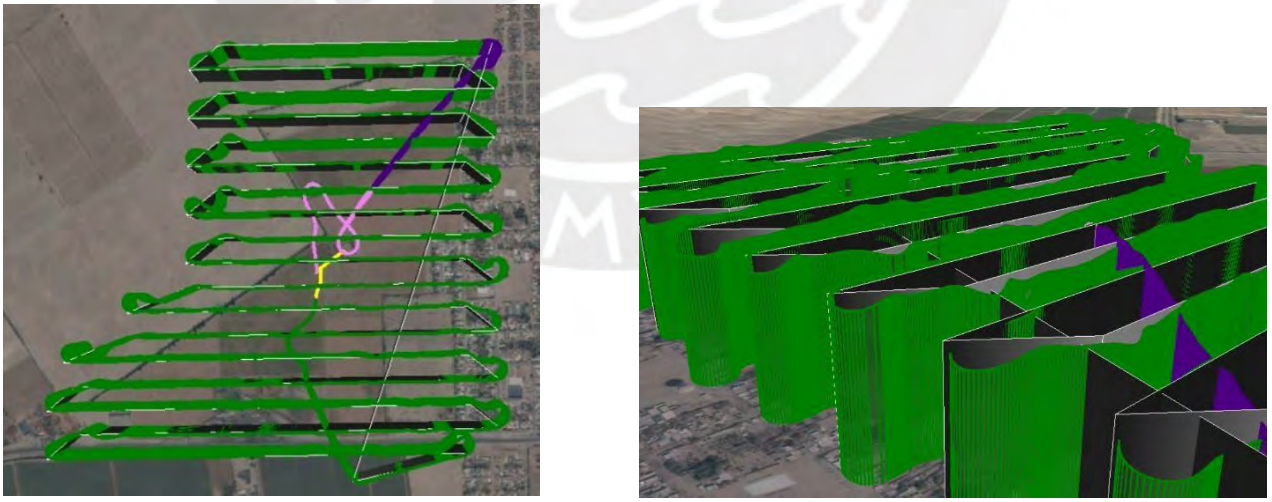
La información que se utilizó para el entrenamiento de la RNA fue de un vuelo realizado en Ica que tuvo una duración de 30 minutos y sirvió para fotografiar un campo de espárragos. El vuelo sigue un patrón para realizar trabajos de "fotogrametría aérea". La ruta de vuelo se muestra en la siguiente figura.

Figura 2. 11 Ruta de Vuelo para Trabajo de Fotogrametría Aérea.

Fuente: Software Mission Planner

La aeronave pasa por cada uno de los puntos verdes (waypoints) en la dirección que indica la línea amarilla. La distancia entre líneas es de 74 metros. La velocidad de vuelo fue de 13m/s (47km/h) a una altitud de 150m sobre el terreno. La información que se consideró es la parte central del vuelo. Se asume que durante este periodo de tiempo el avión a alcanzado una estabilidad constante ya que la fase de despegue a ocurrido unos minutos antes y aún faltan algunos minutos para la fase de aterrizaje. Toda la información que es almacenada por el controlador de vuelo se puede graficar para observar la performance de la aeronave. La figura 2.12a y 2.12b muestran la información graficada utilizando el software Google Earth y se observa como la aeronave ha seguido el plan de vuelo descrito en la figura 2.11

La figura 2.12a muestra, desde una vista de planta, el vuelo de la aeronave al momento de seguir las líneas de vuelo. El patrón está representado por las líneas blancas y el vuelo de la aeronave por la línea verde. Se observa que la aeronave trata de seguir el patrón de vuelo, especialmente en las partes rectas, sin embargo, en las curvas la aeronave tiende a oscilar para ingresar a la siguiente línea. Asimismo, en la figura 2.12b se observa con mayor detalle que durante los giros pierde altitud de 10 a 15 metros, la cual recupera al finalizar el giro e ingresar a la línea de vuelo. También se observa que la altitud de la aeronave oscila durante todo el vuelo. Esto se puede deber a factores externos como vientos locales, cambios de temperatura o presión. Sin embargo, la aeronave siempre busca mantener la altitud de vuelo indicada por el plan de vuelo.



(a) (b)

Figura 2. 12 (a) Ruta de Vuelo Real vista de planta. (b) Ruta de Vuelo Real vista de perfil

Fuente: Propia

De los 30 minutos de vuelo, solo se han tomado aproximadamente 13 minutos que involucran a la aeronave volando el patrón de vuelo. A manera de ejemplo, para mostrar la data que se obtiene de los sensores, se utiliza uno de los estados de la aeronave que se desean modelar.

La información que se va a utilizar incluye por lo menos tres alabeos a la derecha e izquierda. La figura 2.13 muestra los giros de la aeronave durante el recorrido planificado. Se aprecia que la aeronave en la mayoría de los giros excede el instante en el que debe de dejar de girar para ingresar en vuelo nivelado. Por lo tanto, debe de hacer un giro adicional hacia la izquierda para corregir e ingresar correctamente en la siguiente línea de vuelo.

Figura 2. 13 Giro Real a la Derecha de la Aeronave

En la computadora de vuelo, se puede visualizar el horizonte artificial, el cual es un instrumento que tiene como función indicar el ángulo de alabeo al momento de realizar un alabeo. La aeronave tiene la capacidad de inclinarse con un ángulo de alabeo a la derecha o la izquierda de 60° como máximo. La figura 2.14 muestra el horizonte artificial en posición neutral y con un giro a la derecha de 30°.

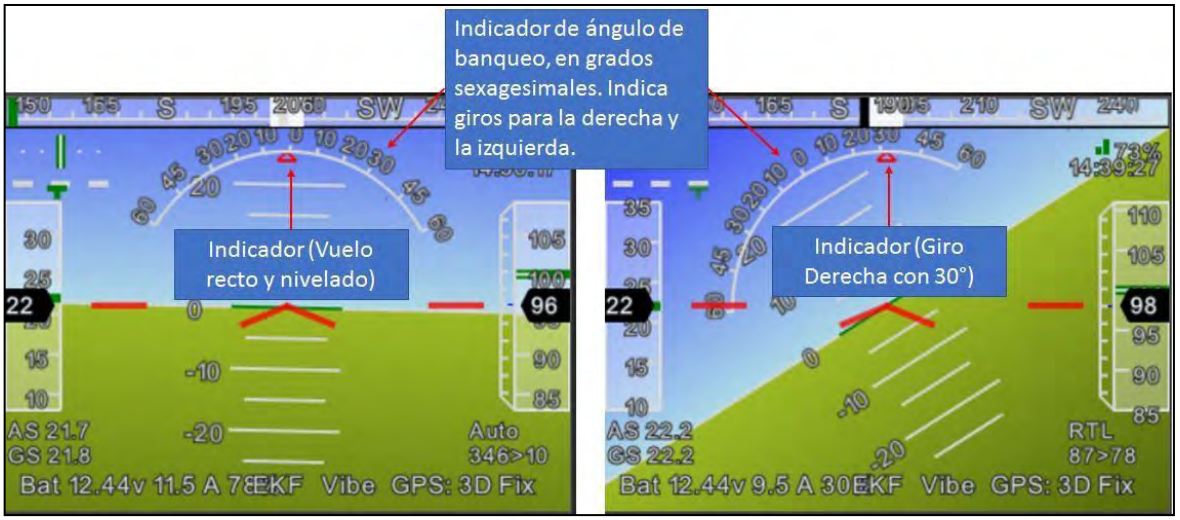


Figura 2. 14 Indicador "Horizonte Artificial" muestra un vuelo recto y nivelado y un giro a la derecha de 30°

Fuente: Propia

Una vez finalizado el vuelo se descarga información de los sensores. La figura 2.15 muestra la información obtenida del IMU correspondiente al ángulo de alabeo para un intervalo de tiempo de 2000 segundos.

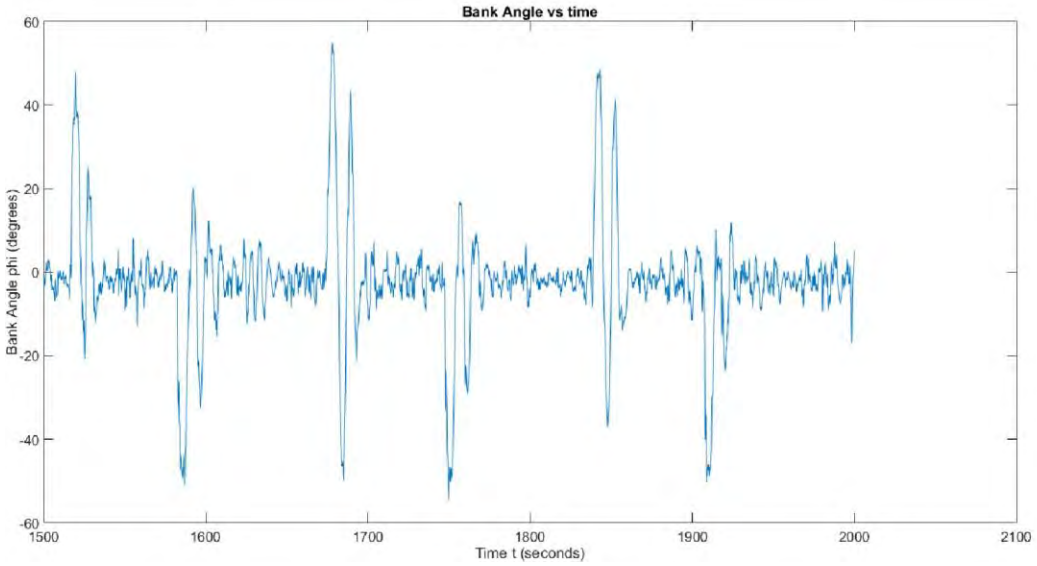


Figura 2. 15 Información del IMU sobre el ángulo de alabeo en un tiempo de 2000 segundos.

Fuente: Software MATLAB

Se utiliza el software MATLAB ® para obtener la data del sensor, segmentarla y graficarla utilizando el código que se puede encontrar en el anexo 01. Lo importante es saber interpretar la información obtenida, a continuación, algunas consideraciones importantes sobre la figura 2.16.

- Los ángulos negativos son giros a la izquierda y los positivos son giros a la derecha.
- Cada cierto tiempo se observan "picos" positivos, negativos o en ambas direcciones, estos indican los giros de la aeronave.
- En los espacios de tiempo donde no se han realizado giros, es decir, en los momentos en los cuales la aeronave volaba de manera nivelada, la figura muestra que la aeronave ha realizado giros menores o correcciones por posibles perturbaciones para mantener la ruta de vuelo.
- Se observa que la data obtenida del sensor tiene ruido que debe de ser filtrado, ya que en la zona de vuelo nivelado la aeronave puede realizar correcciones, pero no del tipo como se observan en la figura 2.15. La figura 2.16 y 2.17 muestran la relación de la data con el vuelo realizado y como se debería de ver la data filtrada obtenida del sensor.

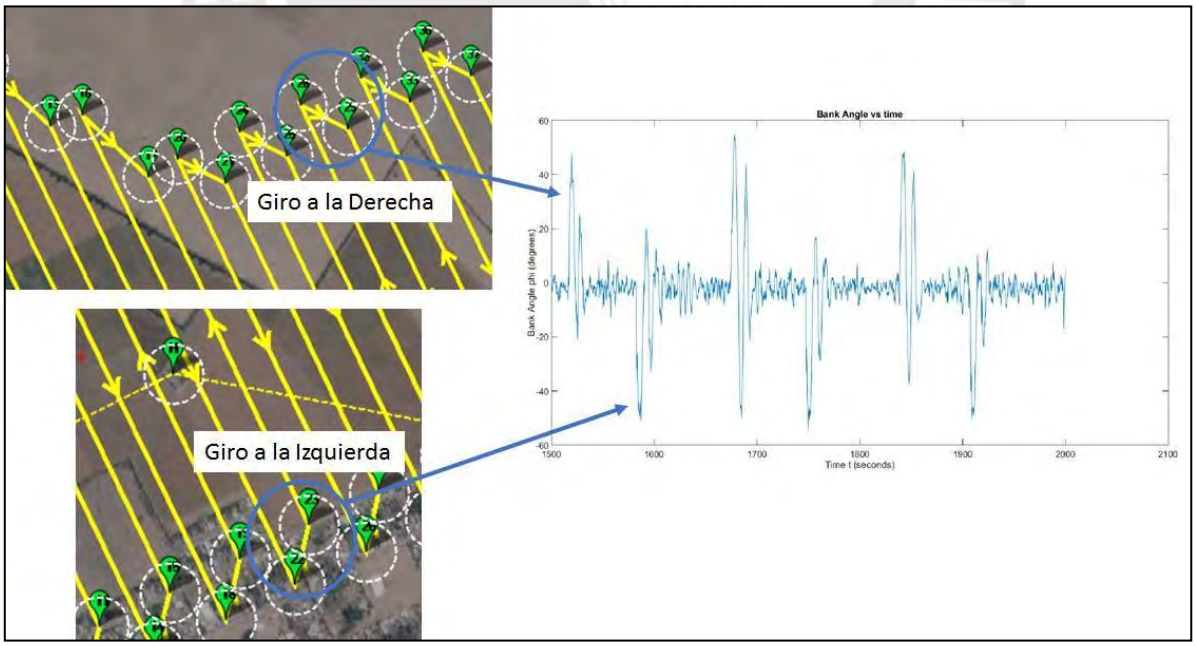


Figura 2. 16 Relación entre la data obtenida del sensor IMU y el vuelo realizado

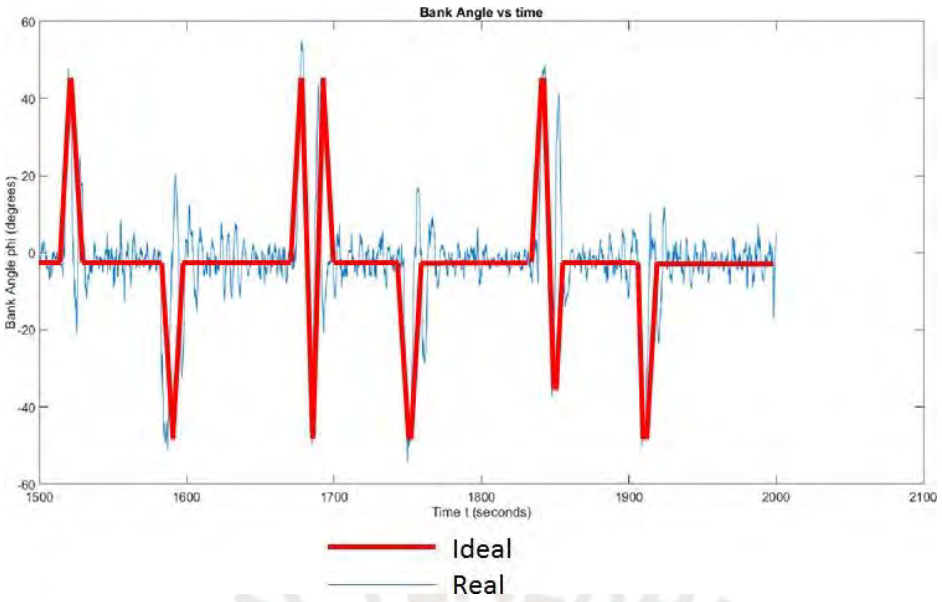


Figura 2. 17 Data Real vs. Data Ideal

En la figura 2.10 se aprecia el ruido en la data que es extraída del sensor. Por lo tanto, se aplicó un filtro digital antes de procesar la información durante el entrenamiento de la RNA.

2.5. Información de Velocidad y Ángulos de la Aeronave.

2.5.1 Medición de la Velocidad

La señal de velocidad proviene del tubo pitot, que es una herramienta que se utiliza para medir la diferencia entra la presión total y la presión estática. Esta diferencia se traduce en presión dinámica que finalmente se convierte en la medición de la velocidad de vuelo.

La siguiente figura muestra la señal que se obtiene del sensor de velocidad (tubo pitot). La señal (línea morada) cuando el aire no está en contacto con el tubo pitot muestra un valor constante en este caso 7m/s. Sin embargo, cuando ingresa aire por el tubo pitot y las mangueras al sensor electrónico, se muestra una medida de velocidad en la pantalla.

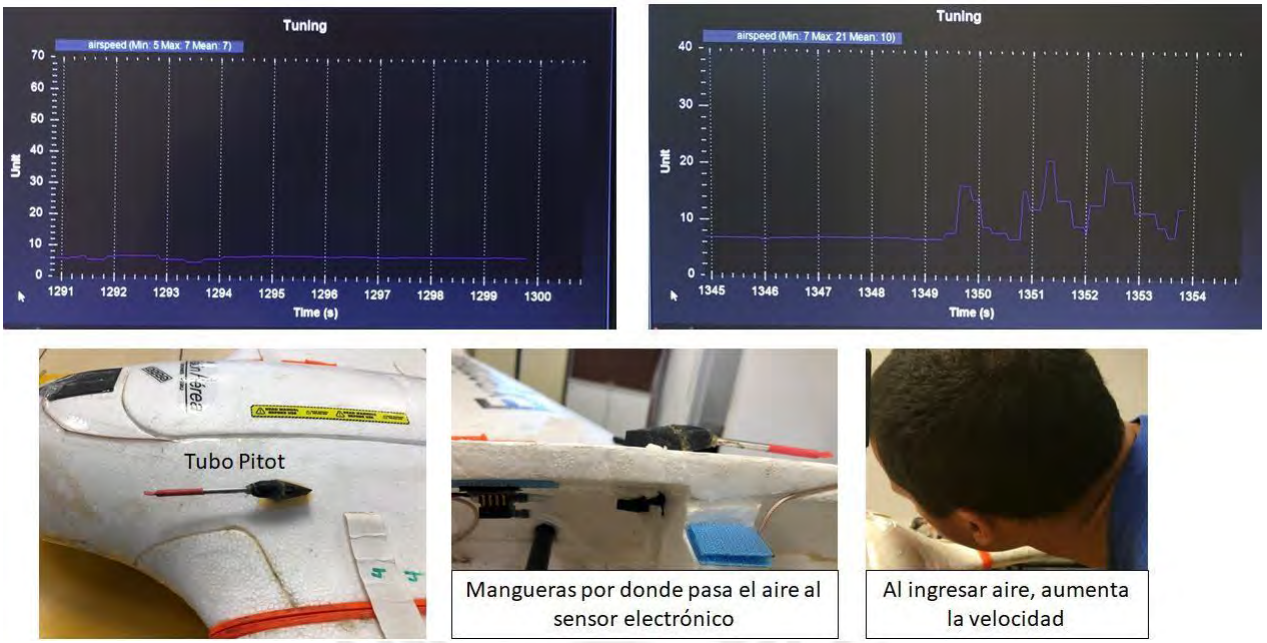


Figura 2. 18 Tubo Pitot y Señal de Velocidad

2.5.2 Giróscopo

El giroscopo es el sensor que mide los movimientos de la aeronave cuando esta rota sobre los ejes descritos en la figura 1.12. La figura 2.19 muestra la medición de este sensor (linea azul) cuando la aeronave realiza un cabeceo positivo (valores positivos) y la figura 2.20 cuando la aeronave realiza un cabeceo negativo (valores negativos).

En ambos casos, la señal se desplaza por un instante de tiempo y luego regresa al estado neutral de equilibrio, lo que significa que el sensor solo mide el instante de movimiento y luego ese desplazamiento se vuelve neutral hasta que ocurre otra perturbación.

Figura 2. 19 Señal de Giroscopo en Cabeceo Positivo

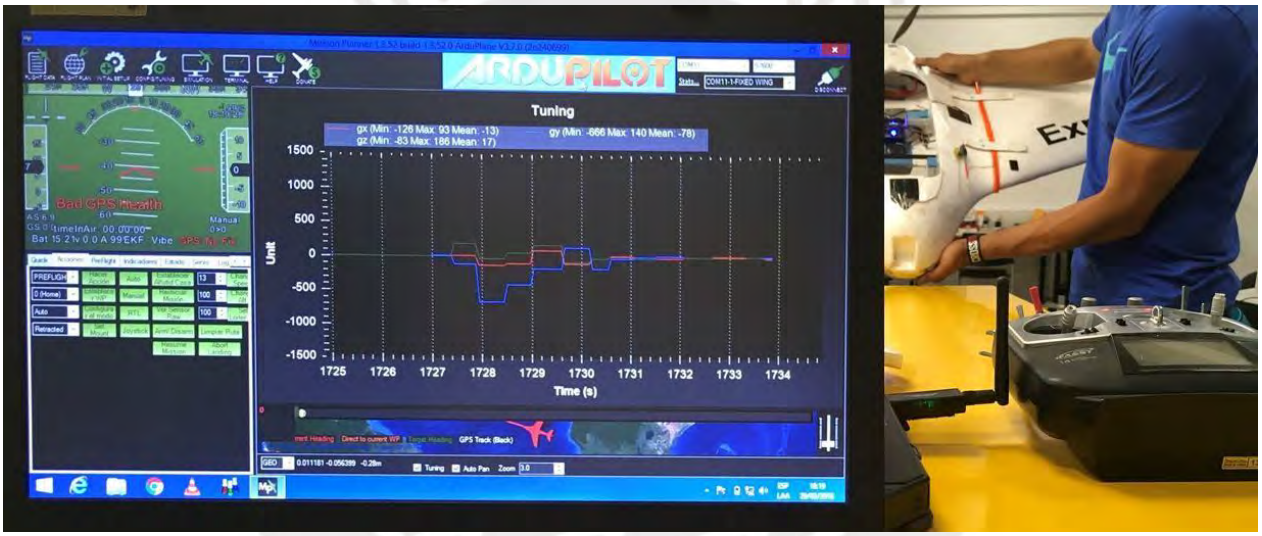


Figura 2. 20 Señal de Giroscopo en Cabeceo Negativo

La figura 2.21 y 2.22 muestra la señal del giróscopo cuando realiza un alabeo a la derecha o izquierda. La señal (línea roja) muestra que un alabeo a la derecha tiene valores positivos y uno a la izquierda tiene valores negativos.

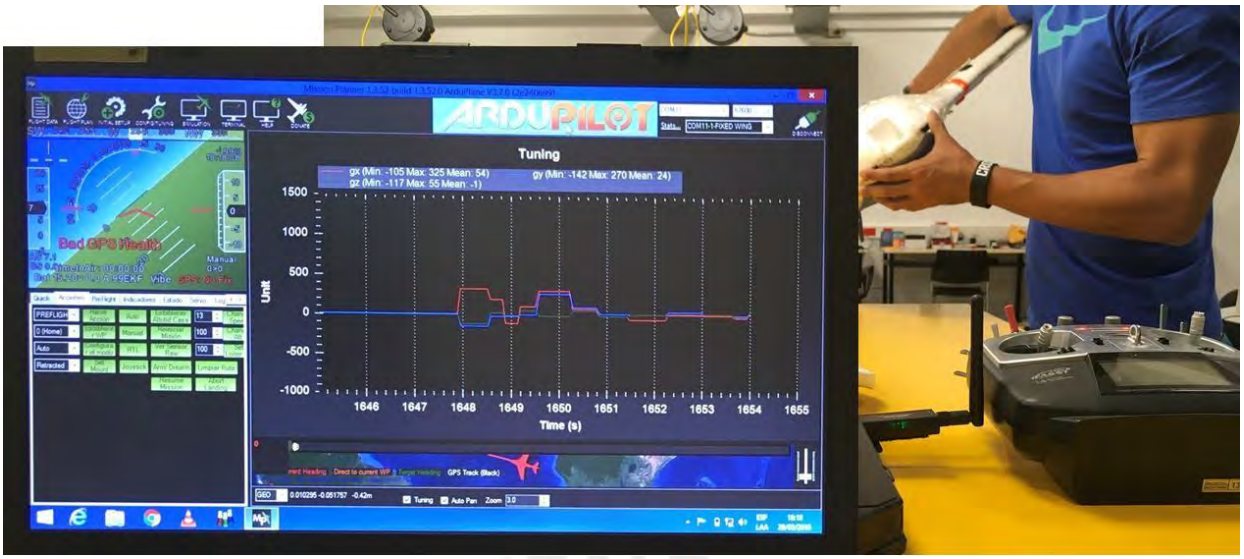


Figura 2. 21 Señal de Giróscopo en Alabeo a la Derecha

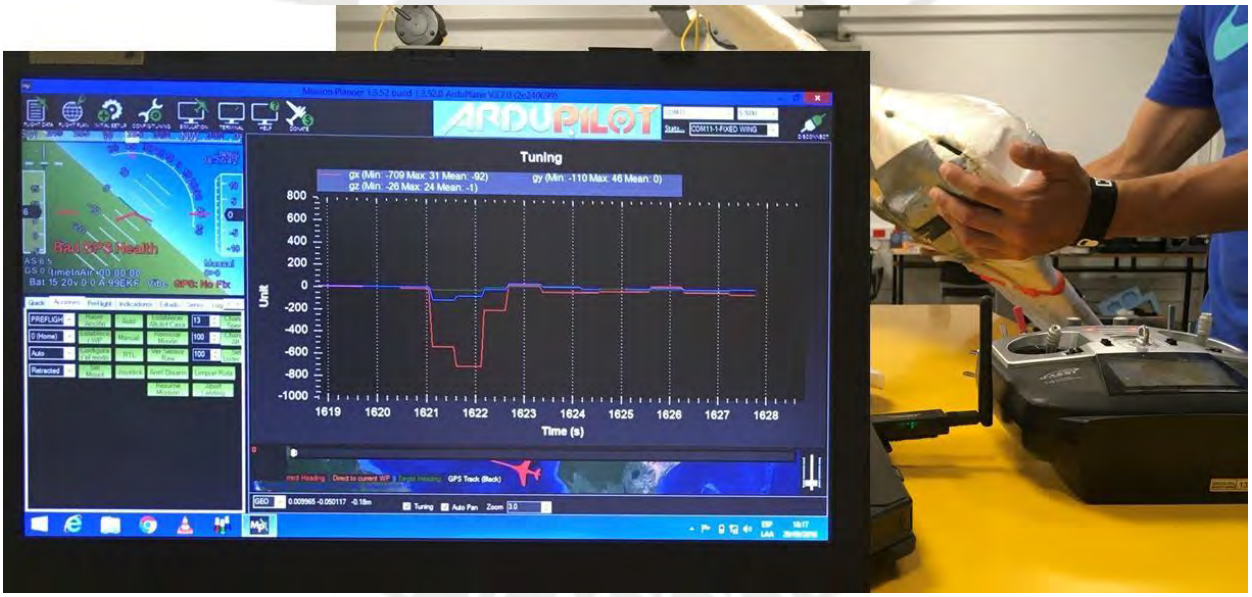


Figura 2. 22 Señal de Giróscopo en Alabeo a la Izquierda

2.5. Diseño de la Red Neuronal Artificial Dinámica.

Este trabajo de investigación utiliza una RNA Dinámica de dos capas intermedias con retroalimentación porque durante la etapa de entrenamiento otorga resultados confiables para la identificación del modelo dinámico de la aeronave.

Durante el proceso de diseño de la RNA se debe de tomar en consideración que esta aeronave es del tipo ala volante, como se ha visto en el capítulo 01. Esto quiere decir que, solo dos superficies controlan todos los movimientos de la aeronave y no posee estabilizador vertical. A diferencia de una aeronave convencional que utiliza cuatro superficies móviles para controlar la aeronave. En este sentido, y como se ha visto en la sección anterior, las señales de control se relacionan y mezclan. Entonces, estas no pueden ser separadas según el eje donde actúan para realizar el entrenamiento de la RNA, como se realiza en otros trabajos de investigación En el documento "Aircraft System Identification using Artificial Neural Networks with Flight Test Data", Harris menciona, los ejes longitudinal y lateral / direccional pueden ser separados y entrenados de manera separada para aeronaves convencionales en ángulos de alabeo cercanos a cero. Asimismo, la dinámica de la altitud no necesita ser aprendida por el algoritmo. También, en el documento "The Development of Neural Networks Techniques for the System Identification of Aircraft Dynamics", John M. Wharington menciona que, la determincación de los parámetros de entrada del modelo va a depender del tipo de aeronave, particularmente de las entradas de las superficies de control y el motor.

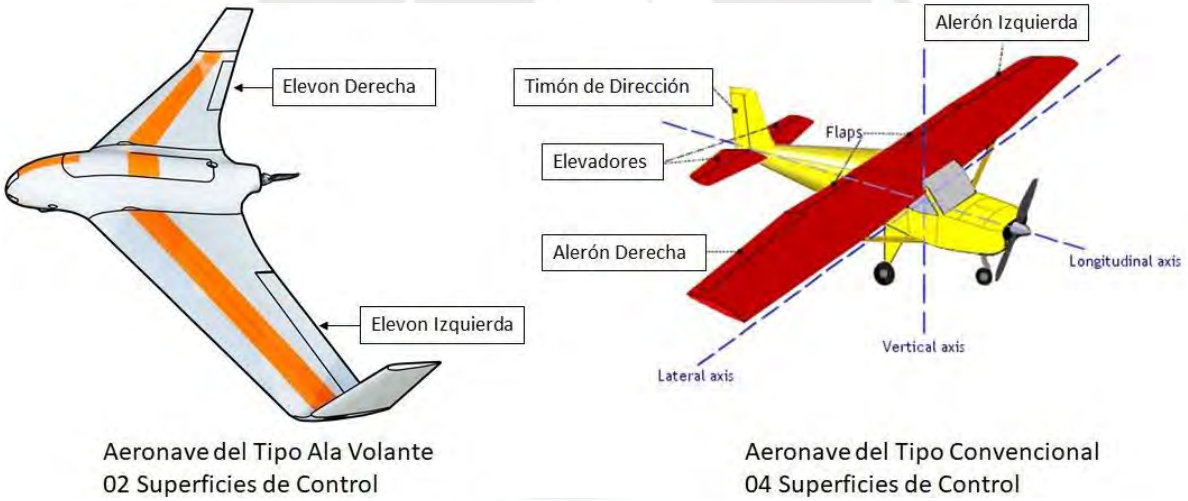


Figura 2. 23 Superficies de Control en Cada Aeronave

Fuente: Propia

Asimismo, otros trabajos de investigación utilizan una RNA lineal para realizar el entrenamiento, como es el caso en el documento "Aircraft System Identification Using Artificial Neural Networks", el autor realizó explica lo realizado de la siguiente manera: es

posible determinar el tiempo discreto de las matrices A y B (sistema lineal) para los movimientos longitudinales y laterales / direccionales en una aeronave dado un modo de vuelo específico. Para determinar el tiempo continuo de las matrices A y B se requiere realizar una transformación utilizando el periodo de muestreo de la data que está siendo procesada. Utilizar una RNA lineal para identificar el modelo de un sistema dinámico como una aeronave no es recomendable. Este trabajo de investigación utiliza un RNA dinámica para realizar el entrenamiento y predicción del modelo dinámico de la aeronave. John M. Wharington lo menciona: la estimación de parámetros es un "subset" del proceso de identificación de la dinámica de la aeronave. Las técnicas de estimación de parámetros de las redes neuronales no están ligadas a una configuración especifica de vuelo, pero requieren que todas las entradas al sistema sean variadas de una manera que es representativa para todo el vuelo.

La figura 2.24 muestra el flujo de información de una RNA estática y dinámica.

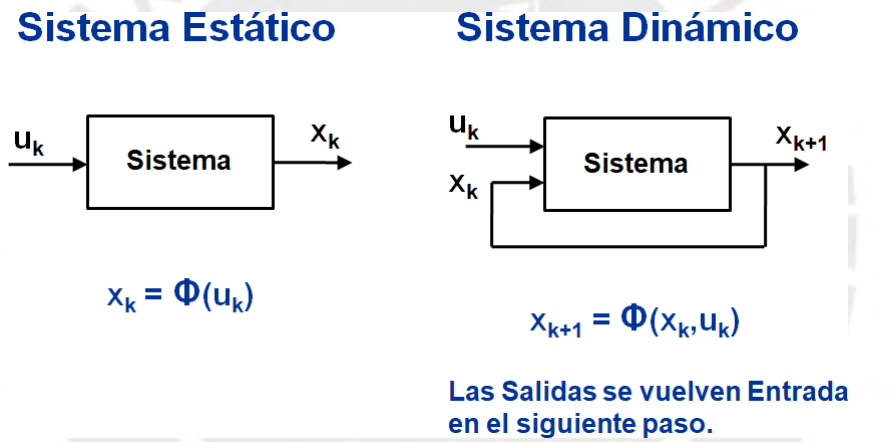


Figura 2. 24 Entrenamiento Sistema Estático Vs. Entrenamiento Sistema Dinámico[8]

2.5.1 Proceso de Entrenamiento de un RNA Dinámica

A continuación, se explica como una RNA Dinámica realiza el entrenamiento y predicción del modelo dinámico de un sistema. La siguiente figura muestra el flujo de información, las variables de entrada (u) y las variables de salida deseada.

[8] Presentación Systems Modeling and Control Using Dynamic Neural Networks and Fuzzy-Neural Networks. Dr. António Moran.

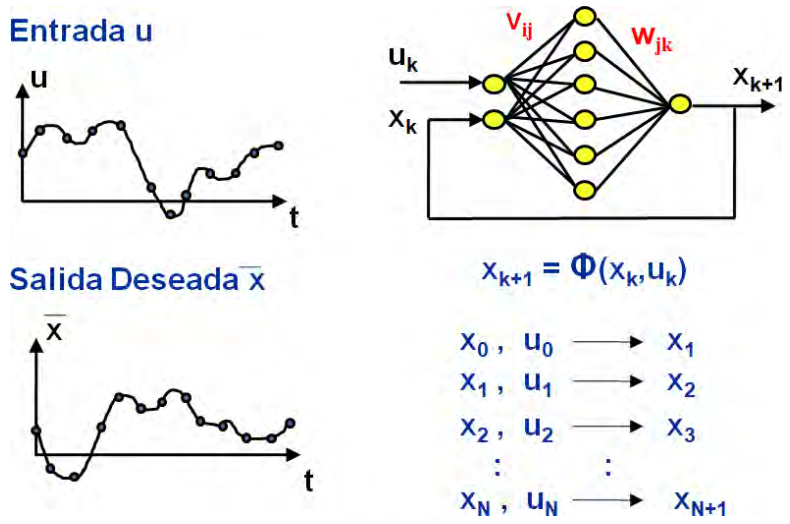


Figura 2. 25 Entrenamiento de una RNA Dinámica: Entrada (u) y Salida Deseada (x)[8]

La RNA Dinámica está preparada para realizar el entrenamiento si existen múltiples entradas y múltiples salidas deseadas. La figura 2.26 muestra el entrenamiento para múltiples entradas.

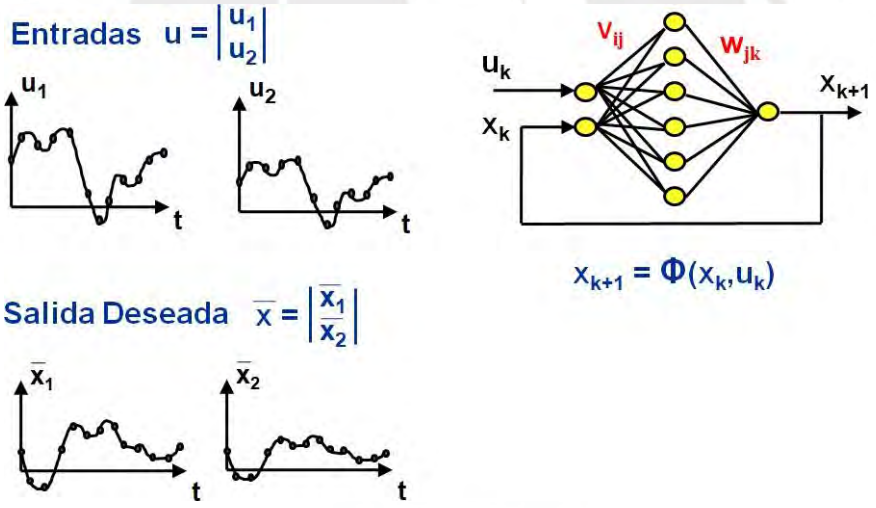
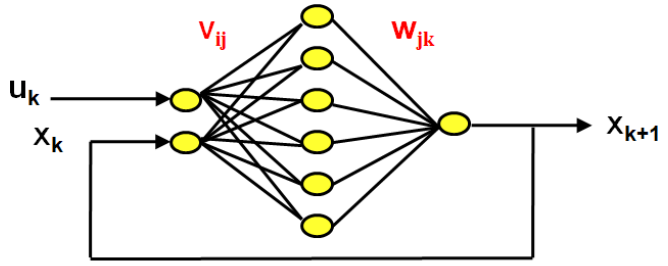


Figura 2. 26 RNA Dinámica con Múltiples Enstradas y Múltiples Salidas[9]

[9] Presentación Systems Modeling and Control Using Dynamic Neural Networks and Fuzzy-Neural Networks. Dr. António Moran.

El entrenamiento debe de ser eficiente y efectivo y para eso se necesita minimizar la función de costo de la RNA a medida que la información fluye durante el entrenamiento. La figura 2.27 muestra la función de costo y la información que se utiliza para ser calculada.



Función de Costo que debe de ser Minimizada

If x is a vector $x = \begin{vmatrix} x_1 \\ x_2 \end{vmatrix}$

$$J = 0.5\,(x_1-\bar{x}_1)^2 + 0.5\,(x_2-\bar{x}_2)^2 + \cdots\cdots + 0.5\,(x_N-\bar{x}_N)^2$$

$$J = 0.5 \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^2$$

$x_k \rightarrow$ Estado (Salida) de la red

$\bar{x}_k \rightarrow$ Salida deseada (data)

Figura 2. 27 Función de Costo de la RNA Dinámica

La derivada parcial total de la función de costo sobre la derivada parcial total de cada una de las funciones de los pesos utilizados (Vij , Wjk) para el entrenamiento es utilizada en el entrenamiento y cálculo de los pesos en el siguiente paso. La figura 2.28 muestra la información descrita.

La figura 2.29 muestra el proceso de entrenamiento de la RNA Dinámica a lo largo del tiempo. Las variables de salida, luego del primero paso de entrenamiento, ingresan nuevamente en el siguiente paso como variables de entrada junto con nueva información de las variables de entrada. Estos pasos se ejecutan tantas veces como sea necesario que el error de entrenamiento se haya minimizado lo suficiente para tener una predicción eficiente del modelo dinámico del sistema.

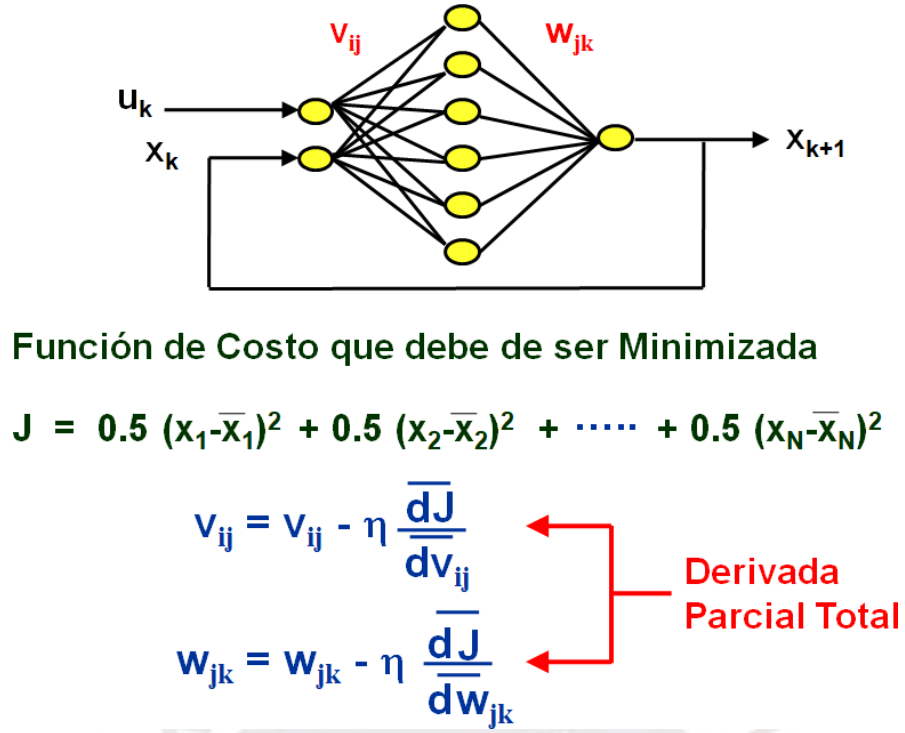


Figura 2. 28 Cálculo de los pesos (Vij , Wjk) utilizando la Derivada Parcial Total de la Función de Costo.

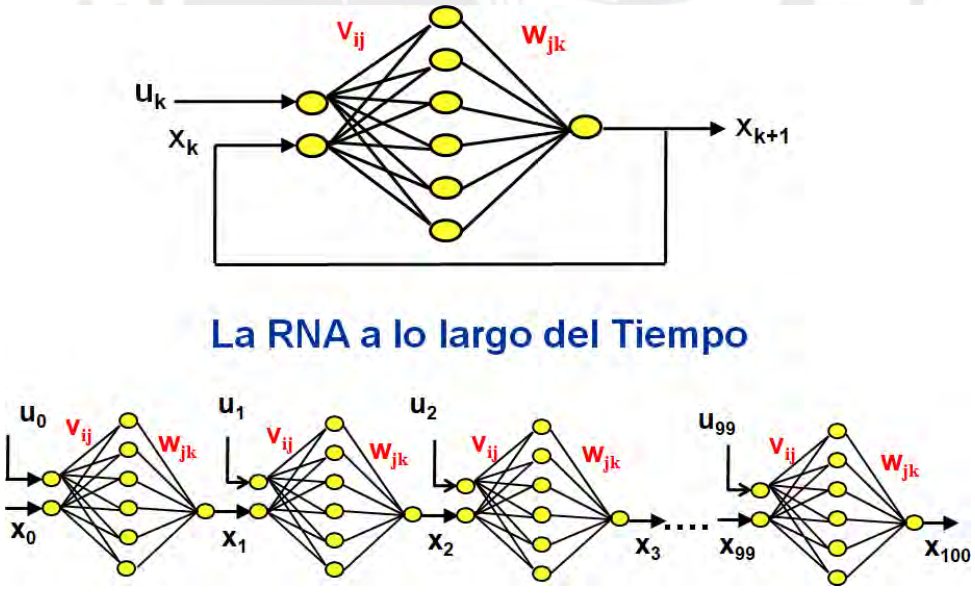


Figura 2. 29 RNA Dinámica a lo largo del Tiempo

2.5.2 RNA Dinámica Utilizada

La figura 2.30 muestra el esquema de la RNA utilizada. En el anexo 02 se muestra el código utilizado en MATLAB implementado la RNA.

Los valores de entrada de la RNA son el vector de entrada, que contiene información que se alimenta al sistema durante el vuelo de la aeronave y las salidas, que son los valores del estado de la aeronave durante el vuelo que predice la RNA, que se retroalimentan para optimizar el cálculo del modelo dinámico de la aeronave. El vector de entradas se muestra en la tabla 2.1, no se utiliza el parámetro de guiñada porque la aeronave no cuenta con un timón de dirección para el control de este eje.

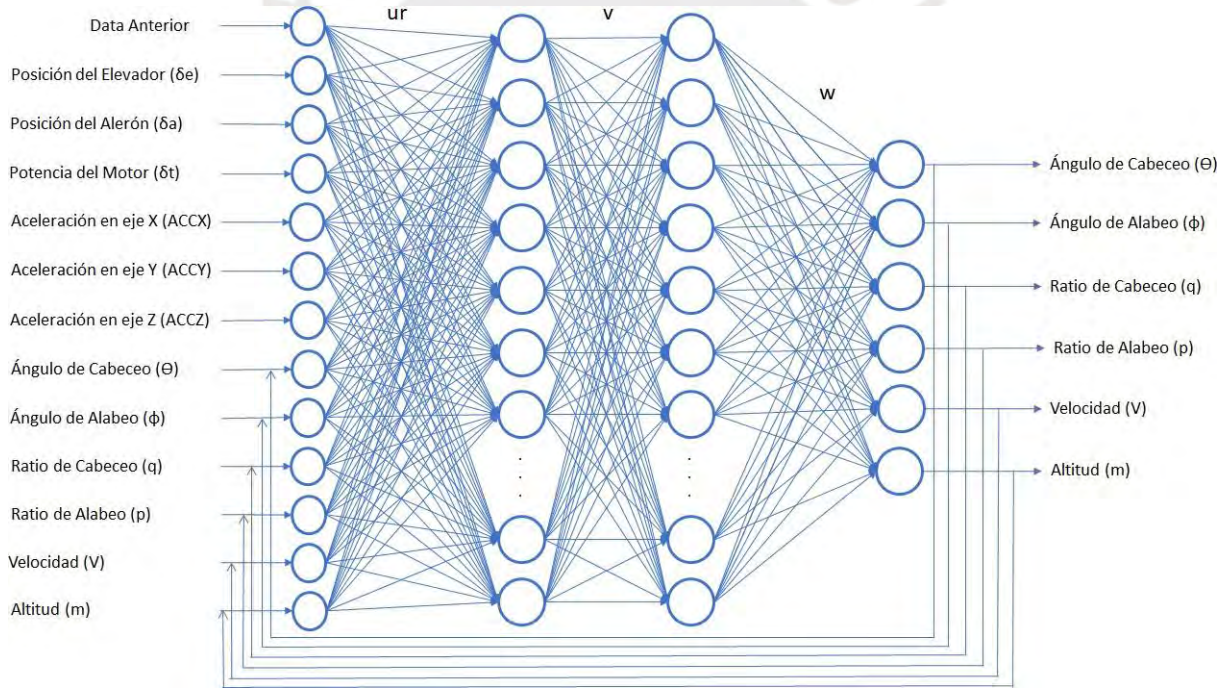


Figura 2. 30 RNA de dos capas

Esta es una RNA del tipo Back Propagation Dinámico y es la que se utilizó para realizar el entrenamiento y la predicción del modelo dinámico de la aeronave.

Tabla 2. 1 Vector de Entrada a la Red Neuronal

Nombre	Símbolo
Posición del Elevador (grados)	δe
Posición de los Alerones (grados)	δa
Empuje del motor en porcentaje de 0% a 100%	δt
Aceleraciones en eje X en m/s^2	ACCX
Aceleraciones en eje Y en m/s^2	ACCY
Aceleraciones en eje Z en m/s^2	ACCZ

Las entradas son los estados de la aeronave que se pueden controlar como es el caso de la posición de las superficies de control y el motor.

Las salidas de la RNA se muestran en la tabla 2.2

Tabla 2. 2 Vector de Salida de la Red Neuronal

Nombre	Símbolo
Ángulo de Cabeceo (Pitch) (grados)	Θ
Ángulo de Alabeo (Roll) (grados)	φ
Ratio de Cabeceo (grados/sec)	q
Ratio de Alabeo (grados/sec)	p
Velocidad en m/s	V
Altitud en m	h

El vector de salida, que se retroalimenta en la RNA para mejorar la predicción del modelo dinámico de la aeronave está compuesto por los ángulos de rotación, la velocidad a la cual estos ángulos cambian en el tiempo, la velocidad y altitud de vuelo. Esta es la información necesaria para el entrenamiento de la RNA y poder modelar el sistema de navegación para controlar el vuelo.

Las capas intermedias utilizan una función sigmoidea bipolar para realizar los cálculos con los pesos (ur, v, w) de la RNA. La función sigmoidea bipolar se muestra en la figura 2.24. La ecuación de la función sigmoide se muestra a continuación:

$$f(x) = \frac{2}{(1+e^{\frac{-(m-c)}{a}-1})}, \qquad 25$$

En donde los valores de “c” y “a” son los valores asignados al centro y pendiente de la curva respectivamente. Si estos valores varían, el punto de origen y la forma de la curva varía.

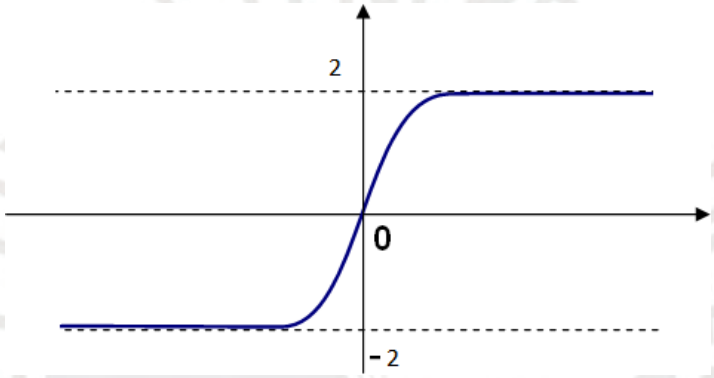


Figura 2. 31 Función Sigmoidea Bipolar

Fuente Propia

Este tipo de función permite la predicción de un modelo dinámico como es el caso de la aeronave. También se puede utilizar otro tipo de funciones como por ejemplo funciones lineales o sigmoideas unipolares, pero durante el entrenamiento de la RNA esta función otorgaba mejores resultados de predicción.

2.6. Preparación de la Información y Entrenamiento de la RNA.

Antes de iniciar el procesamiento en la RNA, se debe de realizar un pre – procesamiento de la información para que la RNA pueda realizar una correcta predicción. Esta etapa previa se refiere a filtrar la data para eliminar cualquier ruido que pueda perjudicar el entrenamiento de la RNA.

2.6.1. Filtrado de la Información

La información que se obtiene de los sensores por lo general posee perturbaciones propias del sensor, a las cuales se le denomina "ruido electromagnético" o simplemente "ruido". En la sección 2.2, la figura 2.14 muestra cómo debería de ser la señal sin el ruido característico de los sensores. Es posible eliminar el ruido de la señal utilizando un filtro digital.

Se utilizo un filtro digital del tipo "zero – phase filter" para eliminar el ruido de las señales de los sensores. Este tipo de filtro es utilizado porque no se tiene una distorsión de la señal al momento de ser filtrada. La señal es filtrada en ambas direcciones, al frente y en reversa, lo cual hace que no haya distorsión de la señal y al mismo tiempo hace que la señal filtrada inicie y termine en el mismo punto que la señal original.

Por ejemplo, la señal del ángulo de alabeo o "roll angle" se muestra en la figura 2.32 sin filtro. Esta señal tiene una frecuencia de corte de 0.07Hz, la cual es calculada utilizando una serie de Fourier.

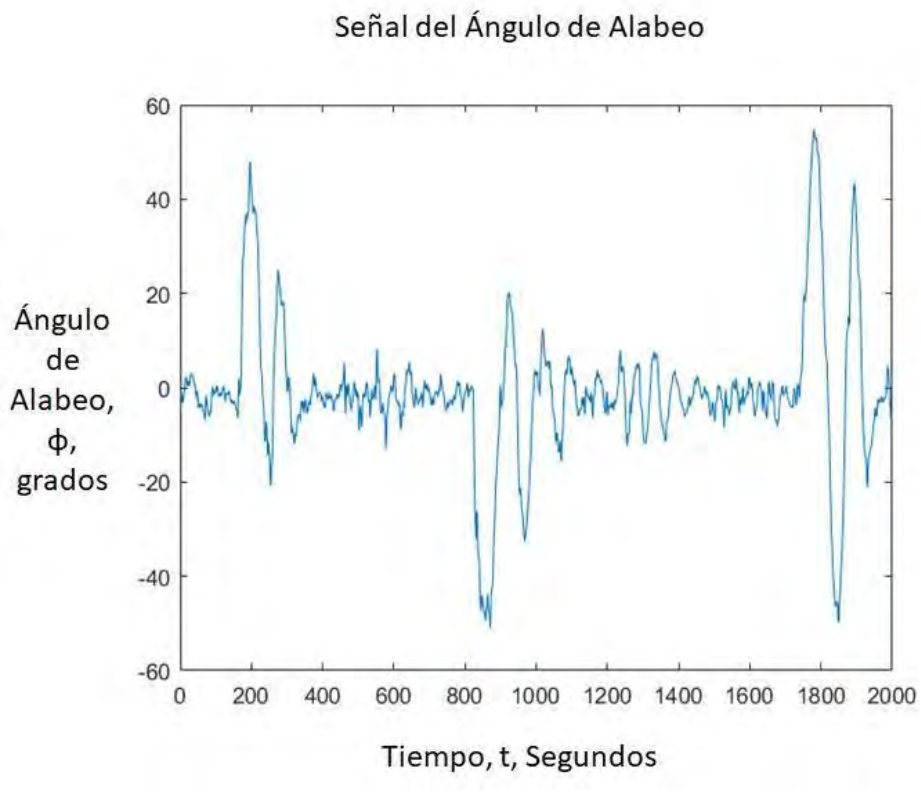


Figura 2. 32 Señal del Ángulo de Alabeo o "Roll Angle" sin filtrar

Si se utilizara un filtro del tipo “pasa bajo” la señal filtrada se vería de la siguiente manera, señal filtrada en color naranja, figura 2.33.

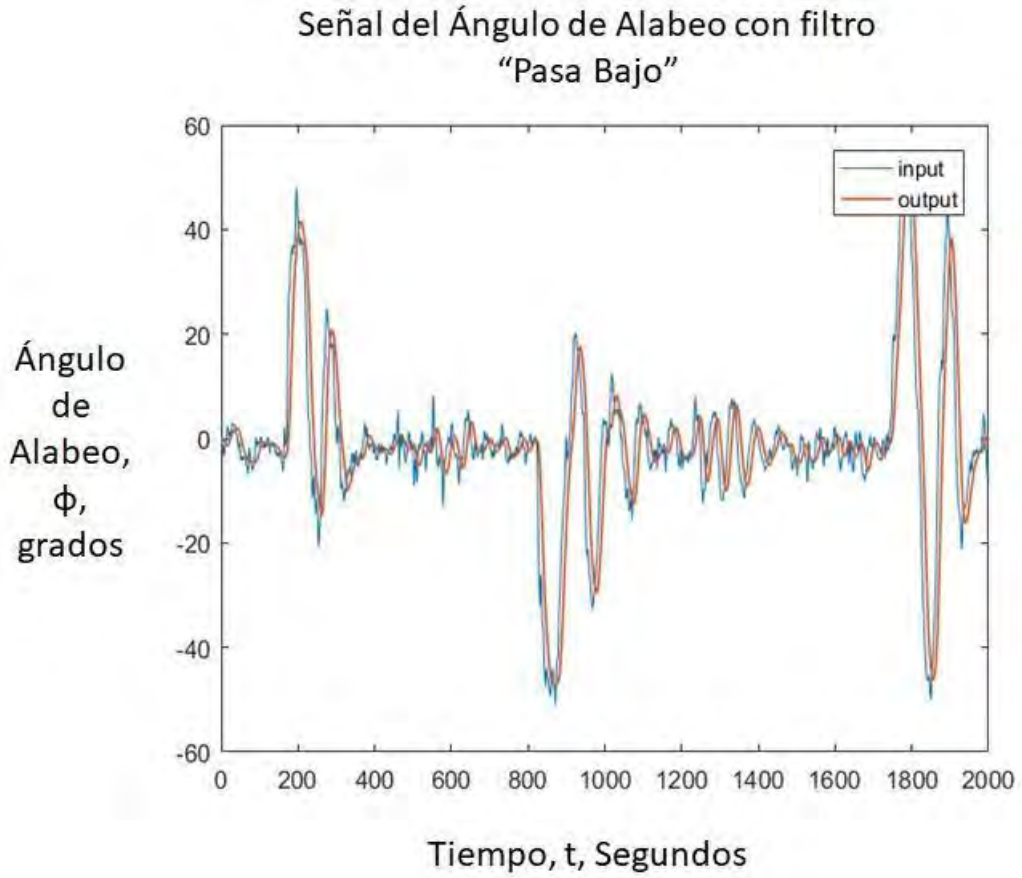


Figura 2. 33 Señal Filtrada sin el Filtro “Zero Phase Filter”

En la figura 2.33 se aprecia que la señal filtrada está desfasada con respecto a la señal original. Por lo tanto, no se puede considerar para el entrenamiento de la RNA. Al aplicar el “Zero Phase Filter” la señal filtrada es corregida y el desfase se vuelve cero. El resultado se muestra en la figura 2.34

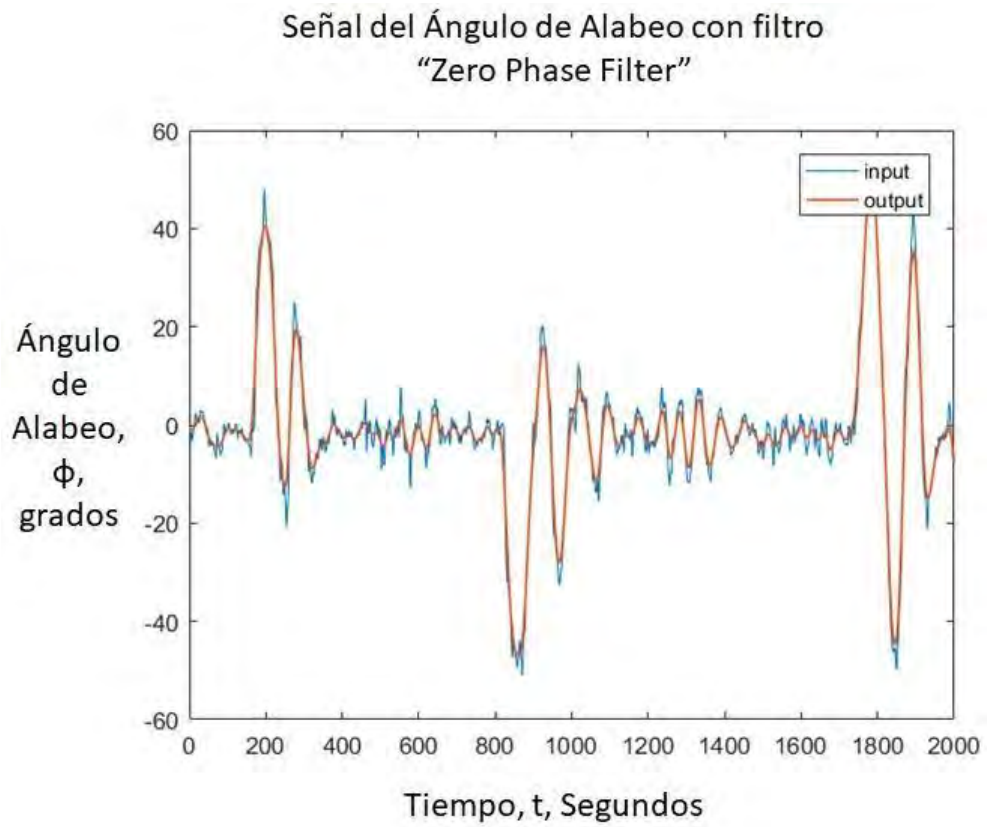


Figura 2. 34 Señal Filtrada con el Filtro Digital "Zero Phase Filter"

Se utilizo la función "filtfilt" del programa MATLAB ® para realizar el filtrado digital "Zero Phase Filter" El código implementado se encuentra en el anexo 03 para el ángulo de alabeo o "roll angle". Sin embargo, este código fue aplicado para todas las señales que se utilizan en la RNA.

Capítulo 03

EL CONTROLADOR DE VUELO

En este capítulo se explica cómo funciona el controlador de vuelo de la aeronave y como el modelo que se ha obtenido puede interactuar con este controlador.

3.1. Controlador de vuelo.

El cerebro de un UAV se denomina controlador de vuelo o autopiloto. El autopiloto, en combinación con otra aviónica, utiliza sensores abordo para estimar la posición y orientación del UAV, realiza el control del vuelo al traducir los comandos de vuelo en comandos para los servomotores de control de la aeronave. Los autopilotos realizan navegación por coordenadas (waypoint navigation), ajuste a la ruta de vuelo durante el vuelo, comunicación entre la aeronave y la estación de control en tierra por telemetría, despegue y aterrizaje autónomo, control y configuración del vuelo por medio de un software y características robustas contra fallas.

La figura 3.1 muestra un diagrama de bloques simplificado que incluye la estimación de estados, algoritmos de control de vuelo y un controlador de misiones. La estimación de estados, como se ha mencionado en el capítulo anterior, consiste en lo que los sensores monitorean y estiman de la cinemática de la aeronave. El algoritmo de control de vuelo automático manipula la potencia y servomotores del control de superficies para realizar los comandos de navegación utilizando la cinemática de la aeronave como retroalimentación. Los comandos de navegación son generados en el control de misión.

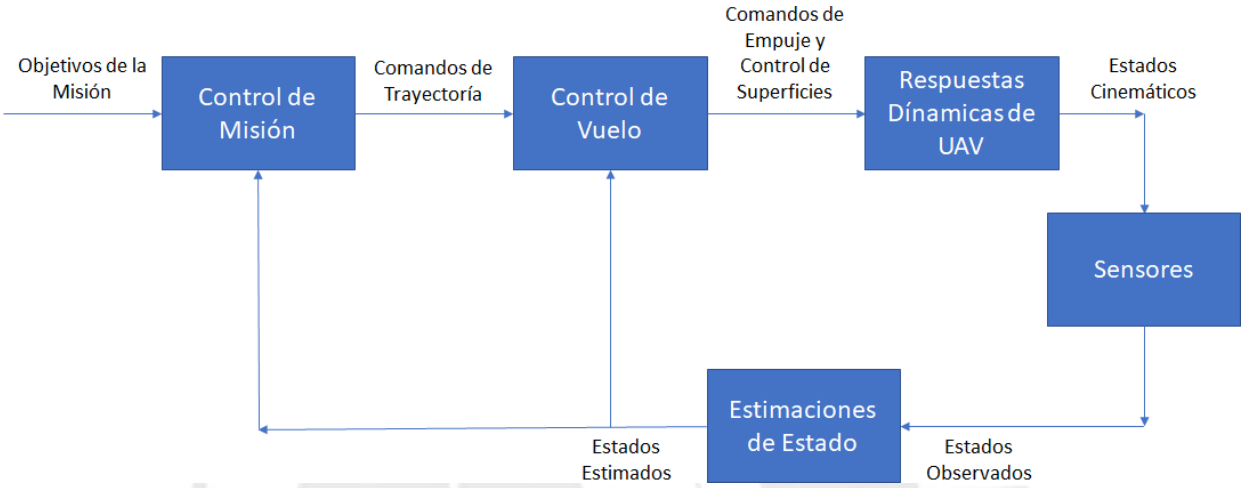


Figura 3. 1 Diagrama Simplificado de Control de Vuelo Autónomo.

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

3.2. La Navegación de la Aeronave

Según lo mencionado en el capítulo 2, el control de vuelo PIXHAWK combina las mediciones del IMU y GPS en un sistema de navegación inercial más completo, como se muestra en la figura 3.2, donde ψ_{ref} se obtiene del sistema de referencia de actitud y rumbo (AHRS, por las siglas en ingles). Este tipo de navegación utiliza un EKF, como se ha explicado en la lógica de control del PIXHAWK.

El algoritmo de control de vuelo compara los comandos de vuelo con los valores estimados y genera los comandos de los canales de control de vuelo. Los cuatro canales de control de vuelo son potencia, elevador, alerones y timón de dirección, sin embargo, como se ha explicado la aeronave que analizaremos no tiene timón de dirección.

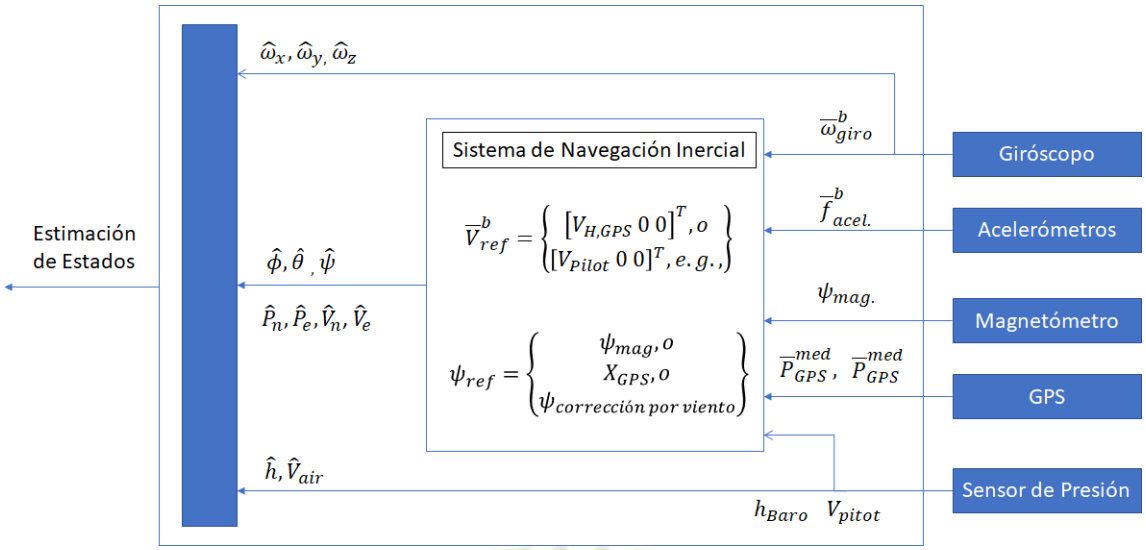


Figura 3. 2 Estimación de Estados: GPS / INS. El EKF es utilizado para fusionar giróscopos, acelerómetros y mediciones de un magnetómetro con GPS para estimar los estados de rotación y traslación. El sensor de presión es utilizado de manera directa para altura y velocidad.

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

El algoritmo de control de vuelo se separa en dos:

- Algoritmo de control lateral para comandos de giros y rutas.
- Algoritmo de control longitudinal para comandos de altitud y velocidad.

La mayoría de UAVs comercial utiliza métodos de control clásico donde un comando de vuelo, por ejemplo, velocidad deseada, es alcanzado al ajustar una señal de control, en este caso empuje del motor, para minimizar el error de retroalimentación entre el comando de vuelo y el valor alcanzado. El método más común es el control proporcional – integral – derivativo (PID), como el que utiliza el controlador de vuelo PIXHAWK. La señal de control es formada como una combinación lineal del error de retroalimentación, la integral de ese error y la derivada de ese error.

La figura 3.3 muestra la implementación del controlador de velocidad utilizando la potencia del motor. La respuesta deseada en un tiempo prudente y con mínima oscilación de la velocidad se puede lograr ajustando los valores escalares de las ganancias K_P, K_I y K_D.

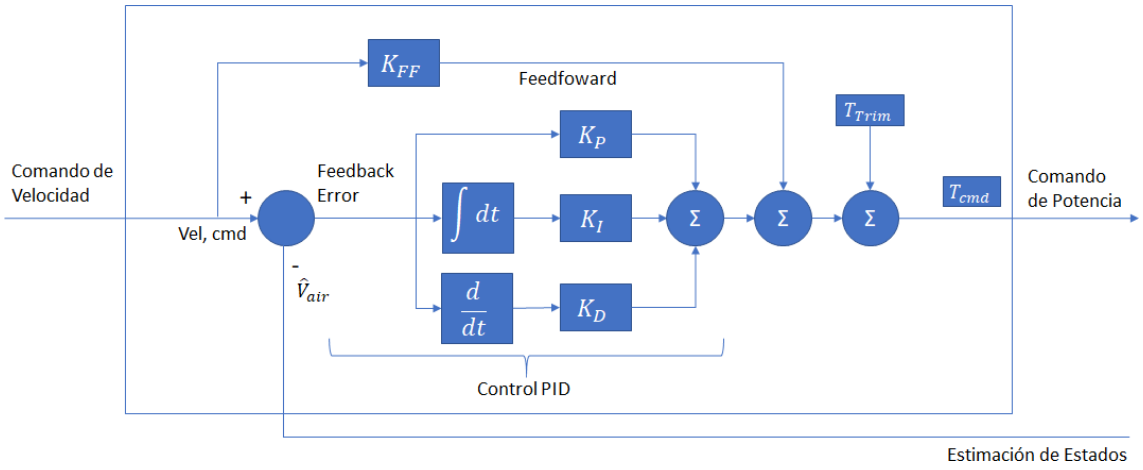


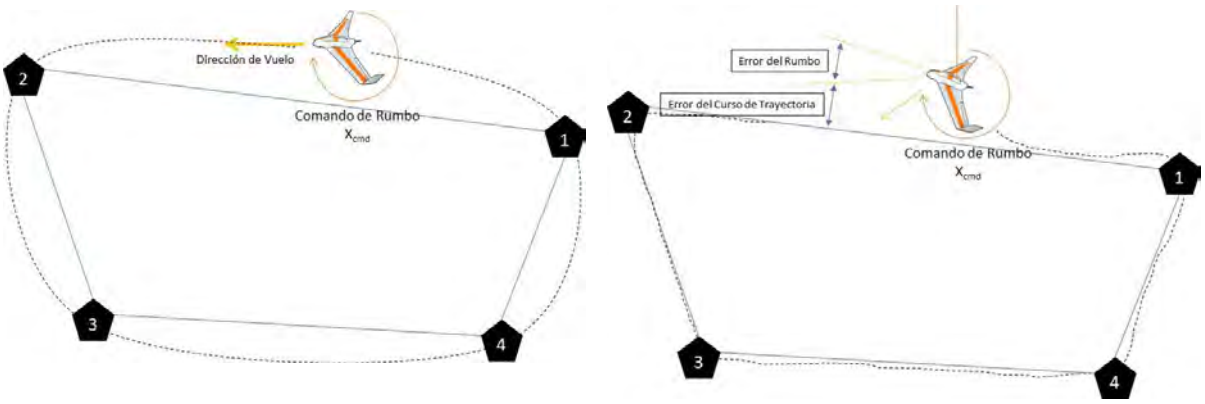
Figura 3. 3 Controlador de velocidad utilizando la potencia del motor. La figura muestra un compensador aumentado de PID para controlar la potencia y alcanzar la velocidad deseada.

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

3.2.1. Control de Vuelo Lateral

Los comandos de vuelo lateral son entregados al autopiloto como segmentos de trayectoria deseada referenciados por waypoints. Un UAV puede volar de dos maneras, directamente a los puntos de chequeo o utilizar una lógica de control para minimizar el error y seguir la trayectoria de vuelo entre dos waypoints. Cuando vuela directamente al punto de chequeo el comando de curso es el curso que debe de seguir la aeronave para llegar al siguiente waypoint. Cuando vuela siguiendo la trayectoria, el comando de curso es una función del error de la trayectoria que debe de seguir y el error del rumbo. La figura 3.4 muestra la diferencia entre ambos vuelos.

El algoritmo de control de trayectoria horizontal convierte los errores de la trayectoria en comandos de rumbo (X_{cmd}). Este comando de rumbo es comparado con el rumbo estimado del UAV ($\hat{\chi}$) para generar un comando de alabeo o roll (Φ_{cmd}). El comando de alabeo alimenta el algoritmo de control de estabilidad horizontal, que genera comandos en los alerones ($\delta_{A,cmd}$) o en el timón de dirección ($\delta_{R,cmd}$) utilizando retroalimentación de los valores estimados de roll ($\hat{\phi}$) y roll-rate o el cambio del roll en el tiempo ($\hat{p}$). El UAV automáticamente controlara el vuelo para alcanzar la ruta deseada. La figura 3.5 ilustra lo descrito.



(a) *(b)*

Figura 3. 4 (a) El UAV vuelo directo a cada Waypoint o (b) el UAV vuela siguiendo el segmento de trayectoria entre cada Waypoint.

Fuente Propia

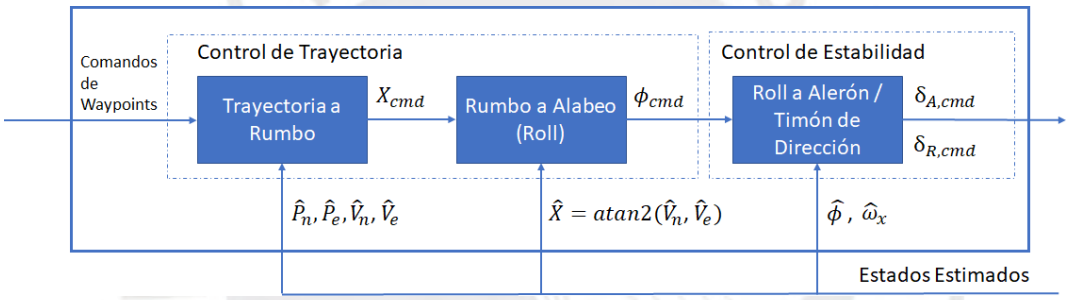


Figura 3. 5 El control de vuelo lateral utiliza los estados estimados como retroalimentación para convertir los comandos de waypoints en comandos de alerones o timón de dirección.

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

3.2.2. Control de Vuelo Longitudinal

En el control longitudinal, la combinación de elevador y potencia del motor es utilizada para controlar la altura y velocidad. Un cambio en la altitud manteniendo la misma potencia del motor inducirá un cambio en la velocidad. Un cambio en la velocidad con un ángulo de cabeceo fijo inducirá un cambio en la altitud porque la magnitud de la sustentación cambia al cambiar la velocidad. Las siguientes dos figuras 3.6 y 3.7 ilustran el control de vuelo longitudinal.

En la figura 3.6 el error de altitud es utilizado para generar un comando de cabeceo (Θ_{cmd}), que genera un comando de elevador utilizando los estados apropiados de retroalimentación.

El error de velocidad es regulado directamente con el comando de potencia. En la figura 3.7, es el error de velocidad que genera un comando de cabeceo, el cual como respuesta genera un comando de elevador utilizando la retroalimentación apropiada. El error de altitud es regulado utilizando el comando de potencia.

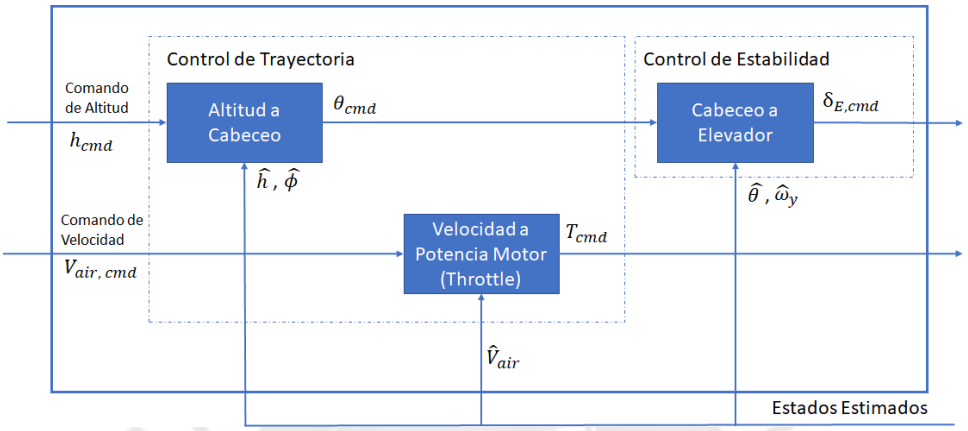


Figura 3. 6 Control de Vuelo Longitudinal donde el cabeceo es utilizado para controlar la altitud y la potencia del motor para controlar la velocidad de vuelo.

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

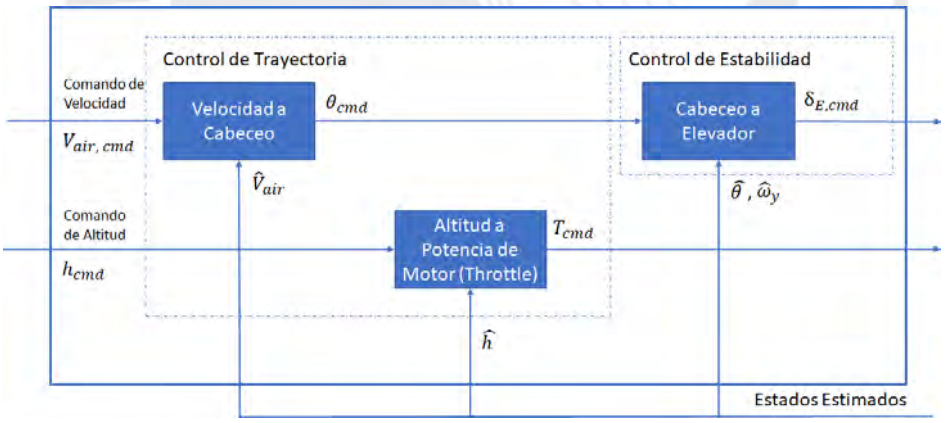


Figura 3. 7 Control de Vuelo Longitudinal donde el cabeceo es utilizado para controlar la velocidad y la potencia del motor para controlar la altitud

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

Utilizar la potencia del motor para corregir el error de altitud es más eficiente que ascender y descender constantemente, por lo tanto, se utilizara el control longitudinal de la figura 3.7 para ascender a una altitud de vuelo más elevada. Sin embargo, el comando del elevador es

generalmente más rápido para corregir pequeños errores en la altitud que la potencia del motor, así que el control longitudinal de la figura 3.6 será utilizada cuando se mantiene una altitud constante.

La figura 3.8 muestra cómo actúa el control de vuelo longitudinal en las diferentes fases del vuelo. Se aprecia que para cada parte del vuelo el control de altitud y velocidad es asumido por la potencia del motor o el cabeceo, según sea más eficiente para el vuelo.

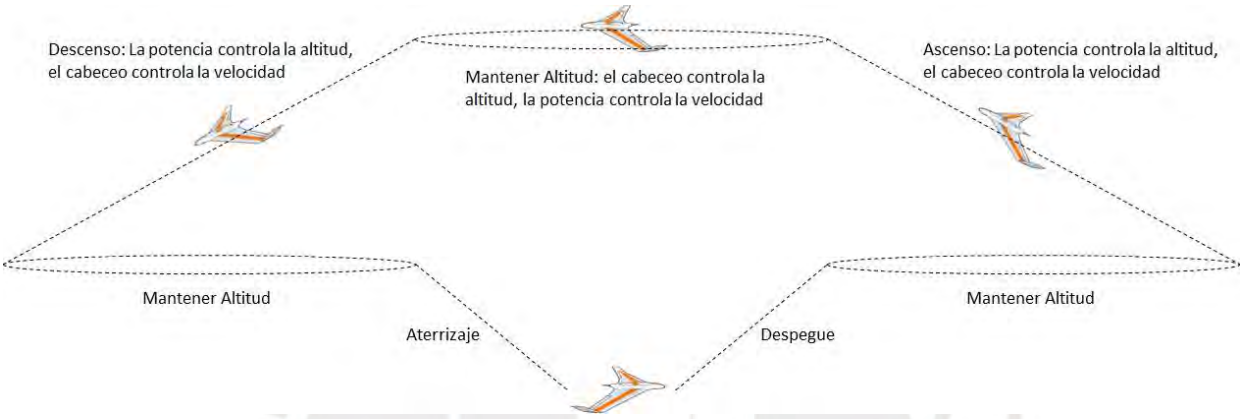


Figura 3. 8 Control de Vuelo Longitudinal en diferentes fases del vuelo.

Fuente Propia

En las figuras de control de vuelo longitudinal y lateral se menciona el control de estabilidad, que es donde el ángulo deseado de alabeo (roll) o cabeceo (pitch) es transformado a una respuesta de la superficie de control de la aeronave. La figura 3.9 y 3.10 muestran el control PID que se utiliza para convertir el ángulo de movimiento deseado en respuesta de la superficie de control.

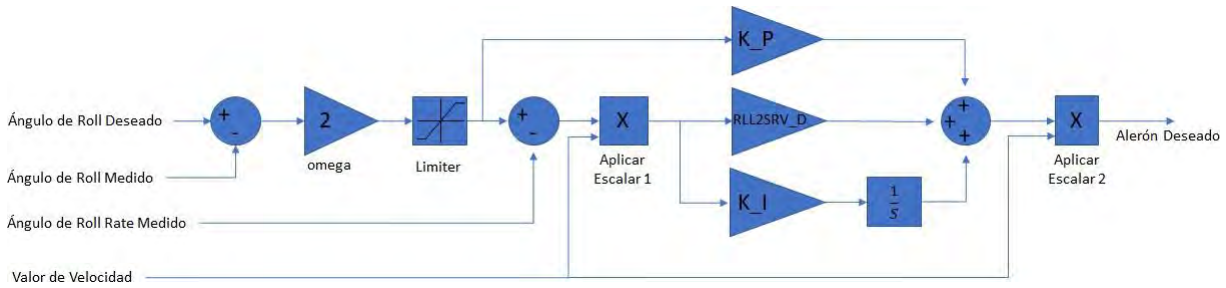


Figura 3. 9 Control de Estabilidad del Alabeo (Roll).

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

La figura 3.9 muestra el controlador PID que toma el ángulo de roll deseado o comando de roll para obtener una respuesta de los alerones y que la aeronave gire a la posición deseada. El ángulo de roll deseado es comparado con el ángulo de roll medido que proviene de los valores estimados por los sensores del autopiloto.

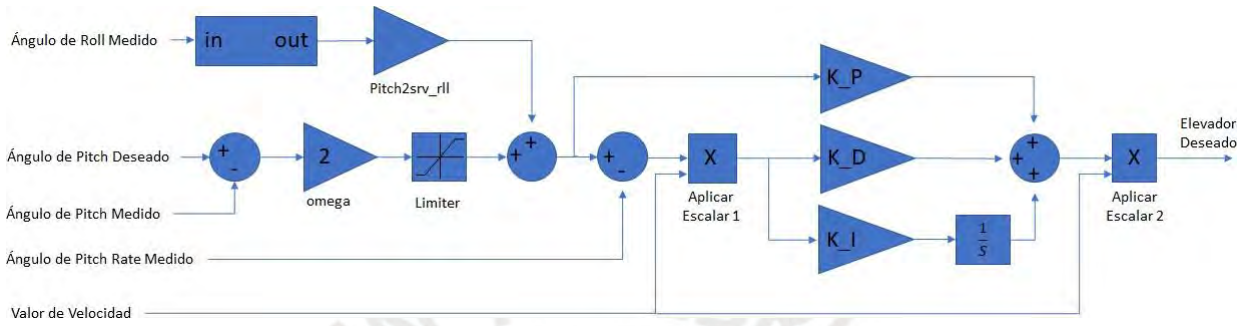


Figura 3. 10 Control de Estabilidad de Cabeceo (Pitch)

Fuente: Fundamental of Small Unmanned Aircraft Flight. J.D. Barton

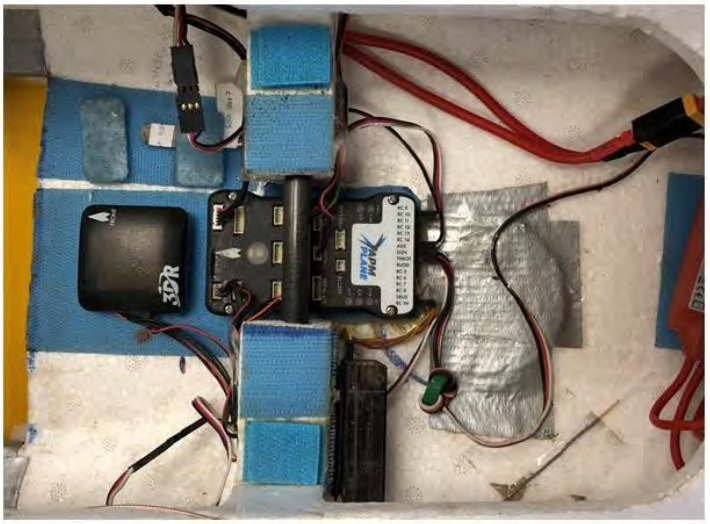
La figura 3.10 muestra el controlador PID que toma el ángulo de pitch deseado o comando de pitch para obtener una respuesta del elevador y que la aeronave gire a la posición deseada. El ángulo de pitch deseado es comparado con el ángulo de pitch medido que proviene de los valores estimados por los sensores del autopiloto.

3.3. Integración de los Sistemas de Control en la Aeronave

La aeronave posee un sistema de navegación que le permite realizar vuelos de manera autónomo, tal cual como se ha descrito en los diagramas de control. La figura 3.11 muestra cómo se realizan las conexiones físicas en la aeronave para que las señales viajen desde el control de vuelo hasta la superficie de control y la aeronave pueda ser controlada.

Servo mecanismo unido a la superficie de control

Las señales de alerones, elevadores y throttle del controlador de vuelo van a los servo mecanismos que mueven las superficies de control (elevons)



Sistema de Navegación Autónomo

Figura 3. 11 Sistema de Navegación Autónomo

Fuente Propia

Asimismo, existen otros sensores que contribuyen con el sistema de navegación autónomo al momento de realizar el vuelo. La figura 3.12 muestra el sensor de viento o tubo pitot. La figura 3.13 muestra el sensor de batería, que permite visualizar el voltaje remanente en la batería y el consumo (amperaje) de todo el sistema. La figura 3.14 muestra el controlador electrónico de velocidad o "ESC" por sus siglas en ingles. El ESC convierte la señal de throttle en potencia para que el motor aumente o disminuya las revoluciones de la hélice.

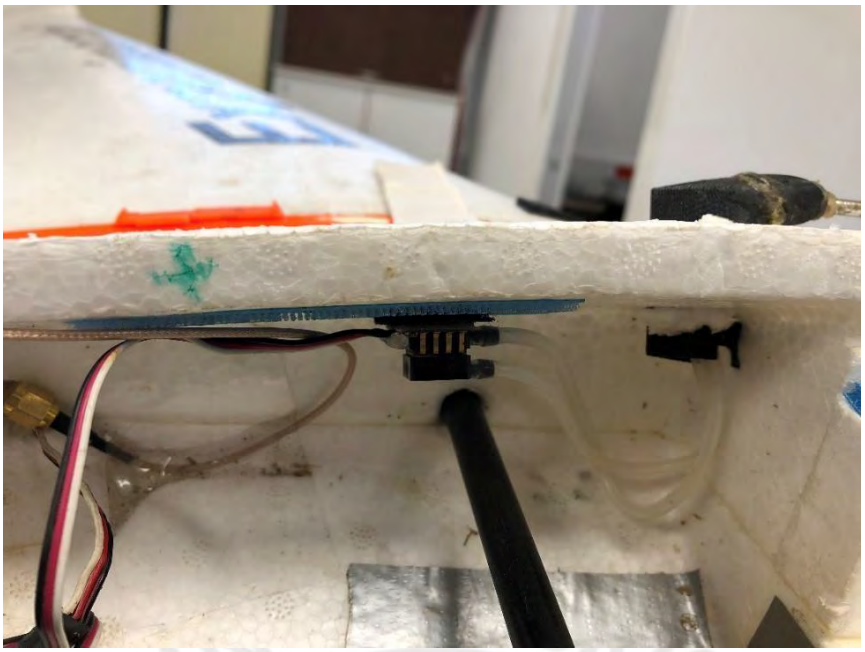


Figura 3. 12 Tubo Pitot para medir la velocidad de vuelo

Fuente Propia

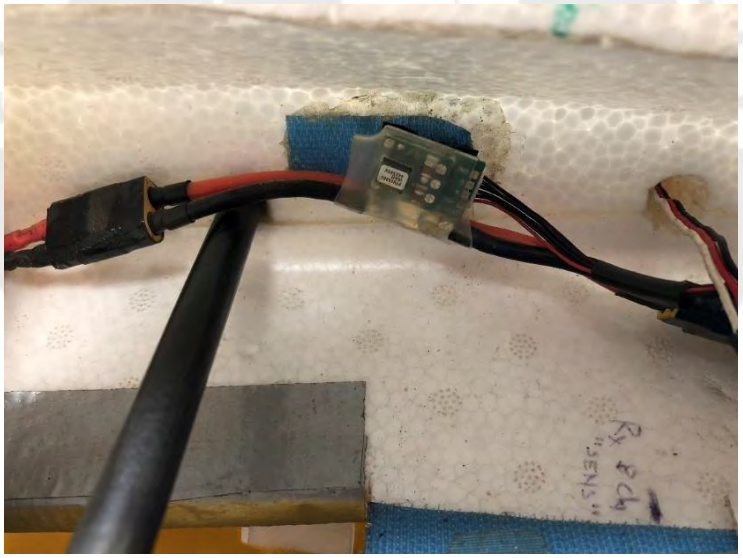


Figura 3. 13 Medidor de Batería

Fuente Propia

Figura 3. 14 Controlador Electrónico de Velocidad o ESC

Fuente Propia

Capítulo 4

RESULTADOS DEL ENTRENAMIENTO Y VALIDACIÓN

En este capítulo se mostrarán los resultados obtenidos del modelamiento, los cuales se han validado con 05 vuelos realizados en diferentes días. Asimismo, se muestra el sistema integrado en la aeronave para explicar la interacción de las partes móviles de la aeronave con el control de vuelo y la dinámica que se logra.

4.1. Entrenamiento y Validación de la RNA.

Luego de que la información de los sensores ha sido filtrada, se procede con el entrenamiento de la RNA. El entrenamiento de la RNA se realizó utilizando una entrada adicional denominada "bias" que es un ruido o perturbación adicional que se ingresa en el entrenamiento para simular alguna perturbación externa como por ejemplo turbulencia durante el vuelo y fuerzas que no se están considerando como el peso de la aeronave.

Se descarto desde un inicio entrenar la RNA separando los ejes de la aeronave al ser un ala volante que mezcla ambos ejes para el vuelo como se ha explicado anteriormente. Asimismo,

la entrada adicional “bias” se mantiene en un valor fijo de 0.5 para todos los entrenamientos. El número de neuronas intermedias se estableció en 10. El tamaño de la RNA, el tipo de topología y el algoritmo de entrenamiento tienen un rol importante para mejorar el aprendizaje y generalización de la RNA. RNA grandes tienen entrenamientos rápidos; sin embargo, la generalización no es acertada, mientras que las RNA pequeñas tienen mejor generalización con entrenamiento más lento. (Roudbari & Saghafi, 2014).

La cantidad de iteraciones para realizar el entrenamiento en todas las pruebas fue de 5000.

4.1.1. Primera Prueba de Entrenamiento

La primera prueba de entrenamiento se realizó para determinar si se utiliza la data obtenida de los sensores o se calcula a partir de las estimaciones de los ángulos de alabeo y cabeceo para los parámetros de ratio de alabeo y ratio de cabeceo. Las ratios se pueden estimar de los sensores de la aeronave, sin embargo, se observa que posee ruido por encima de lo usual, por lo tanto, se evalúa calcular estos parámetros.

Los gráficos que se utilizan muestran 05 líneas de diferentes colores, las cuales representan cada uno de los parámetros que se están entrenando. El eje “y” muestra el porcentaje de error y el eje “x” una escala de tiempo de 2 segundos. El error se mide en el punto de intersección cuando el eje “x” esta en el tiempo de 2 segundos. Estos gráficos se obtienen como resultados del programa desarrollado en el software MATLAB ® para realizar el entrenamiento y validación de la RNA.

La figura 4.1 muestra la grafica con la diferencia de los ratios originales y calculados.

La figura 4.2 muestra el error de entrenamiento cuando se utiliza los ratios calculados u originales. En el entrenamiento de la RNA con las ratios originales, la ratio de alabeo tiene un error de 55.89% mientras que utilizando las ratios calculados el error es de 47.86%. Sin embargo, para la ratio de cabeceo el entrenamiento de la RNA con las ratios originales es de 92.05% y con las ratios calculados es de 99.1%. El Anexo 01 muestra el código utilizado para realizar estos cálculos. Asimismo, el Anexo 02 muestra el código utilizado para calcular los ratios.

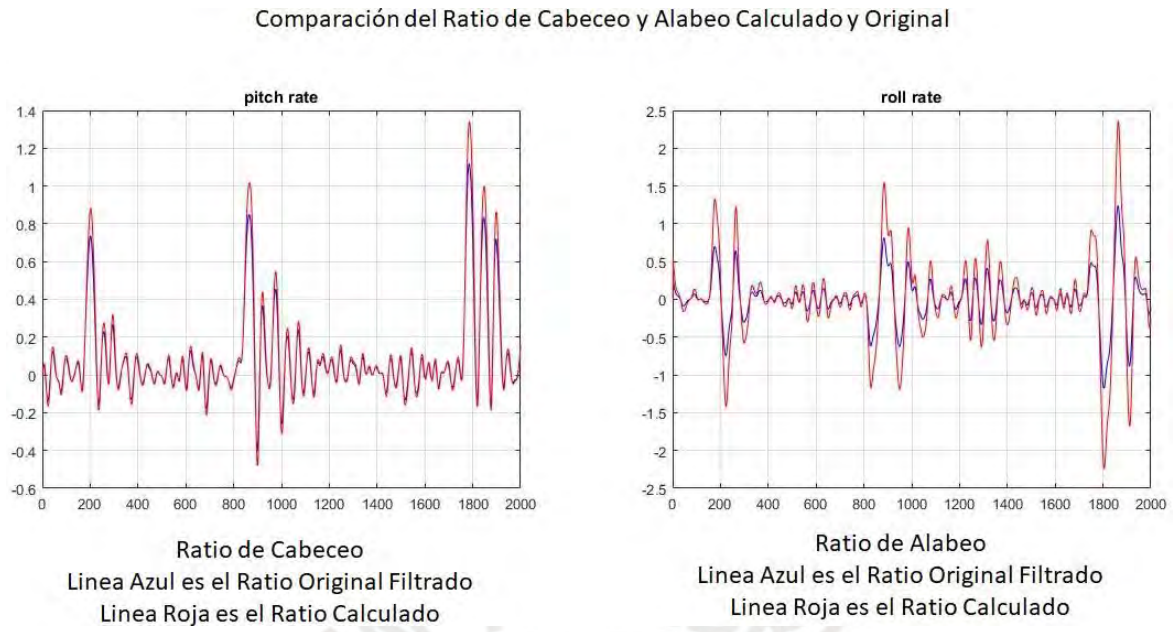


Figura 4. 1 Comparación de Ratio de Cabeceo y Alabeo Calculado y Original

Por lo tanto, solo tomando en consideración el error del entrenamiento de la RNA en las ratios no se puede determinar si se van a utilizar las ratios originales de cabeceo y alabeo o se van a calcular. Entonces, se procede a realizar la validación de las ratios de alabeo y cabeceo, previamente entrenados, en los 05 vuelos diferentes. Las figuras 4.3 a 4.7 muestran los errores solo para las ratios de alabeo y cabeceo en la validación para los 05 vuelos.

Sin embargo, así como en el entrenamiento, el error de las ratios en la validación tampoco permitió concluir si se deben de calcular o utilizar las ratios originales para el alabeo y cabeceo. utilizar para los siguientes entrenamientos, la tabla 5.1 muestra los resultados de los errores en los vuelos de validación, resaltando en verde los errores más bajos.

Entonces, se decidió utilizar el error global en la validación para determinar si se utilizan los ratios calculados u originales. El error global es el error de la validación como un conjunto y no el error de cada uno de los parámetros que son analizados. La table 4.1 resume los errores globales en la validación.

En conclusión y para los siguientes entrenamientos, se tomó la decisión de utilizar la data de las ratios calculados porque en 03 de los 05 vuelos de validación el error global es menor.

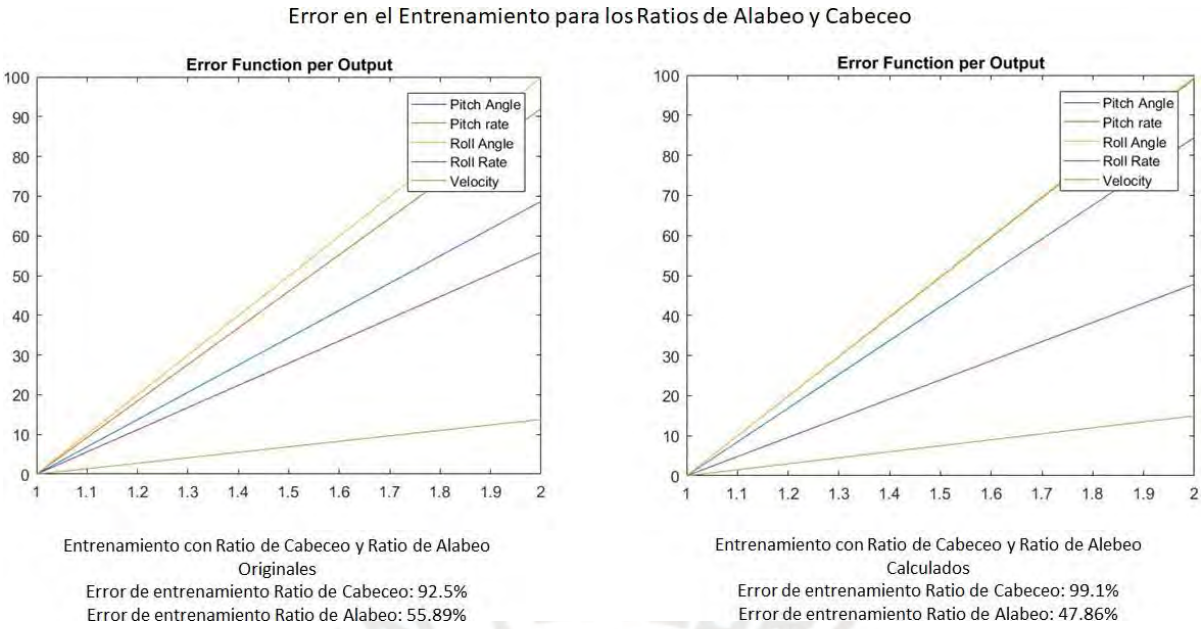


Figura 4. 2 Error en el Entrenamiento para los Ratios de Alabeo y Cabeceo

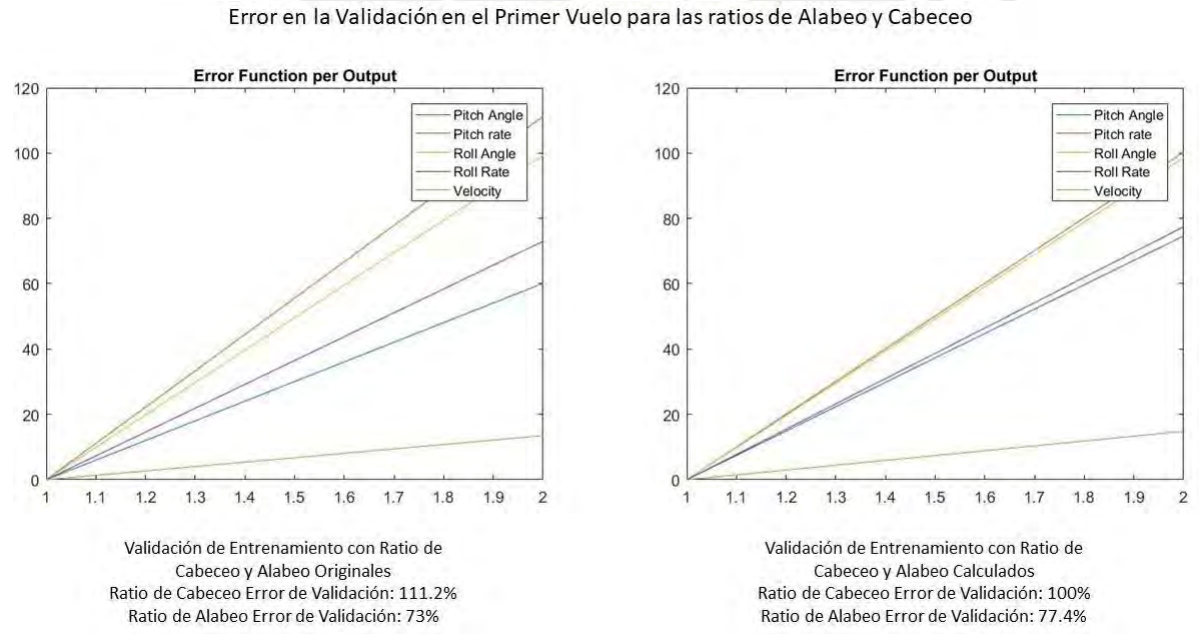


Figura 4. 3 Error en la Validación en el Primer Vuelo para las ratios de Alabeo y Cabeceo

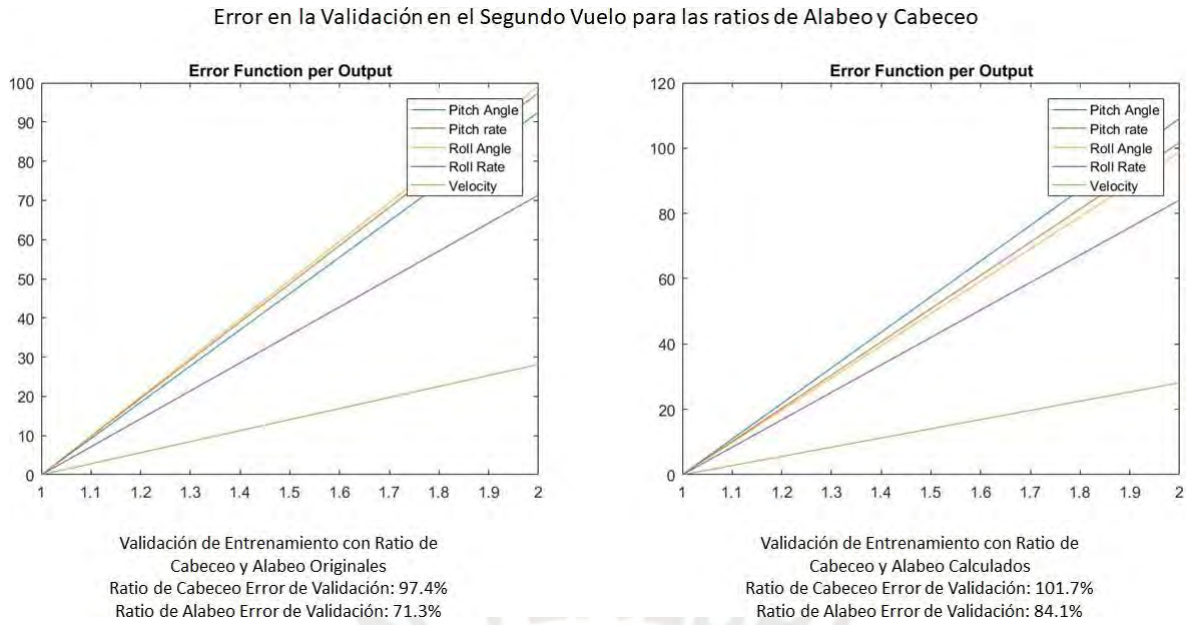


Figura 4. 4 Error en la Validación en el Segundo Vuelo para las ratios de Alabeo y Cabeceo

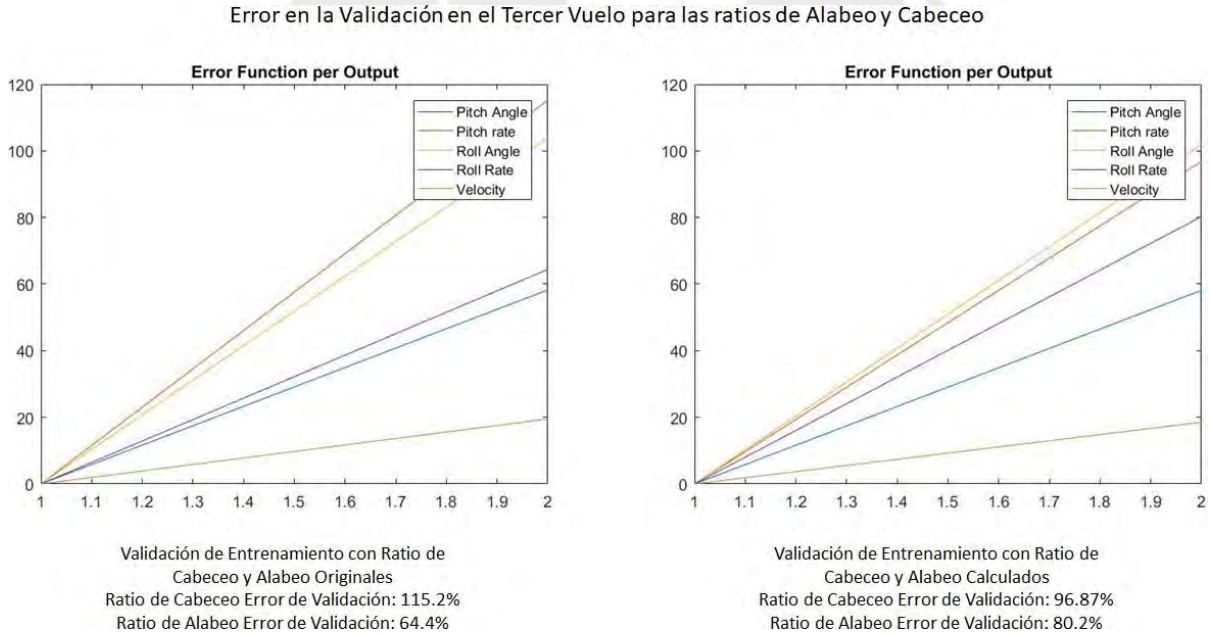
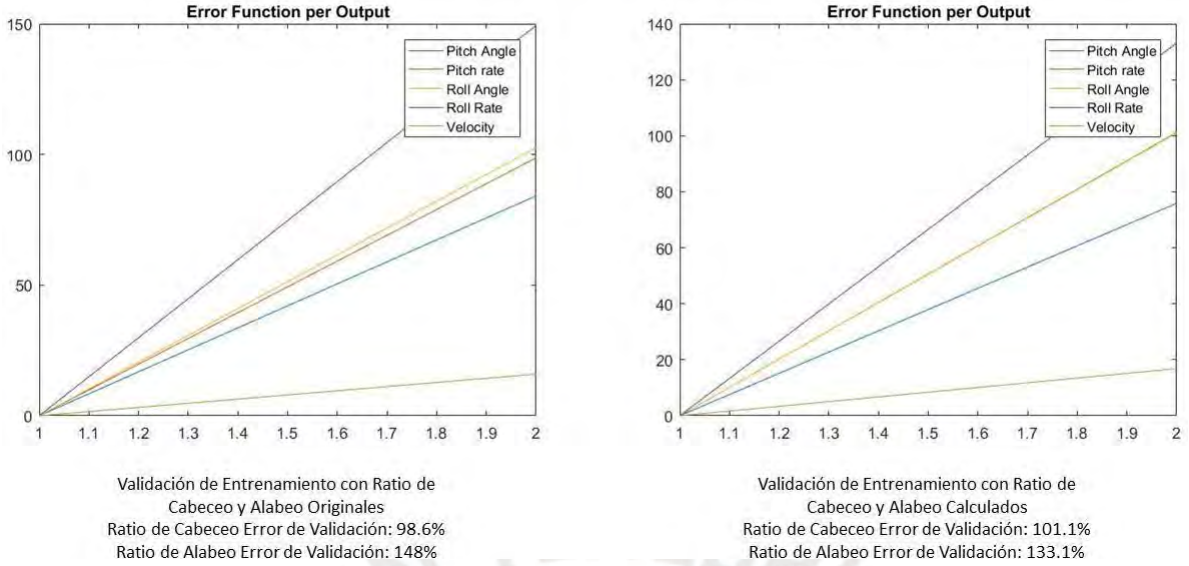


Figura 4. 5 Error en la Validación en el Tercer Vuelo para las ratios de Alabeo y Cabeceo

Error en la Validación en el Cuarto Vuelo para las ratios de Alabeo y Cabeceo

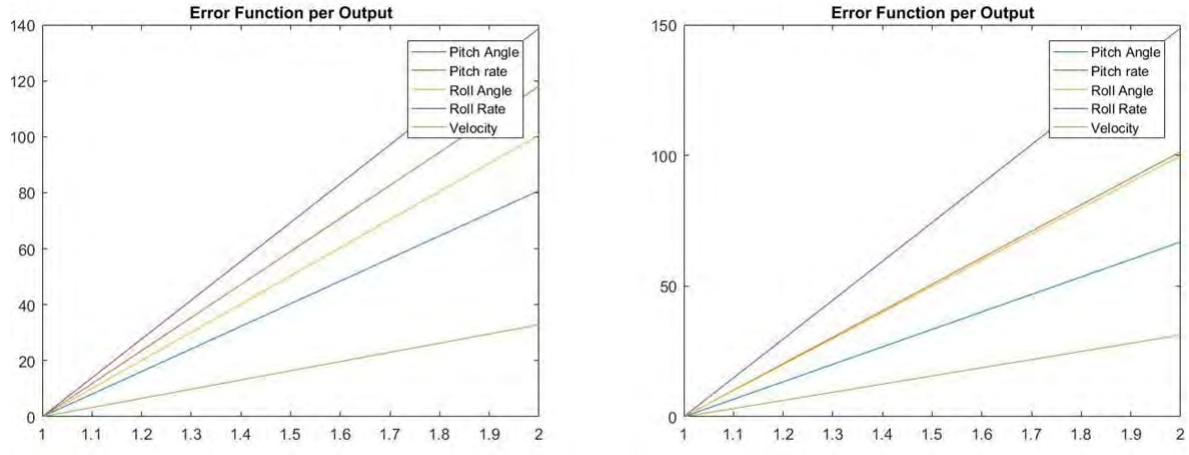


Validación de Entrenamiento con Ratio de Cabeceo y Alabeo Originales
Ratio de Cabeceo Error de Validación: 98.6%
Ratio de Alabeo Error de Validación: 148%

Validación de Entrenamiento con Ratio de Cabeceo y Alabeo Calculados
Ratio de Cabeceo Error de Validación: 101.1%
Ratio de Alabeo Error de Validación: 133.1%

Figura 4. 6 Error en la Validación en el Cuarto Vuelo para las ratios de Alabeo y Cabeceo

Error en la Validación en el Quinto Vuelo para las ratios de Alabeo y Cabeceo



Validación de Entrenamiento con Ratio de Cabeceo y Alabeo Originales
Ratio de Cabeceo Error de Validación: 118.1%
Ratio de Alabeo Error de Validación: 138.7%

Validación de Entrenamiento con Ratio de Cabeceo y Alabeo Calculados
Ratio de Cabeceo Error de Validación: 101.4%
Ratio de Alabeo Error de Validación: 148.5%

Figura 4. 7 Error en la Validación en el Quinto Vuelo para las ratios de Alabeo y Cabeceo

Tabla 4. 1 Errores en la Validación de Pitch Rate y Roll Rate

	Vuelo 01		Vuelo 02		Vuelo 03		Vuelo 04		Vuelo 05	
	Ratio Orig	Ratio Cal	Ratio Orig	Ratio Cal	Ratio Orig	Ratio Cal	Ratio Orig	Ratio Cal	Ratio Orig	Ratio Cals
Ratio de Cabeceo	111.2%	100%	97.4%	101.7%	115.2%	96.87%	98.6%	101.1%	118.1%	101.4%
Ratio de Alabeo	73%	77.4%	71.3%	84.1%	64.4%	80.2%	148%	133.1%	138.7%	148.5%

Tabla 4. 2 Errores Globales de Validación para los 05 Vuelos

	Vuelo 01 Error Global	Vuelo 02 Error Global	Vuelo 03 Error Global	Vuelo 04 Error Global	Vuelo 05 Error Global
Utilizando para el Análisis Rates Originales	43.90%	57.98%	42.82%	61.11%	48.56%
Utilizando para el Análisis Rates Calculados	48.78%	60.50%	40.84%	58.79%	48.27%

4.1.2. Segunda Prueba de Entrenamiento

La segunda prueba de entrenamiento incorpora junto con todos los demás parámetros de entrenamiento en la RNA, el parámetro de la altitud y el cambio de la altitud en el tiempo ("altitude rate"), el cual fue calculado a partir de la altitud y el tiempo de muestreo, ya que este parámetro no puede ser obtenido directamente de los sensores de la aeronave. Roudbari menciona que en adición a los comandos de entrada del piloto, la dinámica de vuelo del vehículo depende de las condiciones de vuelo (velocidad y altitud). Por lo tanto, no se puede esperar que una RNA entrenada a una velocidad y altitud específica puede producir un resultado aceptable a otra velocidad y altitud (Roudbari & Saghafi, 2014).

Sin embargo, el vuelo de entrenamiento y los vuelos de validación fueron realizados a la misma altitud (150 metros) y la misma velocidad (13m/s). Por lo tanto, se toma en consideración ambos parámetros para esta segunda prueba.

La tabla 4.3 muestra el error global y los errores de validación para los 05 vuelos.

Tabla 4. 3 Error Global y de Validación entrenamiento con Altitud y Altitud Rate

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
40.77%	38.29%	41.87%	27.76%	43.65%	32.14%

Al compararse la tabla 4.2 y 4.3 se aprecia que el error global ha disminuido en un 25% aproximadamente y los errores de validación también han mejorado en todos los vuelos. Entonces, incluir los parámetros de altitud y "altitud rate" contribuye con el entrenamiento y validación.

Tomando en consideración lo mencionado por Roudbari acerca de la velocidad y altitud y que el parámetro de "altitud rate" no es utilizado para el control de la aeronave, según los diagramas de bloques descritos en el Capítulo 03, se decidió realizar el entrenamiento de la RNA sin considerar este parametro. La Table 4.4 muestra los resultados.

Tabla 4. 4 Error Global y de Validación entrenamiento con Altitud y sin Altitud Rate

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
30.57%	30.69%	37.47%	25.92%	42.45%	39.14%

Al comparar la tabla 4.3 y 4.4 se observa que el error global del entrenamiento sin "altitud rate" ha disminuido en un 10% aproximadamente y cada uno de los errores de validación, con excepción del vuelo 05, han disminuido.

Por lo tanto, no se utiliza el parámetro de "altitude rate" para los siguientes entrenamientos. Asimismo, se comparó el menor y el mayor error de los vuelos de validación tomando en consideración lo expresado respecto a los parámetros de velocidad y altitud. Las figuras 4.8 y 4.9 muestran los errores para ambas validaciones. Es interesante observar que existe un desfase de la data deseada y la data de salida, producto de cambios en la altitud y velocidad de vuelo en el vuelo de entrenamiento y los de validación.

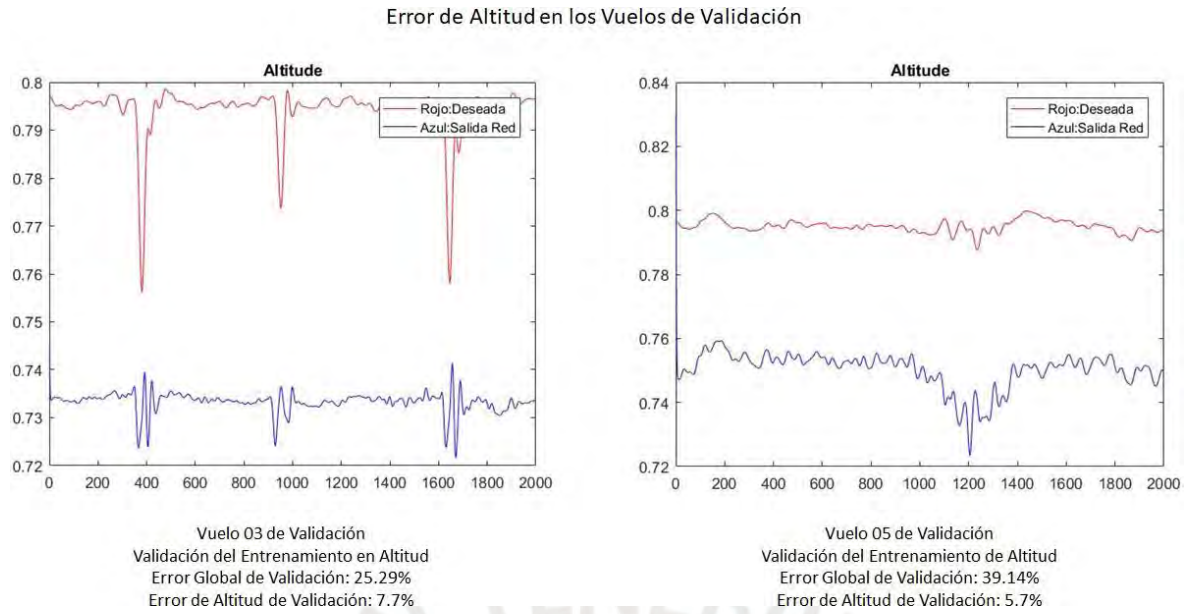


Figura 4. 8 Error de Altitud en los Vuelos de Validación

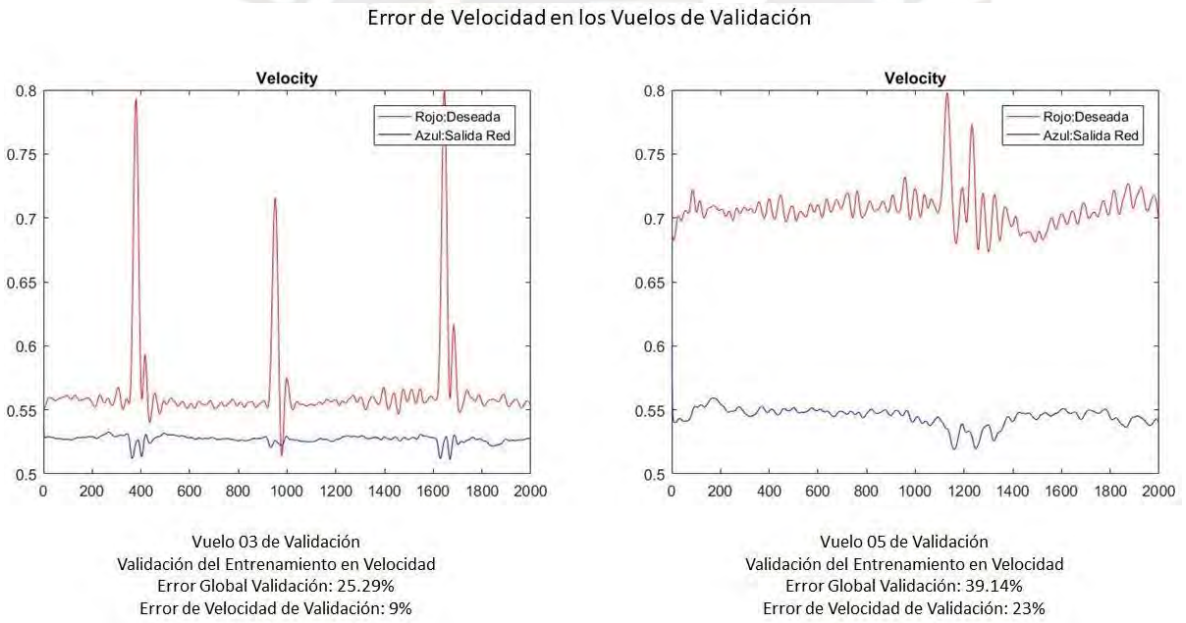


Figura 4. 9 Error de Velocidad en los Vuelos de Validación

4.1.3. Tercera Prueba de Entrenamiento

La tercera prueba de entrenamiento incorpora parámetros adicionales como las aceleraciones en los 3 ejes de la aeronave, así como también entradas adicionales anteriores, pero sin utilizar el parámetro de la altitud.

Utilizar la información del acelerómetro como valores de entrada de la RNA se explica de la siguiente manera. Estos valores son parámetros de la aeronave que en todo momento se están midiendo para identificar la posición de la aeronave y la actitud que esta tiene al momento de volar. El controlador de vuelo utiliza esta información para realizar las correcciones necesarias y que la aeronave pueda realizar un vuelo recto y nivelado.

La table 4.5 muestra los errores globales y de validación de este entrenamiento

Tabla 4. 5 Errores globales y de validación de este entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
42.56%	43.99%	54.85%	42.51%	54.88%	50.41%

Si se compara la tabla 4.4 y 4.5 se observa que al utilizar la información de la aceleración y no considerar la altitud hace que el error global de entrenamiento aumente en un 12% aproximadamente. Asimismo, los errores en la validación aumentan.

Sin embargo, la predicción de los estados de velocidad, ángulo de cabeceo y ángulo de alabeo mejoran en el entrenamiento de la RNA. Las figuras 4.10 a 4.12 muestran la diferencia en error de entrenamiento para cada uno de estos parámetros.

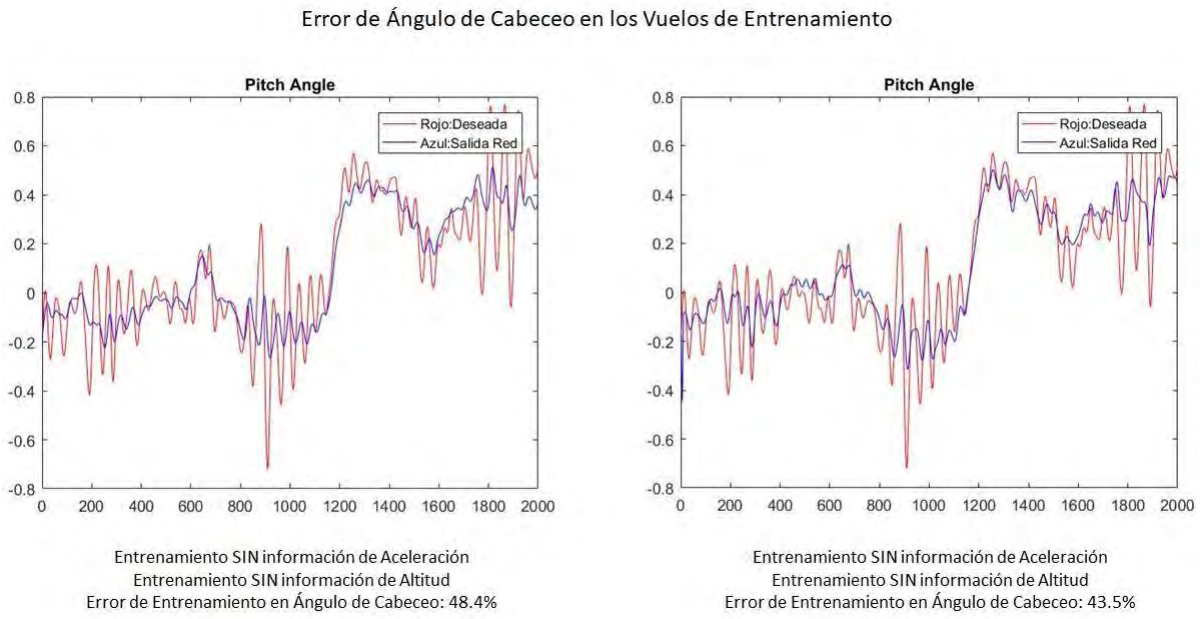


Figura 4. 10 Error de Ángulo de Cabeceo en los Vuelos de Entrenamiento

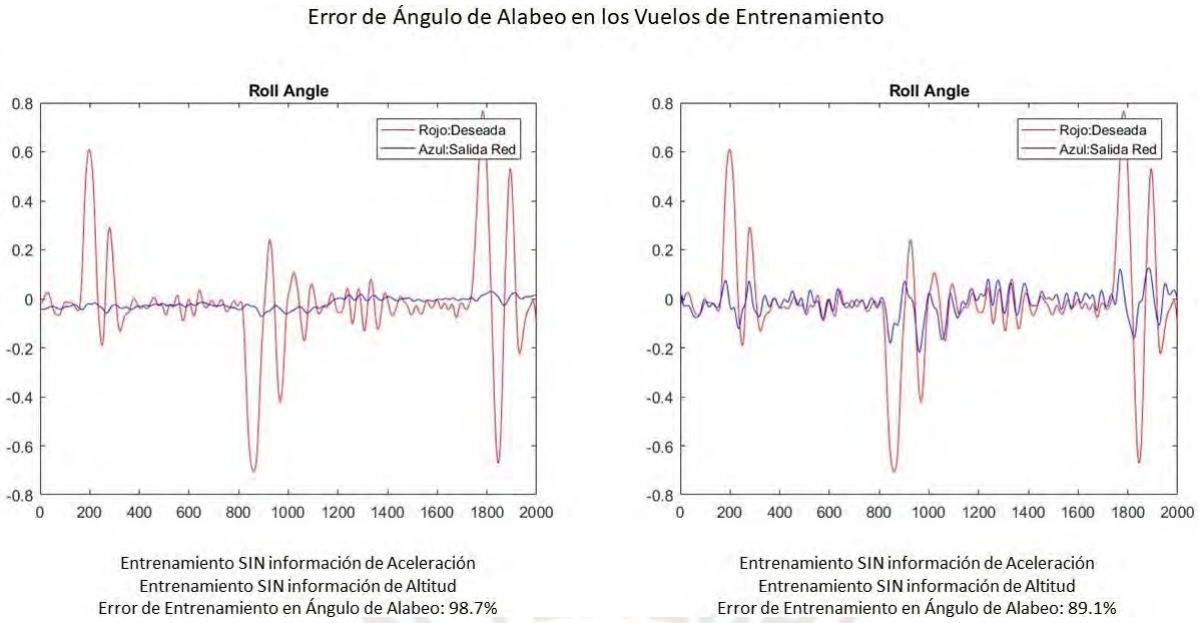


Figura 4. 11 Error de Ángulo de Alabeo en los Vuelos de Entrenamiento

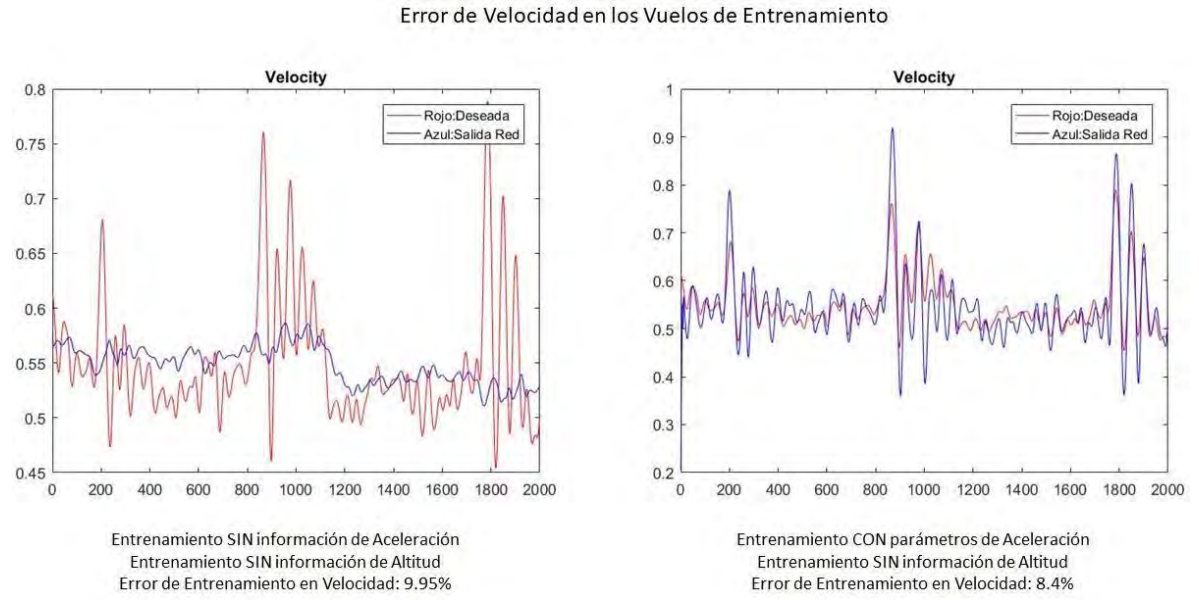


Figura 4. 12 Error de Velocidad en los Vuelos de Entrenamiento

Esta mejora en la predicción de la RNA al momento del entrenamiento permite concluir que es importante considerar los parámetros de aceleración y estados anteriores. La siguiente prueba cuenta con toda la información previamente analizada.

4.1.4. Cuarta Prueba de Entrenamiento

La cuarta prueba de entrenamiento toma en consideración todos lo analizado anteriormente, es decir, la RNA utiliza data de la altitud, valores de aceleración y estados anteriores. Esta prueba se realiza para observar si utilizando todos los parámetros se obtiene un mejor entrenamiento y validación.

La tabla 4.6 muestra el error global de entrenamiento y los errores de validación para cada uno de los cinco vuelos. Este entrenamiento tuvo una mayor cantidad de iteraciones para disminuir el error de entrenamiento. Se observo que para el parámetro de ángulo de cabeceo la RNA utiliza mayor cantidad de iteraciones para realizar la predicción durante el entrenamiento.

Tabla 4. 6 Errores globales y de validación de este entrenamiento Prueba 4 – Primer Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
19.72%	37.70%	35.53%	35.31%	53.81%	45.92%

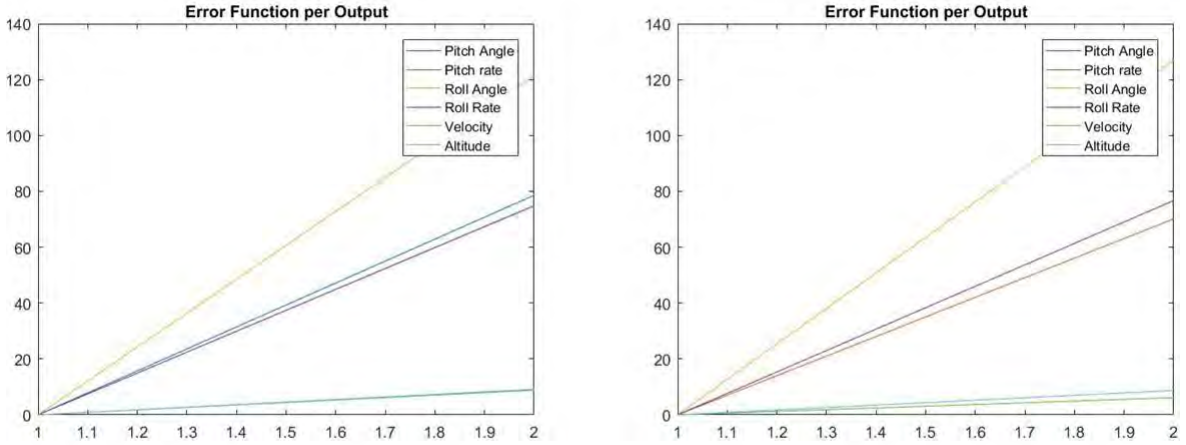
Al continuar con el entrenamiento para mejorar el error de validación y los errores de los vuelos de validación se consiguieron los resultados que se muestran en la tabla 4.7.

Tabla 4. 7 Errores globales y de validación de este entrenamiento Prueba 4 – Segundo Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
19.31%	37.49%	35.91%	34.51%	55.00%	47.05%

Al comparar la tabla 4.6 y 4.7 se observa que para los vuelos de validación 02, 04 y 05 los errores de validación han aumentado entre 0.5% a 1.5% aproximadamente. Las figuras 4.13 a 4.15 muestran que parámetros han crecido en porcentaje de error.

Errores en Vuelo de Validación 02

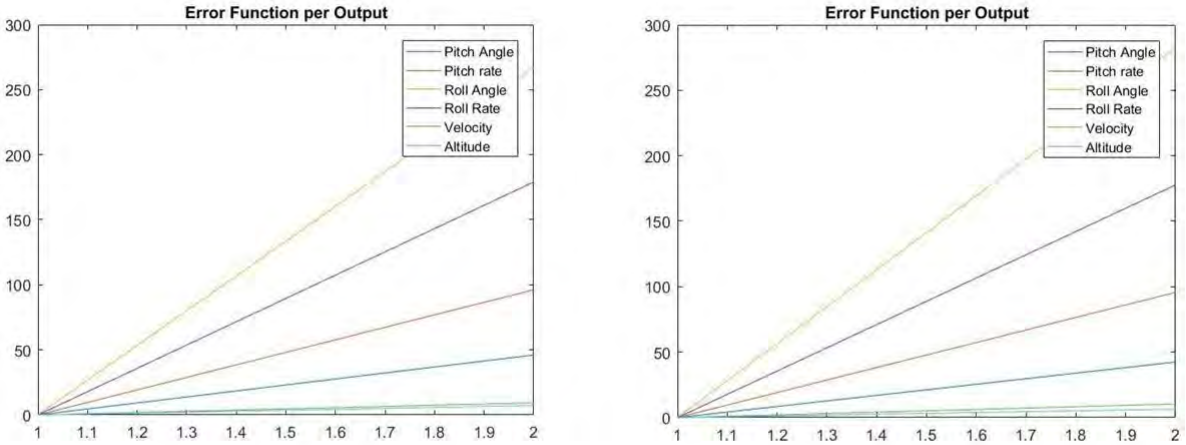


Error de validación Prueba 4 – Primer Entrenamiento
Ángulo de Alabeo: 121.4%
Ángulo de Cabeceo: 78.6%
Altura: 8.8%
Velocidad: 9.1%

Error de validación Prueba 4 – Segundo Entrenamiento
Ángulo de Alabeo: 127%
Ángulo de Cabeceo: 76.7%
Altura: 8.8%
Velocidad: 6.1%

Figura 4. 13 Errores Globales en Vuelo de Validación 02

Errores en Vuelo de Validación 04

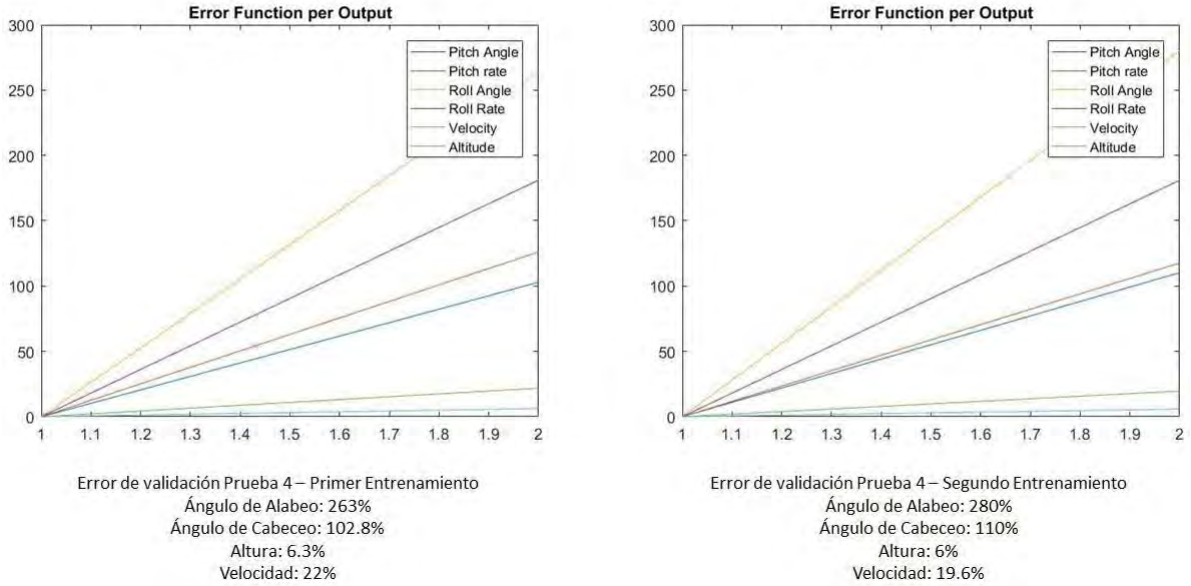


Error de validación Prueba 4 – Primer Entrenamiento
Ángulo de Alabeo: 267%
Ángulo de Cabeceo: 46.1%
Altura: 7.4%
Velocidad: 9.4%

Error de validación Prueba 4 – Segundo Entrenamiento
Ángulo de Alabeo: 282.5%
Ángulo de Cabeceo: 42.4%
Altura: 6.4%
Velocidad: 10.4%

Figura 4. 14 Error Global en Vuelo de Validación 04

Error Globales en Vuelo de Validación 05



Error de validación Prueba 4 – Primer Entrenamiento
Ángulo de Alabeo: 263%
Ángulo de Cabeceo: 102.8%
Altura: 6.3%
Velocidad: 22%

Error de validación Prueba 4 – Segundo Entrenamiento
Ángulo de Alabeo: 280%
Ángulo de Cabeceo: 110%
Altura: 6%
Velocidad: 19.6%

Figura 4. 15 Error Global en Vuelo de Validación 05

En todos los casos es el error de validación en el ángulo de alabeo el que incrementa. En los errores de entrenamiento, este parámetro es el que mayor porcentaje de error posee con un promedio de 61%.

Por lo tanto, se decidió modificar una de las constantes utilizadas durante el entrenamiento. Se modifico el coeficiente de los pesos de la RNA de 0.5 a 0.8 y se obtuvieron los resultados que se muestran en la tabla 4.8.

Tabla 4. 8 Errores globales y de validación de entrenamiento Prueba 4 – Tercer Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
18.95%	37.45%	36.11%	33.92%	55.42%	47.60%

Se compara la tabla 4.7 y 4.8 se observa que el error global de entrenamiento ha disminuido en 1.5%, pero los errores de validación de los vuelos 02, 04 y 05 siguen aumentando. Nuevamente es el parámetro de ángulo de alabeo el que está incrementando el error en la validación.

Se realizó otro cambio en los parámetros de entrenamiento, esta vez se incrementó el coeficiente de las constantes (a y c) de la función bipolar sigmoidea de 5 a 10. Esto ocasiona que el entrenamiento no se detenga en errores locales si no cuando alcanza un error global para que el entrenamiento sea más eficiente. La tabla 4.9 muestra los resultados.

Tabla 4. 9 Errores globales y de validación de entrenamiento Prueba 4 – Cuarto Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
18.64%	37.64%	36.23%	33.55%	55.58%	47.80%

Sin embargo, a pesar de que el error global de entrenamiento descendió en 0.3%, los errores de validación para los vuelos 01, 02, 04 y 05 siguieron aumentando por motivo del ángulo de alabeo que sigue aumentando al momento de validar.

Por lo tanto, se va a realizar un nuevo entrenamiento, pero esta vez se va a utilizar un segundo set de data de otro vuelo para aumentar el contenido de información sobre el comportamiento dinámico de la aeronave.

4.1.5. Quinta Prueba de Entrenamiento: Doble set de información

En esta prueba de entrenamiento se va a utilizar un set de información adicional sobre la aeronave como estados de entrada, el cual incluye los mismos parámetros y estados. Todos los demás parámetros de entrenamiento se mantienen constantes y la cantidad de neuronas intermedias en ambas capas es de 10. La tabla 4.10 muestra los resultados obtenido del entrenamiento.

Tabla 4. 10 Errores globales y de validación de entrenamiento Prueba 5 – Primer Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
27.05%	29.85%	33.81%	27.30%	40.66%	34.01%

Al comparar los resultados de la tabla 4.9 y 4.10 se observa que el error global de entrenamiento aumento en 9% aproximadamente, pero los errores de validación son menores en todos los vuelos. Sin embargo, al continuar con el entrenamiento para mejorar el error de entrenamiento global, sin mover ningún parámetro, se obtuvieron los siguientes resultados.

Tabla 4. 11 Errores globales y de validación de entrenamiento Prueba 5 – Segundo Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
22.70%	31.16%	30.25%	27.42%	42.16%	35.08%

A pesar de que el entrenamiento mejoro el error global en un 5% aproximadamente, los errores de validación aumentaron. Lo mismo que sucedió en las pruebas de entrenamiento previas.

Entonces se decide modificar los parámetros de entrenamiento. El primer parámetro modificado fue el "bias" de 0.5 a 0.3 y se continuó con el entrenamiento. Los resultados se pueden observar en la tabla 4.12.

Tabla 4. 12 Errores globales y de validación de entrenamiento Prueba 5 – Tercer Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
20.74%	35.20%	33.42%	28.03%	47.88%	35.56%

Nuevamente el error global disminuyo, pero los errores de validación aumentaron. Las figuras 4.16 y 4.17 muestran los vuelos de validación 01 y 04, que son los vuelos que tienen más diferencia en el porcentaje de error de validación. En ambos vuelos se muestra que el ángulo de alabeo y la velocidad son los parámetros que han aumentado el error de validación del segundo al tercer entrenamiento.

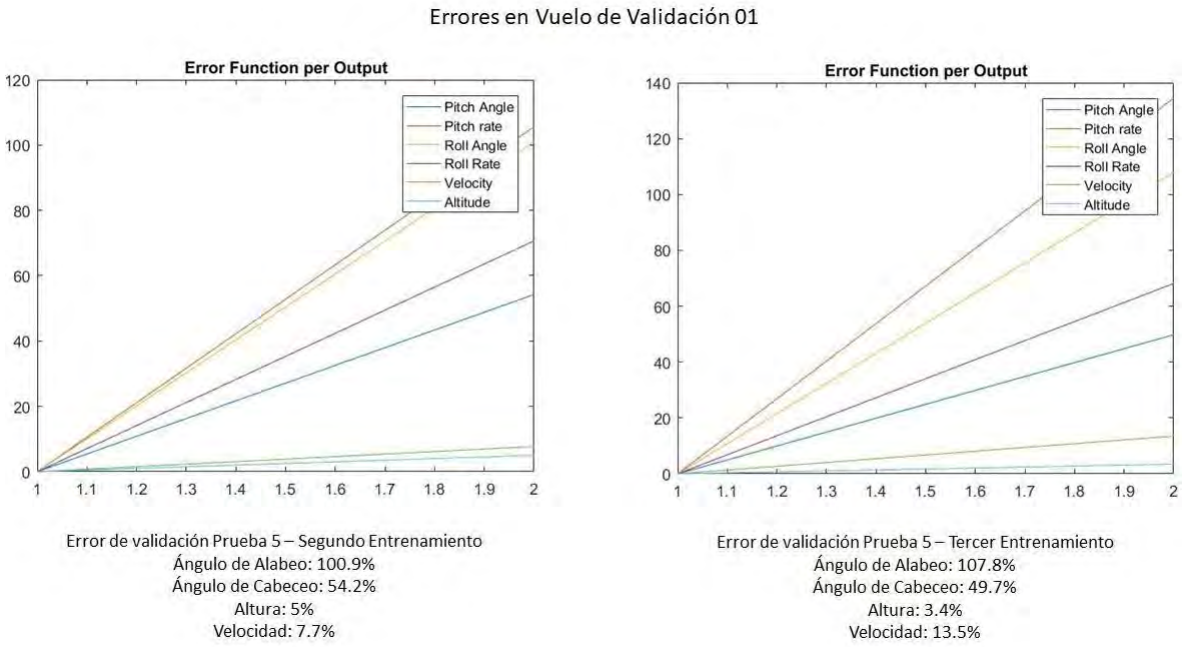


Figura 4. 16 Comparación de los Errores en Vuelo de Validación 01 del segundo y tercer entrenamiento.

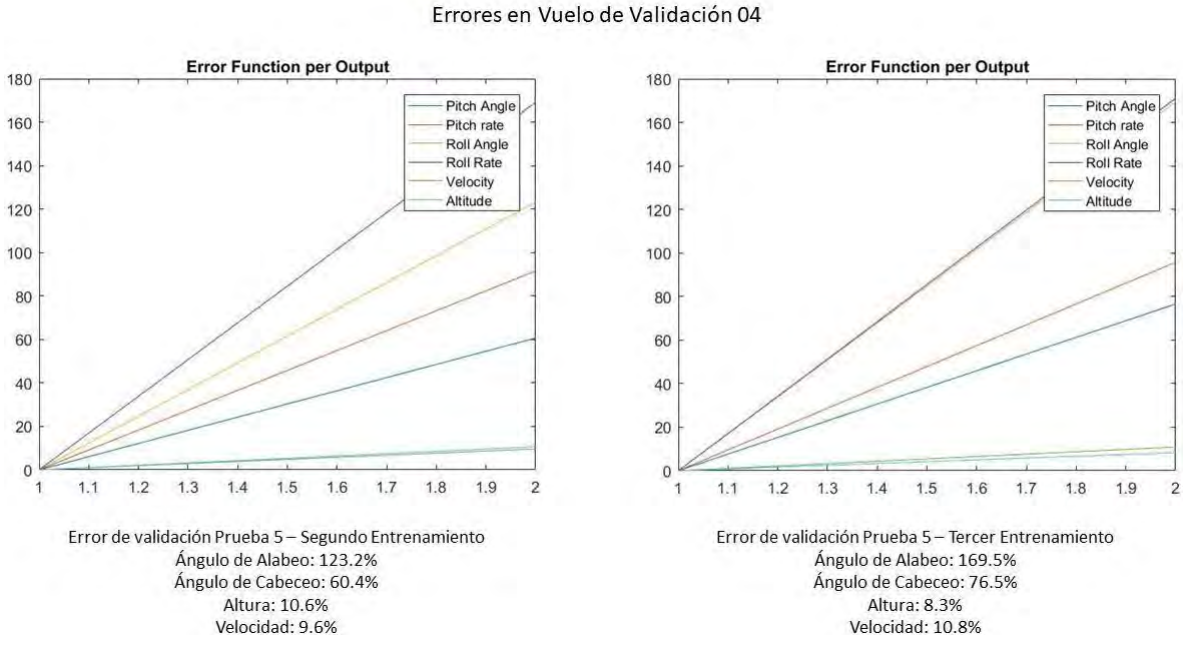


Figura 4. 17 Comparación de los Errores en Vuelo de Validación 04 del segundo y tercer entrenamiento.

Por lo tanto, el parámetro de entrenamiento "bias" fue nuevamente cambiado de 0.3 a 0.8. Se realizó el cuarto entrenamiento de la quinta prueba. Se obtuvieron los resultados que se muestran en la tabla 4.13.

Tabla 4. 13 Errores globales y de validación de la Prueba 5 – Cuarto Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
20.27%	40.74%	34.30%	40.44%	51.34%	46.54%

Esta variación en el parámetro "bias" no mejoró los errores de validación para cada uno de los vuelos. Por lo tanto, se descartó como modificación.

El parámetro "bias" volvió al valor de 0.5 y se decidió cambiar la cantidad de neuronas intermedias de 10 a 50. Se realizó el entrenamiento y se obtuvieron los resultados que se muestran en la tabla 4.14.

Tabla 4. 14 Errores globales y de validación de la Prueba 5 – Quinto Entrenamiento

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
19.90%	36.10%	31.63%	31.62%	49.29%	40.08%

En este entrenamiento los errores de validación disminuyeron. Por lo tanto, el cambio de aumentar las neuronas intermedias de 10 a 50 fue positivo y se mantendrá en los siguientes entrenamientos.

La mejora más significativa es en el vuelo de validación 03 con 9% aproximadamente. La figura 4.18 muestra la comparación en el tercer vuelo de la validación entre el cuarto y quinto entrenamiento.

En este caso, el ángulo de cabeceo y la altitud fueron los parámetros que disminuyeron el error en la validación.

En el siguiente entrenamiento se aumenta el número de neuronas intermedias de 50 a 100.

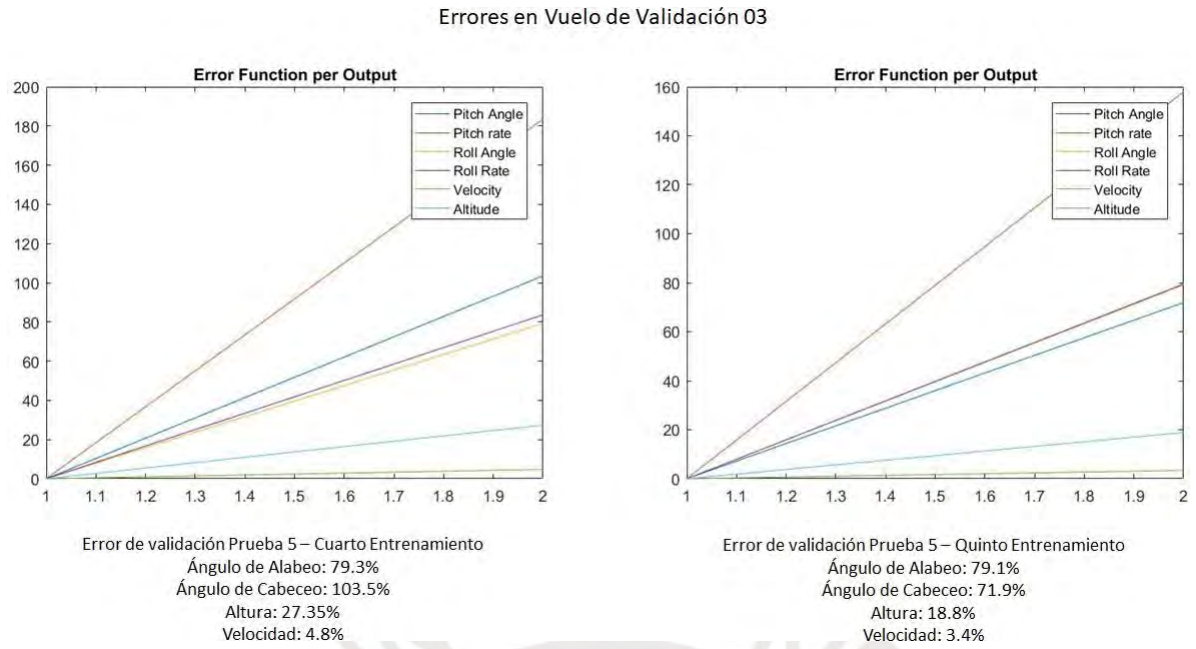


Figura 4. 18 Comparación de los Errores en Vuelo de Validación 03 del cuarto y quinto entrenamiento

4.1.6. Sexta Prueba de Entrenamiento: Doble set de información

En esta prueba de entrenamiento se incrementa el número de neuronas intermedias de 50 a 100 para analizar si mejora el entrenamiento y los errores de validación.

Tabla 4. 15 Errores globales y validación entrenamiento con 100 neuronas intermedias

Error Global Entrenamiento	Error Val. Vuelo 01	Error Val. Vuelo 02	Error Val. Vuelo 03	Error Val. Vuelo 04	Error Val. Vuelo 05
19.35%	35.39%	31.54%	29.41%	48.70%	40.15%

Los errores de validación de los vuelos han mejorado con excepción del vuelo 05, cuando se compara la tabla 4.14 y 4.15. La mejora más significativa ocurre en el vuelo de validación 03. La figura 4.19 muestra los errores por cada variable.

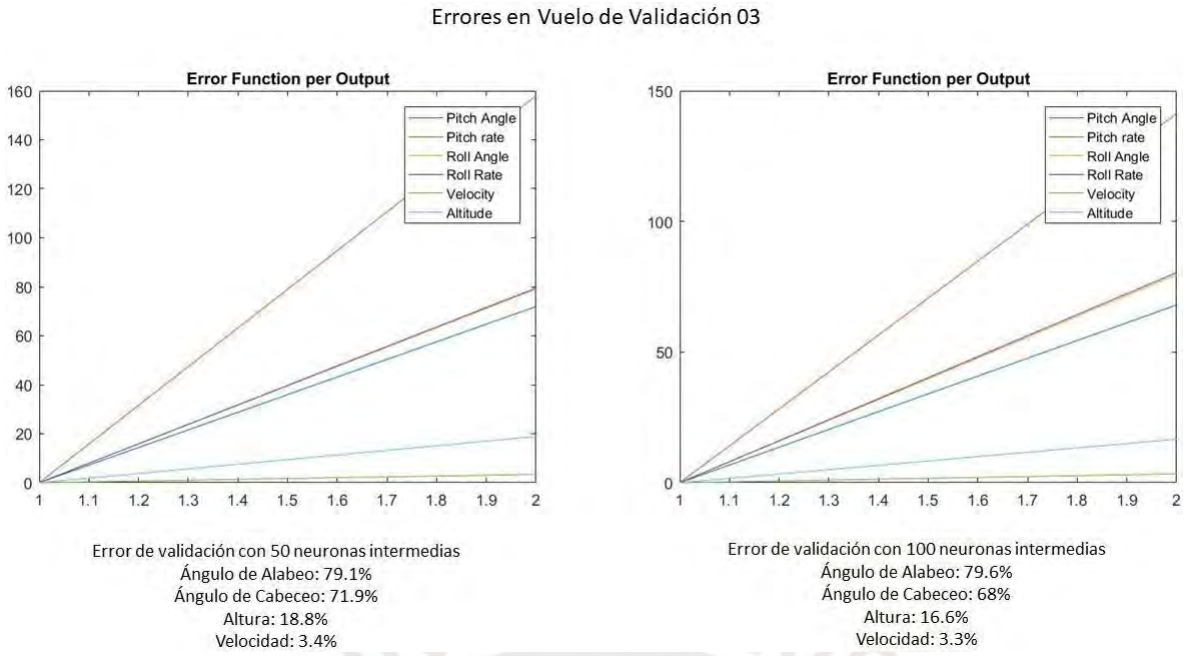


Figura 4. 19 Comparación del Vuelo de Validación 03 para 50 y 100 neuronas intermedias

De la figura 4.19 se aprecia que aumentar el número de neuronas intermedias ha hecho que disminuya el error en el ángulo de cabeceo y altura en el vuelo de validación.

La secuencia de entrenamientos utilizada y los resultados obtenidos muestran que esta metodología de entrenamiento es útil para modelar la dinámica de vuelo de ángulo de cabeceo y la velocidad. Aún esta metodología no es suficiente para realizar un buen entrenamiento del ángulo de alabeo. Los mejores resultados de la predicción de los parámetros dinámicos de vuelo y la validación de los vuelos se presentan a continuación.

La figura 4.20 muestra los errores de los parámetros de interés en el entrenamiento con 100 neuronas.

La figura 4.21 muestra los resultados en el entrenamiento en el ángulo de cabeceo y velocidad para el entrenamiento con 100 neuronas.

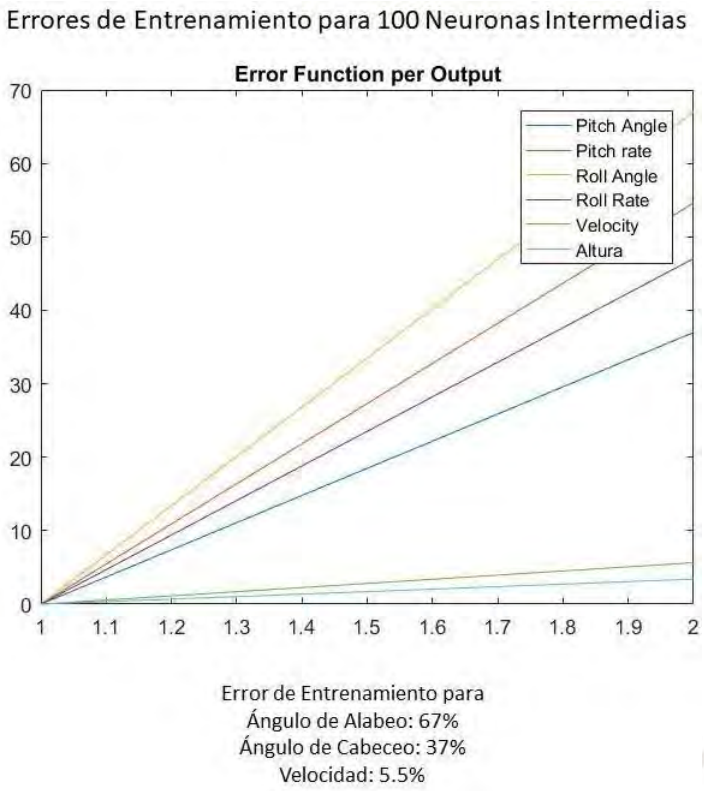


Figura 4. 20 Errores de Entrenamiento para 100 neuronas intermedias

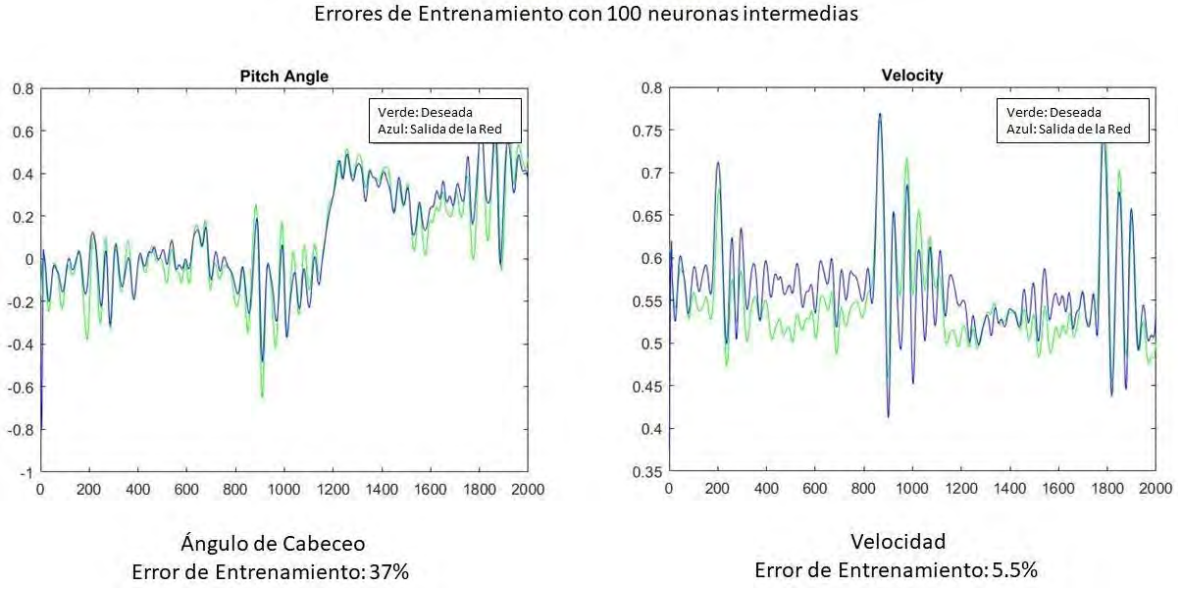


Figura 4. 21 Errores de Entrenamiento con 100 neuronas intermedias

La figura 4.22 muestra la validación del vuelo 03 para los parámetros de ángulo de cabeceo y velocidad

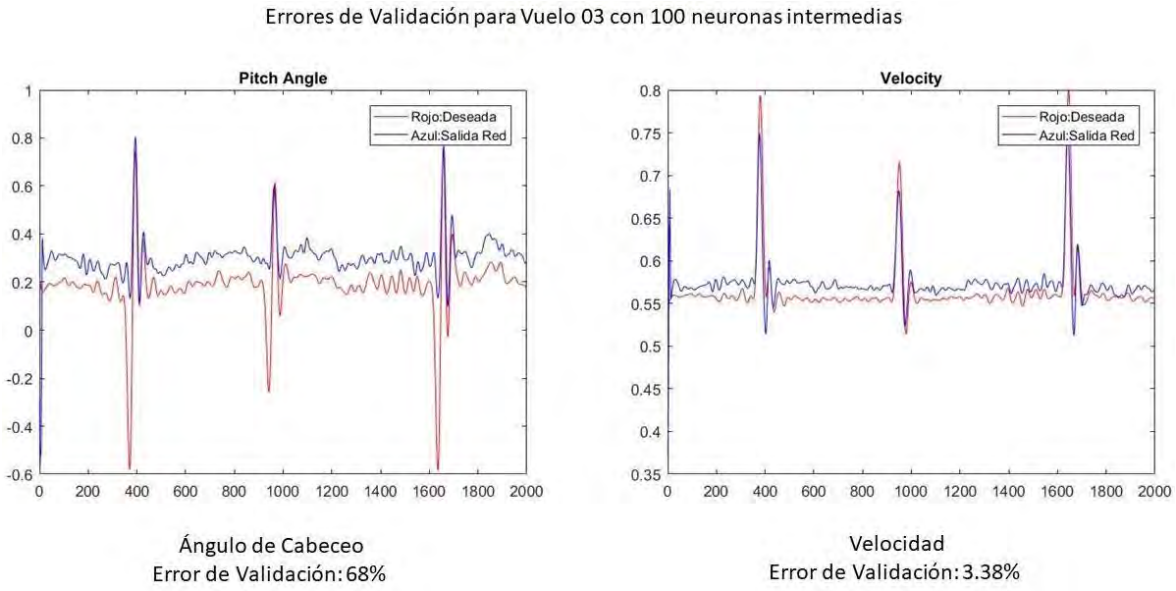


Figura 4. 22 Errores de Validación Vuelo 03 para Ángulo de Cabeceo y de Ataque

4.2. Relación entre el Entrenamiento de la Red Neuronal Artificial para el Problema del Vuelo de la Aeronave

Al momento de analizar la data de vuelo se observa que la aeronave realmente pierde altitud al momento de girar y existe una corrección agresiva de la aeronave para mantener o recuperar la altitud de vuelo. A continuación, se explicará lo que ocurre utilizando los gráficos de la data de entrenamiento de RNA. La figura 4.23 muestra de manera gráfica lo que ocurre al momento que la aeronave realiza un giro.

A continuación, se muestran las figuras 4.24 a 4.28 con la información de los sensores analizando el momento del giro de la aeronave. En las figuras 4.22 a 4.27 se muestra a la aeronave realizando un giro a la izquierda en el instante de tiempo 1585 segundos, en ese instante el ángulo de banqueo es de 46.24°. En ese mismo instante de tiempo, el ángulo de cabeceo es negativo (-14.54°) lo que significa que la aeronave está apuntando para abajo y perdiendo altitud. Por lo tanto, la velocidad aumenta de aproximadamente 12m/s a 14m/s. Sin

embargo, el throttle aumenta de 1100 a 1234, lo que significa que la aeronave intenta recuperar la altitud perdida utilizando solo la fuerza del motor, pero esto hace que aumente la velocidad en los segundos posteriores. Luego, la aeronave utiliza los elevadores para recuperar la actitud de vuelo normal.

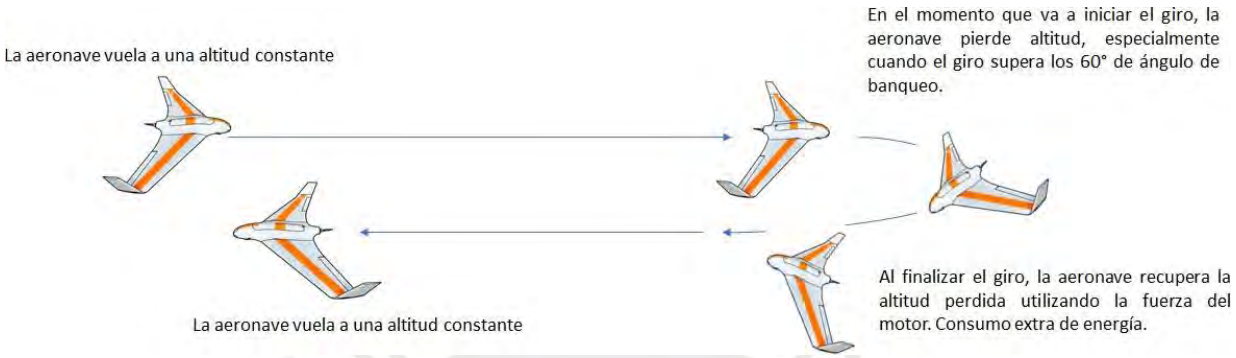


Figura 4. 23 Perdida de Altitud en el Giro

Fuente Propia

Por lo tanto, los resultados obtenidos de poder modelar el cabeceo y la velocidad de la aeronave son útiles para realizar un control preciso durante esta maniobra de vuelo y así mejorar la performance de la aeronave.

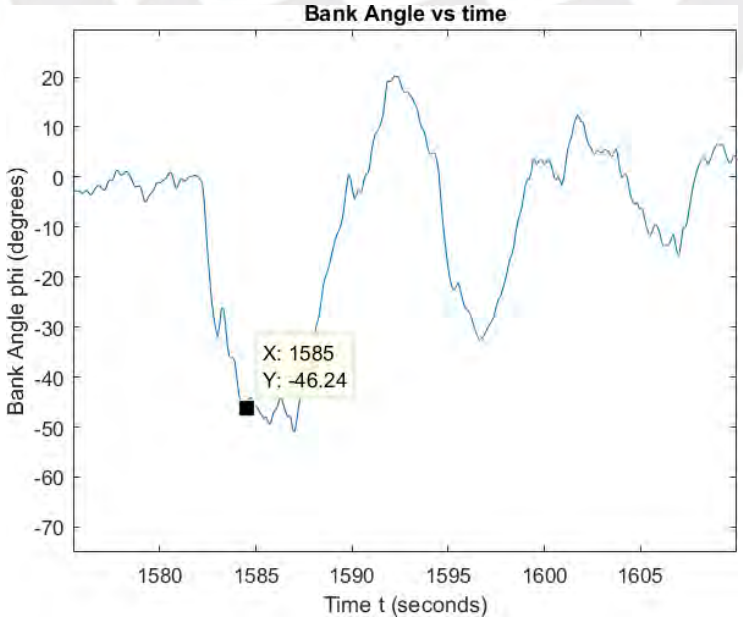


Figura 4. 24 Ángulo de Banqueo de -46.24° en el instante de tiempo 1585 segundos.

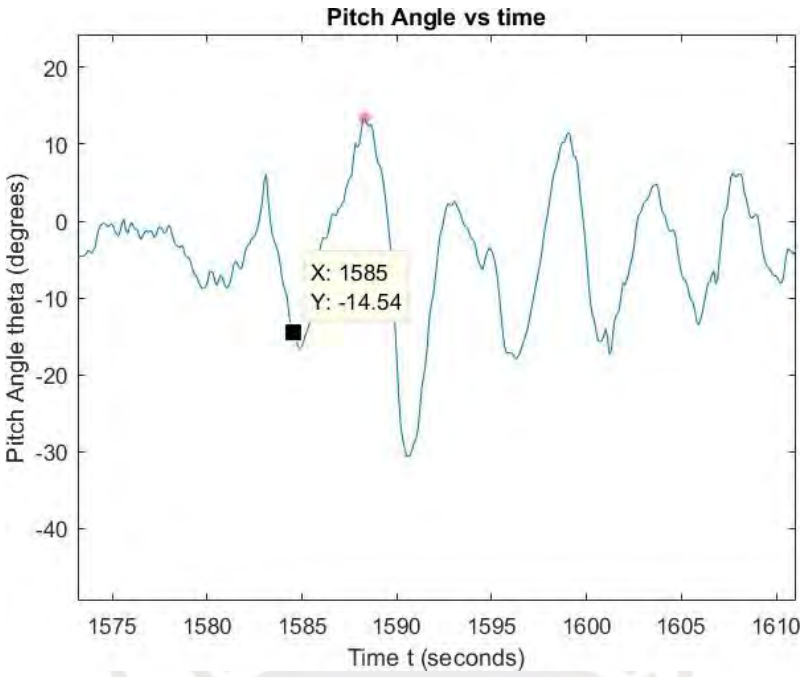


Figura 4. 25 Ángulo de Cabeceo de -14.54° en el instante de tiempo 1585 segundos.

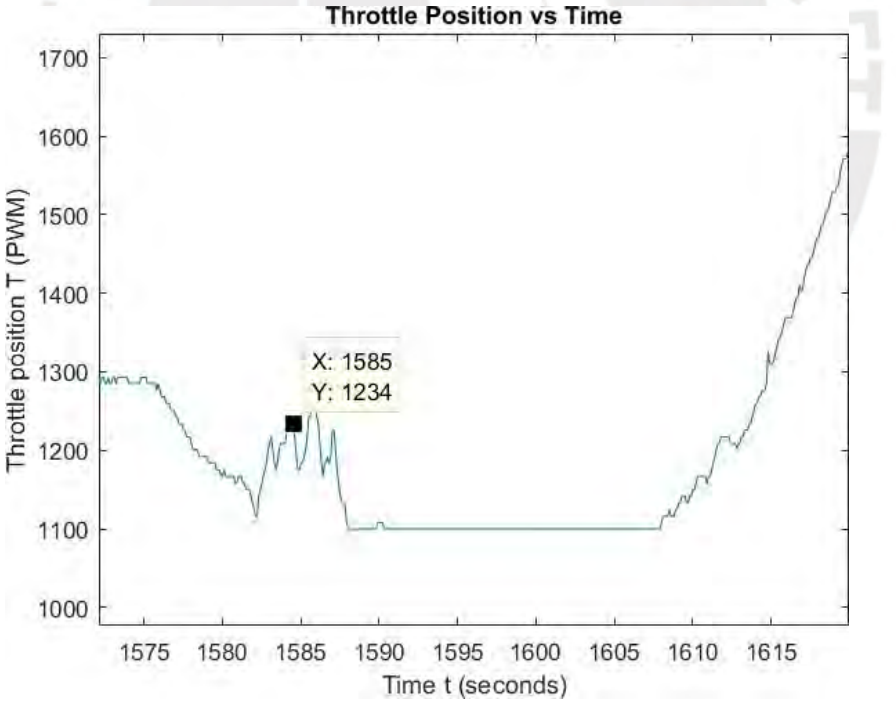


Figura 4. 26 Position de la Potencia del Motor en el tiempo 1585 segundos

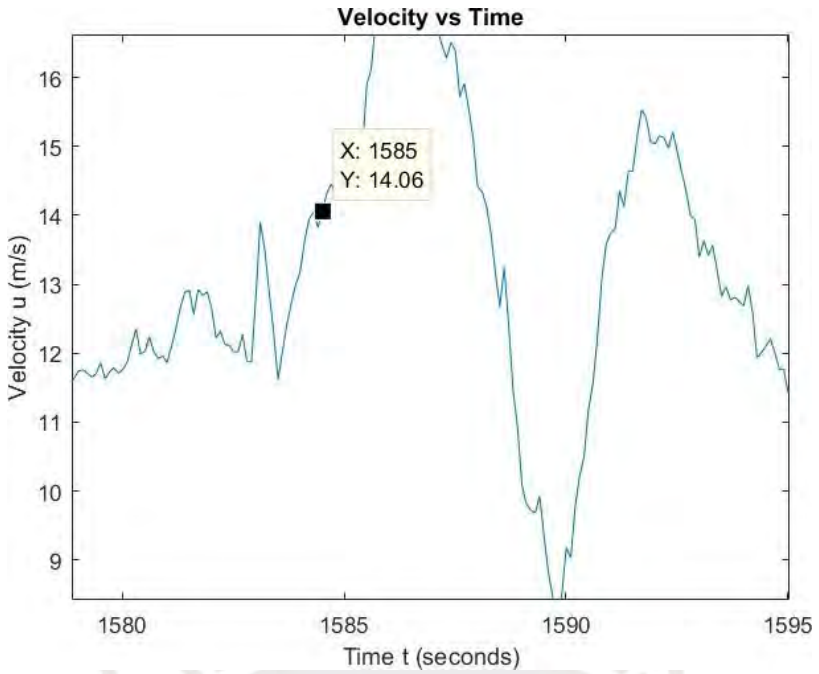


Figura 4. 27 Aumento de la Velocidad en el instante de tiempo 1585 segundos

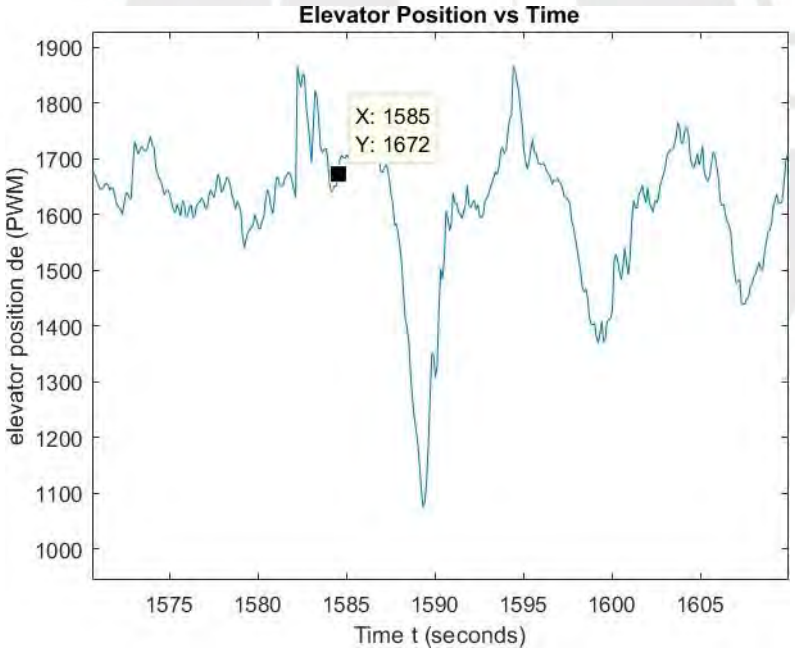


Figura 4. 28 Posición del Elevador en el tiempo 1585 segundos.

La figura 4.29 muestra a la aeronave realizando un vuelo recto y nivelado en modo autónomo.

Figura 4. 29 Aeronave realizando vuelo autónomo

CONCLUSIONES

- El entrenamiento de la Red Neuronal Artificial de dos capas del tipo Back Propagation Dinámico que se utilizó para obtener los resultados finales descritos en el Capítulo 4 ha permitido predecir el modelo dinámico de la aeronave para los parámetros de ángulo de cabeceo y velocidad, a partir de información del EKF que se basa en la información que se obtiene de los sensores a bordo de la aeronave.

- La información que se obtiene de los sensores a bordo de la aeronave debe de ser filtrada porque contiene ruido innecesario que no contribuye con el entrenamiento y la predicción del modelo dinámico de la aeronave. Utilizar un filtro de fase zero (zero phase filter) es la opción utilizada porque puede filtrar la información de manera correcta y sin desfasarla.

- Esta aeronave del tipo ala volante "mezcla" al momento de volar las señales de alabeo y cabeceo, por lo cual no pueden separarse los ejes longitudinales y laterales como en otros entrenamientos de aeronaves convencionales. Es por este motivo, que la

información de ambos ejes debe de ser utilizada en una sola RNA y al momento de realizar la validación.

- Se pudo realizar el modelamiento dinámico de la aeronave con un error de entrenamiento predictivo de 19.35 y un error de validación del modelo con un vuelo distinto de la misma aeronave de 29.41 en el mejor de lo casos. Asimismo, durante el proceso de entrenamiento se introdujo como valor de entrada a la RNA un "bias" para tomar en consideración las perturbaciones o turbulencias que existen durante el vuelo.

- Aumentar el número de neuronas intermedias de 10 a 100, en ambas capas de entrenamiento de la RNA, colaboro a obtener un menor porcentaje de error en el entrenamiento y por lo tanto mejorar los errores de validación con otros vuelos.

- Este trabajo de investigación tomo en consideración tres parámetros (ángulo de cabeceo, ángulo de alabeo y velocidad) que son importantes para sostener un vuelo recto y nivelado. Sin embargo, durante el proceso de entrenamiento se ha visto que no es posible entrenar de manera satisfactoria el ángulo de alabeo, pero es accesible entrenar los parámetros de ángulo de cabeceo y velocidad. Ambos parámetros son importantes para entender el motivo por el cual la aeronave al momento de realizar un giro pierde altitud.

RECOMENDACIONES

- Mejorar el entrenamiento de la RNA para obtener un mejor modelo dinámico de la aeronave. Es importante tomar en consideración la validación después de realizar cada entrenamiento para observar si los cambios realizados validan mejor la información de otros vuelos. El error de entrenamiento debería de disminuir por debajo de un 15%.

- Implementar un controlador que pueda utilizar el modelo dinámico encontrado con el RNA de la aeronave para mejorar el vuelo autónomo de la aeronave, especialmente al momento de los giros para que las superficies de control realicen las acciones correctivas y la aeronave no pierda altitud y utilice energía de manera innecesaria.

- Realizar el entrenamiento de la RNA utilizando la información de los ejes longitudinal y lateral / direccional de la aeronave. Las ecuaciones diferenciales de movimiento de la aeronave combinan valores de ambos ejes, por lo tanto, el entrenamiento debe de predecir todos estos valores que se combinan y utilizarlos para realizar el control de la aeronave.

- Se debería de crear modelos de aeronaves que integren ecuaciones de movimiento y datos experimentales al mismo tiempo.

BIBLIOGRAFÍA

1. Ardupilot, D. (8 de 4 de 2017). *Ardupilot Hardware Options*. Obtenido de Ardupilot Web Site: http://ardupilot.org/copter/docs/common-pixhawk-overview.html
2. Barton, J. D. (2012). Fundamental of Small Unmanned Aircraft Flight. *Johns Hopkins APL Technical Digest*, 132 - 149.
3. Behera, S. K., & Rana, D. (2014). Systema Identification Using Recurrent Neural Networks. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 8111 - 8117.
4. Brandt , S. A., Stiles, R. J., Bertin, J. J., & Whitford, R. (2004). *Introduction to Aeronautics: A Design Perspective.* Virginia: AIAA.
5. Chen, C.-q., & Ji, Y. (2010). Modular Aircraft Simulation Platfomr Based on Simulink. *International Conference on Mechatronics and Automation* (págs. 1454 - 1459). Xi'an, China: IEEE.
6. Dziuk, M., & Jamshidi, M. (2011). Fuzzy Logic Controlled UAV Autopilot Using C-Mean Clustering. *6th International Conference on System of System Engineering* (págs. 305 - 310). Alburquerque, New Mexico: IEEE.
7. Enomoto, M., & Yamamoto, Y. (2015). Modelling, Simulation and Navigation Experiments of Unmanned Aerial Vehicle. *International Conference in Mechatronics and Automation* (págs. 482 - 487). Beijing, China: IEEE.
8. Guzman, I. (06 de 09 de 2016). La PUCP Investiga con Drones a más de 5 mil metros de altura. *PuntoEDU.pe*, pág. 2.
9. Harris, J., Arthurs, F., Henrickson, J. V., & Valasek, J. (2016). Aircraft System Identification using Artificial Neural Networks with Flight Test Data. *2016 International Conference on Unmanned Aircraft Systems* (págs. 679 - 687). Arlington, VA: IEEE.
10. Haykin, S. (2002). *Neural Networks: A comprehensive Foundation".* Pearson Edition Asia.
11. IGP. (20 de 08 de 2016). Por Primera Vez Un Drone Sobrevuela el Volcan Ubinas. *El Comercio Perú*, pág. 2.
12. Jang, J., Sun, C., & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing, A computational Approach to Learning and Machine Inteligence".* Prentice Hall.
13. Jung, D., & Tsiotras, P. (2006). Modeling and Hardware-in-the-Loop Simulation for a Small Unmanned Aerial Vehicle. *America Institute of Aeronautics and Astronautics*, (págs. 1 -13). Atlanta, GA.

14. Kirkpatrick, K., May, J., & Valasek, J. (2013). Aircraft System Identification Using Artificial Neural Networks. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* Dallas/Ft. Worth Region, Texas: AIAA.

15. Kirkpatrick, K., May, J., & Valasek, J. (2013). Aircraft System Identification Using Artificial Neural Networks. *51st AIAA Aerospace Science Meeting including the Nwe Horizons Forum and Aerospace Exposition* (págs. 1 - 12). Grapevine, Texas: AIAA.

16. Louali, R., Bouaziz, S., Elouardi, A., & Abouzahir, M. (2014). Platform Simulation based Unmanned Aircraft System Desing. *2014 Second World Conference on Complex Systems (WCCS)* (págs. 736 - 742). Agadir, Morocco: IEEE.

17. Lung, P., & Geng, Q. (2011). Real -Time Simulation System for UAV based on MATLAB/SIMULINK. *Computing, Control and Industrial Engineering (CCIE)* . IEEE 2nd International Conference.

18. Nusyirwan, I. F. (s.f.). *Engineering Flight Simulator Using MATLAB, Puthon and FLIGHT GEAR.* Malaysia: Universiti Teknologi Malaysia.

19. Ribeiro, L., & Oliveira, N. F. (2010). UAV Autopilot Controllers Test Platform Using Matlab/Simulink and X-Plane. *40th ASEE/IEEE Frontiers in Education Conference* (págs. 1 - 6). Washington, DC: IEEE.

20. Roudbari, A., & Saghafi, F. (2014). Intelligent Modeling and Identification of Aircraft Nonlinear Flight. *Chinese Journal of Aeronautics*, 759 - 771.

21. Stevens, B. L., & Lewis, F. L. (2003). *Aircraft Control and Simulation* (2nd ed.). Hobokren, NJ: John Wiley & Sons.

22. *Understanding RC airplane controls*. (08 de 04 de 2017). Obtenido de R/C Airplane World: http://www.rc-airplane-world.com/rc-airplane-controls.html

23. Warsi, F. A., Hazry, D., Ahmed, F., Joyo, K., Tanveer , H., Kamarudin, H., & Razlan, Z. (2014). Yaw, Pitch and Roll Controller Design for Fixed-wing UAV under uncertainty and perturbed Condition. *10th International Colloquium on Signal Processing and Its Applications* (págs. 151 - 156). Kuala Lumpur, Malaysia: IEEE.

24. Yechout, T. R., Morris, S. L., Bossert, D. E., & Hallgren, W. F. (2003). *Introduction to Aircraft Flight Mechanics.* Virginia: AIAA.

ANEXOS

ANEXO 01

```
%% Data Longitudinal %%
% Grafica de Valores para la información de los vuelos.

clear all
close all
clc

% Permite utilizar la data del IMU del Controlador de Vuelo
data_flight01 = load('20160824_NEX7_V3_ICA_Z12.BIN-1068244.mat');

i = 5; %numero de filas que se utilizan para mostrar.
pitch_angle = data_flight01.ATT([51793:i:76748], [2 6]); % ángulo de cabeceo
pitch_rate = data_flight01.IMU([51787:i:76737], [2 4]); % velocida de cambio del ángulo de cabeceo
velocity = data_flight01.ARSP([10357:15346],[2 3]); % velocidad de vuelo
Elevator = data_flight01.RCOU([10356:15345], [2 4]); % señal PWM del Elevador
Throttle = data_flight01.RCOU([10356:15345], [2 5]); % señal PWM del Throttle


figure(01); % Grafica el ángulo de cabeceo de la aeronave en el tiempo.
plot(pitch_angle(:,1)*1.0e-3, pitch_angle(:,2)); %(X, Y)
title('Pitch Angle vs time')
ylabel('Pitch Angle theta (degrees)')
xlabel('Time t (seconds)')

figure(02); % Grafica la velocidad de cambio del cabeceo en el tiempo.
plot(pitch_rate(:,1)*1.0e-3, pitch_rate(:,2)); %(X, Y)
title('Pitch Rate vs Time')
ylabel('Pitch Rate q (deg/sec)')
xlabel('Time t (seconds)')

figure(03) % Grafica la velocidad de vuelo de la aeronave en el tiempo.
plot(velocity(:,1)*1.0e-3, velocity(:,2)); %(X, Y)
title('Velocity vs Time')
ylabel('Velocity u (m/s)')
xlabel('Time t (seconds)')

figure(04) % Grafica la señal PWM que controla la posición del elevador en el tiempo.
plot(Elevator(:,1)*1.0e-3, Elevator(:,2)); %(X, Y)
title('Elevator Position vs Time')
ylabel('elevator position de (PWM)')
xlabel('Time t (seconds)')

figure(05) % Grafica la señal PWM que controla la posición del Throttle en el tiempo.
plot(Throttle(:,1)*1.0e-3, Throttle(:,2)); %(X, Y)
title('Throttle Position vs Time')
```

```
ylabel('Throttle position T (PWM)')
xlabel('Time t (seconds)')


%% Lateral/Directional Data %%
% Información requerida de la aeronave para el analisis del eje lateral /
% direccional.

i = 5; %numero de filas que se utilizan para mostrar.
bank_angle = data_flight01.ATT([51793:i:76748], [2 4]); % ángulo de banqueo
roll_rate = data_flight01.IMU([51787:i:76737], [2 3]); % velocidad de cambio del ángulo de banqueo
yaw_angle = data_flight01.ATT([51793:i:76748], [2 8]); % ángulo de guiñada
yaw_rate = data_flight01.IMU([51787:i:76737], [2 5]); % velocidad de cambio del ángulo de guiñada
Ailerons = data_flight01.RCOU([10356:15345], [2 3]); % señal PWM de los alerones

figure(06); % grafica el ángulo de banqueo en el tiempo.
plot(bank_angle(:,1)*1.0e-3, bank_angle(:,2)); %(X, Y)
title('Bank Angle vs time')
ylabel('Bank Angle phi (degrees)')
xlabel('Time t (seconds)')

figure(07); % grafica la velocidad de cambio del ángulo de banqueo en el tiempo.
plot(roll_rate(:,1)*1.0e-3, roll_rate(:,2)); %(X, Y)
title('Roll Rate vs Time')
ylabel('Roll Rate p (deg/sec)')
xlabel('Time t (seconds)')

figure(08); % grafica el ángulo de guiñada en el tiempo.
plot(yaw_angle(:,1)*1.0e-3, yaw_angle(:,2)); %(X, Y)
title('Yaw Angle vs time')
ylabel('Yaw Angle psi (degrees)')
xlabel('Time t (seconds)')

figure(09); % grafica la velocidad de cambio de la guiñada en el tiempo.
plot(yaw_rate(:,1)*1.0e-3, yaw_rate(:,2)); %(X, Y)
title('Yaw Rate vs Time')
ylabel('Yaw Rate r (deg/sec)')
xlabel('Time t (seconds)')

figure(10) % grafica la señal PWM de los alerones.
plot(Ailerons(:,1)*1.0e-3, Ailerons(:,2));
title('Ailerons Position vs Time')
ylabel('Ailerons position da (PWM)')
xlabel('Time t (seconds)')
```

ANEXO 02

```
%%Red Neuronal Artificial%%

close all
clc

% Se cargan los datos de cada una de las variables de entrada.
load pitch_angle_filter_V1
load pitch_angle_filter_V3
load velocity_filter_V1
load velocity_filter_V3
load elevator_filter_V1
load elevator_filter_V3
load throttle_filter_V1
load throttle_filter_V3
load roll_angle_filter_V1
load roll_angle_filter_V3
load ailerons_filter_V1
load ailerons_filter_V3
load accel_x_filter_V1
load accel_x_filter_V3
load accel_y_filter_V1
load accel_y_filter_V3
load accel_z_filter_V1
load accel_z_filter_V3
load altitude_filter_V1
load altitude_filter_V3

% Número de puntos que serán tomados para el entrenamiento.
nt = 2000;

% Se grafican todas los estados de entrada y se estandarizan los valores
% para que esten entre -0.8 a 0.8. Este rango debe de ser el mismo para
% todas las señales y se utilizán factores de escalamiento.
figure(1);
pitch_angle_filter_V1 = 0.03*pitch_angle_filter_V1(1:nt,1);
plot(pitch_angle_filter_V1, 'b');
hold on
pitch_angle_filter_V3 = 0.03*pitch_angle_filter_V3(1:nt,1);
plot(pitch_angle_filter_V3,'r');
grid;
title('pitch angle')

% Se calcula el ratio de pitch, es decir la velocidad de cambio del pitch
% en el tiempo.
figure(2);
pitch_rate_filter_V1 = diff(pitch_angle_filter_V1)/0.1;
pitch_rate_filter_V1 = [0; pitch_rate_filter_V1];
pitch_rate_filter_V1 = 1.30*pitch_rate_filter_V1(1:nt,1);
```

```
plot(pitch_rate_filter_V1, 'b');
hold on
pitch_rate_filter_V3 = diff(pitch_angle_filter_V3)/0.1;
pitch_rate_filter_V3 = [0; pitch_rate_filter_V3];
pitch_rate_filter_V3 = 1.30*pitch_rate_filter_V3(1:nt,1);
plot(pitch_rate_filter_V3, 'r');
grid;
title('pitch rate')

figure(3);
velocity_filter_V1 = 0.045*(velocity_filter_V1(1:nt,1));
plot(velocity_filter_V1, 'b');
hold on
velocity_filter_V3 = 0.045*(velocity_filter_V3(1:nt,1));
plot(velocity_filter_V3, 'r');
grid;
title('velocity')

figure(4);
elevator_filter_V1 = 0.047*(elevator_filter_V1(1:nt,1));
plot(elevator_filter_V1, 'b');
hold on
elevator_filter_V3 = 0.047*(elevator_filter_V3(1:nt,1));
plot(elevator_filter_V3, 'r');
grid;
title('elevator')

figure(5);
throttle_filter_V1 = 0.011*(throttle_filter_V1(1:nt,1));
plot(throttle_filter_V1, 'b');
hold on
throttle_filter_V3 = 0.011*(throttle_filter_V3(1:nt,1));
plot(throttle_filter_V3, 'r');
grid;
title('throttle')

figure(6);
roll_angle_filter_V1 = 0.015*roll_angle_filter_V1(1:nt,1);
plot(roll_angle_filter_V1, 'b');
hold on
roll_angle_filter_V3 = 0.015*roll_angle_filter_V3(1:nt,1);
plot(roll_angle_filter_V3, 'r');
grid;
title('roll angle')

figure(7);
roll_rate_filter_V1 = diff(roll_angle_filter_V1)/0.1;
roll_rate_filter_V1 = [0; roll_rate_filter_V1];
roll_rate_filter_V1 = 1.90*roll_rate_filter_V1(1:nt,1);
plot(roll_rate_filter_V1, 'b');
```

```
99 hold on
100 roll_rate_filter_V3 = diff(roll_angle_filter_V3)/0.1;
101 roll_rate_filter_V3 = [0; roll_rate_filter_V3];
102 roll_rate_filter_V3 = 1.90*roll_rate_filter_V3(1:nt,1);
103 plot(roll_rate_filter_V3, 'r');
104 grid;
105 title('roll rate')
106
107 figure(8);
108 ailerons_filter_V1 = 0.047*ailerons_filter_V1(1:nt,1);
109 plot(ailerons_filter_V1, 'b');
110 hold on
111 ailerons_filter_V3 = 0.047*ailerons_filter_V3(1:nt,1);
112 plot(ailerons_filter_V3, 'r');
113 grid;
114 title('ailerons')
115
116 figure(9);
117 accel_x_filter_V1 = 0.2*accel_x_filter_V1(1:nt,1);
118 plot(accel_x_filter_V1, 'b');
119 hold on
120 accel_x_filter_V3 = 0.2*accel_x_filter_V3(1:nt,1);
121 plot(accel_x_filter_V3, 'r');
122 grid;
123 title('accel X')
124
125 figure(10);
126 accel_y_filter_V1 = accel_y_filter_V1(1:nt,1);
127 plot(accel_y_filter_V1, 'b');
128 hold on
129 accel_y_filter_V3 = accel_y_filter_V3(1:nt,1);
130 plot(accel_y_filter_V3, 'r');
131 grid;
132 title('accel Y')
133
134 figure(11);
135 accel_z_filter_V1 = 0.0375*accel_z_filter_V1(1:nt,1);
136 plot(accel_z_filter_V1, 'b');
137 hold on
138 accel_z_filter_V3 = 0.0375*accel_z_filter_V3(1:nt,1);
139 plot(accel_z_filter_V3, 'r');
140 grid;
141 title('accel Z')
142
143 figure(12);
144 altitude_filter_V1 = 0.00465*altitude_filter_V1(1:nt,1);
145 plot(altitude_filter_V1, 'b');
146 hold on
147 altitude_filter_V3 = 0.00465*altitude_filter_V3(1:nt,1);
148 plot(altitude_filter_V3, 'r');
```

```
149 grid;
150 title('altitude')
151
152 % Se han añadido datos "anteriores" para que la RNA pueda utilizar al
153 % momento de realizar la predicción. Al inicio esta data es igual a cero y
154 % se utilizan 10 valores anteriores.
155 elevator_filter_old1_V1(1,1) = 0;
156 elevator_filter_old1_V1(2:nt,1) = elevator_filter_V1(1:nt-1,1);
157
158 throttle_filter_old1_V1(1,1) = 0;
159 throttle_filter_old1_V1(2:nt,1) = throttle_filter_V1(1:nt-1,1);
160
161 ailerons_filter_old1_V1(1,1) = 0;
162 ailerons_filter_old1_V1(2:nt,1) = ailerons_filter_V1(1:nt-1,1);
163
164 accel_x_filter_old1_V1(1,1) = 0;
165 accel_x_filter_old1_V1(2:nt,1) = accel_x_filter_V1(1:nt-1,1);
166
167 accel_y_filter_old1_V1(1,1) = 0;
168 accel_y_filter_old1_V1(2:nt,1) = accel_y_filter_V1(1:nt-1,1);
169
170 accel_z_filter_old1_V1(1,1) = 0;
171 accel_z_filter_old1_V1(2:nt,1) = accel_z_filter_V1(1:nt-1,1);
172
173 elevator_filter_old2_V1(1,1) = 0;
174 elevator_filter_old2_V1(2:nt,1) = elevator_filter_old1_V1(1:nt-1,1);
175
176 throttle_filter_old2_V1(1,1) = 0;
177 throttle_filter_old2_V1(2:nt,1) = throttle_filter_old1_V1(1:nt-1,1);
178
179 ailerons_filter_old2_V1(1,1) = 0;
180 ailerons_filter_old2_V1(2:nt,1) = ailerons_filter_old1_V1(1:nt-1,1);
181
182 accel_x_filter_old2_V1(1,1) = 0;
183 accel_x_filter_old2_V1(2:nt,1) = accel_x_filter_old1_V1(1:nt-1,1);
184
185 accel_y_filter_old2_V1(1,1) = 0;
186 accel_y_filter_old2_V1(2:nt,1) = accel_y_filter_old1_V1(1:nt-1,1);
187
188 accel_z_filter_old2_V1(1,1) = 0;
189 accel_z_filter_old2_V1(2:nt,1) = accel_z_filter_old1_V1(1:nt-1,1);
190
191 elevator_filter_old3_V1(1,1) = 0;
192 elevator_filter_old3_V1(2:nt,1) = elevator_filter_old2_V1(1:nt-1,1);
193
194 throttle_filter_old3_V1(1,1) = 0;
195 throttle_filter_old3_V1(2:nt,1) = throttle_filter_old2_V1(1:nt-1,1);
196
197 ailerons_filter_old3_V1(1,1) = 0;
198 ailerons_filter_old3_V1(2:nt,1) = ailerons_filter_old2_V1(1:nt-1,1);
```

```
199
200 accel_x_filter_old3_V1(1,1) = 0;
201 accel_x_filter_old3_V1(2:nt,1) = accel_x_filter_old2_V1(1:nt-1,1);
202
203 accel_y_filter_old3_V1(1,1) = 0;
204 accel_y_filter_old3_V1(2:nt,1) = accel_y_filter_old2_V1(1:nt-1,1);
205
206 accel_z_filter_old3_V1(1,1) = 0;
207 accel_z_filter_old3_V1(2:nt,1) = accel_z_filter_old2_V1(1:nt-1,1);
208
209 elevator_filter_old4_V1(1,1) = 0;
210 elevator_filter_old4_V1(2:nt,1) = elevator_filter_old3_V1(1:nt-1,1);
211
212 throttle_filter_old4_V1(1,1) = 0;
213 throttle_filter_old4_V1(2:nt,1) = throttle_filter_old3_V1(1:nt-1,1);
214
215 ailerons_filter_old4_V1(1,1) = 0;
216 ailerons_filter_old4_V1(2:nt,1) = ailerons_filter_old3_V1(1:nt-1,1);
217
218 accel_x_filter_old4_V1(1,1) = 0;
219 accel_x_filter_old4_V1(2:nt,1) = accel_x_filter_old3_V1(1:nt-1,1);
220
221 accel_y_filter_old4_V1(1,1) = 0;
222 accel_y_filter_old4_V1(2:nt,1) = accel_y_filter_old3_V1(1:nt-1,1);
223
224 accel_z_filter_old4_V1(1,1) = 0;
225 accel_z_filter_old4_V1(2:nt,1) = accel_z_filter_old3_V1(1:nt-1,1);
226
227 elevator_filter_old5_V1(1,1) = 0;
228 elevator_filter_old5_V1(2:nt,1) = elevator_filter_old4_V1(1:nt-1,1);
229
230 throttle_filter_old5_V1(1,1) = 0;
231 throttle_filter_old5_V1(2:nt,1) = throttle_filter_old4_V1(1:nt-1,1);
232
233 ailerons_filter_old5_V1(1,1) = 0;
234 ailerons_filter_old5_V1(2:nt,1) = ailerons_filter_old4_V1(1:nt-1,1);
235
236 accel_x_filter_old5_V1(1,1) = 0;
237 accel_x_filter_old5_V1(2:nt,1) = accel_x_filter_old4_V1(1:nt-1,1);
238
239 accel_y_filter_old5_V1(1,1) = 0;
240 accel_y_filter_old5_V1(2:nt,1) = accel_y_filter_old4_V1(1:nt-1,1);
241
242 accel_z_filter_old5_V1(1,1) = 0;
243 accel_z_filter_old5_V1(2:nt,1) = accel_z_filter_old4_V1(1:nt-1,1);
244
245 elevator_filter_old6_V1(1,1) = 0;
246 elevator_filter_old6_V1(2:nt,1) = elevator_filter_old5_V1(1:nt-1,1);
247
248 throttle_filter_old6_V1(1,1) = 0;
```

```
249 throttle_filter_old6_V1(2:nt,1) = throttle_filter_old5_V1(1:nt-1,1);
250
251 ailerons_filter_old6_V1(1,1) = 0;
252 ailerons_filter_old6_V1(2:nt,1) = ailerons_filter_old5_V1(1:nt-1,1);
253
254 accel_x_filter_old6_V1(1,1) = 0;
255 accel_x_filter_old6_V1(2:nt,1) = accel_x_filter_old5_V1(1:nt-1,1);
256
257 accel_y_filter_old6_V1(1,1) = 0;
258 accel_y_filter_old6_V1(2:nt,1) = accel_y_filter_old5_V1(1:nt-1,1);
259
260 accel_z_filter_old6_V1(1,1) = 0;
261 accel_z_filter_old6_V1(2:nt,1) = accel_z_filter_old5_V1(1:nt-1,1);
262
263 elevator_filter_old7_V1(1,1) = 0;
264 elevator_filter_old7_V1(2:nt,1) = elevator_filter_old6_V1(1:nt-1,1);
265
266 throttle_filter_old7_V1(1,1) = 0;
267 throttle_filter_old7_V1(2:nt,1) = throttle_filter_old6_V1(1:nt-1,1);
268
269 ailerons_filter_old7_V1(1,1) = 0;
270 ailerons_filter_old7_V1(2:nt,1) = ailerons_filter_old6_V1(1:nt-1,1);
271
272 accel_x_filter_old7_V1(1,1) = 0;
273 accel_x_filter_old7_V1(2:nt,1) = accel_x_filter_old6_V1(1:nt-1,1);
274
275 accel_y_filter_old7_V1(1,1) = 0;
276 accel_y_filter_old7_V1(2:nt,1) = accel_y_filter_old6_V1(1:nt-1,1);
277
278 accel_z_filter_old7_V1(1,1) = 0;
279 accel_z_filter_old7_V1(2:nt,1) = accel_z_filter_old6_V1(1:nt-1,1);
280
281 elevator_filter_old8_V1(1,1) = 0;
282 elevator_filter_old8_V1(2:nt,1) = elevator_filter_old7_V1(1:nt-1,1);
283
284 throttle_filter_old8_V1(1,1) = 0;
285 throttle_filter_old8_V1(2:nt,1) = throttle_filter_old7_V1(1:nt-1,1);
286
287 ailerons_filter_old8_V1(1,1) = 0;
288 ailerons_filter_old8_V1(2:nt,1) = ailerons_filter_old7_V1(1:nt-1,1);
289
290 accel_x_filter_old8_V1(1,1) = 0;
291 accel_x_filter_old8_V1(2:nt,1) = accel_x_filter_old7_V1(1:nt-1,1);
292
293 accel_y_filter_old8_V1(1,1) = 0;
294 accel_y_filter_old8_V1(2:nt,1) = accel_y_filter_old7_V1(1:nt-1,1);
295
296 accel_z_filter_old8_V1(1,1) = 0;
297 accel_z_filter_old8_V1(2:nt,1) = accel_z_filter_old7_V1(1:nt-1,1);
298
```

```
299 elevator_filter_old9_V1(1,1) = 0;
300 elevator_filter_old9_V1(2:nt,1) = elevator_filter_old8_V1(1:nt-1,1);
301
302 throttle_filter_old9_V1(1,1) = 0;
303 throttle_filter_old9_V1(2:nt,1) = throttle_filter_old8_V1(1:nt-1,1);
304
305 ailerons_filter_old9_V1(1,1) = 0;
306 ailerons_filter_old9_V1(2:nt,1) = ailerons_filter_old8_V1(1:nt-1,1);
307
308 accel_x_filter_old9_V1(1,1) = 0;
309 accel_x_filter_old9_V1(2:nt,1) = accel_x_filter_old8_V1(1:nt-1,1);
310
311 accel_y_filter_old9_V1(1,1) = 0;
312 accel_y_filter_old9_V1(2:nt,1) = accel_y_filter_old8_V1(1:nt-1,1);
313
314 accel_z_filter_old9_V1(1,1) = 0;
315 accel_z_filter_old9_V1(2:nt,1) = accel_z_filter_old8_V1(1:nt-1,1);
316
317 elevator_filter_old10_V1(1,1) = 0;
318 elevator_filter_old10_V1(2:nt,1) = elevator_filter_old9_V1(1:nt-1,1);
319
320 throttle_filter_old10_V1(1,1) = 0;
321 throttle_filter_old10_V1(2:nt,1) = throttle_filter_old9_V1(1:nt-1,1);
322
323 ailerons_filter_old10_V1(1,1) = 0;
324 ailerons_filter_old10_V1(2:nt,1) = ailerons_filter_old9_V1(1:nt-1,1);
325
326 accel_x_filter_old10_V1(1,1) = 0;
327 accel_x_filter_old10_V1(2:nt,1) = accel_x_filter_old9_V1(1:nt-1,1);
328
329 accel_y_filter_old10_V1(1,1) = 0;
330 accel_y_filter_old10_V1(2:nt,1) = accel_y_filter_old9_V1(1:nt-1,1);
331
332 accel_z_filter_old10_V1(1,1) = 0;
333 accel_z_filter_old10_V1(2:nt,1) = accel_z_filter_old9_V1(1:nt-1,1);
334
335 elevator_filter_old1_V3(1,1) = 0;
336 elevator_filter_old1_V3(2:nt,1) = elevator_filter_V3(1:nt-1,1);
337
338 throttle_filter_old1_V3(1,1) = 0;
339 throttle_filter_old1_V3(2:nt,1) = throttle_filter_V3(1:nt-1,1);
340
341 ailerons_filter_old1_V3(1,1) = 0;
342 ailerons_filter_old1_V3(2:nt,1) = ailerons_filter_V3(1:nt-1,1);
343
344 accel_x_filter_old1_V3(1,1) = 0;
345 accel_x_filter_old1_V3(2:nt,1) = accel_x_filter_V3(1:nt-1,1);
346
347 accel_y_filter_old1_V3(1,1) = 0;
348 accel_y_filter_old1_V3(2:nt,1) = accel_y_filter_V3(1:nt-1,1);
```

```
349
350 accel_z_filter_old1_V3(1,1) = 0;
351 accel_z_filter_old1_V3(2:nt,1) = accel_z_filter_V3(1:nt-1,1);
352
353 elevator_filter_old2_V3(1,1) = 0;
354 elevator_filter_old2_V3(2:nt,1) = elevator_filter_old1_V3(1:nt-1,1);
355
356 throttle_filter_old2_V3(1,1) = 0;
357 throttle_filter_old2_V3(2:nt,1) = throttle_filter_old1_V3(1:nt-1,1);
358
359 ailerons_filter_old2_V3(1,1) = 0;
360 ailerons_filter_old2_V3(2:nt,1) = ailerons_filter_old1_V3(1:nt-1,1);
361
362 accel_x_filter_old2_V3(1,1) = 0;
363 accel_x_filter_old2_V3(2:nt,1) = accel_x_filter_old1_V3(1:nt-1,1);
364
365 accel_y_filter_old2_V3(1,1) = 0;
366 accel_y_filter_old2_V3(2:nt,1) = accel_y_filter_old1_V3(1:nt-1,1);
367
368 accel_z_filter_old2_V3(1,1) = 0;
369 accel_z_filter_old2_V3(2:nt,1) = accel_z_filter_old1_V3(1:nt-1,1);
370
371 elevator_filter_old3_V3(1,1) = 0;
372 elevator_filter_old3_V3(2:nt,1) = elevator_filter_old2_V3(1:nt-1,1);
373
374 throttle_filter_old3_V3(1,1) = 0;
375 throttle_filter_old3_V3(2:nt,1) = throttle_filter_old2_V3(1:nt-1,1);
376
377 ailerons_filter_old3_V3(1,1) = 0;
378 ailerons_filter_old3_V3(2:nt,1) = ailerons_filter_old2_V3(1:nt-1,1);
379
380 accel_x_filter_old3_V3(1,1) = 0;
381 accel_x_filter_old3_V3(2:nt,1) = accel_x_filter_old2_V3(1:nt-1,1);
382
383 accel_y_filter_old3_V3(1,1) = 0;
384 accel_y_filter_old3_V3(2:nt,1) = accel_y_filter_old2_V3(1:nt-1,1);
385
386 accel_z_filter_old3_V3(1,1) = 0;
387 accel_z_filter_old3_V3(2:nt,1) = accel_z_filter_old2_V3(1:nt-1,1);
388
389 elevator_filter_old4_V3(1,1) = 0;
390 elevator_filter_old4_V3(2:nt,1) = elevator_filter_old3_V3(1:nt-1,1);
391
392 throttle_filter_old4_V3(1,1) = 0;
393 throttle_filter_old4_V3(2:nt,1) = throttle_filter_old3_V3(1:nt-1,1);
394
395 ailerons_filter_old4_V3(1,1) = 0;
396 ailerons_filter_old4_V3(2:nt,1) = ailerons_filter_old3_V3(1:nt-1,1);
397
398 accel_x_filter_old4_V3(1,1) = 0;
```

```
399 accel_x_filter_old4_V3(2:nt,1) = accel_x_filter_old3_V3(1:nt-1,1);
400
401 accel_y_filter_old4_V3(1,1) = 0;
402 accel_y_filter_old4_V3(2:nt,1) = accel_y_filter_old3_V3(1:nt-1,1);
403
404 accel_z_filter_old4_V3(1,1) = 0;
405 accel_z_filter_old4_V3(2:nt,1) = accel_z_filter_old3_V3(1:nt-1,1);
406
407 elevator_filter_old5_V3(1,1) = 0;
408 elevator_filter_old5_V3(2:nt,1) = elevator_filter_old4_V3(1:nt-1,1);
409
410 throttle_filter_old5_V3(1,1) = 0;
411 throttle_filter_old5_V3(2:nt,1) = throttle_filter_old4_V3(1:nt-1,1);
412
413 ailerons_filter_old5_V3(1,1) = 0;
414 ailerons_filter_old5_V3(2:nt,1) = ailerons_filter_old4_V3(1:nt-1,1);
415
416 accel_x_filter_old5_V3(1,1) = 0;
417 accel_x_filter_old5_V3(2:nt,1) = accel_x_filter_old4_V3(1:nt-1,1);
418
419 accel_y_filter_old5_V3(1,1) = 0;
420 accel_y_filter_old5_V3(2:nt,1) = accel_y_filter_old4_V3(1:nt-1,1);
421
422 accel_z_filter_old5_V3(1,1) = 0;
423 accel_z_filter_old5_V3(2:nt,1) = accel_z_filter_old4_V3(1:nt-1,1);
424
425 elevator_filter_old6_V3(1,1) = 0;
426 elevator_filter_old6_V3(2:nt,1) = elevator_filter_old5_V3(1:nt-1,1);
427
428 throttle_filter_old6_V3(1,1) = 0;
429 throttle_filter_old6_V3(2:nt,1) = throttle_filter_old5_V3(1:nt-1,1);
430
431 ailerons_filter_old6_V3(1,1) = 0;
432 ailerons_filter_old6_V3(2:nt,1) = ailerons_filter_old5_V3(1:nt-1,1);
433
434 accel_x_filter_old6_V3(1,1) = 0;
435 accel_x_filter_old6_V3(2:nt,1) = accel_x_filter_old5_V3(1:nt-1,1);
436
437 accel_y_filter_old6_V3(1,1) = 0;
438 accel_y_filter_old6_V3(2:nt,1) = accel_y_filter_old5_V3(1:nt-1,1);
439
440 accel_z_filter_old6_V3(1,1) = 0;
441 accel_z_filter_old6_V3(2:nt,1) = accel_z_filter_old5_V3(1:nt-1,1);
442
443 elevator_filter_old7_V3(1,1) = 0;
444 elevator_filter_old7_V3(2:nt,1) = elevator_filter_old6_V3(1:nt-1,1);
445
446 throttle_filter_old7_V3(1,1) = 0;
447 throttle_filter_old7_V3(2:nt,1) = throttle_filter_old6_V3(1:nt-1,1);
448
```

```
449 ailerons_filter_old7_V3(1,1) = 0;
450 ailerons_filter_old7_V3(2:nt,1) = ailerons_filter_old6_V3(1:nt-1,1);
451
452 accel_x_filter_old7_V3(1,1) = 0;
453 accel_x_filter_old7_V3(2:nt,1) = accel_x_filter_old6_V3(1:nt-1,1);
454
455 accel_y_filter_old7_V3(1,1) = 0;
456 accel_y_filter_old7_V3(2:nt,1) = accel_y_filter_old6_V3(1:nt-1,1);
457
458 accel_z_filter_old7_V3(1,1) = 0;
459 accel_z_filter_old7_V3(2:nt,1) = accel_z_filter_old6_V3(1:nt-1,1);
460
461 elevator_filter_old8_V3(1,1) = 0;
462 elevator_filter_old8_V3(2:nt,1) = elevator_filter_old7_V3(1:nt-1,1);
463
464 throttle_filter_old8_V3(1,1) = 0;
465 throttle_filter_old8_V3(2:nt,1) = throttle_filter_old7_V3(1:nt-1,1);
466
467 ailerons_filter_old8_V3(1,1) = 0;
468 ailerons_filter_old8_V3(2:nt,1) = ailerons_filter_old7_V3(1:nt-1,1);
469
470 accel_x_filter_old8_V3(1,1) = 0;
471 accel_x_filter_old8_V3(2:nt,1) = accel_x_filter_old7_V3(1:nt-1,1);
472
473 accel_y_filter_old8_V3(1,1) = 0;
474 accel_y_filter_old8_V3(2:nt,1) = accel_y_filter_old7_V3(1:nt-1,1);
475
476 accel_z_filter_old8_V3(1,1) = 0;
477 accel_z_filter_old8_V3(2:nt,1) = accel_z_filter_old7_V3(1:nt-1,1);
478
479 elevator_filter_old9_V3(1,1) = 0;
480 elevator_filter_old9_V3(2:nt,1) = elevator_filter_old8_V3(1:nt-1,1);
481
482 throttle_filter_old9_V3(1,1) = 0;
483 throttle_filter_old9_V3(2:nt,1) = throttle_filter_old8_V3(1:nt-1,1);
484
485 ailerons_filter_old9_V3(1,1) = 0;
486 ailerons_filter_old9_V3(2:nt,1) = ailerons_filter_old8_V3(1:nt-1,1);
487
488 accel_x_filter_old9_V3(1,1) = 0;
489 accel_x_filter_old9_V3(2:nt,1) = accel_x_filter_old8_V3(1:nt-1,1);
490
491 accel_y_filter_old9_V3(1,1) = 0;
492 accel_y_filter_old9_V3(2:nt,1) = accel_y_filter_old8_V3(1:nt-1,1);
493
494 accel_z_filter_old9_V3(1,1) = 0;
495 accel_z_filter_old9_V3(2:nt,1) = accel_z_filter_old8_V3(1:nt-1,1);
496
497 elevator_filter_old10_V3(1,1) = 0;
498 elevator_filter_old10_V3(2:nt,1) = elevator_filter_old9_V3(1:nt-1,1);
```

```
499
500 throttle_filter_old10_V3(1,1) = 0;
501 throttle_filter_old10_V3(2:nt,1) = throttle_filter_old9_V3(1:nt-1,1);
502
503 ailerons_filter_old10_V3(1,1) = 0;
504 ailerons_filter_old10_V3(2:nt,1) = ailerons_filter_old9_V3(1:nt-1,1);
505
506 accel_x_filter_old10_V3(1,1) = 0;
507 accel_x_filter_old10_V3(2:nt,1) = accel_x_filter_old9_V3(1:nt-1,1);
508
509 accel_y_filter_old10_V3(1,1) = 0;
510 accel_y_filter_old10_V3(2:nt,1) = accel_y_filter_old9_V3(1:nt-1,1);
511
512 accel_z_filter_old10_V3(1,1) = 0;
513 accel_z_filter_old10_V3(2:nt,1) = accel_z_filter_old9_V3(1:nt-1,1);
514
515 % El bias es una variable de entrada que se introduce para tomar en
516 % consideración las perturbaciones y turbulencias que se originan durante
517 % el vuelo. Algunas consideraciones que se deben de tomar en consideración:
518 % Auemntar el bias mejora la velocidad de entrenamiento, pero no es
519 % eficiente para la validación.
520 % Disminuir el bias disminuye la velocidad de entrenamiento, pero mejora la
521 % validación.
522 bias = 0.5*ones(nt,1);
523
524 % Variables de entrada
525 u1 = [ elevator_filter_V1 throttle_filter_V1 ailerons_filter_V1 accel_x_filter_V1
accel_y_filter_V1
accel_z_filter_V1 elevator_filter_old1_V1 throttle_filter_old1_V1 ailerons_filter_old1_V1
accel_x_filter_old1_V1
accel_y_filter_old1_V1 accel_z_filter_old1_V1 elevator_filter_old2_V1
throttle_filter_old2_V1
ailerons_filter_old2_V1 accel_x_filter_old2_V1 accel_y_filter_old2_V1
accel_z_filter_old2_V1
elevator_filter_old3_V1 throttle_filter_old3_V1 ailerons_filter_old3_V1
accel_x_filter_old3_V1
accel_y_filter_old3_V1 accel_z_filter_old3_V1 elevator_filter_old4_V1
throttle_filter_old4_V1
ailerons_filter_old4_V1 accel_x_filter_old4_V1 accel_y_filter_old4_V1
accel_z_filter_old4_V1
elevator_filter_old5_V1 throttle_filter_old5_V1 ailerons_filter_old5_V1
accel_x_filter_old5_V1
accel_y_filter_old5_V1 accel_z_filter_old5_V1 elevator_filter_old6_V1
throttle_filter_old6_V1
ailerons_filter_old6_V1 accel_x_filter_old6_V1 accel_y_filter_old6_V1
accel_z_filter_old6_V1
elevator_filter_old7_V1 throttle_filter_old7_V1 ailerons_filter_old7_V1
accel_x_filter_old7_V1
accel_y_filter_old7_V1 accel_z_filter_old7_V1 elevator_filter_old8_V1
throttle_filter_old8_V1
```

```
ailerons_filter_old8_V1 accel_x_filter_old8_V1 accel_y_filter_old8_V1
accel_z_filter_old8_V1
elevator_filter_old9_V1 throttle_filter_old9_V1 ailerons_filter_old9_V1
accel_x_filter_old9_V1
accel_y_filter_old9_V1 accel_z_filter_old9_V1 elevator_filter_old10_V1
throttle_filter_old10_V1
ailerons_filter_old10_V1 accel_x_filter_old10_V1 accel_y_filter_old10_V1
accel_z_filter_old10_V1 bias ];
526 u3 = [ elevator_filter_V3 throttle_filter_V3 ailerons_filter_V3 accel_x_filter_V3
accel_y_filter_V3
accel_z_filter_V3 elevator_filter_old1_V3 throttle_filter_old1_V3 ailerons_filter_old1_V3
accel_x_filter_old1_V3
accel_y_filter_old1_V3 accel_z_filter_old1_V3 elevator_filter_old2_V3
throttle_filter_old2_V3
ailerons_filter_old2_V3 accel_x_filter_old2_V3 accel_y_filter_old2_V3
accel_z_filter_old2_V3
elevator_filter_old3_V3 throttle_filter_old3_V3 ailerons_filter_old3_V3
accel_x_filter_old3_V3
accel_y_filter_old3_V3 accel_z_filter_old3_V3 elevator_filter_old4_V3
throttle_filter_old4_V3
ailerons_filter_old4_V3 accel_x_filter_old4_V3 accel_y_filter_old4_V3
accel_z_filter_old4_V3
elevator_filter_old5_V3 throttle_filter_old5_V3 ailerons_filter_old5_V3
accel_x_filter_old5_V3
accel_y_filter_old5_V3 accel_z_filter_old5_V3 elevator_filter_old6_V3
throttle_filter_old6_V3
ailerons_filter_old6_V3 accel_x_filter_old6_V3 accel_y_filter_old6_V3
accel_z_filter_old6_V3
elevator_filter_old7_V3 throttle_filter_old7_V3 ailerons_filter_old7_V3
accel_x_filter_old7_V3
accel_y_filter_old7_V3 accel_z_filter_old7_V3 elevator_filter_old8_V3
throttle_filter_old8_V3
ailerons_filter_old8_V3 accel_x_filter_old8_V3 accel_y_filter_old8_V3
accel_z_filter_old8_V3
elevator_filter_old9_V3 throttle_filter_old9_V3 ailerons_filter_old9_V3
accel_x_filter_old9_V3
accel_y_filter_old9_V3 accel_z_filter_old9_V3 elevator_filter_old10_V3
throttle_filter_old10_V3
ailerons_filter_old10_V3 accel_x_filter_old10_V3 accel_y_filter_old10_V3
accel_z_filter_old10_V3 bias ];
527
528 % Variables de salida
529 z1 = [ pitch_angle_filter_V1 pitch_rate_filter_V1 roll_angle_filter_V1
roll_rate_filter_V1
velocity_filter_V1 altitude_filter_V1 ];
530 z3 = [ pitch_angle_filter_V3 pitch_rate_filter_V3 roll_angle_filter_V3
roll_rate_filter_V3
velocity_filter_V3 altitude_filter_V3 ];
531
532 nzz = 6;
```

```
533 nu1 = length(u1);
534 nu3 = length(u3);
535 ndata = nu1;
536 ndata = nu3;
537 dataoutesc1 = z1;
538 dataoutesc3 = z3;
539
540 % Cantidad de neuronas en las capas de entrada, intermedias y salida
541 ne = 73;
542 nm = 100;
543 np = 100;
544 ns = 6;
545
546 % Coeficientes de inicialización de los pesos de las capas intermedias ur,
547 % v, w, asimismo el centro y pendiente de la ecuación de entrenamiento
548 % sigmoidea 2.
549 ur = 0.5*randn(ne,nm);
550 v = 0.5*randn(nm,np);
551 w = 0.5*randn(np,ns);
552 c1 = zeros(nm,1);
553 a1 = 5*ones(nm,1);
554 c2 = zeros(np,1);
555 a2 = 5*ones(np,1);
556
557 % Se carga la data previamente entrenada para afinar el entrenamiento.
558 load red
559
560 % Se introducen los parametros de aprendizaje.
561 eta = input('Introduce learning rate [v w]: ');
562 etac = input('Introduce learning rate [c: sigmoid center]: ');
563 etaa = input('Introduce learning rate [a: sigmoid slope]: ');
564 beta = input('Introducir momento beta: ');
565
566 % Se introduice cuanto es el error máximo que se aceprta para este
567 % entrenamiento.
568 errormax = input('Introduce maximum value of error function (percentage %) : ' );
569 errormax = errormax/100;
570 % Se introduce el número de iteraciones que hará el entrenamiento antes de
571 % deternerse e indicar un valor de predicción y error.
572 contmax = input('Introduce number of iteration steps: ');
573
574 % Se prepara la data incial
575 outsum2 = sum(dataoutesc1.^2) + sum(dataoutesc3.^2);
576 outsum2 = outsum2';
577 outsum2total = sum(outsum2);
578 cont = 1;
579 erreltotal = 1;
580 dJdw_t_old = 0;
581 dJdv_t_old = 0;
582 dJdu_t_old = 0;
```

```
583
584 while( (erreltotal > errormax) & (cont < contmax) )
585 ersum2 = zeros(ns,1);
586
587 dy1dw_t = zeros(np,ns);
588 dy2dw_t = zeros(np,ns);
589 dy3dw_t = zeros(np,ns);
590 dy4dw_t = zeros(np,ns);
591 dy5dw_t = zeros(np,ns);
592 dy6dw_t = zeros(np,ns);
593
594 dy1dv_t = zeros(nm,np);
595 dy2dv_t = zeros(nm,np);
596 dy3dv_t = zeros(nm,np);
597 dy4dv_t = zeros(nm,np);
598 dy5dv_t = zeros(nm,np);
599 dy6dv_t = zeros(nm,np);
600
601 dy1du_t = zeros(ne,nm);
602 dy2du_t = zeros(ne,nm);
603 dy3du_t = zeros(ne,nm);
604 dy4du_t = zeros(ne,nm);
605 dy5du_t = zeros(ne,nm);
606 dy6du_t = zeros(ne,nm);
607
608 dJdw_t = zeros(np,ns);
609 dJdv_t = zeros(nm,np);
610 dJdu_t = zeros(ne,nm);
611
612 dJdw_t = zeros(np,ns);
613 dJdv_t = zeros(nm,np);
614 dJdu_t = zeros(ne,nm);
615
616 % Se realiza el primer entrenamiento
617
618 x = dataoutesc1(1,:); % Estado Inicial
619 x = x';
620 for k = 1:ndata-1
621 in_red = [ x
622 (u1(k,:))' ];
623 m = ur'*in_red;
624 n = 2.0./(1 + exp(-(m-c1)./a1)) - 1; % Sigmoidea 2
625 p = v'*n;
626 q = 2.0./(1 + exp(-(p-c2)./a2)) - 1; % Sigmoidea 2
627
628 out_red = w'*q;
629 outputesc(k,:) = out_red';
630 dndm = diag((1 - n.*n)./(2*a1)); % derivada de la Sigmoidea 2
631 dqdp = diag((1 - q.*q)./(2*a2)); % derivada de la Sigmoidea 2
632
```

```
633 dy1dw_s = [ q zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1)];
634 dy2dw_s = [ zeros(np,1) q zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1)];
635 dy3dw_s = [ zeros(np,1) zeros(np,1) q zeros(np,1) zeros(np,1) zeros(np,1)];
636 dy4dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) q zeros(np,1) zeros(np,1)];
637 dy5dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q zeros(np,1)];
638 dy6dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q ];
639
640 dy1dv_s = n*w(:,1)'*dqdp;
641 dy2dv_s = n*w(:,2)'*dqdp;
642 dy3dv_s = n*w(:,3)'*dqdp;
643 dy4dv_s = n*w(:,4)'*dqdp;
644 dy5dv_s = n*w(:,5)'*dqdp;
645 dy6dv_s = n*w(:,6)'*dqdp;
646
647 dy1du_s = in_red*w(:,1)'*dqdp*v'*dndm;
648 dy2du_s = in_red*w(:,2)'*dqdp*v'*dndm;
649 dy3du_s = in_red*w(:,3)'*dqdp*v'*dndm;
650 dy4du_s = in_red*w(:,4)'*dqdp*v'*dndm;
651 dy5du_s = in_red*w(:,5)'*dqdp*v'*dndm;
652 dy6du_s = in_red*w(:,6)'*dqdp*v'*dndm;
653
654 jacob = w'*dqdp*v'*dndm*(ur(1:nzz,:))';
655 dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t +
jacob(1,3).*dy3dw_t + jacob(1,4).
*dy4dw_t + jacob(1,5).*dy5dw_t + jacob(1,6).*dy6dw_t ;
656 dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t +
jacob(2,3).*dy3dw_t + jacob(2,4).
*dy4dw_t + jacob(2,5).*dy5dw_t + jacob(2,6).*dy6dw_t ;
657 dy3dw_t = dy3dw_s + jacob(3,1).*dy1dw_t + jacob(3,2).*dy2dw_t +
jacob(3,3).*dy3dw_t + jacob(3,4).
*dy4dw_t + jacob(3,5).*dy5dw_t + jacob(3,6).*dy6dw_t ;
658 dy4dw_t = dy4dw_s + jacob(4,1).*dy1dw_t + jacob(4,2).*dy2dw_t +
jacob(4,3).*dy3dw_t + jacob(4,4).
*dy4dw_t + jacob(4,5).*dy5dw_t + jacob(4,6).*dy6dw_t ;
659 dy5dw_t = dy5dw_s + jacob(5,1).*dy1dw_t + jacob(5,2).*dy2dw_t +
jacob(5,3).*dy3dw_t + jacob(5,4).
*dy4dw_t + jacob(5,5).*dy5dw_t + jacob(5,6).*dy6dw_t ;
660 dy6dw_t = dy6dw_s + jacob(6,1).*dy1dw_t + jacob(6,2).*dy2dw_t +
jacob(6,3).*dy3dw_t + jacob(6,4).
*dy4dw_t + jacob(6,5).*dy5dw_t + jacob(6,6).*dy6dw_t ;
661
662 dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t + jacob(1,3).*dy3dv_t
+ jacob(1,4).
*dy4dv_t + jacob(1,5).*dy5dv_t + jacob(1,6).*dy6dv_t ;
663 dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t + jacob(2,3).*dy3dv_t
+ jacob(2,4).
*dy4dv_t + jacob(2,5).*dy5dv_t + jacob(2,6).*dy6dv_t ;
664 dy3dv_t = dy3dv_s + jacob(3,1).*dy1dv_t + jacob(3,2).*dy2dv_t + jacob(3,3).*dy3dv_t
+ jacob(3,4).
*dy4dv_t + jacob(3,5).*dy5dv_t + jacob(3,6).*dy6dv_t ;
```

```
665 dy4dv_t = dy4dv_s + jacob(4,1).*dy1dv_t + jacob(4,2).*dy2dv_t + jacob(4,3).*dy3dv_t
+ jacob(4,4).
*dy4dv_t + jacob(4,5).*dy5dv_t + jacob(4,6).*dy6dv_t ;
666 dy5dv_t = dy5dv_s + jacob(5,1).*dy1dv_t + jacob(5,2).*dy2dv_t + jacob(5,3).*dy3dv_t
+ jacob(5,4).
*dy4dv_t + jacob(5,5).*dy5dv_t + jacob(5,6).*dy6dv_t ;
667 dy6dv_t = dy6dv_s + jacob(6,1).*dy1dv_t + jacob(6,2).*dy2dv_t + jacob(6,3).*dy3dv_t
+ jacob(6,4).
*dy4dv_t + jacob(6,5).*dy5dv_t + jacob(6,6).*dy6dv_t ;
668
669 dy1du_t = dy1du_s + jacob(1,1).*dy1du_t + jacob(1,2).*dy2du_t + jacob(1,3).*dy3du_t
+ jacob(1,4).
*dy4du_t + jacob(1,5).*dy5du_t + jacob(1,6).*dy6du_t ;
670 dy2du_t = dy2du_s + jacob(2,1).*dy1du_t + jacob(2,2).*dy2du_t + jacob(2,3).*dy3du_t
+ jacob(2,4).
*dy4du_t + jacob(2,5).*dy5du_t + jacob(2,6).*dy6du_t ;
671 dy3du_t = dy3du_s + jacob(3,1).*dy1du_t + jacob(3,2).*dy2du_t + jacob(3,3).*dy3du_t
+ jacob(3,4).
*dy4du_t + jacob(3,5).*dy5du_t + jacob(3,6).*dy6du_t ;
672 dy4du_t = dy4du_s + jacob(4,1).*dy1du_t + jacob(4,2).*dy2du_t + jacob(4,3).*dy3du_t
+ jacob(4,4).
*dy4du_t + jacob(4,5).*dy5du_t + jacob(4,6).*dy6du_t ;
673 dy5du_t = dy5du_s + jacob(5,1).*dy1du_t + jacob(5,2).*dy2du_t + jacob(5,3).*dy3du_t
+ jacob(5,4).
*dy4du_t + jacob(5,5).*dy5du_t + jacob(5,6).*dy6du_t ;
674 dy6du_t = dy6du_s + jacob(6,1).*dy1du_t + jacob(6,2).*dy2du_t + jacob(6,3).*dy3du_t
+ jacob(6,4).
*dy4du_t + jacob(6,5).*dy5du_t + jacob(6,6).*dy6du_t ;
675
676 out_des = dataoutesc1(k+1,:);
677 out_des = out_des';
678 er = (out_red - out_des);
679 erJ = (out_red - out_des);
680 qq1 = 1; qq2 = 1; qq3 = 1; qq4 = 1; qq5 = 1; qq6 = 1;
681 dJdw_t = dJdw_t + qq1*erJ(1,1).*dy1dw_t + qq2*erJ(2,1).*dy2dw_t +
qq3*erJ(3,1).*dy3dw_t + qq4*erJ
(4,1).*dy4dw_t + qq5*erJ(5,1).*dy5dw_t + qq6*erJ(6,1).*dy6dw_t ;
682 dJdv_t = dJdv_t + qq1*erJ(1,1).*dy1dv_t + qq2*erJ(2,1).*dy2dv_t +
qq3*erJ(3,1).*dy3dv_t + qq4*erJ
(4,1).*dy4dv_t + qq5*erJ(5,1).*dy5dv_t + qq6*erJ(6,1).*dy6dv_t ;
683 dJdu_t = dJdu_t + qq1*erJ(1,1).*dy1du_t + qq2*erJ(2,1).*dy2du_t +
qq3*erJ(3,1).*dy3du_t + qq4*erJ
(4,1).*dy4du_t + qq5*erJ(5,1).*dy5du_t + qq6*erJ(6,1).*dy6du_t ;
684 ersum2 = ersum2 + er.^2;
685 x = out_red; % output turns to be input for the next step
686 end
687
688 % Se realiza el segundo entrenamiento
689
690 dy1dw_t = zeros(np,ns);
```

```
691 dy2dw_t = zeros(np,ns);
692 dy3dw_t = zeros(np,ns);
693 dy4dw_t = zeros(np,ns);
694 dy5dw_t = zeros(np,ns);
695 dy6dw_t = zeros(np,ns);
696
697 dy1dv_t = zeros(nm,np);
698 dy2dv_t = zeros(nm,np);
699 dy3dv_t = zeros(nm,np);
700 dy4dv_t = zeros(nm,np);
701 dy5dv_t = zeros(nm,np);
702 dy6dv_t = zeros(nm,np);
703
704 dy1du_t = zeros(ne,nm);
705 dy2du_t = zeros(ne,nm);
706 dy3du_t = zeros(ne,nm);
707 dy4du_t = zeros(ne,nm);
708 dy5du_t = zeros(ne,nm);
709 dy6du_t = zeros(ne,nm);
710
711 x = dataoutesc3(1,:); % Estado inicial
712 x = x';
713 for k = 1:ndata-1
714 in_red = [ x
715 (u3(k,:))' ];
716 m = ur'*in_red;
717 n = 2.0./(1 + exp(-(m-c1)./a1)) - 1; % Sigmoidea 2
718 p = v'*n;
719 q = 2.0./(1 + exp(-(p-c2)./a2)) - 1; % Sigmoidea 2
720
721 out_red = w'*q;
722 outputesc(k,:) = out_red';
723 dndm = diag((1 - n.*n)./(2*a1)); % Sigmoidea 2
724 dqdp = diag((1 - q.*q)./(2*a2)); % Sigmoidea 2
725
726 dy1dw_s = [ q zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) ];
727 dy2dw_s = [ zeros(np,1) q zeros(np,1) zeros(np,1) zeros(np,1) ];
728 dy3dw_s = [ zeros(np,1) zeros(np,1) q zeros(np,1) zeros(np,1) ];
729 dy4dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) q zeros(np,1) ];
730 dy5dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q ];
731 dy6dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q ];
732
733 dy1dv_s = n*w(:,1)'*dqdp;
734 dy2dv_s = n*w(:,2)'*dqdp;
735 dy3dv_s = n*w(:,3)'*dqdp;
736 dy4dv_s = n*w(:,4)'*dqdp;
737 dy5dv_s = n*w(:,5)'*dqdp;
738 dy6dv_s = n*w(:,6)'*dqdp;
739
740 dy1du_s = in_red*w(:,1)'*dqdp*v'*dndm;
```

```
741 dy2du_s = in_red*w(:,2)'*dqdp*v'*dndm;
742 dy3du_s = in_red*w(:,3)'*dqdp*v'*dndm;
743 dy4du_s = in_red*w(:,4)'*dqdp*v'*dndm;
744 dy5du_s = in_red*w(:,5)'*dqdp*v'*dndm;
745 dy6du_s = in_red*w(:,6)'*dqdp*v'*dndm;
746
747 jacob = w'*dqdp*v'*dndm*(ur(1:nzz,:))';
748 dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t +
jacob(1,3).*dy3dw_t + jacob(1,4).
*dy4dw_t + jacob(1,5).*dy5dw_t + jacob(1,6).*dy6dw_t ;
749 dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t +
jacob(2,3).*dy3dw_t + jacob(2,4).
*dy4dw_t + jacob(2,5).*dy5dw_t + jacob(2,6).*dy6dw_t ;
750 dy3dw_t = dy3dw_s + jacob(3,1).*dy1dw_t + jacob(3,2).*dy2dw_t +
jacob(3,3).*dy3dw_t + jacob(3,4).
*dy4dw_t + jacob(3,5).*dy5dw_t + jacob(3,6).*dy6dw_t ;
751 dy4dw_t = dy4dw_s + jacob(4,1).*dy1dw_t + jacob(4,2).*dy2dw_t +
jacob(4,3).*dy3dw_t + jacob(4,4).
*dy4dw_t + jacob(4,5).*dy5dw_t + jacob(4,6).*dy6dw_t ;
752 dy5dw_t = dy5dw_s + jacob(5,1).*dy1dw_t + jacob(5,2).*dy2dw_t +
jacob(5,3).*dy3dw_t + jacob(5,4).
*dy4dw_t + jacob(5,5).*dy5dw_t + jacob(5,6).*dy6dw_t ;
753 dy6dw_t = dy6dw_s + jacob(6,1).*dy1dw_t + jacob(6,2).*dy2dw_t +
jacob(6,3).*dy3dw_t + jacob(6,4).
*dy4dw_t + jacob(6,5).*dy5dw_t + jacob(6,6).*dy6dw_t ;
754
755 dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t + jacob(1,3).*dy3dv_t
+ jacob(1,4).
*dy4dv_t + jacob(1,5).*dy5dv_t + jacob(1,6).*dy6dv_t ;
756 dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t + jacob(2,3).*dy3dv_t
+ jacob(2,4).
*dy4dv_t + jacob(2,5).*dy5dv_t + jacob(2,6).*dy6dv_t ;
757 dy3dv_t = dy3dv_s + jacob(3,1).*dy1dv_t + jacob(3,2).*dy2dv_t + jacob(3,3).*dy3dv_t
+ jacob(3,4).
*dy4dv_t + jacob(3,5).*dy5dv_t + jacob(3,6).*dy6dv_t ;
758 dy4dv_t = dy4dv_s + jacob(4,1).*dy1dv_t + jacob(4,2).*dy2dv_t + jacob(4,3).*dy3dv_t
+ jacob(4,4).
*dy4dv_t + jacob(4,5).*dy5dv_t + jacob(4,6).*dy6dv_t ;
759 dy5dv_t = dy5dv_s + jacob(5,1).*dy1dv_t + jacob(5,2).*dy2dv_t + jacob(5,3).*dy3dv_t
+ jacob(5,4).
*dy4dv_t + jacob(5,5).*dy5dv_t + jacob(5,6).*dy6dv_t ;
760 dy6dv_t = dy6dv_s + jacob(6,1).*dy1dv_t + jacob(6,2).*dy2dv_t + jacob(6,3).*dy3dv_t
+ jacob(6,4).
*dy4dv_t + jacob(6,5).*dy5dv_t + jacob(6,6).*dy6dv_t ;
761
762 dy1du_t = dy1du_s + jacob(1,1).*dy1du_t + jacob(1,2).*dy2du_t + jacob(1,3).*dy3du_t
+ jacob(1,4).
*dy4du_t + jacob(1,5).*dy5du_t + jacob(1,6).*dy6du_t ;
763 dy2du_t = dy2du_s + jacob(2,1).*dy1du_t + jacob(2,2).*dy2du_t + jacob(2,3).*dy3du_t
+ jacob(2,4).
```

```
*dy4du_t + jacob(2,5).*dy5du_t + jacob(2,6).*dy6du_t ;
764 dy3du_t = dy3du_s + jacob(3,1).*dy1du_t + jacob(3,2).*dy2du_t + jacob(3,3).*dy3du_t
+ jacob(3,4).
*dy4du_t + jacob(3,5).*dy5du_t + jacob(3,6).*dy6du_t ;
765 dy4du_t = dy4du_s + jacob(4,1).*dy1du_t + jacob(4,2).*dy2du_t + jacob(4,3).*dy3du_t
+ jacob(4,4).
*dy4du_t + jacob(4,5).*dy5du_t + jacob(4,6).*dy6du_t ;
766 dy5du_t = dy5du_s + jacob(5,1).*dy1du_t + jacob(5,2).*dy2du_t + jacob(5,3).*dy3du_t
+ jacob(5,4).
*dy4du_t + jacob(5,5).*dy5du_t + jacob(5,6).*dy6du_t ;
767 dy6du_t = dy6du_s + jacob(6,1).*dy1du_t + jacob(6,2).*dy2du_t + jacob(6,3).*dy3du_t
+ jacob(6,4).
*dy4du_t + jacob(6,5).*dy5du_t + jacob(6,6).*dy6du_t ;
768
769 out_des = dataoutesc3(k+1,:);
770 out_des = out_des';
771 er = (out_red - out_des);
772 erJ = (out_red - out_des);
773 qq1 = 1; qq2 = 1; qq3 = 1; qq4 = 1; qq5 = 1; qq6 = 1;
774 dJdw_t = dJdw_t + qq1*erJ(1,1).*dy1dw_t + qq2*erJ(2,1).*dy2dw_t +
qq3*erJ(3,1).*dy3dw_t + qq4*erJ
 (4,1).*dy4dw_t + qq5*erJ(5,1).*dy5dw_t + qq6*erJ(6,1).*dy6dw_t ;
775 dJdv_t = dJdv_t + qq1*erJ(1,1).*dy1dv_t + qq2*erJ(2,1).*dy2dv_t +
qq3*erJ(3,1).*dy3dv_t + qq4*erJ
(4,1).*dy4dv_t + qq5*erJ(5,1).*dy5dv_t + qq6*erJ(6,1).*dy6dv_t ;
776 dJdu_t = dJdu_t + qq1*erJ(1,1).*dy1du_t + qq2*erJ(2,1).*dy2du_t +
qq3*erJ(3,1).*dy3du_t + qq4*erJ
(4,1).*dy4du_t + qq5*erJ(5,1).*dy5du_t + qq6*erJ(6,1).*dy6du_t ;
777 ersum2 = ersum2 + er.^2;
778 x = out_red;
779 end
780
781 % Calculo de Error Total
782
783 dJdw_t = dJdw_t/(2*ndata);
784 dJdv_t = dJdv_t/(2*ndata);
785 dJdu_t = dJdu_t/(2*ndata);
786
787 ersum2total = sum(ersum2);
788 cont = cont + 1;
789 if ( rem(cont,1) == 0 )
790 errorrel(cont/1,:) = sqrt(ersum2'./outsum2');
791 errorreltotal(cont/1,1) = sqrt(ersum2total/outsum2total);
792 erreltotal = errorreltotal(cont/1,1);
793 cont;
794 erreltotal
795 end
796
797 w = w - eta*dJdw_t + beta*dJdw_t_old;
798 v = v - eta*dJdv_t + beta*dJdv_t_old;
```

```
799 ur = ur - eta*dJdu_t + beta*dJdu_t_old;
800 dJdw_t_old = dJdw_t;
801 dJdv_t_old = dJdv_t;
802 dJdu_t_old = dJdu_t;
803 end
804
805 % Graficar las figuras luego de realizar la predicción, mostrando el error
806 % de entrenamiento.
807 figure(12);
808 plot(errorreltotal*100);
809 title('Total Error Function');
810
811 figure(13);
812 plot(errorrel*100);
813 title('Error Function per Output');
814 legend('Pitch Angle','Pitch rate','Roll Angle','Roll Rate','Velocity','Altitude');
815
816 figure(14);
817 plot(z3(:,1),'-g');
818 hold on;
819 plot(outputesc(:,1),'-b');
820 title('Pitch Angle');
821 legend('Rojo:Deseada', 'Verde:Deseada' ,'Azul:Salida Red');
822
823 figure(15);
824 plot(z3(:,2),'-g');
825 hold on;
826 plot(outputesc(:,2),'-b');
827 title('Pitch Rate');
828 legend('Rojo:Deseada', 'Verde:Deseada' ,'Azul:Salida Red');
829
830 figure(16);
831 plot(z3(:,3),'-g');
832 hold on;
833 plot(outputesc(:,3),'-b');
834 title('Roll Angle');
835 legend('Rojo:Deseada', 'Verde:Deseada' ,'Azul:Salida Red');
836
837 figure(17);
838 plot(z3(:,4),'-g');
839 hold on;
840 plot(outputesc(:,4),'-b');
841 title('Roll Rate');
842 legend('Rojo:Deseada', 'Verde:Deseada' ,'Azul:Salida Red');
843
844 figure(18);
845 plot(z3(:,5),'-g');
846 hold on;
847 plot(outputesc(:,5),'-b');
848 title('Velocity');
```

```
849 legend('Rojo:Deseada', 'Verde:Deseada' ,'Azul:Salida Red');
850
851 figure(19);
852 plot(z3(:,6),'-r');
853 hold on;
854 plot(outputesc(:,6),'-b');
855 title('Altitude');
856 legend('Rojo:Deseada','Azul:Salida Red');
857
858 figure20);
859 title('Input Signal');
860 plot(u1,'-b');
861 hold on
862 plot(u3,'-g');
863 title('Input Signal u');
864 xlabel('Time [sec]');
865
866 % Saving: number of input, hidden and output neurons, coefficients v and
867 % w, center c, and slope c
868
869 % save red ne nm np ns ur v w a1 c1 a2 c2 bias
```

ANEXO 03

```
%% Filtrado de la Información %%

clear all
close all
clc

% Se carga la data del vuelo que se va a utilizar.
data_flight = load('20160915_NEX7_V1_ICA_Z12.BIN-1091899.mat');
% Este valor "i" es utilizado para que el programa sepa cada cuantos
% espacios debe de recoger un valor de información, en este caso cada cinco
% espacios se utiliza un valor de información.
i = 5;
% Data específica del acelerador. Localizada en el campo IMU de la data de
% vuelo.
accel_x = data_flight.IMU([51787:i:76737], [2 6]);
nt = 2000;
accel_x = accel_x(1:nt,2);
L = length(accel_x);

rng default;
Fs = 10;
figure(1);
plot(accel_x)

% serie de fourier para encontrar frecuencia de corte
Z = fft(accel_x);
P2 = abs(Z/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
figure(2);
plot(f,P1)

% Se introduce el valor de la frecuencia de corte, según lo obtenido de la
fc = input('Introduce cut-off frequency: ');

% Frecuencia normalizada
Wn = (2/Fs)*fc;

% Coeficiente de filtrado
b = fir1(20, Wn, 'low', kaiser(21, 3));

Hd =
designfilt('lowpassfir','FilterOrder',20,'CutoffFrequency',fc,'DesignMethod','window','Windo
w',{@kaiser,
3},'SampleRate',Fs);
yout = filter(Hd, accel_x);
xin = accel_x;
```

```
figure(3);
plot(xin)
hold on
plot(yout, 'LineWidth',1)
legend('input','output');

fvtool(Hd, 'Analysis','grpdelay')

% Aplicación del filtro "zero - phase filter"
accel_x_filter_V1 = filtfilt(Hd, accel_x);

% Graficar la figura final con el filtrado aplicado
figure(4);
plot(xin)
hold on
plot(accel_x_filter_V1, 'LineWidth',1)
legend('input','output');

save accel_x_filter_V1
```

ANEXO 04

```
%%Red Neuronal Artificial%%

% Validación%

clear all
close all
clc

load pitch_angle_filter_V5
load velocity_filter_V5
load elevator_filter_V5
load throttle_filter_V5
load roll_angle_filter_V5
load ailerons_filter_V5
load accel_x_filter_V5
load accel_y_filter_V5
load accel_z_filter_V5
load altitude_filter_V5

nt = 2000;

figure(1);
pitch_angle_filter_V5 = 0.049*pitch_angle_filter_V5(1:nt,1);
plot(pitch_angle_filter_V5, 'b');
grid;
title('pitch angle')

figure(2);
pitch_rate_filter_V5 = diff(pitch_angle_filter_V5)/0.2;
pitch_rate_filter_V5 = [0; pitch_rate_filter_V5];
pitch_rate_filter_V5 = 2.8*pitch_rate_filter_V5(1:nt,1);
plot(pitch_rate_filter_V5, 'b');
grid;
title('pitch rate')

figure(3);
velocity_filter_V5 = 0.043*(velocity_filter_V5(1:nt,1));
plot(velocity_filter_V5, 'b');
grid;
title('velocity')

figure(4);
elevator_filter_V5 = 0.05*(elevator_filter_V5(1:nt,1));
plot(elevator_filter_V5, 'b');
grid;
title('elevator')

figure(5);
```

```
throttle_filter_V5 = 0.018*(throttle_filter_V5(1:nt,1));
plot(throttle_filter_V5, 'b');
grid;
title('throttle')

figure(6);
roll_angle_filter_V5 = 0.014*roll_angle_filter_V5(1:nt,1);
plot(roll_angle_filter_V5, 'b');
grid;
title('roll angle')

figure(7);
roll_rate_filter_V5 = diff(roll_angle_filter_V5)/0.2;
roll_rate_filter_V5 = [0; roll_rate_filter_V5];
roll_rate_filter_V5 = 2.8*roll_rate_filter_V5(1:nt,1);
plot(roll_rate_filter_V5, 'b');
grid;
title('roll rate')

figure(8);
ailerons_filter_V5 = 0.047*ailerons_filter_V5(1:nt,1);
plot(ailerons_filter_V5, 'b');
grid;
title('ailerons')

figure(9);
accel_x_filter_V5 = 0.18*accel_x_filter_V5(1:nt,1);
plot(accel_x_filter_V5, 'b');
grid;
title('accel X')

figure(10);
accel_y_filter_V5 = 1.6*accel_y_filter_V5(1:nt,1);
plot(accel_y_filter_V5, 'b');
grid;
title('accel Y')

figure(11);
accel_z_filter_V5 = 0.04*accel_z_filter_V5(1:nt,1);
plot(accel_z_filter_V5, 'b');
grid;
title('accel Z')

figure(12);
altitude_filter_V5 = 0.00382*altitude_filter_V5(1:nt,1);
plot(altitude_filter_V5, 'b');
grid;
title('altitude')

elevator_filter_old1_V5(1,1) = 0;
```

```
99 elevator_filter_old1_V5(2:nt,1) = elevator_filter_V5(1:nt-1,1);
100
101 throttle_filter_old1_V5(1,1) = 0;
102 throttle_filter_old1_V5(2:nt,1) = throttle_filter_V5(1:nt-1,1);
103
104 ailerons_filter_old1_V5(1,1) = 0;
105 ailerons_filter_old1_V5(2:nt,1) = ailerons_filter_V5(1:nt-1,1);
106
107 accel_x_filter_old1_V5(1,1) = 0;
108 accel_x_filter_old1_V5(2:nt,1) = accel_x_filter_V5(1:nt-1,1);
109
110 accel_y_filter_old1_V5(1,1) = 0;
111 accel_y_filter_old1_V5(2:nt,1) = accel_y_filter_V5(1:nt-1,1);
112
113 accel_z_filter_old1_V5(1,1) = 0;
114 accel_z_filter_old1_V5(2:nt,1) = accel_z_filter_V5(1:nt-1,1);
115
116 elevator_filter_old2_V5(1,1) = 0;
117 elevator_filter_old2_V5(2:nt,1) = elevator_filter_old1_V5(1:nt-1,1);
118
119 throttle_filter_old2_V5(1,1) = 0;
120 throttle_filter_old2_V5(2:nt,1) = throttle_filter_old1_V5(1:nt-1,1);
121
122 ailerons_filter_old2_V5(1,1) = 0;
123 ailerons_filter_old2_V5(2:nt,1) = ailerons_filter_old1_V5(1:nt-1,1);
124
125 accel_x_filter_old2_V5(1,1) = 0;
126 accel_x_filter_old2_V5(2:nt,1) = accel_x_filter_old1_V5(1:nt-1,1);
127
128 accel_y_filter_old2_V5(1,1) = 0;
129 accel_y_filter_old2_V5(2:nt,1) = accel_y_filter_old1_V5(1:nt-1,1);
130
131 accel_z_filter_old2_V5(1,1) = 0;
132 accel_z_filter_old2_V5(2:nt,1) = accel_z_filter_old1_V5(1:nt-1,1);
133
134 elevator_filter_old3_V5(1,1) = 0;
135 elevator_filter_old3_V5(2:nt,1) = elevator_filter_old2_V5(1:nt-1,1);
136
137 throttle_filter_old3_V5(1,1) = 0;
138 throttle_filter_old3_V5(2:nt,1) = throttle_filter_old2_V5(1:nt-1,1);
139
140 ailerons_filter_old3_V5(1,1) = 0;
141 ailerons_filter_old3_V5(2:nt,1) = ailerons_filter_old2_V5(1:nt-1,1);
142
143 accel_x_filter_old3_V5(1,1) = 0;
144 accel_x_filter_old3_V5(2:nt,1) = accel_x_filter_old2_V5(1:nt-1,1);
145
146 accel_y_filter_old3_V5(1,1) = 0;
147 accel_y_filter_old3_V5(2:nt,1) = accel_y_filter_old2_V5(1:nt-1,1);
148
```

```
149 accel_z_filter_old3_V5(1,1) = 0;
150 accel_z_filter_old3_V5(2:nt,1) = accel_z_filter_old2_V5(1:nt-1,1);
151
152 elevator_filter_old4_V5(1,1) = 0;
153 elevator_filter_old4_V5(2:nt,1) = elevator_filter_old3_V5(1:nt-1,1);
154
155 throttle_filter_old4_V5(1,1) = 0;
156 throttle_filter_old4_V5(2:nt,1) = throttle_filter_old3_V5(1:nt-1,1);
157
158 ailerons_filter_old4_V5(1,1) = 0;
159 ailerons_filter_old4_V5(2:nt,1) = ailerons_filter_old3_V5(1:nt-1,1);
160
161 accel_x_filter_old4_V5(1,1) = 0;
162 accel_x_filter_old4_V5(2:nt,1) = accel_x_filter_old3_V5(1:nt-1,1);
163
164 accel_y_filter_old4_V5(1,1) = 0;
165 accel_y_filter_old4_V5(2:nt,1) = accel_y_filter_old3_V5(1:nt-1,1);
166
167 accel_z_filter_old4_V5(1,1) = 0;
168 accel_z_filter_old4_V5(2:nt,1) = accel_z_filter_old3_V5(1:nt-1,1);
169
170 elevator_filter_old5_V5(1,1) = 0;
171 elevator_filter_old5_V5(2:nt,1) = elevator_filter_old4_V5(1:nt-1,1);
172
173 throttle_filter_old5_V5(1,1) = 0;
174 throttle_filter_old5_V5(2:nt,1) = throttle_filter_old4_V5(1:nt-1,1);
175
176 ailerons_filter_old5_V5(1,1) = 0;
177 ailerons_filter_old5_V5(2:nt,1) = ailerons_filter_old4_V5(1:nt-1,1);
178
179 accel_x_filter_old5_V5(1,1) = 0;
180 accel_x_filter_old5_V5(2:nt,1) = accel_x_filter_old4_V5(1:nt-1,1);
181
182 accel_y_filter_old5_V5(1,1) = 0;
183 accel_y_filter_old5_V5(2:nt,1) = accel_y_filter_old4_V5(1:nt-1,1);
184
185 accel_z_filter_old5_V5(1,1) = 0;
186 accel_z_filter_old5_V5(2:nt,1) = accel_z_filter_old4_V5(1:nt-1,1);
187
188 elevator_filter_old6_V5(1,1) = 0;
189 elevator_filter_old6_V5(2:nt,1) = elevator_filter_old5_V5(1:nt-1,1);
190
191 throttle_filter_old6_V5(1,1) = 0;
192 throttle_filter_old6_V5(2:nt,1) = throttle_filter_old5_V5(1:nt-1,1);
193
194 ailerons_filter_old6_V5(1,1) = 0;
195 ailerons_filter_old6_V5(2:nt,1) = ailerons_filter_old5_V5(1:nt-1,1);
196
197 accel_x_filter_old6_V5(1,1) = 0;
198 accel_x_filter_old6_V5(2:nt,1) = accel_x_filter_old5_V5(1:nt-1,1);
```

```
199
200 accel_y_filter_old6_V5(1,1) = 0;
201 accel_y_filter_old6_V5(2:nt,1) = accel_y_filter_old5_V5(1:nt-1,1);
202
203 accel_z_filter_old6_V5(1,1) = 0;
204 accel_z_filter_old6_V5(2:nt,1) = accel_z_filter_old5_V5(1:nt-1,1);
205
206 elevator_filter_old7_V5(1,1) = 0;
207 elevator_filter_old7_V5(2:nt,1) = elevator_filter_old6_V5(1:nt-1,1);
208
209 throttle_filter_old7_V5(1,1) = 0;
210 throttle_filter_old7_V5(2:nt,1) = throttle_filter_old6_V5(1:nt-1,1);
211
212 ailerons_filter_old7_V5(1,1) = 0;
213 ailerons_filter_old7_V5(2:nt,1) = ailerons_filter_old6_V5(1:nt-1,1);
214
215 accel_x_filter_old7_V5(1,1) = 0;
216 accel_x_filter_old7_V5(2:nt,1) = accel_x_filter_old6_V5(1:nt-1,1);
217
218 accel_y_filter_old7_V5(1,1) = 0;
219 accel_y_filter_old7_V5(2:nt,1) = accel_y_filter_old6_V5(1:nt-1,1);
220
221 accel_z_filter_old7_V5(1,1) = 0;
222 accel_z_filter_old7_V5(2:nt,1) = accel_z_filter_old6_V5(1:nt-1,1);
223
224 elevator_filter_old8_V5(1,1) = 0;
225 elevator_filter_old8_V5(2:nt,1) = elevator_filter_old7_V5(1:nt-1,1);
226
227 throttle_filter_old8_V5(1,1) = 0;
228 throttle_filter_old8_V5(2:nt,1) = throttle_filter_old7_V5(1:nt-1,1);
229
230 ailerons_filter_old8_V5(1,1) = 0;
231 ailerons_filter_old8_V5(2:nt,1) = ailerons_filter_old7_V5(1:nt-1,1);
232
233 accel_x_filter_old8_V5(1,1) = 0;
234 accel_x_filter_old8_V5(2:nt,1) = accel_x_filter_old7_V5(1:nt-1,1);
235
236 accel_y_filter_old8_V5(1,1) = 0;
237 accel_y_filter_old8_V5(2:nt,1) = accel_y_filter_old7_V5(1:nt-1,1);
238
239 accel_z_filter_old8_V5(1,1) = 0;
240 accel_z_filter_old8_V5(2:nt,1) = accel_z_filter_old7_V5(1:nt-1,1);
241
242 elevator_filter_old9_V5(1,1) = 0;
243 elevator_filter_old9_V5(2:nt,1) = elevator_filter_old8_V5(1:nt-1,1);
244
245 throttle_filter_old9_V5(1,1) = 0;
246 throttle_filter_old9_V5(2:nt,1) = throttle_filter_old8_V5(1:nt-1,1);
247
248 ailerons_filter_old9_V5(1,1) = 0;
```

```
249 ailerons_filter_old9_V5(2:nt,1) = ailerons_filter_old8_V5(1:nt-1,1);
250
251 accel_x_filter_old9_V5(1,1) = 0;
252 accel_x_filter_old9_V5(2:nt,1) = accel_x_filter_old8_V5(1:nt-1,1);
253
254 accel_y_filter_old9_V5(1,1) = 0;
255 accel_y_filter_old9_V5(2:nt,1) = accel_y_filter_old8_V5(1:nt-1,1);
256
257 accel_z_filter_old9_V5(1,1) = 0;
258 accel_z_filter_old9_V5(2:nt,1) = accel_z_filter_old8_V5(1:nt-1,1);
259
260 elevator_filter_old10_V5(1,1) = 0;
261 elevator_filter_old10_V5(2:nt,1) = elevator_filter_old9_V5(1:nt-1,1);
262
263 throttle_filter_old10_V5(1,1) = 0;
264 throttle_filter_old10_V5(2:nt,1) = throttle_filter_old9_V5(1:nt-1,1);
265
266 ailerons_filter_old10_V5(1,1) = 0;
267 ailerons_filter_old10_V5(2:nt,1) = ailerons_filter_old9_V5(1:nt-1,1);
268
269 accel_x_filter_old10_V5(1,1) = 0;
270 accel_x_filter_old10_V5(2:nt,1) = accel_x_filter_old9_V5(1:nt-1,1);
271
272 accel_y_filter_old10_V5(1,1) = 0;
273 accel_y_filter_old10_V5(2:nt,1) = accel_y_filter_old9_V5(1:nt-1,1);
274
275 accel_z_filter_old10_V5(1,1) = 0;
276 accel_z_filter_old10_V5(2:nt,1) = accel_z_filter_old9_V5(1:nt-1,1);
277
278 load Red
279
280 u = [ elevator_filter_V5 throttle_filter_V5 ailerons_filter_V5 accel_x_filter_V5
accel_y_filter_V5
accel_z_filter_V5 elevator_filter_old1_V5 throttle_filter_old1_V5 ailerons_filter_old1_V5
accel_x_filter_old1_V5
accel_y_filter_old1_V5 accel_z_filter_old1_V5 elevator_filter_old2_V5
throttle_filter_old2_V5
ailerons_filter_old2_V5 accel_x_filter_old2_V5 accel_y_filter_old2_V5
accel_z_filter_old2_V5
elevator_filter_old3_V5 throttle_filter_old3_V5 ailerons_filter_old3_V5
accel_x_filter_old3_V5
accel_y_filter_old3_V5 accel_z_filter_old3_V5 elevator_filter_old4_V5
throttle_filter_old4_V5
ailerons_filter_old4_V5 accel_x_filter_old4_V5 accel_y_filter_old4_V5
accel_z_filter_old4_V5
elevator_filter_old5_V5 throttle_filter_old5_V5 ailerons_filter_old5_V5
accel_x_filter_old5_V5
accel_y_filter_old5_V5 accel_z_filter_old5_V5 elevator_filter_old6_V5
throttle_filter_old6_V5
```

```
ailerons_filter_old6_V5 accel_x_filter_old6_V5 accel_y_filter_old6_V5
accel_z_filter_old6_V5
elevator_filter_old7_V5 throttle_filter_old7_V5 ailerons_filter_old7_V5
accel_x_filter_old7_V5
accel_y_filter_old7_V5 accel_z_filter_old7_V5 elevator_filter_old8_V5
throttle_filter_old8_V5
ailerons_filter_old8_V5 accel_x_filter_old8_V5 accel_y_filter_old8_V5
accel_z_filter_old8_V5
elevator_filter_old9_V5 throttle_filter_old9_V5 ailerons_filter_old9_V5
accel_x_filter_old9_V5
accel_y_filter_old9_V5 accel_z_filter_old9_V5 elevator_filter_old10_V5
throttle_filter_old10_V5
ailerons_filter_old10_V5 accel_x_filter_old10_V5 accel_y_filter_old10_V5
accel_z_filter_old10_V5 bias ];
281 z = [ pitch_angle_filter_V5 pitch_rate_filter_V5 roll_angle_filter_V5 roll_rate_filter_V5
velocity_filter_V5
altitude_filter_V5];
282
283 nzz = 6;
284 nu = length(u);
285 ndata = nu;
286 dataoutesc = z;
287
288 % Introducing learning parameters
289 eta = input('Introduce learning rate [v w]: '); %disminuir el eta de entrenamiento
desacelera el
entrenamiento
290 etac = input('Introduce learning rate [c: sigmoid center]: ');
291 etaa = input('Introduce learning rate [a: sigmoid slope]: ');
292 beta = input('Introducir momento beta: '); %reducir el beta
293
294 errormax = input('Introduce maximum value of error function (percentage %) : ' );
295 errormax = errormax/100;
296 contmax = input('Introduce number of iteration steps: ');
297
298 outsum2 = sum(dataoutesc.^2);
299 outsum2 = outsum2';
300 outsum2total = sum(outsum2);
301 cont = 1;
302 erreltotal = 1;
303 dJdw_t_old = 0;
304 dJdv_t_old = 0;
305 dJdu_t_old = 0;
306
307 while( (erreltotal > errormax) & (cont < contmax) )
308 ersum2 = zeros(ns,1);
309 dJdw = 0;
310 dJdv = 0;
311 dJdu = 0;
312
```

```
dy1dw_t = zeros(np,ns);
dy2dw_t = zeros(np,ns);
dy3dw_t = zeros(np,ns);
dy4dw_t = zeros(np,ns);
dy5dw_t = zeros(np,ns);
dy6dw_t = zeros(np,ns);

dy1dv_t = zeros(nm,np);
dy2dv_t = zeros(nm,np);
dy3dv_t = zeros(nm,np);
dy4dv_t = zeros(nm,np);
dy5dv_t = zeros(nm,np);
dy6dv_t = zeros(nm,np);

dy1du_t = zeros(ne,nm);
dy2du_t = zeros(ne,nm);
dy3du_t = zeros(ne,nm);
dy4du_t = zeros(ne,nm);
dy5du_t = zeros(ne,nm);
dy6du_t = zeros(ne,nm);

dJdw_t = zeros(np,ns);
dJdv_t = zeros(nm,np);
dJdu_t = zeros(ne,nm);

x = dataoutesc(1,:); % Initial state
x = x';
for k = 1:ndata-1
in_red = [ x
(u(k,:))' ];
m = ur'*in_red;
n = 2.0./(1 + exp(-(m-c1)./a1)) - 1; % Sigmoidea 2
p = v'*n;
q = 2.0./(1 + exp(-(p-c2)./a2)) - 1; % Sigmoidea 2

out_red = w'*q;
outputesc(k,:) = out_red';
dndm = diag((1 - n.*n)./(2*a1)); % Sigmoidea 2
dqdp = diag((1 - q.*q)./(2*a2)); % Sigmoidea 2

dy1dw_s = [ q zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) ];
dy2dw_s = [ zeros(np,1) q zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) ];
dy3dw_s = [ zeros(np,1) zeros(np,1) q zeros(np,1) zeros(np,1) zeros(np,1) ];
dy4dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) q zeros(np,1) zeros(np,1) ];
dy5dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q zeros(np,1) ];
dy6dw_s = [ zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) zeros(np,1) q ];

dy1dv_s = n*w(:,1)'*dqdp;
dy2dv_s = n*w(:,2)'*dqdp;
dy3dv_s = n*w(:,3)'*dqdp;
```

```
363 dy4dv_s = n*w(:,4)'*dqdp;
364 dy5dv_s = n*w(:,5)'*dqdp;
365 dy6dv_s = n*w(:,6)'*dqdp;
366
367 dy1du_s = in_red*w(:,1)'*dqdp*v'*dndm;
368 dy2du_s = in_red*w(:,2)'*dqdp*v'*dndm;
369 dy3du_s = in_red*w(:,3)'*dqdp*v'*dndm;
370 dy4du_s = in_red*w(:,4)'*dqdp*v'*dndm;
371 dy5du_s = in_red*w(:,5)'*dqdp*v'*dndm;
372 dy6du_s = in_red*w(:,6)'*dqdp*v'*dndm;
373
374 jacob = w'*dqdp*v'*dndm*(ur(1:nzz,:))';
375 dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t +
jacob(1,3).*dy3dw_t + jacob(1,4).
*dy4dw_t + jacob(1,5).*dy5dw_t + jacob(1,6).*dy6dw_t ;
376 dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t +
jacob(2,3).*dy3dw_t + jacob(2,4).
*dy4dw_t + jacob(2,5).*dy5dw_t + jacob(2,6).*dy6dw_t ;
377 dy3dw_t = dy3dw_s + jacob(3,1).*dy1dw_t + jacob(3,2).*dy2dw_t +
jacob(3,3).*dy3dw_t + jacob(3,4).
*dy4dw_t + jacob(3,5).*dy5dw_t + jacob(3,6).*dy6dw_t ;
378 dy4dw_t = dy4dw_s + jacob(4,1).*dy1dw_t + jacob(4,2).*dy2dw_t +
jacob(4,3).*dy3dw_t + jacob(4,4).
*dy4dw_t + jacob(4,5).*dy5dw_t + jacob(4,6).*dy6dw_t ;
379 dy5dw_t = dy5dw_s + jacob(5,1).*dy1dw_t + jacob(5,2).*dy2dw_t +
jacob(5,3).*dy3dw_t + jacob(5,4).
*dy4dw_t + jacob(5,5).*dy5dw_t + jacob(5,6).*dy6dw_t ;
380 dy6dw_t = dy6dw_s + jacob(6,1).*dy1dw_t + jacob(6,2).*dy2dw_t +
jacob(6,3).*dy3dw_t + jacob(6,4).
*dy4dw_t + jacob(6,5).*dy5dw_t + jacob(6,6).*dy6dw_t ;
381
382 dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t + jacob(1,3).*dy3dv_t
+ jacob(1,4).
*dy4dv_t + jacob(1,5).*dy5dv_t + jacob(1,6).*dy6dv_t ;
383 dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t + jacob(2,3).*dy3dv_t
+ jacob(2,4).
*dy4dv_t + jacob(2,5).*dy5dv_t + jacob(2,6).*dy6dv_t ;
384 dy3dv_t = dy3dv_s + jacob(3,1).*dy1dv_t + jacob(3,2).*dy2dv_t + jacob(3,3).*dy3dv_t
+ jacob(3,4).
*dy4dv_t + jacob(3,5).*dy5dv_t + jacob(3,6).*dy6dv_t ;
385 dy4dv_t = dy4dv_s + jacob(4,1).*dy1dv_t + jacob(4,2).*dy2dv_t + jacob(4,3).*dy3dv_t
+ jacob(4,4).
*dy4dv_t + jacob(4,5).*dy5dv_t + jacob(4,6).*dy6dv_t ;
386 dy5dv_t = dy5dv_s + jacob(5,1).*dy1dv_t + jacob(5,2).*dy2dv_t + jacob(5,3).*dy3dv_t
+ jacob(5,4).
*dy4dv_t + jacob(5,5).*dy5dv_t + jacob(5,6).*dy6dv_t ;
387 dy6dv_t = dy6dv_s + jacob(6,1).*dy1dv_t + jacob(6,2).*dy2dv_t + jacob(6,3).*dy3dv_t
+ jacob(6,4).
*dy4dv_t + jacob(6,5).*dy5dv_t + jacob(6,6).*dy6dv_t ;
388
```

```
389 dy1du_t = dy1du_s + jacob(1,1).*dy1du_t + jacob(1,2).*dy2du_t + jacob(1,3).*dy3du_t
+ jacob(1,4).
*dy4du_t + jacob(1,5).*dy5du_t + jacob(1,6).*dy6du_t ;
390 dy2du_t = dy2du_s + jacob(2,1).*dy1du_t + jacob(2,2).*dy2du_t + jacob(2,3).*dy3du_t
+ jacob(2,4).
*dy4du_t + jacob(2,5).*dy5du_t + jacob(2,6).*dy6du_t ;
391 dy3du_t = dy3du_s + jacob(3,1).*dy1du_t + jacob(3,2).*dy2du_t + jacob(3,3).*dy3du_t
+ jacob(3,4).
*dy4du_t + jacob(3,5).*dy5du_t + jacob(3,6).*dy6du_t ;
392 dy4du_t = dy4du_s + jacob(4,1).*dy1du_t + jacob(4,2).*dy2du_t + jacob(4,3).*dy3du_t
+ jacob(4,4).
*dy4du_t + jacob(4,5).*dy5du_t + jacob(4,6).*dy6du_t ;
393 dy5du_t = dy5du_s + jacob(5,1).*dy1du_t + jacob(5,2).*dy2du_t + jacob(5,3).*dy3du_t
+ jacob(5,4).
*dy4du_t + jacob(5,5).*dy5du_t + jacob(5,6).*dy6du_t ;
394 dy6du_t = dy6du_s + jacob(6,1).*dy1du_t + jacob(6,2).*dy2du_t + jacob(6,3).*dy3du_t
+ jacob(6,4).
*dy4du_t + jacob(6,5).*dy5du_t + jacob(6,6).*dy6du_t ;
395
396 out_des = dataoutesc(k+1,:);
397 out_des = out_des';
398 er = (out_red - out_des);
399 erJ = (out_red - out_des);
400 qq1 = 1; qq2 = 1; qq3 = 1; qq4 = 1; qq5 = 1; qq6 = 1;
401 dJdw_t = dJdw_t + qq1*erJ(1,1).*dy1dw_t + qq2*erJ(2,1).*dy2dw_t +
qq3*erJ(3,1).*dy3dw_t + qq4*erJ
(4,1).*dy4dw_t + qq5*erJ(5,1).*dy5dw_t + qq6*erJ(6,1).*dy6dw_t ;
402 dJdv_t = dJdv_t + qq1*erJ(1,1).*dy1dv_t + qq2*erJ(2,1).*dy2dv_t +
qq3*erJ(3,1).*dy3dv_t + qq4*erJ
(4,1).*dy4dv_t + qq5*erJ(5,1).*dy5dv_t + qq6*erJ(6,1).*dy6dv_t ;
403 dJdu_t = dJdu_t + qq1*erJ(1,1).*dy1du_t + qq2*erJ(2,1).*dy2du_t +
qq3*erJ(3,1).*dy3du_t + qq4*erJ
(4,1).*dy4du_t + qq5*erJ(5,1).*dy5du_t + qq6*erJ(6,1).*dy6du_t ;
404 ersum2 = ersum2 + er.^2;
405 x = out_red; % output turns to be input for the next step
406 end
407
408 dJdw_t = dJdw_t/(2*ndata);
409 dJdv_t = dJdv_t/(2*ndata);
410 dJdu_t = dJdu_t/(2*ndata);
411
412 ersum2total = sum(ersum2);
413 cont = cont + 1;
414 if ( rem(cont,1) == 0 )
415 errorrel(cont/1,:) = sqrt(ersum2'./outsum2');
416 errorreltotal(cont/1,1) = sqrt(ersum2total/outsum2total);
417 erreltotal = errorreltotal(cont/1,1);
418 cont;
419 erreltotal
420 end
```

```
421
422 w = w - eta*dJdw_t + beta*dJdw_t_old;
423 v = v - eta*dJdv_t + beta*dJdv_t_old;
424 ur = ur - eta*dJdu_t + beta*dJdu_t_old;
425
426 dJdw_t_old = dJdw_t;
427 dJdv_t_old = dJdv_t;
428 dJdu_t_old = dJdu_t;
429 end
430
431 figure(14);
432 plot(errorreltotal*100);
433 title('Total Error Function');
434
435 figure(15);
436 plot(errorrel*100);
437 title('Error Function per Output');
438 legend('Pitch Angle','Pitch rate','Roll Angle','Roll Rate','Velocity','Altitude');
439
440 figure(16);
441 % title('Desired Output (red) and Network Output (blue)');
442 plot(z(:,1),'-r');
443 hold on;
444 plot(outputesc(:,1),'-b');
445 title('Pitch Angle');
446 legend('Rojo:Deseada','Azul:Salida Red');
447
448 figure(17);
449 % title('Desired Output (red) and Network Output (blue)');
450 plot(z(:,2),'-r');
451 hold on;
452 plot(outputesc(:,2),'-b');
453 title('Pitch Rate');
454 legend('Rojo:Deseada','Azul:Salida Red');
455
456 figure(18);
457 % title('Desired Output (red) and Network Output (blue)');
458 plot(z(:,3),'-r');
459 hold on;
460 plot(outputesc(:,3),'-b');
461 title('Roll Angle');
462 legend('Rojo:Deseada','Azul:Salida Red');
463
464 figure(19);
465 % title('Desired Output (red) and Network Output (blue)');
466 plot(z(:,4),'-r');
467 hold on;
468 plot(outputesc(:,4),'-b');
469 title('Roll Rate');
470 legend('Rojo:Deseada','Azul:Salida Red');
```

```
471
472 figure(20);
473 % title('Desired Output (red) and Network Output (blue)');
474 plot(z(:,5),'-r');
475 hold on;
476 plot(outputesc(:,5),'-b');
477 title('Velocity');
478 legend('Rojo:Deseada','Azul:Salida Red');
479
480 figure(21);
481 plot(z(:,6),'-r');
482 hold on;
483 plot(outputesc(:,6),'-b');
484 title('Altitude');
485 legend('Rojo:Deseada','Azul:Salida Red');
486
487 figure(23);
488 % title('Input Signal');
489 plot(u,'-b');
490 title('Input Signal u');
491 xlabel('Time [sec]');
```