UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"VEHÍCULO AÉREO NO TRIPULADO BASADO EN CONTROL DE LÓGICA DIFUSA"

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS CON MENCIÓN EN AUTOMÁTICA E INSTRUMENTACIÓN

ELABORADO POR

FABRICIO HUMBERTO PAREDES LARROCA

ASESOR

M.Sc. VÍCTOR SOTELO NEYRA

LIMA - PERÚ

2012

ABSTRACT

The objective of this thesis is to control aeromodel, through the application of Automatic Control techniques. It will be necessary to use the tools of Engineering in several of their fields, including the system analysis, the automatic control, and instrumentation as well as the electronic, mechanical and electronic design. All of these areas were involved in this project.

To achieve the objective is to be used sensors for altitude, speed, position, forces G e inclination were used especially designed for measurement and control mechanisms based on fuzzy logic. The final product consists of a low-cost pilot platform of, geared to development of advanced control strategies.

It has been also used a global positioning system (GPS), the gps coordinates obtained were compared with a flight route previously established. The idea is to replicate this route as the set point to follow using a logic of control based primarily with the theories of fuzzy logic within a computer system known as DSP.

The behavior obtained with the strategies were carried out by flight simulation programs such as the “flight simulator™”. This project includes the flight of airplane with all the electronic described above, but do not include the maneuvers of takeoff and landing because it puts at risk not only the plane the project, in doing so these processes nullifies any activity on the DSP and takes the manual command through the radio control of the aeromodel.

RESUMEN DEL TEMA DE TESIS

El objetivo de esta Tesis es controlar la dinámica de un avión de aeromodelismo, específicamente la altitud y la velocidad del avión, utilizando sensores especialmente diseñados y mecanismos de control basados en sistemas difusos. El producto final consiste en una plataforma experimental de bajo costo, orientada al desarrollo de estrategias de control avanzado.

Los sensores utilizan circuitos electrónicos de tecnología avanzada, estos son: un GPS para medir la velocidad del avión, un giroscopio para conocer la posición del avión en el espacio, un compás para conocer la dirección de vuelo.

La información de los sensores se procesa con circuitos lineales y microprocesadores especialmente programados, El sistema de control analiza la información proveniente de los sensores y las restricciones impuestas por las necesidades del vuelo, para generar las variables manipuladas que corresponden a la potencia del motor y la deflexión de los elevadores, cuyos valores se transmiten al avión utilizando un DSP para el control.

El sistema de control considera la estrategia de un control difuso. El comportamiento obtenido con la estrategia se comparará por simulación, empleando un modelo matemático del avión desarrollado para diferentes condiciones de vuelo, y haciendo volar el avión en forma experimental bajo diferentes condiciones.

El proyecto consiste en el desarrollo e implementación del control autónomo de un aeromodelo, mediante la aplicación de técnicas de Control Automático. Para ello, será necesario utilizar las herramientas de la Ingeniería en varios de sus campos, incluyendo el análisis de sistemas, el control automático, la instrumentación, la electrónica, la mecánica y el diseño eléctrico. Todas estas áreas se combinan y formaran parte en este trabajo.

Para lograr el objetivo planteado se empleará sensores de: altitud, velocidad, posición e inclinación angular.

Se plantea también la utilización de un sistema de posicionamiento global (GPS), estas coordenadas obtenidas serán comparadas con una ruta de vuelo previamente establecida, la idea es poder replicar esta, con la que se generaría al momento del vuelo y tratar de aproximarla con una lógica de control basado fundamentalmente con las teorías de la lógica difusa, esto enmarcado dentro de un sistema computacional conocido como DSP.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS Y DIAGRAMAS.

Pág

INTRODUCCIÓN.

El presente trabajo práctico consiste en el desarrollo de un una avión que pueda volar y seguir una ruta de forma autónoma. Para ello se debe programar en lenguaje ANSI C una unidad de navegación inercial. Esta se encontrará compuesta por un conjunto de sensores tales como: Giróscopos, acelerómetros, brújula, Gps para conocer la posición del avión en el espacio. Los actuadotes serán el servo motor que se encuentran manipulando tanto el elevador como el timón de cola. La teoría que logrará controlarlo será la de lógica difusa, hay que tener en cuenta que el modelo matemático no se conoce, es por eso que esta teoría de control es la más idónea.

Por otro lado se ha realizado simulaciones con modelos aproximados utilizando las teorías de control óptimo, trabajando con algunas trayectorias suaves (ondas sinusoidales) y otras bastante bruscas (ondas cuadradas) teniendo éxito la simulación. Las pruebas de campo fueron bastante importantes porque a través de ellas se fue perfeccionando el sistema de lógica difusa.

Hay que destacar que para llegar a tomar la decisión de utilizar la unidad inercial se hicieron varias pruebas con otros dispositivos, estos no funcionaron y otros tuvieron problemas de comunicación. Otros tuvieron fallas de fábrica y fueron devueltos al país de procedencia.

1.- PLANTEAMIENTO DEL PROBLEMA

El vuelo no tripulado no es un tema nuevo. Esto tiene como antecedentes los conflictos bélicos surgidos en las dos últimas décadas. La idea de realizar este proyecto consiste en crear soluciones nuevas de bajo costo para poder ser implementadas en nuestro país.

La principal aplicación que me motivo a realizar este trabajo fue la de optimizar las vías de las carreteras hacia el sur por ejemplo en el verano, en donde el gasto de combustible y tiempo innecesario para trasladarse fuera de Lima. Otra de las tantas aplicaciones que se dan a nivel internacional es el uso de esta tecnología para detectar en la selva los cultivos de plantaciones ilegales o la presencia de insurgentes en un área no permitida o simplemente como sistemas de transmisión de sonidos por altoparlantes para decir algo específico a una población. Finalmente el trabajo práctico busca solucionar el problema de tripular una aeronave de tal manera que no existan los altos costos de combustibles o los de operación o también la exposición de vidas como la del piloto en zonas donde existe un peligro latente. Si cumple con guiar un aeromodelo entonces podría ser el principio de control para un aeronave mucho más grande.

1.1.-ANTECEDENTES

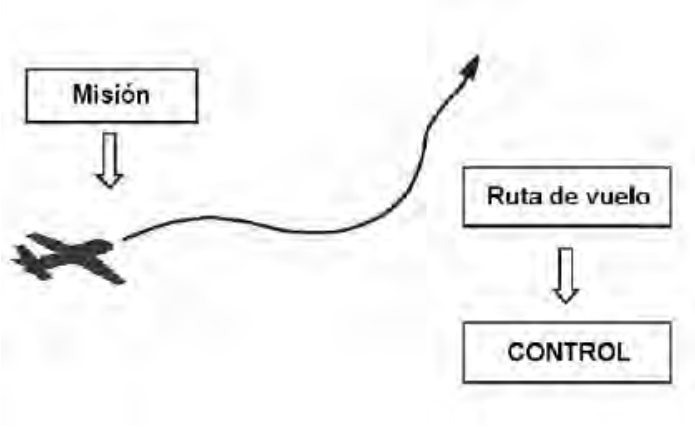
El ejemplo más antiguo fue desarrollado después de la primera guerra mundial, y se emplearon durante la segunda guerra mundial para entrenar a los operarios de los cañones antiaéreos. Sin embargo, no es hasta poco más que a finales del siglo XX cuando operan los '*UAV* (por sus siglas del inglés **UAV** *Unmanned Aerial Vehicle*) mediante radio control con todas las características de autonomía.

Los UAV han demostrado sobradamente en diferentes escenarios y especialmete en la Guerra del Golfo y en la Guerra de Bosnia, el gran potencial que pueden tener. En cuanto a la obtención, manejo y transmisión de la información, gracias a la aplicación de nuevas técnicas de protección de la misma (Guerra electrónica, criptografía) resulta posible conseguir comunicaciones más seguras, más difíciles de detectar e interferir.

Se pueden aplicar en ambientes de alta toxicidad química y radiológicos en desastres tipo Chernobyl, en los que sea necesario tomar muestras con alto peligro de vidas humanas y realizar tareas de control de ambiente. Además, pueden cooperar en misiones de control del narcotráfico y contra el terrorismo. También podrían grabar videos de alta calidad para ser empleados como medios de prueba.

1.2.- OBJETIVO

Se desea que el aeromodelo tenga la capacidad de llevar a cabo misiones en forma autónoma. Una misión consiste en una serie de puntos geográficos, estos puntos definidos previamente por el usuario, forman la ruta de vuelo deseada. Lo que tiene que hacer el sistema de control entonces es, dada una condición inicial (no se tienen en cuenta despegues y aterrizajes), generar acciones sobre las variables de control para que el aeromodelo permanezca dentro de la ruta y pase por los puntos objetivo.



Misión-Ruta de Vuelo.

FIG.1

Definición de misión del aeromodelo. Los puntos objetivos predeterminados definen la ruta de vuelo. El control se encarga de que el aeromodelo siga la ruta.

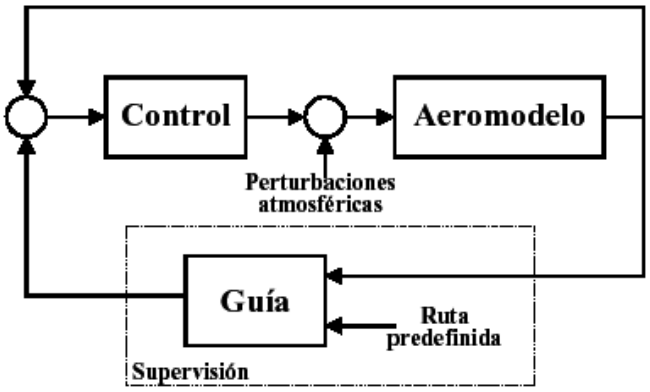


Diagrama de bloques del sistema de control propuesto.

FIG.2

Diagrama de bloques del sistema de control propuesto. El bloque de control regula la dinámica del aeromodelo de acuerdo a su estado actual y a las señales de referencia generadas por el bloque guía de acuerdo al plan de vuelo previamente definido.

2.-FUERZAS SOBRE EL AVIÓN.

Son básicamente cuatro las fuerzas que influyen en el movimiento de un avión típico. Estas cuatro fuerzas interactúan de diferentes formas para dar lugar a cada una de las situaciones más comunes de vuelo, como lo son el despegue, el aterrizaje, el ascenso, el descenso y el vuelo a nivel. Estas fuerzas se describen a continuación[Springer01].

• Sustentación: Esta fuerza de origen aerodinámico es provocada principalmente por las alas del avión, y permite que éste se eleve.

• Arrastre: Ésta también tiene su origen en la aerodinámica del avión, y se opone a su movimiento. Lo ideal es que esta fuerza sea pequeña.

• Propulsión: Es la fuerza que produce el motor del avión para impulsarlo. Esta fuerza se opone al arrastre, y cuando ambas son iguales, el avión mantiene su Velocidad.

• Fuerza de Gravedad: Es provocada por el campo gravitacional de la Tierra, y se mantiene relativamente constante en vuelos dentro de la atmósfera.

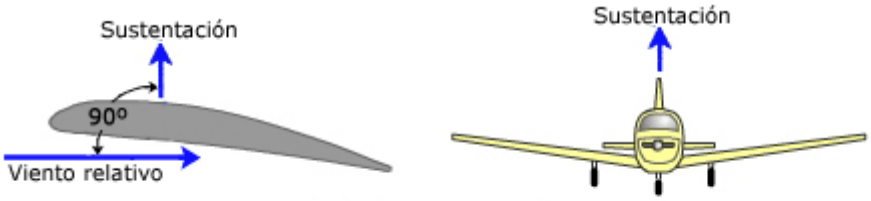
Fuerzas que actuan en vuelo.

FIG.3

Además de las fuerzas básicas anteriores, existen los momentos (torques) del avión, que se producen por la acción de las fuerzas en torno a un punto. Estos momentos se generan en torno a los tres ejes del avión, y dan lugar a los tres tipos de giro que pueden producirse (cabeceo, balanceo, guiñada) [AFD98].

En el vuelo de un avión la fuerza más importante es la sustentación, que permite que el avión se eleve venciendo la fuerza de gravedad.

La fuerza de sustentación se produce por las diferencias de presión que existen entre la cara superior e inferior del ala. Estas diferencias de presión, aplicadas sobre una superficie importante (como la del ala) generan las fuerzas como para elevar el avión. Para producir tales diferencias de presión, según el principio de Bernoulli es necesario que el flujo de aire en la parte superior del ala viaje más rápido que el de su parte inferior. Con esto, la presión arriba disminuye con respecto a la de abajo y la fuerza que se produce tiende a contrarrestar la gravedad [AFD98].



Perpendicularidad de la sustentación.
FIG.4

2.1.-ACTITUD DEL AVIÓN. Este término se refiere a la orientación o referencia angular de los ejes longitudinal y transversal del avión con respecto al horizonte, y se especifica en términos de: posición de morro (pitch) y posición de las alas (bank); por ejemplo: el avión esta volando con 5º de morro arriba y 15º de alabeo a la izquierda.

2.2.-TRAYECTORIA DE VUELO. Es la dirección seguida por el perfil aerodinámico durante su desplazamiento en el aire; es decir es la trayectoria que siguen las alas y por tanto el avión.

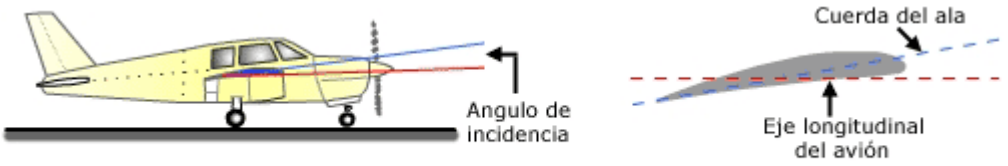
2.3.-VIENTO RELATIVO. Es el flujo de aire que produce el avión al desplazarse. El viento relativo es paralelo a la trayectoria de vuelo y de dirección opuesta. Su velocidad es la relativa del avión con respecto a la velocidad de la masa de aire.

Trayectoria de vuelo y viento relativo.

FIG.5

Es importante destacar que no debe asociarse la trayectoria de vuelo, ni por tanto el viento relativo, con la actitud de morro del avión; por ejemplo, una trayectoria de vuelo recto y nivelado puede llevar aparejada una actitud de morro ligeramente elevada.

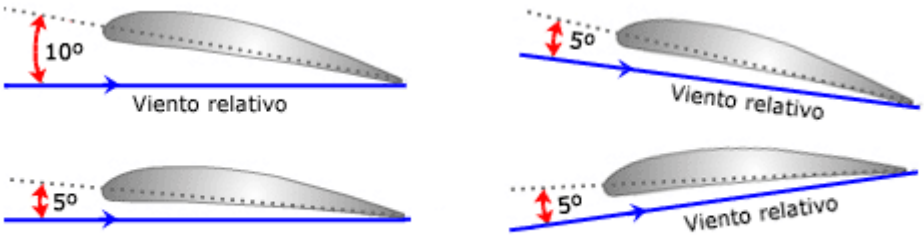
2.4.-ÁNGULO DE INCIDENCIA. El ángulo de incidencia es el ángulo agudo formado por la cuerda del ala con respecto al eje longitudinal del avión. Este ángulo es fijo, pues responde a consideraciones de diseño y no es modificable por el piloto. [Airlow]



Ángulo de incidencia.

FIG.6

2.5.-ÁNGULO DE ATAQUE. El ángulo de ataque es el ángulo agudo formado por la cuerda del ala y la dirección del viento relativo. Este ángulo es variable, pues depende de la dirección del viento relativo y de la posición de las alas con respecto a este, ambos extremos controlados por el piloto. Es conveniente tener muy claro el concepto de ángulo de ataque pues el vuelo está directa y estrechamente relacionado con el mismo.



Ángulo de ataque y viento relativo.

FIG.7

Es importante notar, tal como muestra, que el ángulo de ataque se mide respecto al viento relativo y no en relación a la línea del horizonte. En la parte de la izquierda el avión mantiene una trayectoria horizontal (el viento relativo también lo es) con diferentes ángulos de ataque (5º y 10º); a la derecha y arriba, el avión mantiene una trayectoria ascendente con un ángulo de ataque de 5º, mientras que a la derecha y abajo la trayectoria es descendente también con un ángulo de ataque de 5º. Dada la importancia de este concepto. [Airlow]

Se muestran distintas fases de un avión en vuelo, en cada una de las cuales podemos apreciar de una manera gráfica los conceptos definidos: la trayectoria; el viento relativo, paralelo y de dirección opuesta a la trayectoria, y la sustentación, perpendicular al viento relativo.

La sustentación es perpendicular al viento relativo, que es paralelo y opuesto a la trayectoria.

FIG.8

2.6.-FACTORES QUE AFECTAN A LA SUSTENTACIÓN.

La forma del perfil del ala. Hasta cierto límite, a mayor curvatura del perfil mayor diferencia de velocidad entre las superficies superior e inferior del ala y por tanto mayor diferencia de presión, o lo que es igual mayor fuerza de sustentación. No obstante no hay que confundirse pensando que es necesario que el ala sea curvada por arriba y plana o cóncava por abajo para producir sustentación, pues un ala con un perfil simétrico también la produce. Lo que ocurre es que un ala ligeramente curvada entra en pérdida con un ángulo de ataque mucho mayor que un ala simétrica, lo que significa que tanto su coeficiente de sustentación como su resistencia a la pérdida son mayores [AFD98].

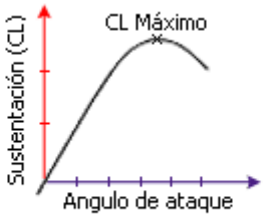
La curvatura de un ala típica moderna es solo de un 1% o un 2%. La razón por la cual no se hace más curvada, es que un incremento de esta curvatura requeriría una superficie inferior cóncava, lo cual ofrece dificultades de construcción. Otra razón, es que una gran curvatura solo es realmente beneficiosa en velocidades cercanas a la pérdida (despegue y aterrizaje), y para tener más sustentación en esos momentos es suficiente con extender los flaps.

La superficie alar. Cuanto más grandes sean las alas mayor será la superficie sobre la que se ejerce la fuerza de sustentación. Pero hay que tener en cuenta que perfiles muy curvados o alas muy grandes incrementan la resistencia del avión al ofrecer mayor superficie enfrentada a la corriente de aire. En cualquier caso, tanto la forma como la superficie del ala dependen del criterio del diseñador, que tendrá que adoptar un compromiso entre todos los factores según convenga a la funcionalidad del avión.

La densidad del aire. Cuanto mayor sea la densidad del aire, mayor es el número de partículas por unidad de volumen que cambian velocidad por presión y producen sustentación [ACS03].

La velocidad del viento relativo. A mayor velocidad sobre el perfil, mayor es la sustentación. La sustentación es proporcional al cuadrado de la velocidad (factor $\mathbf{v^2}$ del teorema de Bernoulli), siendo por tanto este factor el que comparativamente más afecta a la sustentación.

El ángulo de ataque. Si se aumenta el ángulo de ataque es como si se aumentara la curvatura de la parte superior del perfil, o sea el estrechamiento al flujo de aire, y por tanto la diferencia de presiones y en consecuencia la sustentación. En el siguiente gráfico se ve de forma general como aumenta el coeficiente de sustentación (CL) con el ángulo de ataque hasta llegar al CL máximo, a partir del cual la sustentación disminuye con el ángulo de ataque. Los valores y la forma de la curva en la gráfica dependerán de cada perfil concreto [AFD98].



Coeficiente de sustentación vs. Angulo de ataque.
FIG.9

En resumen, la sustentación creada por el ala está en función de:

- El coeficiente aerodinámico (Forma del perfil).
- La superficie alar.
- La densidad del aire.
- La velocidad del viento relativo.
- El ángulo de ataque.

La fórmula correspondiente sería: **L=CL*q*S** donde **CL** es el coeficiente de sustentación, dependiente del tipo de perfil y del ángulo de ataque; **q** la presión aerodinámica (*1/2dv²* siendo *d* la densidad y *v* la velocidad del viento relativo) y **S** la superficie alar. [WingS]

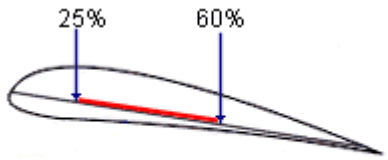
2.7.-CENTRO DE PRESIONES.

Se denomina centro de presiones al punto teórico del ala donde se considera aplicada toda la fuerza de sustentación. La figura muestra un ejemplo de distribución de presiones sobre un perfil moviéndose en el aire. A efectos teóricos, aunque la presión actúa sobre todo el perfil, se considera que toda la fuerza de sustentación se ejerce sobre un punto en la línea de la cuerda (resultante).

Centro de presiones.

FIG.10

A medida que aumenta o disminuye el ángulo de ataque se modifica la distribución de presiones alrededor del perfil, desplazándose el centro de presiones, dentro de unos límites, hacia adelante o atrás respectivamente. El margen de desplazamiento suele estar entre el 25% y el 60% de la cuerda, y puesto que afecta a la estabilidad de la aeronave es conveniente que sea el menor posible. [WingS]



Límites de desplazamiento del centro de presiones.

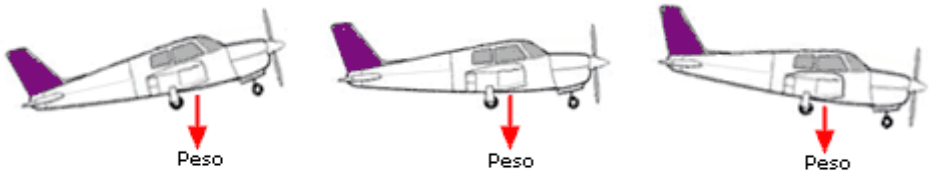
FIG.11

Mediante métodos empíricos se ha demostrado que a medida que se incrementa el ángulo de ataque, el Centro de Presiones se desplaza gradualmente hacia adelante. En un punto más allá del ángulo de ataque para vuelo ordinario, comienza a moverse hacia atrás de nuevo; cuando llega a un punto lo suficientemente atrás, el morro del avión cae porque el ala está en pérdida.

2.8.-PESO.

El peso es la fuerza de atracción gravitatoria sobre un cuerpo, siendo su dirección perpendicular a la superficie de la tierra, su sentido hacia abajo, y su intensidad proporcional a la masa de dicho cuerpo. Esta fuerza es la que atrae al avión hacia

la tierra y ha de ser contrarrestada por la fuerza de sustentación para mantener al avión en el aire.



Dirección y sentido del peso.

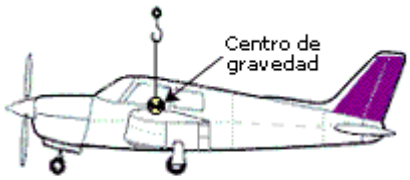
FIG .12

Dependiendo de sus características, cada avión tiene un peso máximo que no debe ser sobrepasado, como debe efectuarse la carga de un avión para no exceder sus limitaciones.

2.9.-CENTRO DE GRAVEDAD.

Es el punto donde se considera ejercida toda la fuerza de gravedad, es decir el peso. El C.G es el punto de balance de manera que si se pudiera colgar el avión por ese punto específico este quedaría en perfecto equilibrio. El avión realiza todos sus movimientos pivotando sobre el C.G [ACS03].

La situación del centro de gravedad respecto al centro de presiones tiene una importancia enorme en la estabilidad y controlabilidad del avión.



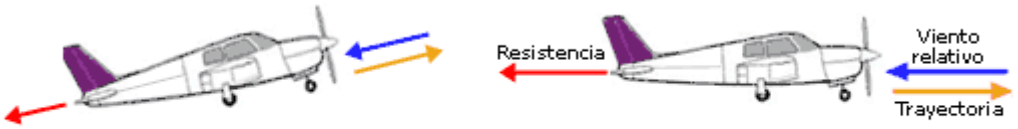
Centro de gravedad

FIG .13

2.10.-RESISTENCIA.

La resistencia es la fuerza que impide o retarda el movimiento de un aeroplano. La resistencia actúa de forma paralela y en la misma dirección que el viento

relativo, aunque también podríamos afirmar que la resistencia es paralela y de dirección opuesta a la trayectoria.



Dirección y sentido de la resistencia.

FIG. 14

Desde un punto de vista aerodinámico, cuando un ala se desplaza a través del aire hay dos tipos de resistencia: (a) resistencia debida a la fricción del aire sobre la superficie del ala, y (b) resistencia por la presión del propio aire oponiéndose al movimiento.

La resistencia por fricción es proporcional a la viscosidad, que en el aire es muy baja, de manera que la mayoría de las veces esta resistencia es pequeña comparada con la producida por la presión, mientras que la resistencia debida a la presión depende de la densidad de la masa de aire.

Ambas resistencias crean una fuerza proporcional al área sobre la que actúan y al cuadrado de la velocidad. Una parte de la resistencia por presión que produce un ala depende de la cantidad de sustentación producida; a esta parte se le denomina resistencia inducida, denominándose resistencia parásita a la suma del resto de resistencias.

La fórmula de la resistencia (en ingles "drag") tiene la misma forma que la de la sustentación: **D=CD*q*S** donde **CD** es el coeficiente de resistencia, dependiente del tipo de perfil y del ángulo de ataque; **q** la presión aerodinámica ($1/2dv^2$ siendo *d* la densidad y *v* la velocidad del viento relativo) y **S** la superficie alar [ACS03].

La resistencia total del avión es pues la suma de dos tipos de resistencia: la resistencia inducida y la resistencia parásita.

2.11.-RESISTENCIA INDUCIDA.

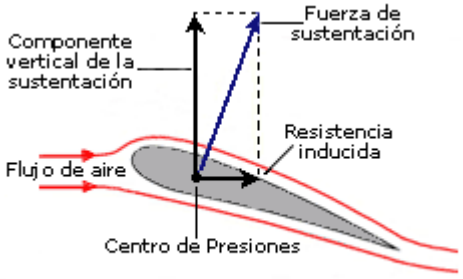
La resistencia inducida, indeseada pero inevitable, es un producto de la sustentación, y se incrementa en proporción directa al incremento del ángulo de ataque.

Al encontrarse en la parte posterior del ala la corriente de aire que fluye por arriba con la que fluye por debajo, la mayor velocidad de la primera deflecta hacia abajo a la segunda haciendo variar ligeramente el viento relativo, y este efecto crea una resistencia. Este efecto es más acusado en el extremo del ala, pues el aire que fluye por debajo encuentra una vía de escape hacia arriba donde hay menor presión, pero la mayor velocidad del aire fluyendo por arriba deflecta esa corriente hacia abajo produciéndose resistencia adicional. Este movimiento de remolino crea vórtices que absorben energía del avión. [Airlow]

Deflexión del flujo de aire.

FIG. 15

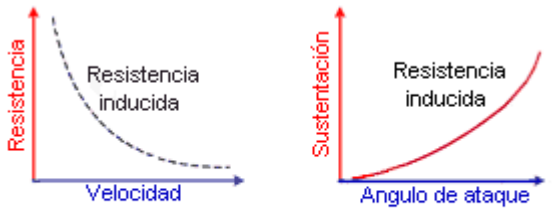
Representadas de forma gráfica la sustentación y la resistencia, la fuerza aerodinámica se descompone en dos fuerzas: una aprovechable de sustentación y otra no deseada pero inevitable de resistencia [ACS03].



Resistencia inducida.

FIG. 16

De la explicación dada se deduce claramente que la resistencia inducida aumenta a medida que aumenta el ángulo de ataque. Pero si para mantener la misma sustentación ponemos más velocidad y menos ángulo de ataque, la resistencia inducida será menor, de lo cual deducimos que la resistencia inducida disminuye con el aumento de velocidad. La siguiente figura nos muestra la relación entre la resistencia inducida, la velocidad, y el ángulo de ataque. [Airlow]



Variación de la resistencia inducida con la velocidad y el ángulo de ataque.

FIG. 17

En la resistencia inducida también tiene influencia la forma de las alas; un ala alargada y estrecha tiene menos resistencia inducida que un ala corta y ancha.

2.12.-RESISTENCIA PARÁSITA.

Es la producida por las demás resistencias no relacionadas con la sustentación, como son: resistencia al avance de las partes del avión que sobresalen (fuselaje, tren de aterrizaje no retráctil, antenas de radio, etc.); entorpecimiento del flujo del aire en alas sucias por impacto de insectos, rozamiento o fricción superficial con el aire; interferencia del flujo de aire a lo largo del fuselaje con el flujo de las alas; el flujo de aire canalizado al compartimiento del motor para refrigerarlo (que puede suponer en algunos aeroplanos cerca del 30% de la resistencia total).También, la superficie total del ala y la forma de esta afecta a la resistencia parásita; un ala más alargada presenta mayor superficie al viento, y por ello mayor resistencia parásita, que un ala más corta. Lógicamente, cuanto mayor sea la velocidad mayor será el efecto de la resistencia parásita: la resistencia parásita aumenta con la velocidad.

Resistencia parásita vs. Velocidad.

FIG. 18

Si la resistencia inducida es un producto de la sustentación, y en la resistencia parásita tienen influencia la superficie alar y la forma del ala, prácticamente todos los factores que afectan a la sustentación afectan en mayor o menor medida a la resistencia.

Si con el aumento de velocidad disminuye la resistencia inducida y se incrementa la resistencia parásita, tiene que haber un punto en que la suma de ambas (resistencia total) sea el menor posible.

Resistencia total

FIG. 19

A baja velocidad la mayoría de la resistencia es inducida, debido al incremento del ángulo de ataque para producir suficiente sustentación para soportar el peso del avión. A medida que la velocidad sigue bajando, la resistencia inducida se incrementa rápidamente y la resistencia parásita apenas tiene influencia. Por el contrario, a alta velocidad la resistencia parásita es la dominante mientras que la inducida es irrelevante.

Resumiendo:

- A mayor velocidad menor resistencia inducida.

- A mayor ángulo de ataque mayor resistencia inducida.
- A mayor velocidad mayor resistencia parásita.

2.13.- EMPUJE O TRACCIÓN.

Para vencer la inercia del avión parado, acelerarlo en la carrera de despegue o en vuelo, mantener una tasa de ascenso adecuada, vencer la resistencia al avance, se necesita una fuerza: el empuje o tracción. Esta fuerza se obtiene acelerando una masa de aire a una velocidad mayor que la del aeroplano. La reacción, de igual intensidad pero de sentido opuesto (3ª ley del movimiento de Newton), mueve el avión hacia adelante. En aviones de hélice, la fuerza de propulsión la genera la rotación de la hélice, movida por el motor.

Esta fuerza se ejerce en la misma dirección a la que apunta el eje del sistema propulsor, que suele ser más o menos paralela al eje longitudinal del avión.

Dirección y sentido de empuje.

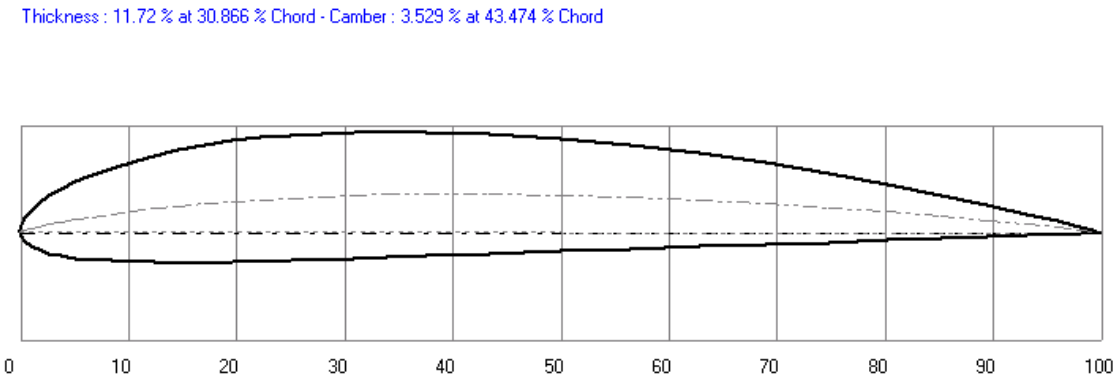
FIG. 20

Puesto que potencia es equivalente a energía por unidad de tiempo, a mayor potencia mayor capacidad de aceleración.

La potencia es el factor más importante a la hora de determinar la tasa de ascenso de un avión. De hecho la tasa máxima de ascenso de un avión no está relacionada con la sustentación sino con la potencia disponible descontada la necesaria para mantener un vuelo nivelado [ACS03].

En el estudio del aeromodelo se ha obtenido los valores del perfil (Clark Y) a través de simuladores y datos de manuales sobre perfiles, estos me han permitido resolver las ecuaciones diferenciales que se deben de tener en cuenta para poder

desarrollar el modelo matemático. Las simulaciones de este perfil se apreciarán en las siguientes figuras.



Perfil tipo Clark Y- Utilizado en el ala del avión Cessna 182[Airfoil].

FIG. 21

Xupper	Yupper	Xlower	Ylower
0	.047	0	.047
.107	.616	.107	-.453
.428	1.254	.428	-.898
.961	1.943	.961	-1.296
1.704	2.652	1.704	-1.651
2.653	3.352	2.653	-1.959
3.806	4.027	3.806	-2.214
5.156	4.677	5.156	-2.414
6.699	5.313	6.699	-2.567
8.427	5.939	8.427	-2.68
10.332	6.552	10.332	-2.763
12.408	7.134	12.408	-2.816
14.645	7.66	14.645	-2.839
17.033	8.113	17.033	-2.832
19.562	8.483	19.562	-2.795
22.221	8.774	22.221	-2.734
25	8.996	25	-2.653
27.886	9.158	27.886	-2.559
30.866	9.266	30.866	-2.458
33.928	9.318	33.928	-2.351
37.059	9.312	37.059	-2.242
43.474	9.128	43.474	-2.018
50	8.719	50	-1.792
56.526	8.105	56.526	-1.566
62.941	7.319	62.941	-1.345
69.134	6.405	69.134	-1.131

Coordenadas con las que se traza el perfil Clark Y[Airfoil].

Xupper, Yupper, cordenadas superiores del perfil.
Xupper, Yupper, cordenadas inferiores del perfil.
Unidades en coordenadas polares.

FIG. 22

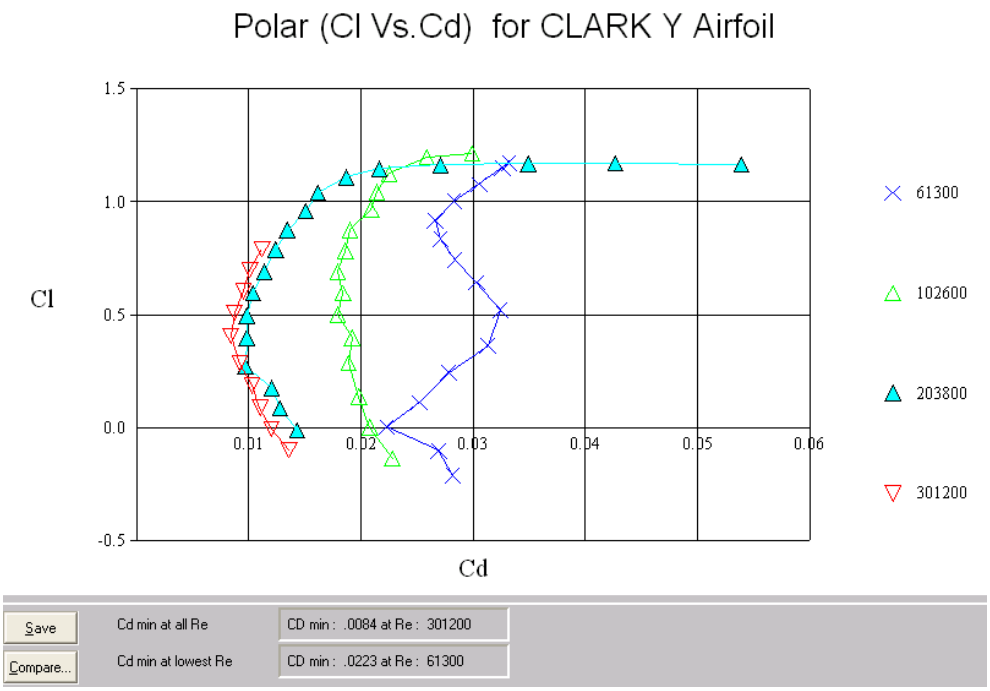
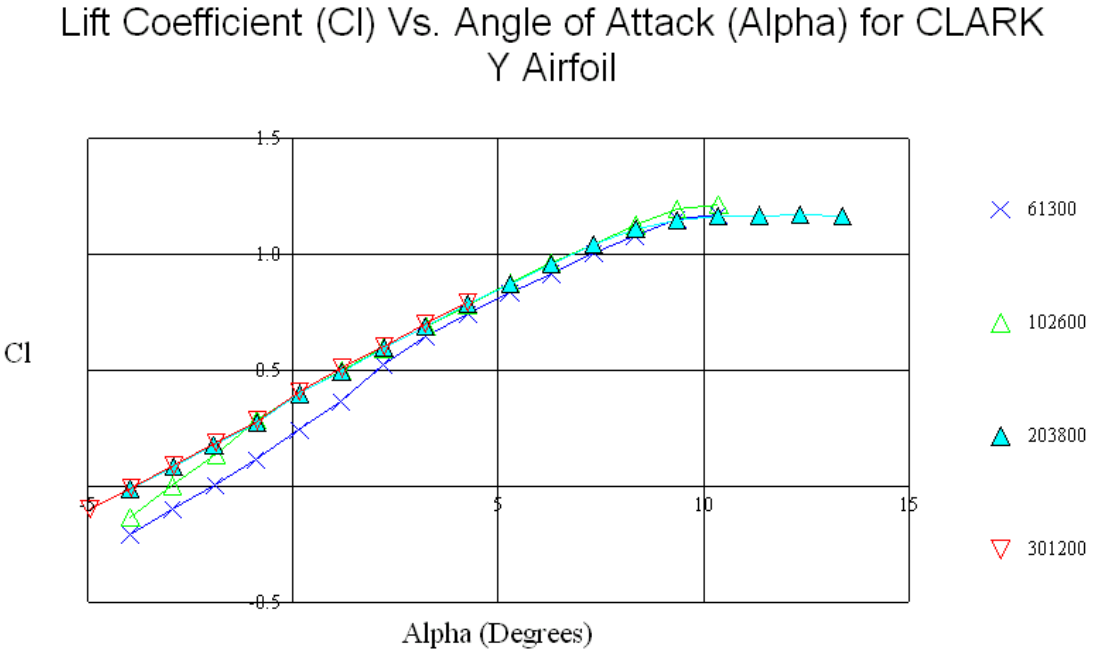


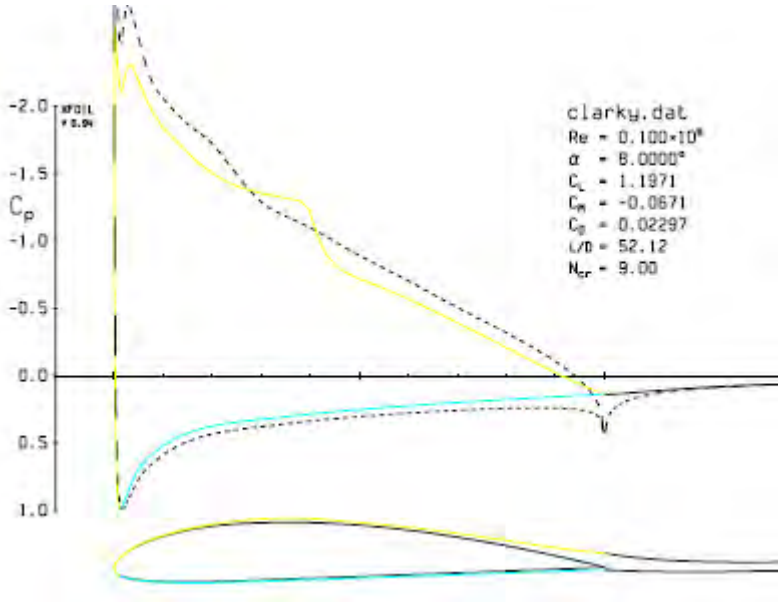
Gráfico de coordenadas polares en donde se compara CI Vs Cd para un perfil Clark Y [Airfoil]

FIG.23



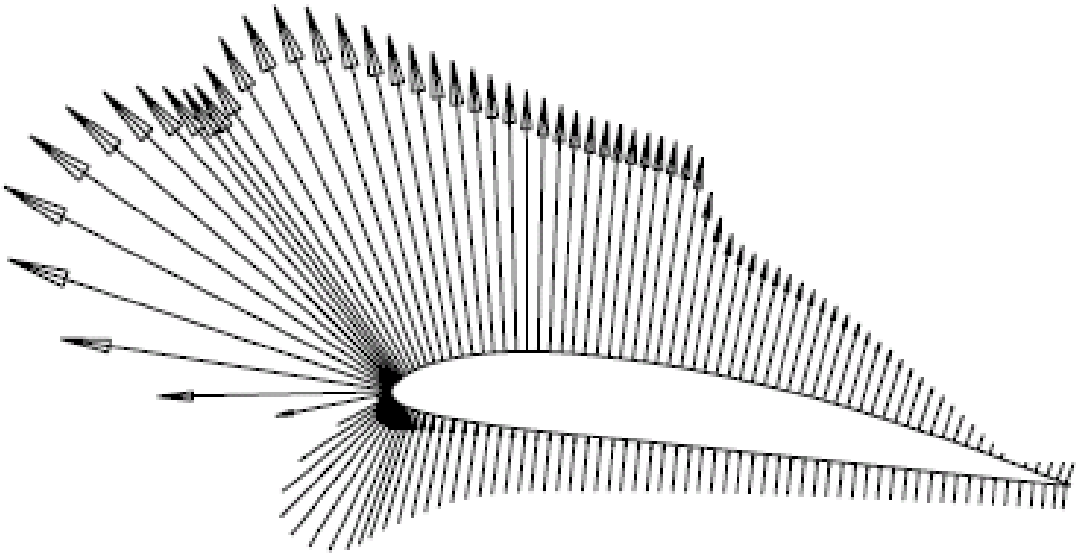
Coeficiente CI Vs. el ángulo de ataque para el perfil Clark Y [Airfoil].

FIG. 24



Simulación del perfil Clark Y con un Reynolds de 102600 [Airfoil].

FIG. 25



Presiones que actúan en el perfil Clark Y con un Reynolds de 102600 [Airfoil].

FIG. 26

3.- DISEÑO DE CONTROLADORES Y OBSERVADORES DE SISTEMAS DE MULTIPLE ENTRADA Y MULTIPLE (MIMO) EN EL ESPACIO DE ESTADOS

3.1.- INTRODUCION

Al diseñar sistemas de control de múltiple entrada y múltiple (multivariable) salida usando el método de ubicación de polos en el plano *Z* se presenta el inconveniente de obtener soluciones múltiples, cada solución presenta diversidad de formas en el transitorio y generalmente no cumplen con las especificaciones de tiempo elegidas, por lo tanto el escoger la solución más próxima o sea la mejor es un gran inconveniente [CA01].

Otro método a usar es el *Control Optimo* que consiste en optimar u optimizar el valor de una función seleccionada como el *índice de desempeño* y por lo tanto es de gran interés para los ingenieros de control.

Al diseñar un controlador multivariable mediante sistemas de control óptimo, se necesita encontrar una regla que determine la decisión de control presente, sujeta a ciertas restricciones, para minimizar la desviación del comportamiento ideal, esta medida es prevista, generalmente, por el índice de desempeño seleccionado que es una función cuyo valor se considera una indicación de qué tanto se parece el desempeño del sistema real al desempeñado deseado.

3.2.-FORMULACIÓN DE UN SISTEMA DE CONTROL MULTIVARIABLE.

Considerando un sistema de control lineal invariante en el tiempo de múltiple entrada y múltiple salida descrito mediante las ecuaciones de estado y de salida como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) \tag{3.1}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{3.2}$$

x*(k)* : vector de estado de *n×1*

y*(k)* : vector de salida de *m×1*

u*(k)* : vector de entrada o de control de *r×1*

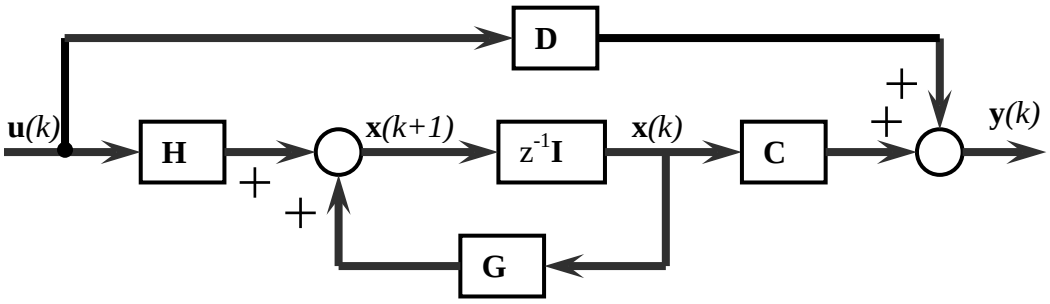
G : matriz de estado de *n×n*

H : matriz de entrada de *n×r*

C : matriz de salida de *m×n*

D : matriz de transmisión directa de *m×r*

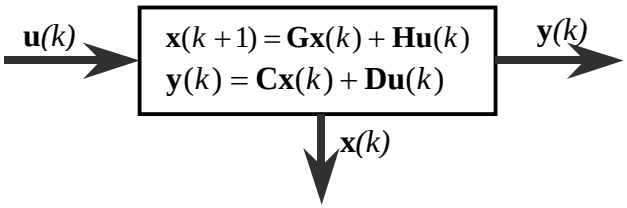
La representación gráfica de las ecuaciones (3.1) y (3.2) se muestra en la figura 21.



Sistema de control multivariable en lazo abierto.

Fig. 27

El diagrama de bloques del sistema de control de la figura 21 se puede simplificar mediante la representación de gráfica de la figura 22



Sistema de control simplificado en lazo abierto.

Fig. 28

3.3.-CONTROLABILIDAD DE UN SISTEMA DE CONTROL MULTIVARIABLE.

Un sistema de control es de estado completamente controlable, si es posible transferir el sistema de un estado inicial arbitrario a cualquier otro estado deseado arbitrario en un tiempo finito.

La matriz de controlabilidad M de dimensión *n*×*rn* es

$$\mathbf{M} = \begin{bmatrix} \mathbf{H} & \mathbf{GH} & \mathbf{G}^2\mathbf{H} & \cdots & \mathbf{G}^{n-1}\mathbf{H} \end{bmatrix} \qquad (3.3)$$

El sistema es controlable en estado completo si la matriz M tiene rango = *n*

3.4.-OBSERVABILIDAD DE UN SISTEMA DE CONTROL MULTIVARIABLE.

Un sistema de control es de estado completamente observable, si cualquier estado inicial x*(0)* se puede determinar a partir de la observación de *y(kT)* sobre un número finito de periodos de muestreo. El sistema, por lo tanto, es completamente observable, si cualquier transición del estado de manera eventual afecta a todos los elementos del vector de salida.

La matriz de observabilidad N de dimensión *nm×n* es

$$\mathbf{N} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CG} \\ \mathbf{CG}^2 \\ \vdots \\ \mathbf{CG}^{n-1} \end{bmatrix} \tag{3.4}$$

El sistema es observable en estado completo si la matriz N tiene rango = *n*

3.5.- DISEÑO DE UN CONTROLADOR (REGULADOR) CON ENTRADA DE REFERENCIA NULA.

En el diseño de controladores en realimentación del vector de estado x*(k)* de dimensión *n×1* cuando el vector de control u*(k)* de dimensión *r×1* para un sistema controlable en estado completo y descrito por las ecuaciones (3.1) y (3.2), se elige el vector de control expresado como

$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k) \tag{3.5}$$

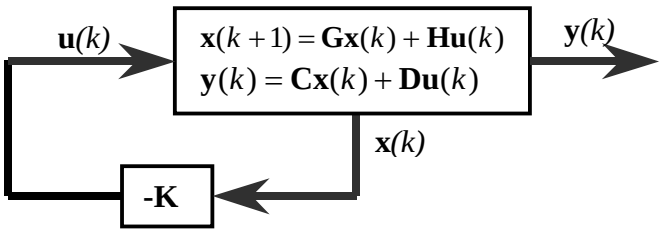
Para determinar la matriz de ganancias de estados K de dimensión *r×n* se siguen los siguientes pasos:

1. Determinar si el par (G, H) es controlable en estado completo, es decir si la matriz de controlabilidad M de dimensión *n×rn* tiene rango *n*.
2. Obtener aleatoriamente una matriz K_S de dimensión *r×n* y definir G_0=G-HK_S de tal modo que G_0 presente valores propios distintos, sino elegir otra matriz K_s .
3. Realizar una combinación lineal aleatoria de las columnas de H para generar H_0=Hv de tal manera que el par (G_0, H_0) representa un sistema

equivalente de entrada simple. Es decir G_0, tiene dimensión *n×n,* H_0 dimensión *n×1* y el vector v debe tener dimensión *r×1*

4. Asignar el espectro o conjunto deseado de polos ubicados en lazo cerrado μ_1, μ_2, ..., μ_n y determinar la matriz de realimentación de estados K_0 usando cualquier método de diseño mediante ubicación de polos de sistemas de regulación con entrada simple tal que los valores propios de $(G_0-H_0K_0)$ sean iguales a los polos deseados en lazo cerrado.
5. Determinar la matriz deseada de realimentación K haciendo $K=K_s+vK_0$
6. Comprobar que la matriz (G-HK) presente el mismo espectro deseado.

La representación gráfica del sistema de control simplificado expresado mediante las ecuaciones (3.1), (3.2) y (3.5) se muestra en la figura 23



Sistema de control en lazo cerrado.

Fig. 29

Al reemplazar la ecuación (3.5) en la ecuación (3.1) se obtiene

$$\mathbf{x}(k+1) = (\mathbf{G} - \mathbf{HK})\mathbf{x}(k) \tag{3.6}$$

La solución de la ecuación de estado es

$$\mathbf{x}(k) = (\mathbf{G} - \mathbf{HK})^k \mathbf{x}(0) \tag{3.7}$$

Reemplazando la ecuación (3.5) en la ecuación (3.2) resulta

$$y(k) = \mathbf{Cx}(k) - \mathbf{DKx}(k) = (\mathbf{C} - \mathbf{DK})\mathbf{x}(k) \tag{3.8}$$

3.6.- DISEÑO DE OBSERVADORES DE ESTADO DE MÚLTIPLE SALIDA MEDIANTE EL METODO DE UBICACION DE POLOS.

Cuando el vector real x*(k)* no está disponible o no es medible en forma directa de diseña un estimador de estados para que un vector estimado $\tilde{\mathbf{x}}(k)$ sea lo más cercano al vector de estados real x*(k)*.

Considerando el sistema de control completamente observable descrito por las ecuaciones (3.1) y (3.2) definidas por

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

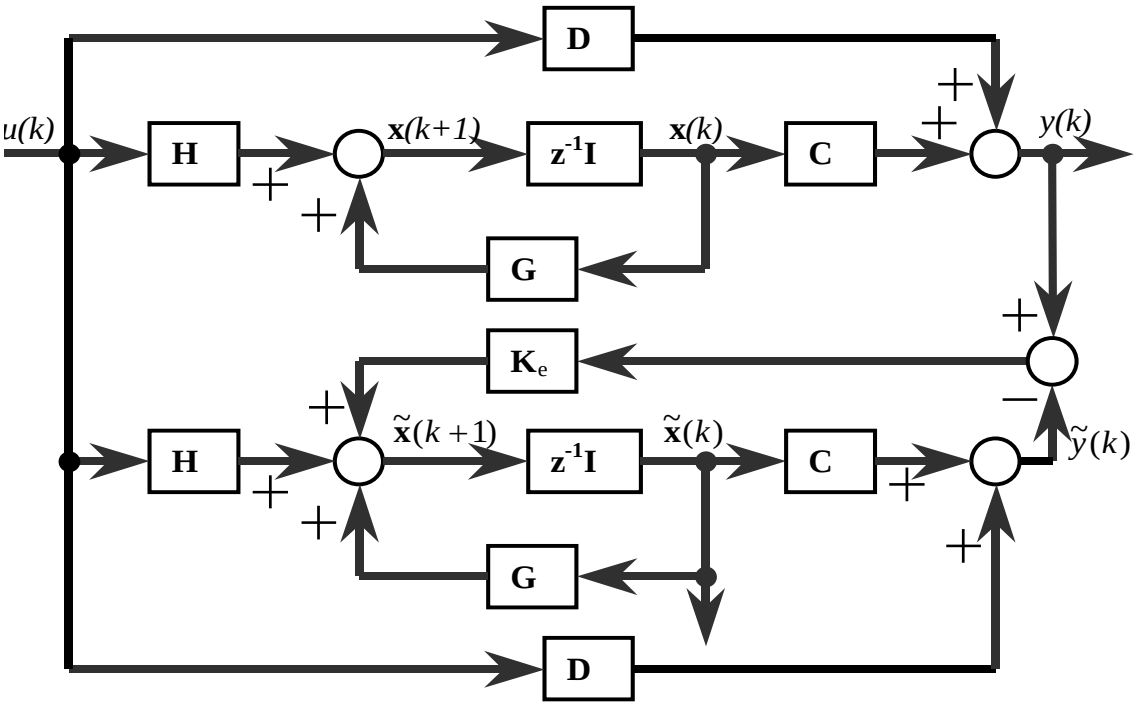
Teniendo en cuenta que las señales y*(k)* e $\tilde{\mathbf{y}}(k)$ se pueden medir por lo tanto la ecuación dinámica del observador de estados, se escribe como

$$\tilde{\mathbf{x}}(k+1) = \mathbf{G}\tilde{\mathbf{x}}(k) + \mathbf{H}\mathbf{u}(k) + \mathbf{K}_e(\mathbf{y}(k) - \tilde{\mathbf{y}}(k)) \qquad (3.9)$$

$$\tilde{\mathbf{y}}(k) = \mathbf{C}\tilde{\mathbf{x}}(k) + \mathbf{D}\mathbf{u}(k) \qquad (3.10)$$

donde K_e es una matriz de ponderación de estados de dimensión *n×m.*

Las ecuaciones (3.1), (3.2), (3.9) y (3.10) se representan en un diagrama de bloques en la figura 24



Sistema de control con estimador predicción de estados.

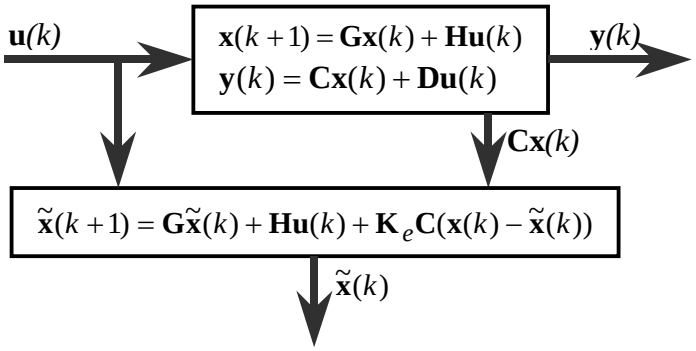
Fig.30

Reemplazando las ecuaciones (3.2) y (3.10) en la ecuación (3.9) y operando se obtiene

$$\tilde{\mathbf{x}}(k+1) = \mathbf{G}\tilde{\mathbf{x}}(k) + \mathbf{H}\mathbf{u}(k) + \mathbf{K}_e(\mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) - \mathbf{C}\tilde{\mathbf{x}}(k) - \mathbf{D}\mathbf{u}(k))$$

$$\tilde{\mathbf{x}}(k+1) = \mathbf{G}\tilde{\mathbf{x}}(k) + \mathbf{H}\mathbf{u}(k) + \mathbf{K}_e\mathbf{C}(\mathbf{x}(k) - \tilde{\mathbf{x}}(k)) \qquad (3.11)$$

El sistema de control representado y expresado mediante las ecuaciones (3.1), (3.2) y (3.11) se puede representar mediante un esquema simplificado en la figura



Sistema de control reducido con estimador de estados.

Fig. 31

Para determinar los valores de la matriz K_e se obtiene la ecuación de error del observador restando la ecuación (3.1) de la ecuación (3.11) y se obtiene

$$\mathbf{x}(k+1) - \tilde{\mathbf{x}}(k+1) = \mathbf{G}(\mathbf{x}(k) - \tilde{\mathbf{x}}(k)) - \mathbf{K}_e\mathbf{C}(\mathbf{x}(k) - \tilde{\mathbf{x}}(k)) \qquad (3.12)$$

Definiendo un vector e*(k)* que representa la diferencia entre x*(k)* y $\tilde{\mathbf{x}}(k)$

$$\mathbf{e}(k) = \mathbf{x}(k) - \tilde{\mathbf{x}}(k)$$

Empleando el vector *e(k)* en la ecuación (3.12) se determina la ecuación siguiente

$$\mathbf{e}(k+1) = \mathbf{G}\mathbf{e}(k) - \mathbf{K}_e\mathbf{C}\mathbf{e}(k)$$

$$\mathbf{e}(k+1) = (\mathbf{G} - \mathbf{K}_e\mathbf{C})\mathbf{e}(k) \qquad (3.13)$$

La ecuación (3.13) describe la dinámica del observador por lo que se debe diseñar la ganancia K_e de forma muy apropiada para que el vector del error e*(k)* tienda asintóticamente a cero a una velocidad lo suficientemente rápido. Para determinar el valor de K_e se puede utilizar el método de la ubicación de polos de tal manera que los valores propios de la matriz G-K_eC sean estables.

$$\left|z\mathbf{I} - \mathbf{G} + \mathbf{K}_e\mathbf{C}\right| = (z-\mu_1)(z-\mu_2)\cdots(z-\mu_n) \qquad (3.14)$$

Sistema dual: Mediante el principio de "*no unicidad de la representación en el espacio de estado"* de la sección 2.3 se puede representar un sistema de control *dual* al sistema original descrito por las ecuaciones (3.1) y (3.2) como

$$\hat{\mathbf{x}}(k+1) = \mathbf{G}^T \hat{\mathbf{x}}(k) + \mathbf{C}^T \hat{\mathbf{u}}(k) \tag{3.15}$$

$$\hat{\mathbf{y}}(k) = \mathbf{H}^T \hat{\mathbf{x}}(k) + \mathbf{D}\hat{\mathbf{u}}(k) \tag{3.16}$$

Las matrices G, H, C, D son las mismas que las matrices de las ecuaciones (3.1) y (3.2). Considerando al sistema descrito por las ecuaciones (3.15) y (3.16) como sistema de regulación definido, entonces la señal de control se expresa como

$$\hat{\mathbf{u}}(k) = -\mathbf{K}\hat{\mathbf{x}}(k) \tag{3.17}$$

Reemplazando la ecuación (3.17) en la ecuación (3.15) se obtiene

$$\hat{\mathbf{x}}(k+1) = (\mathbf{G}^T - \mathbf{C}^T\mathbf{K})\hat{\mathbf{x}}(k) \tag{3.18}$$

El polinomio característico del sistema dado por la ecuación (3.18) se da por

$$\left| z\mathbf{I} - \mathbf{G}^T + \mathbf{C}^T\mathbf{K} \right| = (z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_n) \tag{3.19}$$

Aplicando la propiedad del determinante $|A| = |A^T|$ entonces la ecuación (3.19) también se expresa como

$$\left| z\mathbf{I} - \mathbf{G} + \mathbf{K}^T\mathbf{C} \right| = (z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_n) \tag{3.20}$$

Al comparar las ecuaciones (3.14) y (3.20) y si los polos de lazo cerrado son iguales es decir $\mu_i = \lambda_i$ entonces igualando las ecuaciones (3.14) y (3.20) se obtiene

$$\left| z\mathbf{I} - \mathbf{G} + \mathbf{K}_e\mathbf{C} \right| = \left| z\mathbf{I} - \mathbf{G} + \mathbf{K}^T\mathbf{C} \right| \tag{3.21}$$

De la ecuación (3.21) se deduce que

$$\mathbf{K}_e = \mathbf{K}^T \tag{3.22}$$

Si se calcula la matriz K usando el algoritmo de solución de un sistema de control multivariable, entonces se determina la matriz de ganancia de estimación K_e

3.7.- SISTEMAS DE CONTROLADOR ÓPTIMO.

Un sistema de control óptimo minimiza (o maximiza) el índice de desempeño seleccionado, este índice, en realidad determina la configuración del sistema. Un sistema de control que es óptimo bajo un índice de desempeño por lo general no es óptimo bajo otro índice de desempeño.

Los problemas de control óptimo que se pueden resolver en forma analítica, dan una buena visión de las estructuras y algoritmos óptimos que se pueden aplicar a casos prácticos.

3.8.-FORMULACIÓN DE UN SISTEMA DE OPTIMIZACIÓN.

El problema de un sistema de optimización de un sistema de control se puede formular si se tiene la siguiente información: Ecuaciones del sistema, clase de vectores de control permitidos, índice de desempeño y parámetros del sistema.
La solución de un problema de control óptimo consiste en determinar el vector de control óptimo dentro de la clase de vectores de control permitidos. El vector de control depende de la naturaleza del índice de desempeño, de la naturaleza de restricciones, del estado inicial o salida inicial y del estado deseado o salida deseada.
Considerando un sistema de control multivariable lineal invariante en el tiempo descrito mediante las ecuaciones de estado (3.1) y de salida (3.2) como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

Donde (*k = 0, 1, 2, ..., N-1*) y el estado inicial arbitrario es $\mathbf{x}(0) = \mathbf{c}$ y la secuencia u*(0)*, u*(1)*, u*(2)*, *...*, u*(N-1)* son los vectores de control óptimos que minimiza un índice de desempeño cuadrático.

3.9.- CONTROL ÓPTIMO CUADRÁTICO NO ESTACIONARIO

Ley de control óptimo
La ley de control óptimo para un sistema de regulación se determina mediante la ecuación

$$\mathbf{u}(k) = -\mathbf{K}(k)\mathbf{x}(k) \qquad (3.23)$$

Donde K*(k)* es una matriz de *r*×*n* variante en el tiempo. Si *N* = ∞, entonces K*(k)* es una matriz constante de *r*×*n.*

3.9.1.-INDICE DE DESEMPEÑO CUADRÁTICO.

En el problema de control óptimo cuadrático se desea determinar una ley para el vector de control u*(k)* tal que un índice de desempeño cuadrático se minimice. Un ejemplo de índice de desempeño cuadrático es

$$J = \frac{1}{2}\mathbf{x}^T(N)\mathbf{S}\mathbf{x}(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)\right] \qquad (3.24)$$

Donde las matrices S y Q son matrices de ponderación simétricas definidas positivas o semidefinidas positivas y la matriz R es una matriz de ponderación, simétrica definida positiva. Estas matrices se seleccionan para valorar la importancia relativa de la contribución en el desempeño debido al vector de estado x*(k)* (*k = 0, 1, 2, ... , N-1*). Al vector de control u*(k)* (*k = 0, 1, 2, ... , N-1*) y al estado final x*(N)*, respectivamente.

El primer término dentro de los corchetes de la sumatoria toma en cuenta la importancia relativa del error durante el proceso de control y el segundo término toma en cuenta el gasto de energía de la señal de control.

Al emplear un conjunto de multiplicadores de Lagrange λ*(1),* λ*(2), ...,* λ*(N)*, se define un nuevo índice de desempeño *L* como:

$$L = J + \frac{1}{2}\sum_{k=0}^{N-1}\left\{\lambda^T(k+1)\left[\mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) - \mathbf{x}(k+1)\right] + \left[\mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) - \mathbf{x}(k+1)\right]^T\boldsymbol{\lambda}(k+1)\right\} \qquad (3.25)$$

Donde J está expresado mediante la ecuación (4.24)

$$J = \frac{1}{2}\mathbf{x}^T(N)\mathbf{S}\mathbf{x}(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)\right]$$

Para minimizar la función *L*, se necesita diferenciar L respecto a cada uno de los componentes de los vectores x*(k)*, u*(k)* y λ*(k)* e igualar los resultados a cero. Por lo tanto se tiene

$$\frac{\partial \mathbf{L}}{\partial \mathbf{x}(k)} = \mathbf{0} \;\Rightarrow\; \mathbf{Q}\mathbf{x}(k) + \mathbf{G}^T\boldsymbol{\lambda}(k+1) - \boldsymbol{\lambda}(k) = \mathbf{0}, \quad k = 1, 2, \cdots, N-1 \qquad (3.26)$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{x}(N)} = \mathbf{0} \;\Rightarrow\; \mathbf{S}\mathbf{x}(N) - \boldsymbol{\lambda}(N) = \mathbf{0} \qquad (3.27)$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}(k)} = \mathbf{0} \;\Rightarrow\; \mathbf{R}\mathbf{u}(k) + \mathbf{H}^T\boldsymbol{\lambda}(k+1) = \mathbf{0}, \quad k = 0, 1, 2, \cdots, N-1 \qquad (3.28)$$

$$\frac{\partial \mathbf{L}}{\partial \boldsymbol{\lambda}(k)} = \mathbf{0} \;\Rightarrow\; \mathbf{G}\mathbf{x}(k-1) + \mathbf{H}\mathbf{u}(k-1) - \mathbf{x}(k) = \mathbf{0}, \quad k = 1, 2, \cdots, N \tag{3.29}$$

Modificando adecuadamente la ecuación (3.26) se tiene

$$\boldsymbol{\lambda}(k) = \mathbf{Q}\mathbf{x}(k) + \mathbf{G}^T \boldsymbol{\lambda}(k+1), \quad k = 1, 2, \cdots, N-1 \tag{3.30}$$

Con la condición final $\boldsymbol{\lambda}(N) = \mathbf{S}\mathbf{x}(N)$

Al resolver la ecuación (3.28) para u*(k)* y sabiendo que R^{-1} existe, se obtiene

$$\mathbf{u}(k) = -\mathbf{R}^{-1}\mathbf{H}^T \boldsymbol{\lambda}(k+1), \quad k = 0, 1, 2, \cdots, N-1 \tag{3.31}$$

La ecuación (3.29) resulta ser la ecuación de estado y se puede escribir como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k), \quad k = 0, 1, 2, \cdots, N-1 \tag{3.32}$$

Al sustituir la ecuación (3.31) en la ecuación (3.32) se obtiene

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) - \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T \boldsymbol{\lambda}(k+1) \tag{4.33}$$

Con la condición inicial $\mathbf{x}(0) = \mathbf{c}$.

Para obtener la solución al problema de minimización, se necesitan resolver las ecuaciones (3.30) y (3.33) en forma simultánea con dos puntos de frontera $\boldsymbol{\lambda}(N) = \mathbf{S}\mathbf{x}(N)$ y $\mathbf{x}(0) = \mathbf{c}$ respectivamente determinando el valor óptimo para el vector x*(k)* y para el vector λ*(k)* y el vector de control óptimo u*(k)* se puede obtener en la forma de lazo abierto. Sin embargo, si se emplea la ecuación de Riccati, el vector u*(k)* se puede obtener en la forma de lazo cerrado como en la ecuación (3.23) es decir

$$\mathbf{u}(k) = -\mathbf{K}(k)\mathbf{x}(k)$$

Si se supone que el vector de multiplicadores de Lagrange λ*(k)* se puede aplicar la transformación de Riccati al definir λ*(k)* como

$$\boldsymbol{\lambda}(k) = \mathbf{P}(k)\mathbf{x}(k) \tag{3.34}$$

Donde P*(k)* es una matriz simétrica definida positiva o semidefinida positiva de dimensión *n*×*n*.

Al sustituir la ecuación (3.34) en la ecuación (3.30) resulta

$$\mathbf{P}(k)\mathbf{x}(k) = \mathbf{Q}\mathbf{x}(k) + \mathbf{G}^T \mathbf{P}(k+1)\mathbf{x}(k+1) \tag{3.35}$$

Al sustituir la ecuación (3.34) en la ecuación (3.33) se obtiene

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) - \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T \mathbf{P}(k+1)\mathbf{x}(k+1) \tag{3.36}$$

La ecuación (3.36) se puede escribir como

$$\left[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T \mathbf{P}(k+1)\right]\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) \tag{3.37}$$

La ecuación (3.37) se convierte en

$$\mathbf{x}(k+1) = \left[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)\right]^{-1}\mathbf{G}\mathbf{x}(k) \tag{3.38}$$

Al sustituir la ecuación (3.38) en la ecuación (3.35), se obtiene

$$\mathbf{P}(k)\mathbf{x}(k) = \mathbf{Q}\mathbf{x}(k) + \mathbf{G}^T\mathbf{P}(k+1)[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)]^{-1}\mathbf{G}\mathbf{x}(k) \tag{3.39}$$

La ecuación (3.39) también se puede escribir como

$$\left\{\mathbf{P}(k) - \mathbf{Q} - \mathbf{G}^T\mathbf{P}(k+1)[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)]^{-1}\mathbf{G}\right\}\mathbf{x}(k) = \mathbf{0} \tag{3.40}$$

En la última ecuación se debe cumplir para todo vector x*(k)*. Por lo tanto se obtiene

$$\mathbf{P}(k) = \mathbf{Q} + \mathbf{G}^T\mathbf{P}(k+1)[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)]^{-1}\mathbf{G} \tag{3.41}$$

La ecuación (3.41) se puede modificar al utilizar el lema de inversión de matrices

$$(\mathbf{A} + \mathbf{B}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{I} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{D}\mathbf{A}^{-1}$$

Al hacer las sustituciones

$$\mathbf{A} = \mathbf{I}\ , \quad \mathbf{B} = \mathbf{H}\mathbf{R}^{-1}\ , \quad \mathbf{D} = \mathbf{H}^T\mathbf{P}(k+1)$$

Se obtiene

$$[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)]^{-1} = \mathbf{I} - \mathbf{H}\mathbf{R}^{-1}[\mathbf{I} + \mathbf{H}^T\mathbf{P}(k+1)\mathbf{H}\mathbf{R}^{-1}]^{-1}\mathbf{H}^T\mathbf{P}(k+1)$$

$$[\mathbf{I} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)]^{-1} = \mathbf{I} - \mathbf{H}[\mathbf{R} + \mathbf{H}^T\mathbf{P}(k+1)\mathbf{H}]^{-1}\mathbf{H}^T\mathbf{P}(k+1)$$

En consecuencia la ecuación (3.41) se puede modificar a

$$\mathbf{P}(k) = \mathbf{Q} + \mathbf{G}^T\mathbf{P}(k+1)\mathbf{G} - \mathbf{G}^T\mathbf{P}(k+1)\mathbf{H}[\mathbf{R} + \mathbf{H}^T\mathbf{P}(k+1)\mathbf{H}]^{-1}\mathbf{H}^T\mathbf{P}(k+1)\mathbf{G} \tag{3.42}$$

La ecuación (3.42) se denomina ecuación de Riccati.

En referencia a las ecuaciones (3.27) y (3.34), para el valor de *k* = *N* se obtiene $\mathbf{P}(N)\mathbf{x}(N) = \boldsymbol{\lambda}(N) = \mathbf{S}\mathbf{x}(N)$. Entonces $\mathbf{P}(N) = \mathbf{S}$

Mediante la ecuación (3.34) se puede obtener

$$\boldsymbol{\lambda}(k+1) = \mathbf{P}(k+1)\mathbf{x}(k+1) \tag{3.43}$$

Reemplazando la ecuación de estado (3.32) en la ecuación (3.43)

$$\boldsymbol{\lambda}(k+1) = \mathbf{P}(k+1)\left[\mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)\right] \tag{3.44}$$

Sustituyendo la ecuación (3.44) en la ecuación de control (3.31) se obtiene

$$\mathbf{u}(k) = -\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)\mathbf{G}\mathbf{x}(k) - \mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)\,\mathbf{H}\mathbf{u}(k) \tag{3.45}$$

Despejando el vector de control óptimo u*(k)* obteniéndose como

$$\mathbf{u}(k) = -[\mathbf{I} + \mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)\,\mathbf{H}]^{-1}\mathbf{R}^{-1}\mathbf{H}^T\mathbf{P}(k+1)\mathbf{G}\mathbf{x}(k) \tag{3.46}$$

Modificando la ecuación (3.46) se obtiene la siguiente ley de control óptima como

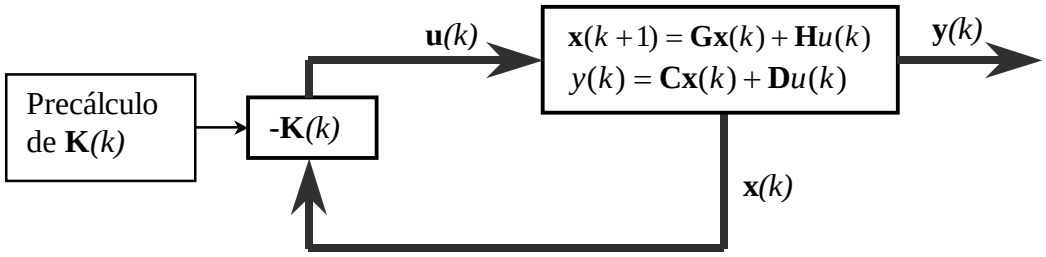
$$\mathbf{u}(k) = -[\mathbf{R} + \mathbf{H}^T \mathbf{P}(k+1)\mathbf{H}]^{-1}\mathbf{H}^T \mathbf{P}(k+1)\mathbf{G}\mathbf{x}(k) \qquad (3.47)$$

Relacionado la ecuación (3.23) con la ecuación (3.47) se obtiene la expresión para la matriz de realimentación K*(k)* como sigue

$$\mathbf{K}(k) = [\mathbf{R} + \mathbf{H}^T \mathbf{P}(k+1)\mathbf{H}]^{-1}\mathbf{H}^T \mathbf{P}(k+1)\mathbf{G} \qquad (3.48)$$

La ecuación (3.48) indica que la matriz de realimentación K*(k)* variante en el tiempo se puede calcular antes de que el proceso comience, conociendo las matrices G, H, Q, R y S y el valor de *k=N* [CA01].

Existen otras formas equivalentes para representar la matriz de realimentación K*(k)*. En la figura 26 se muestra el esquema de control óptimo del sistema regulador no estacionario basado en el índice de desempeño cuadrático.



Sistema regulador óptimo basado en el índice de desempeño cuadrático.

Fig.32

Al realizar manipulaciones se puede evaluar el valor mínimo del índice de desempeño. Es decir

$$\min(J) = \min\left\{\frac{1}{2}\mathbf{x}^T(N)\mathbf{S}\mathbf{x}(N) + \frac{1}{2}\sum_{k=0}^{N-1}\left[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)\right]\right\} \qquad (3.49)$$

Obteniéndose el valor mínimo del índice de desempeño *J* como

$$J_{\min} = \frac{1}{2}\mathbf{x}^T(0)\mathbf{P}(0)\mathbf{x}(0) \qquad (3.50)$$

3.10.- CONTROL OPTIMO CUADRÁTICO EN ESTADO ESTACIONARIO SE UN SISTEMA REGULADOR.

Cuando se considera el sistema de control óptimo cuadrático con el proceso sin limitaciones, o cuando $N \to \infty$ (proceso de etapas infinitas). Como $N \to \infty$, la solución de control óptimo se convierte en una solución de estado estacionario, y la matriz de ganancia variante en el tiempo K*(k)* se convierte en una matriz de ganancia constante K y se llama matriz de ganancia en estado estacionario.

Considerando el sistema descrito por las ecuaciones (3.1) y (3.2) como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

Teniendo en cuenta el control óptimo cuadrático en estado estacionario de un sistema regulador, el índice de desempeño *J* se modifica a

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)\right] \qquad (3.51)$$

El índice de desempeño *J,* cuando $N = \infty$, converge a una constante, $x(\infty) = 0$ por lo tanto $(1/2)\,\mathbf{x}^T(\infty)\mathbf{S}\mathbf{x}(\infty) = 0$

La matriz de estado estacionario P*(k)* se define como P (matriz real simétrica definida positiva). Entonces la ecuación de Ricatti (3.42) se convierte en

$$\mathbf{P} = \mathbf{Q} + \mathbf{G}^T\mathbf{P}\mathbf{G} - \mathbf{G}^T\mathbf{P}\mathbf{H}[\mathbf{R} + \mathbf{H}^T\mathbf{P}\mathbf{H}]^{-1}\mathbf{H}^T\mathbf{P}\mathbf{G} \qquad (3.52)$$

La matriz de ganancia en estado estacionario K se obtiene modificando la ecuación (3.48) como

$$\mathbf{K} = [\mathbf{R} + \mathbf{H}^T\mathbf{P}\,\mathbf{H}]^{-1}\mathbf{H}^T\mathbf{P}\mathbf{G} \qquad (3.53)$$

La ley de control óptimo para la operación en estado estacionario está dada por

$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k) \qquad (3.54)$$

El índice de desempeño *J* asociando la ley de control óptimo en estado estacionario se obtiene de la ecuación (3.50) como

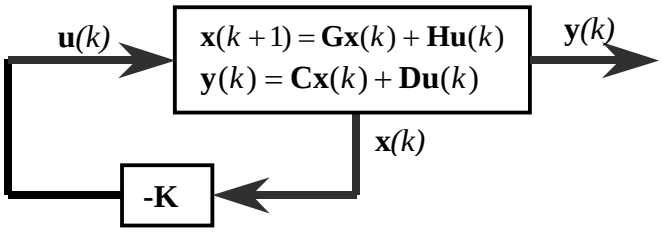
$$J_{\min} = \frac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0) \qquad (3.55)$$

Al implantar el controlador óptimo en estado estacionario (invariante en el tiempo) se requiere solucionar en estado estacionario la ecuación de Ricatti. Una forma es resolviendo la ecuación de Ricatti de la expresión (3.52) o también invirtiendo la dirección del tiempo de la ecuación de Ricatti (3.42) modificándose a la forma siguiente

$$\mathbf{P}(k+1) = \mathbf{Q} + \mathbf{G}^T\mathbf{P}(k)\mathbf{G} - \mathbf{G}^T\mathbf{P}(k)\mathbf{H}[\mathbf{R} + \mathbf{H}^T\mathbf{P}(k)\mathbf{H}]^{-1}\mathbf{H}^T\mathbf{P}(k)\mathbf{G} \qquad (3.56)$$

La solución de la ecuación (3.56) comienza con P*(0)* = 0, e iterar la ecuación hasta obtener una solución en estado estacionario, es decir obtener una matriz P constante.

La representación gráfica del sistema de control óptimo en estado estacionario se muestra en la figura 4.7



Sistema de control en lazo cerrado.

Fig. 33

3.11.- DISEÑO DE UN CONTROL ÓPTIMO CUADRÁTICO EN ESTADO ESTACIONARIO DE SEGUIMIENTO (PROPROCIONAL).

Sea el sistema de control de múltiple entrada descrito por las ecuaciones (3.1) y (3.2) definas como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$$

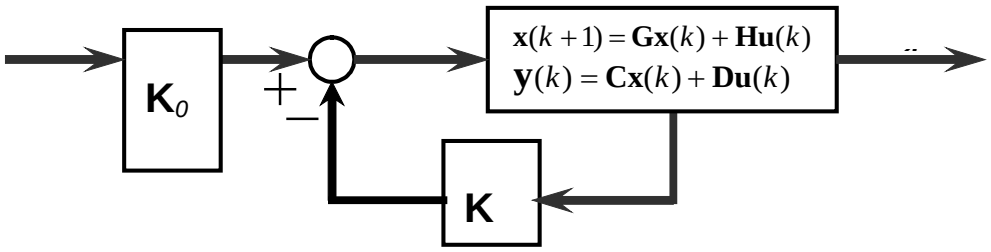
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

La variable de control u*(k)* no acotada se puede expresar como

$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k) + \mathbf{K}_0\mathbf{r}(k) \qquad (3.57)$$

Donde el vector r*(k)* es el vector de referencia *m×1* del mismo tamaño que el vector de salida y*(k)*, la matriz K es la matriz de ganancia de realimentación de estado de dimensión *r×n*, y la matriz K_0 es una matriz de compensación ajustable proporcional *r×m*.

En la figura 28 se muestra el diagrama de bloques correspondiente a las ecuaciones (3.1), (3.2), y (3.57), que representa a un sistema de control óptimo en lazo cerrado.



Sistema de control óptimo proporcional y pre-compensador.

Fig. 34

Para determinar la ganancia óptima K en estado estacionario del sistema representado en la figura 28 se debe de llevar a un sistema de regulación óptimo cuadrático alrededor de $x(k_\infty)$ evaluando al sistema de control cuando $k \to \infty$ para un sistema de control seguimiento a un escalón con pre-compensador de simple entrada obteniéndose la matriz óptima de ganancia K en estado estacionario dada en la expresión (3.53) y escrita como

$$\mathbf{K} = [\mathbf{R} + \mathbf{H}^T \mathbf{P}\, \mathbf{H}]^{-1} \mathbf{H}^T \mathbf{P}\mathbf{G}$$

Donde la matriz P se obtiene de la matriz de Ricatti dada en la ecuación (3.52) y volviéndola a escribir como

$$\mathbf{P} = \mathbf{Q} + \mathbf{G}^T \mathbf{P}\mathbf{G} - \mathbf{G}^T \mathbf{P}\mathbf{H}[\mathbf{R} + \mathbf{H}^T \mathbf{P}\mathbf{H}]^{-1} \mathbf{H}^T \mathbf{P}\mathbf{G}$$

La solución de la ecuación de Ricatti en estado estacionario se obtiene mediante la ecuación (3.56) como

$$\mathbf{P}(k+1) = \mathbf{Q} + \mathbf{G}^T \mathbf{P}(k)\mathbf{G} - \mathbf{G}^T \mathbf{P}(k)\mathbf{H}[\mathbf{R} + \mathbf{H}^T \mathbf{P}(k)\mathbf{H}]^{-1} \mathbf{H}^T \mathbf{P}(k)\mathbf{G}$$

La ganancia de prealimentación K_0 se selecciona para que el sistema en lazo cerrado tenga una ganancia unitaria en estado permanente, eliminado el error en estado estacionario ante una entrada escalón, $r(k)$=r. Obteniéndose como en la ecuación (3.41) pero en forma matricial como

$$\mathbf{K}_0 = \left((\mathbf{C} - \mathbf{D}\mathbf{K})\left[\mathbf{I} - \mathbf{G} + \mathbf{H}\mathbf{K}\right]^{-1}\mathbf{H} + \mathbf{D}\right)^{-1} \qquad (3.58)$$

3.12.- DISEÑO DE UN CONTROL ÓPTIMO CUADRÁTICO EN ESTADO ESTACIONARIO DE SEGUIMIENTO (PROPORCIONAL-INTEGRAL).

Sea el sistema de control de múltiple entrada descrito por las ecuaciones (3.1) y (3.2) definas como

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k), \quad \mathbf{D} = \mathbf{0}$$

La variable de control u*(k)* no acotada se puede expresar como

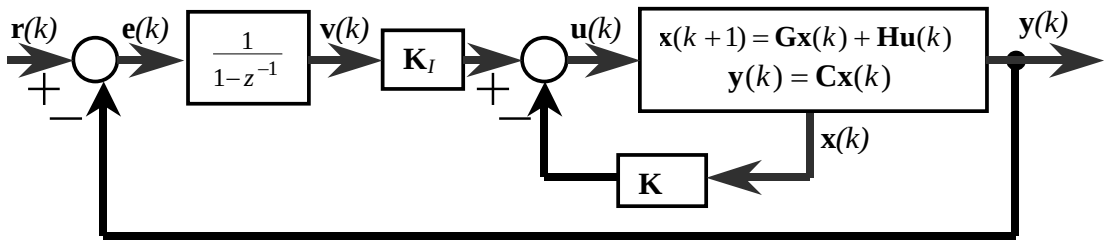
$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k) + \mathbf{K}_I\mathbf{v}(k) \tag{3.59}$$

Donde el vector v*(k)* es un vector de *m×1* del mismo tamaño que el vector de salida y*(k)*, la matriz K es la matriz de ganancia de realimentación de estado de dimensión *r×n*, y la matriz K_I es una matriz de ganancia integral de *r×m*.

La ecuación del integrador se escribe como

$$\mathbf{v}(k) = \mathbf{v}(k-1) + (\mathbf{r}(k) - \mathbf{y}(k)) \tag{3.60}$$

En la figura 4.9 se muestra el diagrama de bloques correspondiente a las ecuaciones (3.1), (3.2), (3.59) y (3.60) que representa un sistema de control óptimo en lazo cerrado.



Sistema de control óptimo de seguimiento proporcional – integral.

FIG 35

La ecuación (3.60) en base a las ecuaciones (3.1) y (3.2) se puede modificar como

$$\mathbf{v}(k+1) = -\mathbf{C}\mathbf{G}\mathbf{x}(k) + \mathbf{v}(k) - \mathbf{C}\mathbf{H}\mathbf{u}(k) + \mathbf{r}(k+1) \tag{3.61}$$

De las ecuaciones (3.1), (3.2) y (3.61) se obtiene un sistema de control óptimo ampliado como sigue:

Ecuación de estado

$$\begin{bmatrix}\mathbf{x}(k+1)\\ \mathbf{v}(k+1)\end{bmatrix} = \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ -\mathbf{C}\mathbf{G} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{v}(k)\end{bmatrix} + \begin{bmatrix}\mathbf{H}\\ -\mathbf{C}\mathbf{H}\end{bmatrix}\mathbf{u}(k) + \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{r}(k+1) \tag{3.62}$$

Ecuación de salida

$$\mathbf{y}(k) = \begin{bmatrix}\mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{v}(k)\end{bmatrix} \tag{3.63}$$

Ecuación de control

$$\mathbf{u}(k) = -\begin{bmatrix}\mathbf{K} & -\mathbf{K}_I\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{v}(k)\end{bmatrix} \tag{3.64}$$

Definiendo las siguientes expresiones

$\hat{\mathbf{x}}(k) = \begin{bmatrix}\mathbf{x}(k)\\ \mathbf{v}(k)\end{bmatrix}$ de dimensión *(n+m)×1*

$\hat{\mathbf{G}} = \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ -\mathbf{CG} & \mathbf{I}\end{bmatrix}$ de dimensión *(n+m)×(n+m)*

$\hat{\mathbf{H}} = \begin{bmatrix}\mathbf{H}\\ -\mathbf{CH}\end{bmatrix}$ de dimensión *(n+m)×r*

$\hat{\mathbf{H}} = \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}$ de dimensión *(n+m)×r*

$\hat{\mathbf{C}} = \begin{bmatrix}\mathbf{C} & \mathbf{0}\end{bmatrix}$ de dimensión *m×(n+m)*

$\hat{\mathbf{D}} = \mathbf{0}$ de dimensión *m×r*

$\hat{\mathbf{K}} = \begin{bmatrix}\mathbf{K} & -\mathbf{K}_I\end{bmatrix}$ de dimensión *r×(n+m)*

Las ecuaciones (3.62), (3.63) y (3.64) se pueden escribir como

$$\hat{\mathbf{x}}(k+1) = \hat{\mathbf{G}}\hat{\mathbf{x}}(k) + \hat{\mathbf{H}}\mathbf{u}(k) + \hat{\mathbf{H}}\mathbf{r}(k+1) \tag{3.65}$$

$$\mathbf{y}(k) = \hat{\mathbf{C}}\hat{\mathbf{x}}(k) + \hat{\mathbf{D}}\mathbf{r}(k) \tag{3.66}$$

$$\mathbf{u}(k) = -\hat{\mathbf{K}}\hat{\mathbf{x}}(k) \tag{3.67}$$

Para diseñar la ganancia óptima en estado estacionario $\hat{\mathbf{K}}$ del sistema de control descrito por las ecuaciones (3.65), (3.66) y (3.67) se transforma a un sistema de regulación alrededor de $\hat{\mathbf{x}}(k_\infty)$ evaluando al sistema de control cuando $k\to\infty$ como sigue

$$\hat{\mathbf{x}}(k_\infty+1) = \hat{\mathbf{G}}\hat{\mathbf{x}}(k_\infty) + \hat{\mathbf{H}}\mathbf{u}(k_\infty) + \hat{\mathbf{H}}\mathbf{r}(k_\infty+1) \tag{3.68}$$

$$\mathbf{u}(k_\infty) = -\hat{\mathbf{K}}\hat{\mathbf{x}}(k_\infty) \tag{3.69}$$

Si se define la variación del estado actual con el estado en el infinito del sistema como

$$\hat{\mathbf{x}}_e(k) = \hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k_\infty), \qquad \mathbf{u}_e(k) = \mathbf{u}(k) - \mathbf{u}(k_\infty)$$

La entrada *r(k)* de referencia se considera un escalón por lo tanto se obtiene

$$\mathbf{r}(k) = \mathbf{r}(k+1) = \mathbf{r}(k_\infty) = \mathbf{r} \tag{3.70}$$

Restando las ecuaciones (3.65) y (3.68) se obtiene

$$\hat{\mathbf{x}}(k+1)-\hat{\mathbf{x}}(k_\infty+1)=\hat{\mathbf{G}}(\hat{\mathbf{x}}(k)-\hat{\mathbf{x}}(k_\infty))+\hat{\mathbf{H}}(u(k)-u(k_\infty))+\hat{\overline{\mathbf{H}}}(r(k+1)-r(k_\infty+1)) \tag{3.71}$$

Evaluando con las variaciones de estados y entrada se obtiene

$$\hat{\mathbf{x}}_e(k+1)=\hat{\mathbf{G}}\hat{\mathbf{x}}_e(k)+\hat{\mathbf{H}}\mathbf{u}_e(k) \tag{3.72}$$

Restando las ecuaciones (3.67) y (3.69) se obtiene

$$\mathbf{u}(k)-\mathbf{u}(k_\infty)=-\hat{\mathbf{K}}(\hat{\mathbf{x}}(k)-\hat{\mathbf{x}}(k_\infty)) \tag{3.73}$$

Evaluando con las variaciones de estados y entrada se obtiene

$$\mathbf{u}_e(k)=-\hat{\mathbf{K}}\hat{\mathbf{x}}_e(k) \tag{3.74}$$

Las ecuaciones (3.72) y (3.74) representan un sistema de regulación de un sistema de control óptimo alrededor de $x(k_\infty)$. Por lo tanto, el diseño se convierte en determinar la matriz $\hat{\mathbf{K}}$ tal que minimice al siguiente índice de desempeño cuadrático discreto en estado estacionario J como sigue

$$J=\frac{1}{2}\sum_{k=0}^{\infty}\left[\hat{\mathbf{x}}_e^T(k)\hat{\mathbf{Q}}\hat{\mathbf{x}}_e(k)+\mathbf{u}_e^T(k)\hat{\mathbf{R}}\mathbf{u}_e(k)\right] \tag{3.75}$$

La matriz de ponderación $\hat{\mathbf{Q}}$, es simétrica definida positiva o semi definida positiva de $(n+m)\times(n+m)$ y la matriz de ponderación $\hat{\mathbf{R}}$, es simétrica y siempre definida positiva de $r\times r$.

Por lo tanto la matriz de ganancia óptima en estado estacionario $\hat{\mathbf{K}}$ de la ecuación (3.74) de control óptimo del sistema de control descrito por la ecuación que minimiza el índice de desempeño *J* descrito por la ecuación (3.76) se determina mediante la siguiente expresión

$$\hat{\mathbf{K}}=[\hat{\mathbf{R}}+\hat{\mathbf{H}}^T\hat{\mathbf{P}}\,\hat{\mathbf{H}}]^{-1}\hat{\mathbf{H}}^T\hat{\mathbf{P}}\hat{\mathbf{G}} \tag{3.76}$$

Donde la matriz $\hat{\mathbf{P}}$ sigue siendo una matriz simétrica definida positiva de dimensión $(n+m)\times(n+m)$ se obtiene de la matriz de Ricatti descrita como

$$\hat{\mathbf{P}}=\hat{\mathbf{Q}}+\hat{\mathbf{G}}^T\hat{\mathbf{P}}\hat{\mathbf{G}}-\hat{\mathbf{G}}^T\hat{\mathbf{P}}\hat{\mathbf{H}}[\hat{\mathbf{R}}+\hat{\mathbf{H}}^T\hat{\mathbf{P}}\hat{\mathbf{H}}]^{-1}\hat{\mathbf{H}}^T\hat{\mathbf{P}}\hat{\mathbf{G}} \tag{3.77}$$

La solución de la ecuación de Ricatti se obtiene mediante la ecuación siguiente

$$\hat{\mathbf{P}}(k+1)=\hat{\mathbf{Q}}+\hat{\mathbf{G}}^T\hat{\mathbf{P}}(k)\hat{\mathbf{G}}-\hat{\mathbf{G}}^T\hat{\mathbf{P}}(k)\hat{\mathbf{H}}[\hat{\mathbf{R}}+\hat{\mathbf{H}}^T\hat{\mathbf{P}}(k)\hat{\mathbf{H}}]^{-1}\hat{\mathbf{H}}^T\hat{\mathbf{P}}(k)\hat{\mathbf{G}} \tag{3.78}$$

Al reemplazar la ecuación (3.67) en la ecuación (3.65) se obtiene ecuación de estado el lazo cerrado como

$$\hat{\mathbf{x}}(k+1)=\hat{\mathbf{G}}\hat{\mathbf{x}}(k)-\hat{\mathbf{H}}\hat{\mathbf{K}}\hat{\mathbf{x}}(k)+\hat{\overline{\mathbf{H}}}\mathbf{r}(k+1)$$

$$\hat{\mathbf{x}}(k+1) = (\hat{\mathbf{G}} - \hat{\mathbf{H}}\hat{\mathbf{K}})\hat{\mathbf{x}}(k) + \hat{\mathbf{H}}\mathbf{r}(k) \tag{3.79}$$

En un sistema lineal invariante en el tiempo si se aplica una entrada constante, los estados del sistema llegan a un punto de equilibrio en estado permanente, es decir

$$\hat{\mathbf{x}}(k_\infty + 1) = \hat{\mathbf{x}}(k_\infty) \Rightarrow \mathbf{v}(k_\infty + 1) = \mathbf{v}(k_\infty) \tag{3.80}$$

Teniendo en cuenta las ecuaciones (3.60) y (3.80) se demuestra que el error en estado estacionario es cero ante una entrada tipo escalón del siguiente modo:

$$\mathbf{v}(k_\infty + 1) = \mathbf{v}(k_\infty) = \mathbf{v}(k_\infty) + \mathbf{e}(k_\infty) \rightarrow \mathbf{e}(k_\infty) = \mathbf{y}(k_\infty) - \mathbf{r} = \mathbf{0} \rightarrow \mathbf{y}(k_\infty) = \mathbf{r}$$

4.- DINÁMICA LONGITUDINAL DEL AVIÓN

4.1.-INTRODUCCIÓN

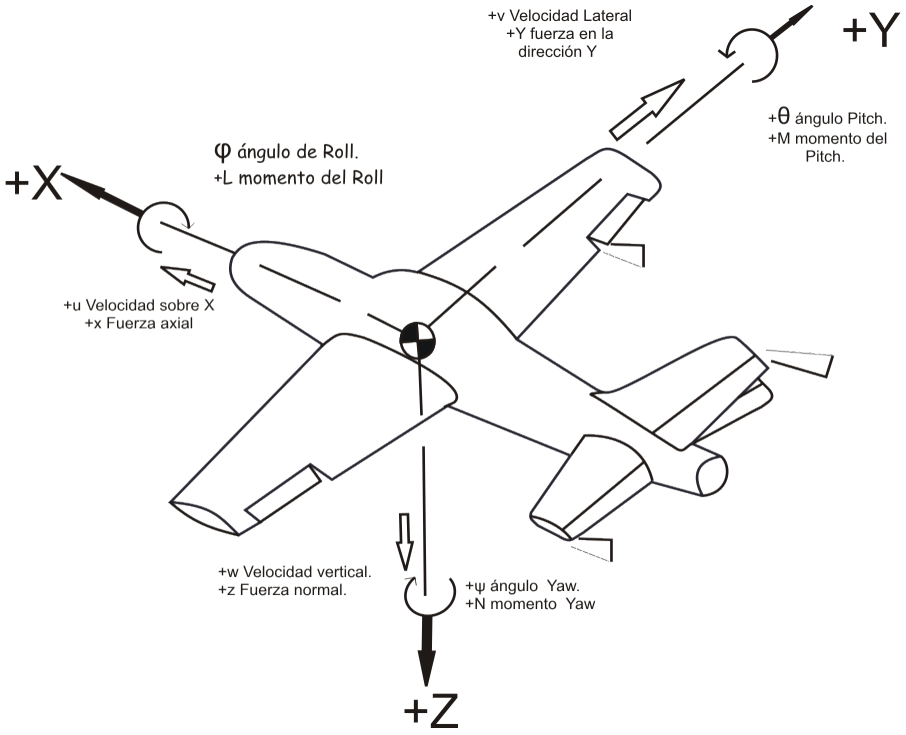
Es necesario conocer el modelo matemático del avión y realizar las pruebas de control en el simulador. Esto permitirá estudiar el comportamiento de los diferentes sistemas de control, así como las características cualitativas de las respuestas del avión en diferentes condiciones de vuelo. Estos modelos en general no lineales me permiten estudiar su comportamiento en un régimen pequeño de condiciones de vuelo. Los modelos obtenidos me permite ajustar los coeficientes con las series de Taylor.

4.2.- SISTEMA DE REFERENCIA DEL VIENTO.

Este sistema tiene su eje longitudinal X paralelo al vector velocidad del avión. De este modo existen fuerza perpendiculares a la dirección del viento, que son las de sustentación o lift, y fuerzas paralelas a la dirección del viento llamadas arrastre o drag [Springer01].

4.3 SISTEMAS DE REFERENCIA DEL AVIÓN

Su origen se encuentra en el centro de gravedad del avión, su eje longitudinal +u es paralelo al fuselaje, el eye +v apunta en la dirección del ala derecha, y el eje +w hacia la parte inferior del avión. Existe una única fuerza apreciable que se desprende de este sistema es la propulsión o thrust. El ángulo entre +u y la proyección de X sobre +u+v es el ángulo de ataque, es el ángulo que el sistema forma con el de la tierra y se conoce el ángulo de cabeceo o pitch [Springer01].



Sistema de referencia del avión.

FIG.36

4.4.- DESCRIPCIÓN DE LAS FUERZAS INVOLUCRADAS EN EL VUELO.

4.4.1.- FUERZA DE GRAVEDAD.

Como la fuerza de gravedad deriva de la energía potencial gravitatoria, disminuye a la medida que nos alejamos del centro de la tierra y siempre apunta hacia él. Además esta en función de la masa del avión, esta no genera momentos en el avión.

4.4.2.- FUERZA DE PROPULSIÓN.

Esta fuerza es generada por el motor del avión y además genera un momento sobre el centro de gravedad del avión.

4.4.3.- FUERZA AERODINÁMICA.

Son producidas por el aire alrededor de un cuerpo. Según la ecuación de Bernoulli, en un fluido ideal en régimen permanente a lo largo de una línea de corriente se obtiene:

$$\frac{p}{\rho} + g * z + \frac{V^2}{2} = cte$$

donde P es la presión, ρ es la densidad de flujo, g es la aceleración de la gravedad, z es la altura y V es la velocidad con la que se mueve el fluido. Si bien esta ecuación supone además fluido incompresible se tiene:

$$\frac{p}{\rho} + \frac{V^2}{2} = cte$$

Si se aplica la ecuación anterior por el flujo de aire que pasa por el ala del perfil del aeromodelo se obtiene la presión que la presión debajo del ala es mayor que la presión superior, la ecuación que describe la diferencia de presiones se puede calcular:

$$\Delta P = P_1 - P_2 = \rho * \frac{\left(V_2^{\,2} - V_1^{\,2}\right)}{2} = k\frac{\rho V^2}{2}$$

Donde k depende de la forma del perfil del ala, P_1 y V_1 corresponden a la parte inferior del ala y P_2 y V_2 a la parte superior, y con ello se describe la fuerza de sustentación que en ella se produce.

El análisis de fuerzas y los momentos aerodinámicos se consideran generalmente parámetros adimensionales llamados coeficientes de fuerzas y de momentos C_L y $C_{D.}$ De este modo se obtiene que las fuerzas aerodinámicas se modelan según:

$$F = C_F * \frac{1}{2} * \rho * V^2 * S = C_F * q * S$$

donde q es la presión dinámica a la que se somete la superficie. De igual modo los momentos aerodinámicos se modelan según la ecuación:

$$M = C_M * \frac{1}{2} * \rho * V^2 * S * x = C_M * q * S * x$$

donde x es la distancia entre el punto en el que se calcula el momento y el centro aerodinámico, lugar donde se efectúa las fuerzas aerodinámicas [AFD98].

4.5.- DINÁMICA LONGITUDINAL DEL AVIÓN

4.5.1.- INTRODUCCIÓN

Al obtener el modelo matemático del avión, se llevaran estas ecuaciones diferenciales al matlab, luego se simulará los parámetros y con estos parámetros se deben llevar al simulador de vuelo luego para conocer más las características cualitativas de la respuesta del avión en diferentes condiciones de vuelo (modelo global).

Una característica muy importante del modelo, indispensable para la técnica de control utilizada, es que sus ecuaciones son derivadas directamente de la mecánica newtoniana básica. Es decir, la base de todo son sumatorias de fuerzas y de momentos que, sin llegar a ser muy complejas, muestran claramente las características de la dinámica del sistema. Estas ecuaciones se combinan para crear un espacio de estados no lineal que representa al sistema:

$$\dot{x} = f(x, F_{TOT}(t), M_{TOT}(t))$$

Donde $\dot{X}$ representa las variables de estado, F_{TOT} la fuerza translacional total y M_{TOT} la fuerza rotacional total. Es claro que este sistema es acoplado dado que las fuerzas y los momentos dependen de las variables de estado. Un inconveniente un poco más grande es que en algunos casos las ecuaciones que representan el sistema son implícitas, es decir, las fuerzas y los momentos no solo dependen del vector de estados sino de sus derivadas; la solución es dividir la ecuación en una parte explícita y una implícita, que generalmente resulta ser lineal, que puede ser resuelta con métodos numéricos [Springer01].

El vector de estados completo consta de doce elementos: tres componentes de velocidad lineal, tres componentes de velocidad angular, tres ángulos que definen la actitud del aeromodelo con respecto a un marco de referencia terrestre y dos coordenadas geográficas y una altura que definen la posición del aeromodelo con respecto a la tierra. Para facilitar la solución de las ecuaciones implícitas es mejor usar la velocidad real, el ángulo de ataque y el ángulo de deslizamiento lateral en lugar de las componentes del vector de velocidad lineal.

4.5.2.- LISTA DE VARIABLES DE ESTADO DEL MODELO.

Lista de las doce variables de estado del modelo con sus respectivas unidades [Springer01].

V : Velocidad real del aire [m/s].
alpha(α) : Ángulo de ataque [rad].
beta (β) : Ángulo de deslizamiento lateral [rad].
p : Tasa de rotación [rad/s].
q : Tasa de elevación [rad/s].
r : Tasa de giro [rad/s].
psi(ψ) : Ángulo de giro [rad].
theta(θ) : Ángulo de elevación [rad]
phi(.) : Ángulo de rotación [rad]
xe : Coordenada X, marco de referencia inercial [m].
ye : Coordenada Y, marco de referencia inercial [m].
H : Altura sobre el nivel del mar [m]

4.5.3.-DINÁMICA DEL AVIÓN:

Se explica la dinámica del modelo matemático, que consiste en 12 estados [Springer01].

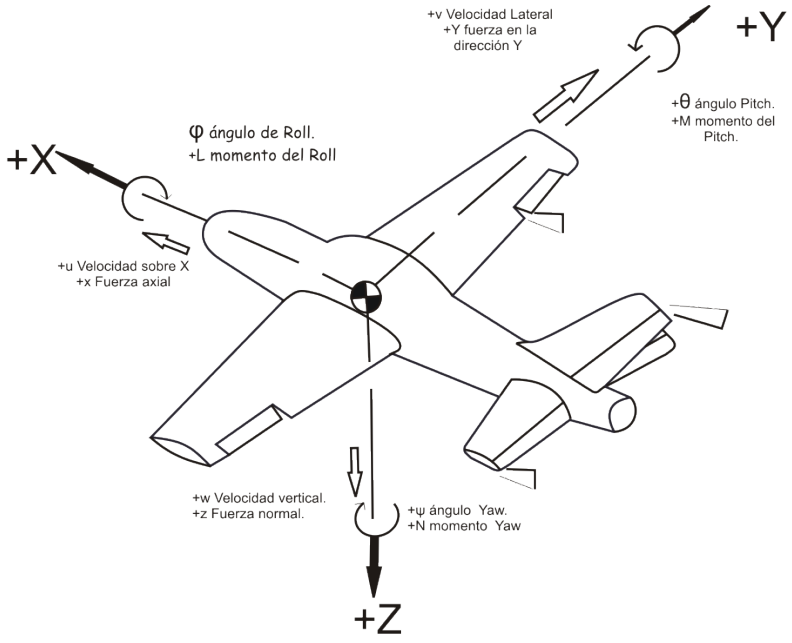
$[u, v, w, p, q, r, \phi, \theta, \psi, z', y', z']$ donde:

$[u, v, w]$ = Velocidad del fuselaje del modelo.

$[p, q, r]$ = Velocidad angular del modelo.

$[\phi, \theta, \psi]$ =Ángulo de Euler, transformación entre el modelo y la referencia terrestre (roll,pitch,yaw).

$[z', y', z']$ = Posición del avión respecto a un punto fijo terrestre.



Velocidad, fuerzas y momentos del aeromodelo.

FIG. 37

4.5.4.-ECUACIONES DINÁMICAS NO LINEALES DEL MODELO

4.5.5.-ECUACIONES DE MOVIMIENTO

$$\begin{Bmatrix} M_x \\ M_y \\ M_z \end{Bmatrix} = \begin{Bmatrix} I_x\dot{p} - I_{xz}\dot{r} \\ I_y\dot{q} \\ I_z\dot{r} - I_{xz}\dot{p} \end{Bmatrix} + \begin{Bmatrix} qr(I_z - I_y) - pqI_{xz} \\ pr(I_x - I_z) + (p^2 - r^2)I_{xz} \\ pq(I_y - I_x) + qrI_{xz} \end{Bmatrix}$$

Donde:

m = masa del aeromodelo.

[X,Y,Z] = fuerzas aerodinámicas del aeromodelo.

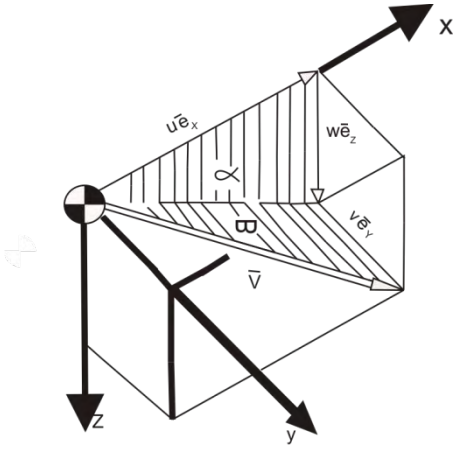
[L,M,N] = momentos aerodinámicos del aeromodelo.

I_u = momentos de inercia del aeromodelo.

T = acelerador del aeromodelo.

Los momentos aerodinámicos y las fuerzas se encuentran en función de los estados, y en función de las entradas donde: $[u, v, w, p, q, r, \delta_\alpha, \delta_e, \delta_r, \delta_t]$, donde: δ_α = deflexión del alerón, δ_e = deflexión del elevador, δ_r = deflexión de cola, δ_t =

acelerador. Las principales variables a controlar son: Velocidad del aeromodelo (V), el ángulo de ataque (α), y el ángulo de deriva (β).[Springer01].



Componentes de la velocidad.

FIG. 38

La relación entre estas componentes de la velocidad es:

$$V = \sqrt{u^2 + v^2 + w^2}$$

$$\alpha = tg^{-1}\left(\frac{w}{\sqrt{u^2 + v^2}}\right)$$

$$\beta = tg^{-1}\left(\frac{u}{u}\right)$$

El modelo se encuentra en equilibrio, a continuación se describen las Ecuaciones Longitudinales Dinámicas y las Ecuaciones Laterales-Direcionales Dinámicas.

[Springer01].

4.5.6.- ECUACIONES LONGITUDINALES DINÁMICAS

Se define el vector longitudinal de estados:

$\{x\} = \begin{bmatrix} \frac{u}{V} & \alpha & q & \theta \end{bmatrix}^T$, se describe el siguiente sistema de ecuaciones linealizadas: [Springer01].

$$[I_n]\{\dot{x}\} = [A]\{x\} + \{B_n\}\delta$$

$$[I_n] = \begin{pmatrix} V & 0 & 0 & 0 \\ 0 & (V - Z_{\dot{\alpha}}) & 0 & 0 \\ 0 & M_{\dot{\alpha}} & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$; Matriz inercial

$$[A_n] = \begin{bmatrix} VX_u & X_\alpha & 0 & -g\cos\Theta_0 \\ VZ_u & Z_\alpha & (V + Z_q) & -g\,sen\,\Theta_0 \\ VM_u & M_\alpha & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\{B_n\} = \begin{bmatrix} X_\delta & Z_\delta & M_\delta & 0 \end{bmatrix}^T$$

$$\{\dot{x}\} = [A]\{x\} + \{B\}\delta$$

$[A]$ = Matriz de la planta longitudinal.
$[B]$ = Matriz de control longitudinal.

4.5.7.-DERIVADAS DE ESTABILIDAD DIMENSIONAL

Tmo. Descripción.

$$X_u = -\frac{QS}{mV}(2C_D + M\frac{\partial C_D}{\partial M})$$

$$X_\alpha = \frac{QS}{m}\left(C_L - \frac{\partial C_D}{\partial \alpha}\right)$$

$$X_{\dot\alpha} = -\frac{QS}{m}\left(\frac{c}{2V}\right)\frac{\partial C_D}{\partial(\frac{\dot\alpha c}{2v})}$$

$$X_q = -\frac{QS}{m}\left(\frac{c}{2V}\right)\frac{\partial C_D}{\frac{\partial qc}{2v}}$$

$$X_\delta = -\frac{QS}{m}\frac{\partial C_D}{\partial \delta}$$

$$Z_u = -\frac{QS}{mV}\left(2C_L + M\frac{\partial C_L}{\partial M}\right)$$

$$Z_\alpha = -\frac{QS}{mV}(2C_L + M\frac{\partial C_L}{\partial M})$$

$$Z_{\dot\alpha} = -\frac{QS}{m}\left(\frac{c}{2V}\right)\frac{\partial C_L}{\partial(\frac{\dot\alpha c}{2v})}$$

$$Z_q = -\frac{QS}{m}\left(\frac{c}{2V}\right)\frac{\partial C_L}{\partial(\frac{qc}{2v})}$$

$$Z_\delta = -\frac{QS}{m}\frac{\partial C_L}{\partial}$$

$$M_u = \frac{QSc}{I_y V}M\frac{\partial C_m}{\partial M}$$

$$M_\alpha = \frac{QSc}{I_y}\frac{\partial C_m}{\partial M}$$

$$M_{\dot\alpha} = \frac{QSc}{I_y}\left(\frac{c}{2V}\right)\frac{\partial C_m}{\partial\left(\frac{\dot\alpha c}{2V}\right)}$$

$$M_q = \frac{QSc}{I_y}\left(\frac{c}{2V}\right)\frac{\partial C_m}{\partial\left(\frac{qc}{2V}\right)}$$

$$M_\delta = \frac{QSc}{I_y}\frac{\partial C_m}{\partial \delta}$$

[Springer01].

4.5.7.1-NOMENCLATURA DE LAS VARIABLES DIMENSIONALES.

Q = Presión dinámica.

C_D = Vector positivo con respecto a la velocidad.

C_L = Vector de velocidad normal.

M = Momento.

ά = Ratio del ángulo de ataque.

δ = Ángulo del pich.

I_y = Momento de inercia con respecto al pich.

S = Área de referencia.

b = Largo del ala del avión.

c = Cuerda del ala del avión.

V = Velocidad del avión.

α = Ángulo de ataque del ala del avión.

m = Masa del fuselaje.

u = Velocidad de perturbación.

C_m = Dirección hacia arriba de la nariz del avión.

X_u, Z_u, M_u = Derivadas en función de u/v.

$X_\alpha, Z_\alpha, M_\alpha$ = Derivadas en función de α.

X_q, Z_q, M_q = Derivadas en función de q.

4.5.8.-ECUACIONES LATERALES-DIRECIONALES DINÁMICAS

Se define el vector de estado direccional por $\{x\} = [\beta, p, \phi, r]^T$ donde el control del término δ es el manipulado, dependiendo si es alerón (δ_α) o cola (δ_r).

$$[I_n]\{\dot{x}\} = [A]\{x\} + \{B_n\}\delta$$

$$[I_n] = \begin{pmatrix} V & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{I_{xz}}{I_x} \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{I_{xz}}{I_z} & 1 & 1 \end{pmatrix}$$

$$[A_n] = \begin{bmatrix} Y_\beta & Y_P & g\cos\Theta_0 & (Y_r - V) \\ L_\beta & L_P & 0 & L_r \\ 0 & 1 & 0 & 0 \\ N_\beta & N_P & 0 & N_r \end{bmatrix}$$

$$\{B_n\} = \begin{bmatrix} Y_\delta & L_\delta & 0 & N_\delta \end{bmatrix}^T$$

$$\{\dot{x}\} = [A]\{x\} + \{B\}\delta$$

$[A]$ = Matriz lateral de dirección lateral de la planta. $[I_n]^{-1}[A_n]$

$[B]$ = Vector de control de dirección lateral de la planta. $[I_n]^{-1}[B_n]$

[Springer01].

4.5.9.-DERIVADAS DE ESTABILIDAD DIMENSIONAL

Tmo. Descripción.

$$Y_\beta = \frac{QS}{m}\frac{\partial C_y}{\partial \beta})$$

$$Y_p = \frac{QS}{m}\left(\frac{b}{2V}\right)\frac{\partial Cy}{\partial(\frac{pb}{2v})}$$

$$Y_r = \frac{QS}{m}\left(\frac{b}{2V}\right)\frac{\partial C_y}{\partial(\frac{rb}{2v})}$$

$$Y_\delta = \frac{QS}{m}\frac{\partial C_y}{\frac{\partial \delta}{2v}}$$

$$Y_\delta = -\frac{QS}{m}\frac{\partial C_D}{\partial \delta}$$

$$L_\beta = \frac{QSb}{I_x}\frac{\partial C_t}{\partial \beta}$$

$$L_p = \frac{QSb}{I_x}\left(\frac{b}{2V}\right)\frac{\partial C_t}{\partial(\frac{pb}{2v})}$$

$$L_r = -\frac{QSb}{I_x}\left(\frac{b}{2V}\right)\frac{\partial C_t}{\partial(\frac{rb}{2v})}$$

$$L_\delta = -\frac{QSb}{I_z}\frac{\partial C_t}{\partial \beta}$$

$$N_\beta = -\frac{QSb}{I_z}\frac{\partial C_n}{\partial \delta}$$

$$N_p = \frac{QSb}{I_z}\left(\frac{b}{2V}\right)\frac{\partial C_n}{\partial(\frac{pb}{2v})}$$

$$N_r = \frac{QSb}{I_z}\left(\frac{b}{2V}\right)\frac{\partial C_n}{\partial(\frac{pb}{2v})}$$

$$N_\delta = \frac{QSb}{I_z}\frac{\partial C_n}{\partial \delta}$$

[Springer01].

Aunque en general el número de reglas que describen eficientemente la dinámica del aeromodelo crece proporcionalmente al grado de precisión y detalle que se requiera, es sorprendente que para obtener un resultado satisfactorio en el sistema de control y guía, sea suficiente con un conjunto tan pequeño como el siguiente:

- Si sube un alerón (baja el otro), el avión gira en ese sentido.
- Si el avión gira, pierde sustentación.
- Si el timón se inclina a la derecha (izquierda), el avión gira hacia la derecha (izquierda).
- Si baja (sube) el elevador, aumenta (disminuye) el ángulo de ataque y aumenta (disminuye) la sustentación.
- Si los flaps suben (bajan), disminuye (aumenta) el ángulo de ataque y disminuye (aumenta) la sustentación.
- Si la aceleración aumenta, aumenta la sustentación.
- Si la sustentación aumenta (disminuye), el avión se eleva (desciende).
- Si el ángulo de ataque es muy grande, el avión se queda estático en el aire.

4.6.-SIMULACIÓN EN MATLAB DEL MODELO MATEMÁTICO PRODUCTO DE LAS ECUACIONES DIFERENCIALES:

```
home
clear all
close all

 % Dinámica longitudinal del avión en tiempo continuo -  lazo abierto
 % State vector: x = [u w q theta h Omega]
 % Input vector: u = [elevator throttle]
 % Output vector: y = [Va-x  h-z]

 % Matriz de estados de lazo abiero
A1=[
    -0.2197    0.6002   -1.4882   -9.7969   -0.0001    0.0108
   -0.5820   -4.1207   22.4024   -0.6460    0.0009         0
    0.4823   -4.5286   -4.7515         0   -0.0000   -0.0084
         0         0    1.0000         0         0         0
    0.0658   -0.9978         0   22.9997         0         0
   32.1028    2.1170         0         0   -0.0294   -2.7813

    ];

 % Matriz de control

B1=[  0.3246         0
     -2.1521         0
    -29.8230         0
         0           0
         0           0
         0      448.6010  ];

 % Matriz de observación (de salida)

C1=[ 0.9978    0.0658         0         0         0         0
     0         0              0        0          1         0
    ];

% Matriz de transferencia directa
D1=[0 0
    0 0 ];

% Dinámica de dirección lateral del avión en tiempo continuo -  lazo
abierto

% State vector: x = [v p r phi psi]
% Input vector: u = [aileron rudder]
% Output vector: y = [phi-x psi-z]




 % Matriz de estados

A2=[ -0.6373    1.5134  -22.9499    9.7969         0
```

```
   -4.1921  -20.6293    9.9287         0         0
    0.6799   -2.6759   -1.0377         0         0
         0    1.0000    0.0659    0.0000         0
         0         0    1.0022    0.0000         0

    ];


 % Matriz de control

B2=[   -1.2511    3.1933
 -109.8428    1.9764
   -4.3309  -20.1764
         0         0
         0         0

    ];
% Matriz de observación
C2=[      0         0         0    1.0000         0
         0         0         0         0    1.0000
    ];

D2=[0 0
   0 0];


% Simulación de la dinámica del avión en tiempo continuo en lazo abierto
%------------------------------------------------------------------------

% Tiempo de simulación: 120 seg = 2 minutos
t=0:0.001:120;

% ====================================
% Simulación del Movimiento longitudinal
% ====================================
% Respuesta ante la entrada u1 = elevador de unidad unitaria tipo escalón
Y1e=step(A1,B1,C1,D1,1,t);
% Respuesta ante entrada u2 = throtlle de unidad unitaria tipo escalo
Y1a=step(A1,B1,C1,D1,2,t);
% Respuesta de Y(t) longitudinal ante u1 y u2
Y1=Y1e+Y1a;

%  Simulación longitudinal continua
figure(1)
subplot(2,1,1)
plot(t,Y1(:,1))
title('Velocidad del avión el la dirección +X')
ylabel('Va(m/s)')
xlabel('t(seg)')
grid
subplot(2,1,2)
plot(t,Y1(:,2))
title('Altitud del avión el la dirección -Z')
ylabel('H(m)')
xlabel('t(seg)')
grid

% ================================
% Simulación del Movimiento lateral
% ================================
```

```
% Respuesta ante la entrada u1 = alerón de unidad unitaria tipo escalón
Y2a=step(A2,B2,C2,D2,1,t);
% Respuesta ante la entrada u2 = rudder de unidad unitaria tipo escalón
Y2c=step(A2,B2,C2,D2,2,t);
% Respuesta de Y(t) lateral ante u1 y u2
Y2=Y2a+Y2c;

%  Simulación lateral discreta
figure(2)
subplot(2,1,1)
plot(t,Y2(:,1))
title('Angulo de giro del avión respecto del eje +X')
ylabel('Va(m/s)')
xlabel('t(seg)')
grid
subplot(2,1,2)
plot(t,Y2(:,2))
title('Angulo de derive del avión respecto del eje -Z')
ylabel('Va(m/s)')
xlabel('t(seg)')
grid

%=====================================
%Discretización de la dinámica del avión
%=====================================
% periodo de muestreo T
T=0.01;
% Discretización longitudinal
[G1,H1,C1,D1]=c2dm(A1,B1,C1,D1,T);
% Discretización lateral
[G2,H2,C2,D2]=c2dm(A2,B2,C2,D2,T);

% Controlabilidad longitudinal
M1=ctrb(G1,H1);
r1=rank(M1)
% Controlabilidad lateral
M2=ctrb(G2,H2);
r2=rank(M2)

% Simulación del modelo linealizado de lazo cerrado en tiempo discreto
% Numero de muestras = N
N=12000;
k=0:N-1;
% t = kT (seg)
% Respuesta ante la entrada u1 = elevador de unidad unitaria tipo escalón
Y1e=dstep(G1,H1,C1,D1,1,N);
% Respuesta ante entrada u2 = throtlle de unidad unitaria tipo escalón
Y1a=dstep(G1,H1,C1,D1,2,N);
% Respuesta de Y(kT) longitudinal ante u1(kT) y u2(kT)
Y1=Y1e+Y1a;

%  Simulación longitudinal discreta
figure(3)
subplot(2,1,1)
stairs(k*T,Y1(:,1))
title('Velocidad del avión el la dirección +X')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y1(:,2))
```

```
title('Altitud del avión el la dirección -Z')
ylabel('H(m)')
xlabel('kT(seg)')
grid

% Respuesta ante la entrada u1 = alerón de unidad unitaria tipo escalón
Y2a=dstep(G2,H2,C2,D2,1,N);
% Respuesta ante entrada u2 = throtlle de unidad unitaria tipo escalón
Y2c=dstep(G2,H2,C2,D2,2,N);
% Respuesta de Y(kT) later ante u1(kT) y u2(kT)
Y2=Y2a+Y2c;

%  Simulación lateral discreta
figure(4)
subplot(2,1,1)
stairs(k*T,Y2(:,1))
title('Angulo de giro del avión respecto del eje +X')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y2(:,2))
title('Angulo de derive del avión respecto del eje -Z')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid


% Diseño de Kampliado
% Sistema de control optimo para el modelo linealizado del avión
% utilizando integradores de error en lazo cerrado
% Numero de muestras N
N=12000;
k=0:N-1;

%========================
% Movimiento longitudinal
%------------------------
% Modelo entendido del avión-integradores
G1e=[G1, zeros(6,2);-C1*G1 eye(2)];
H1e=[H1;-C1*H1];
% Controlabilidad del sistema
M1e=ctrb(G1e,H1e);
r=rank(M1e)

% matrices de ponderación para la minimización del índice de rendimiento
Q1e=diag([1000000 100000 10000 1000 1000 1000 1000 1000]);
R1=diag([10^12 10^12]);
% Calculo de la matriz extendida de ganancia de realimentación
[K1e P1e E1e]=dlqr(G1e,H1e,Q1e,R1);
% Matriz de ganancia de realimentación de estados
K1=K1e(:,1:6)
% Matriz de ganancia de integración de error
KI1=-K1e(:,7:8)
% Valor absoluto del los autovalores
aE1e=abs(E1e)

% Matriz de estado ampliado en lazo cerrado
G1ea=G1e-H1e*K1e;
% Matriz de referencia para el control del sistema de lazo cerrado
H1ea=[0 0
```

```
      0 0
      0 0
      0 0
      0 0
      0 0
      1 0
      0 1];
% Matriz de salida ampliada en lazo cerrado
C1ea=[C1 zeros(2,2)]
% Matriz de transferencia directa
D1ea=zeros(2,2)
% Respuesta del sistema para una señal de referencia de velocidad r1=Var
[Y1r1,X1r1]=dstep(G1ea,25*H1ea,C1ea,D1ea,1,N);
% Respuesta del sistema para una señal de referencia de velocidad r2=Ha
[Y1r2,X1r2]=dstep(G1ea,-200*H1ea,C1ea,D1ea,2,N);
% Respuesta ante las dos señales de referencia
Y1r=Y1r1+Y1r2;
% Vector de estados en lazo cerrado
X1r=X1r1+X1r2;

% Simulación en lazo cerrado
figure(5)
subplot(2,1,1)
stairs(k*T,Y1r(:,1))
title('Velocidad del avión el la dirección +X')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y1r(:,2))
title('Altitud del avión el la dirección -Z')
ylabel('H(m)')
xlabel('kT(seg)')
grid

figure(6)
subplot(6,1,1)
stairs(k*T,X1r(:,1))
ylabel('U')
title('Vector de estados movimiento longitudinal')
grid
subplot(6,1,2)
stairs(k*T,X1r(:,2))
ylabel('W')
grid
subplot(6,1,3)
stairs(k*T,X1r(:,3))
ylabel('q')
grid
subplot(6,1,4)
stairs(k*T,X1r(:,4))
ylabel('theta')
grid
subplot(6,1,5)
stairs(k*T,X1r(:,5))
ylabel('h')
grid
subplot(6,1,6)
stairs(k*T,X1r(:,6))
ylabel('Engine')
xlabel('kT(seg)')
grid
```

```
%=========================
% Movimiento lateral
%-------------------------
% Modelo entendido del avión-integradores

G2e=[G2, zeros(5,2);-C2*G2 eye(2)];
H2e=[H2;-C2*H2];
% Controlabilidad del sistema
M2e=ctrb(G2e,H2e);

r2=rank(M2e)
% matrices de ponderación
Q2e=diag([1000000 100000 1000 1000 100 1000 1000]);
R2=diag([10^12 10^12]);

[K2e P2e E2e]=dlqr(G2e,H2e,Q2e,R2);
K2=K2e(:,1:5)
KI2=-K2e(:,6:7)
aE2e=abs(E2e)
G2ea=G2e-H2e*K2e;
H2ea=[0 0
      0 0
      0 0
      0 0
      0 0
      1 0
      0 1];
C2ea=[C2 zeros(2,2)]
D2ea=zeros(2,2)
N=12000;
k=0:N-1;

[Y2r1,X2r1]=dstep(G2ea,pi/16*H2ea,C2ea,D2ea,1,N);
[Y2r2,X2r2]=dstep(G2ea,-pi/2*H2ea,C2ea,D2ea,2,N);
Y2r=Y2r1+Y2r2;
X2r=X2r1+X2r2;
figure(7)
subplot(2,1,1)
stairs(k*T,Y2r(:,1))
title('Angulo de giro del avión respecto del eje +X')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y2r(:,2))
title('Angulo de derive del avión respecto del eje -Z')
ylabel('Va(m/s)')
xlabel('kT(seg)')
grid

figure(8)
subplot(5,1,1)
stairs(k*T,X2r(:,1))
title('Vector de estados movimiento lateral')
ylabel('betha')
grid
subplot(5,1,2)
stairs(k*T,X2r(:,2))
ylabel('p')
```

```
grid
subplot(5,1,3)
stairs(k*T,X2r(:,3))
ylabel('r')
grid
subplot(5,1,4)
stairs(k*T,X2r(:,4))
ylabel('phi')
grid
subplot(5,1,5)
stairs(k*T,X2r(:,5))
ylabel('psi')
xlabel('kT(seg)')
grid
```

4.7.-CONTROL ÓPTIMO UTILIZANDO EL MODELO LINEALIZADO EN TIEMPO DISCRETO.

Proceso:

$$x(k+1) = Gx(k) + Hu(k)$$
$$y(k) = Cx(k)$$
$$D = [0]$$

Ley de control.

$$u(k) = -kx(k) + k_T u(k)$$

Integrales de error en adelanto [CA01].

$$V(z) = \frac{1}{1 - Z^{-1}} E(z)$$

osea

$$V(k) - V(k-1) = E(k)$$
$$V(k+1) = V(k) + E(k) = V(k) + r(K+1) - y(k+1)$$
$$V(k) + r(k+1) - CX(k+1)$$

como $r(k)$ *escalón*:

$$r(k+1) = r(k) = r(0) = r(\infty)$$
$$V(k+1) = -CG(k) + V(k) - CHu(k) + r(k)$$

4.8.-DIAGRAMA DE BLOQUES.

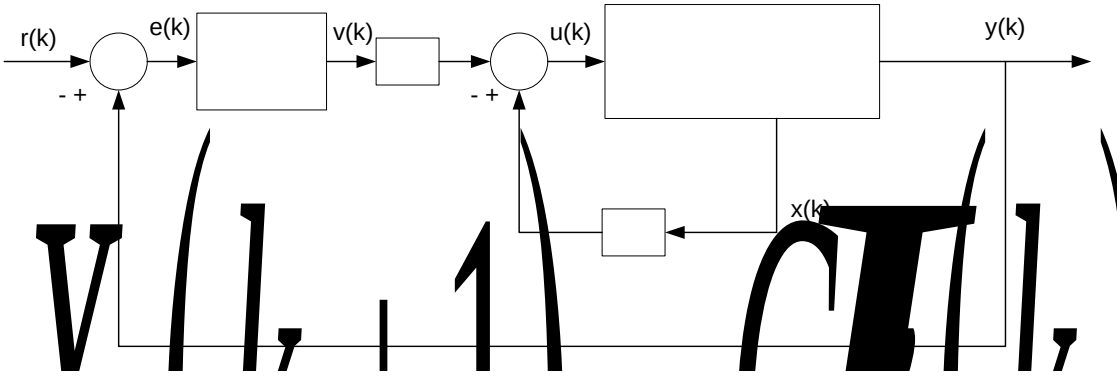


Diagrama de bloques del control óptimo linealizado en tiempo discreto.

FIG. 39

De las ecuaciones anteriores se obtiene:

$$\begin{bmatrix} x(k+1) \\ V(k) \end{bmatrix} = \begin{bmatrix} G & 0 \\ -CG & I \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} H \\ -CH \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(k)$$

$$y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix}$$

$$u(k) = -\begin{bmatrix} k & -kI \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix}$$

$$\xi(k) \square \begin{bmatrix} x(k) \\ v(k) \end{bmatrix}$$

entonces:

$$\xi(k+1) = \tilde{G}\xi(k) + \tilde{H}u(k) + \tilde{E}r(k)$$
$$y(k) = \tilde{C}\xi(k) + [0]r(k)$$
$$u(k) = -\tilde{K}\xi(k)$$

$$\tilde{G} = \begin{bmatrix} G & 0 \\ -CG & I \end{bmatrix}, \quad \tilde{H} = \begin{bmatrix} H \\ -CH \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad \tilde{K} = \begin{bmatrix} K & -KI \end{bmatrix}$$

Cálculo de $\tilde{K}$, se debe encontrar de tal modo que minimice el índice de rendimiento J.

$J = \frac{1}{2}\sum_{J=0}^{\infty} \xi^T(0) \quad \tilde{Q}\xi(k) + u^T(k)Ru(k)$, la ecuación que minimiza a J es la ecuación de Ricatti.

$$\tilde{P}(k+1) = \tilde{Q} + \tilde{G}^T P(k)\tilde{G} - \tilde{G}^T P(k)\tilde{H}\left(R + \tilde{H}^T P(k)H\right)^{-1} H^T P(k)\tilde{G}$$

También usando la sentencia del matlab dlqr se toma en cuenta el sistema de lazo cerrado.

$$\xi(k+1) = \left(\tilde{G} - \tilde{H}\tilde{K}\right)\xi(k) + \tilde{E}r(k)$$
$$y(k) = \tilde{C}\xi(k)$$

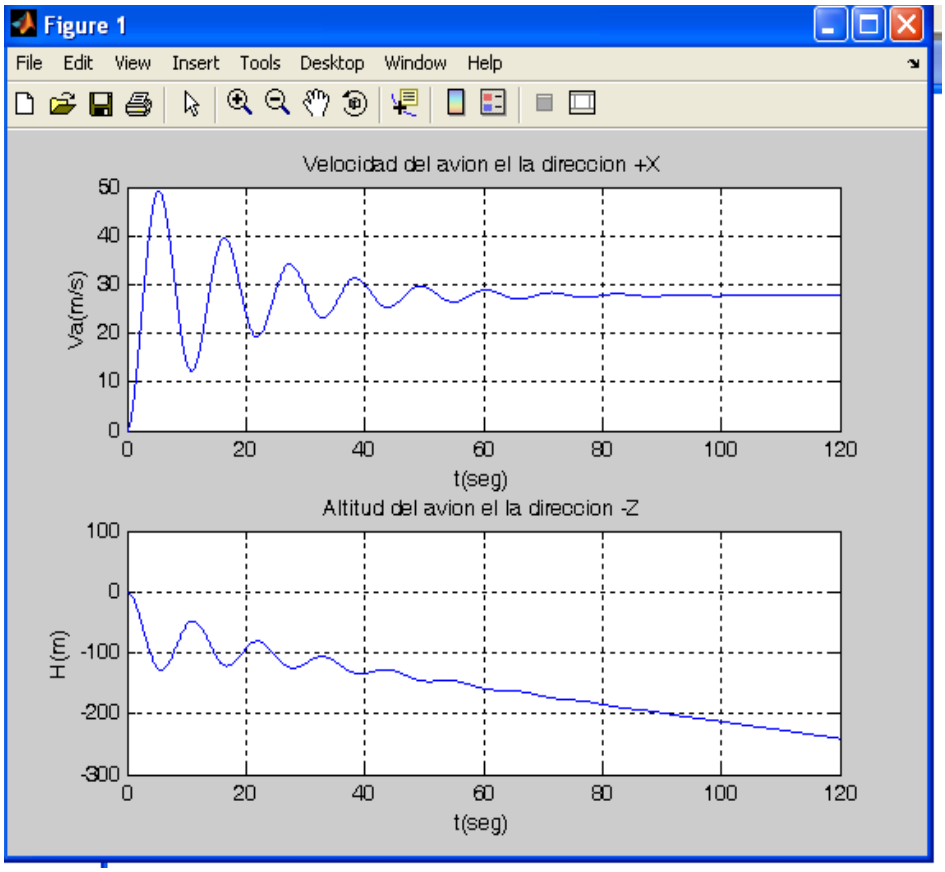
Al ejecutar el programa se obtiene la simulación de respuesta del sistema tanto en lazo abierto del sistema linealizado así como del sistema en lazo abierto.

En la figura 40 se muestra la simulación de la respuesta de la velocidad vx(t) en la dirección del eje x y de la altura h(t) en lazo abierto (sin control) en tiempo continuo. Observando grande oscilaciones las cuales se deben corregir incluyendo el controlador.

En la figura 41 se muestra la simulación de los ángulos de giro (respecto del eje x) y el Angulo de derive (respecto del eje -z), el cual dichos ángulos varían indefinidamente sin control. Para la simulación se está considerando al sistema linealizado.

4.9.-SIMULACIÓN EN MATLAB DEL MODELO LINEALIZADO

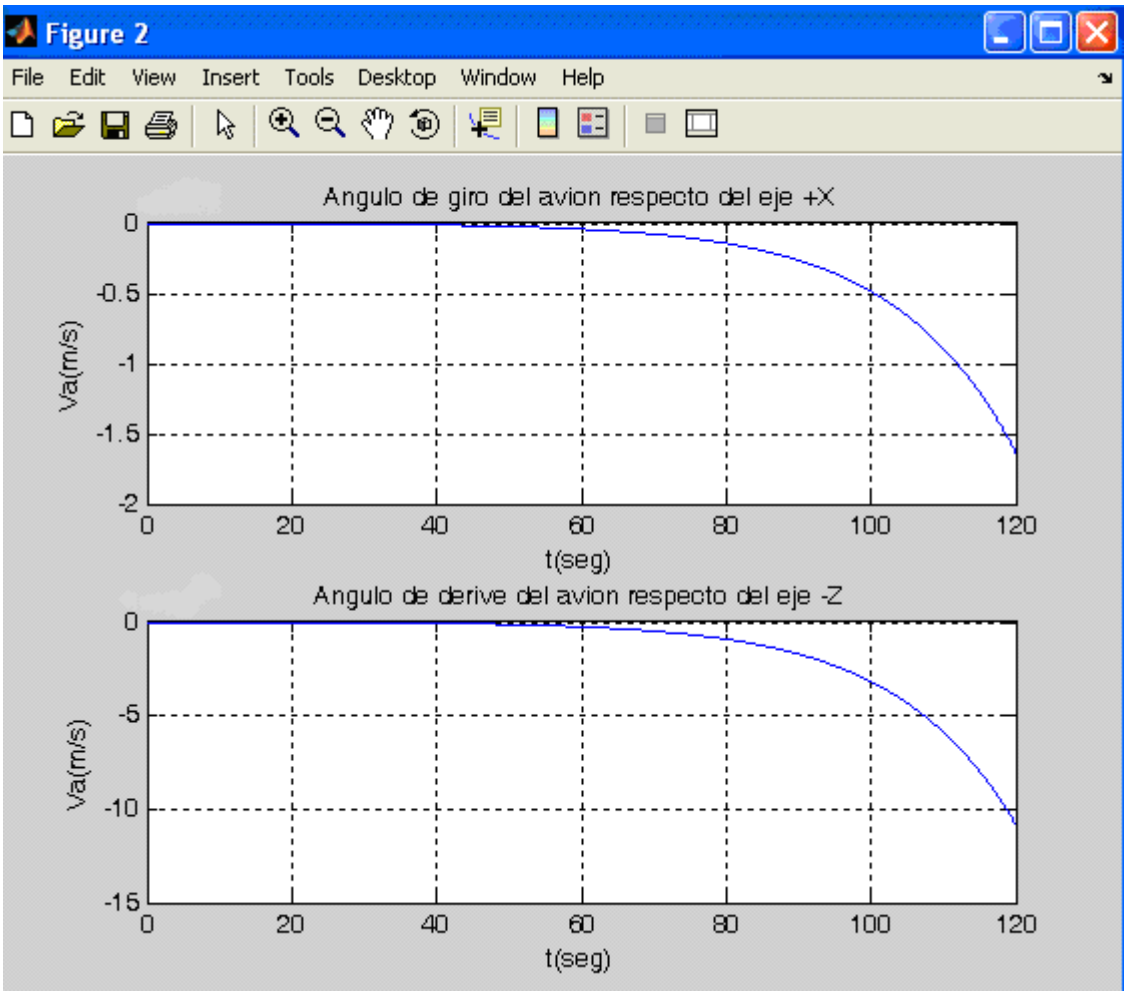
En la siguiente gráfica se muestra la simulación del movimiento longitudinal ante la entrada u_1, la respuesta al escalón unitario en elevador. Esta simulación es en lazo abierto: La velocidad del avión es en eje +x y la altitud del avión en el –z, en donde comienza a subir a más de 200 metros.



Velocidad (m/s) y H (m) del modelo linealizado.

FIG. 40

En esta gráfica se muestra la respuesta al impulso en el movimiento lateral. Está simulación lateral discreta se realiza en la cola del avión y nos muestra el giro del avión respecto del eje +x con un ángulo de deriva respecto al eje –z.

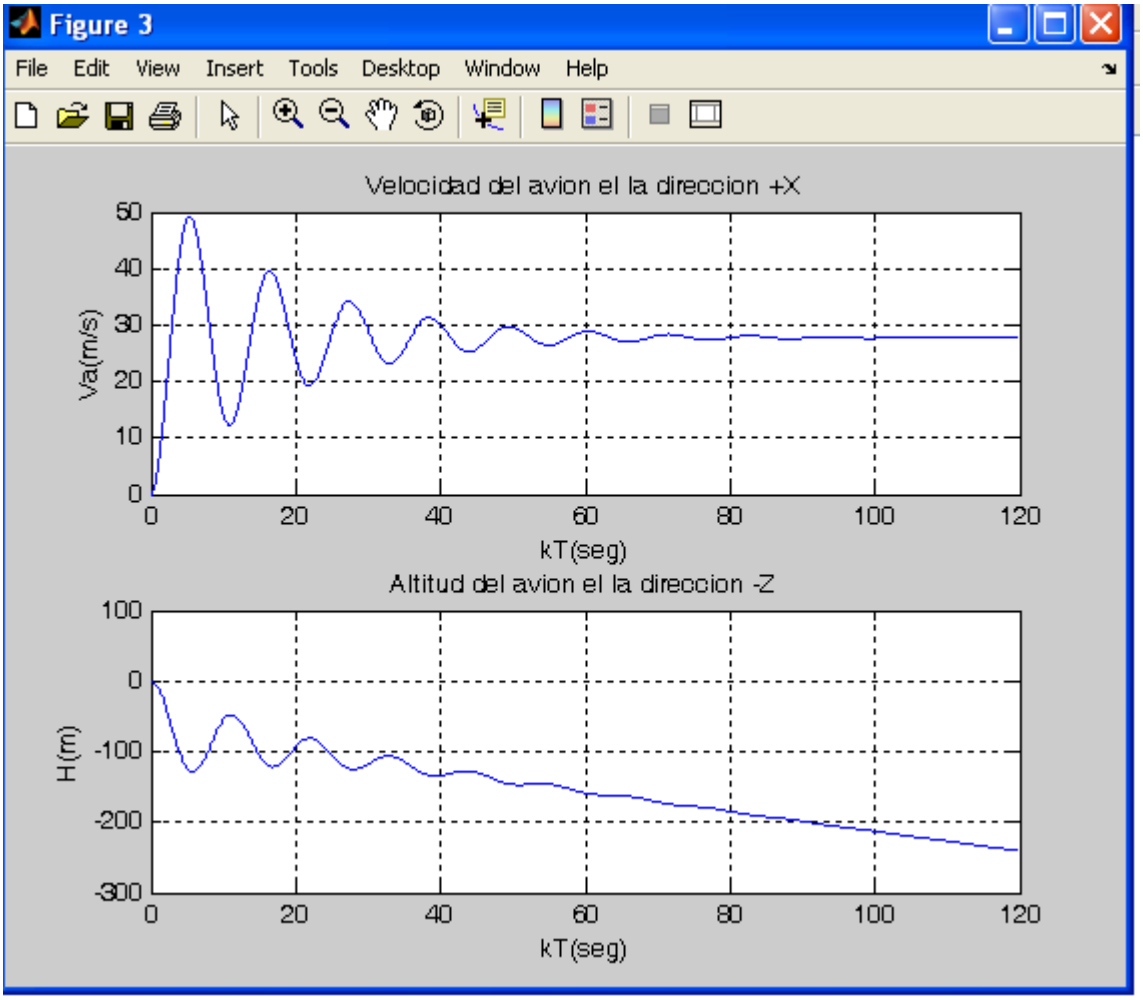


Velocidad en (m/s) de giro del aeromodelo en el eje X y en el eje Y respectivamente.

FIG. 41

En la gráfica siguiente nos muestra la dinámica del avión en lazo cerrado en tiempo discreto, dado un impulso al elevador de tipo escalón unitario.

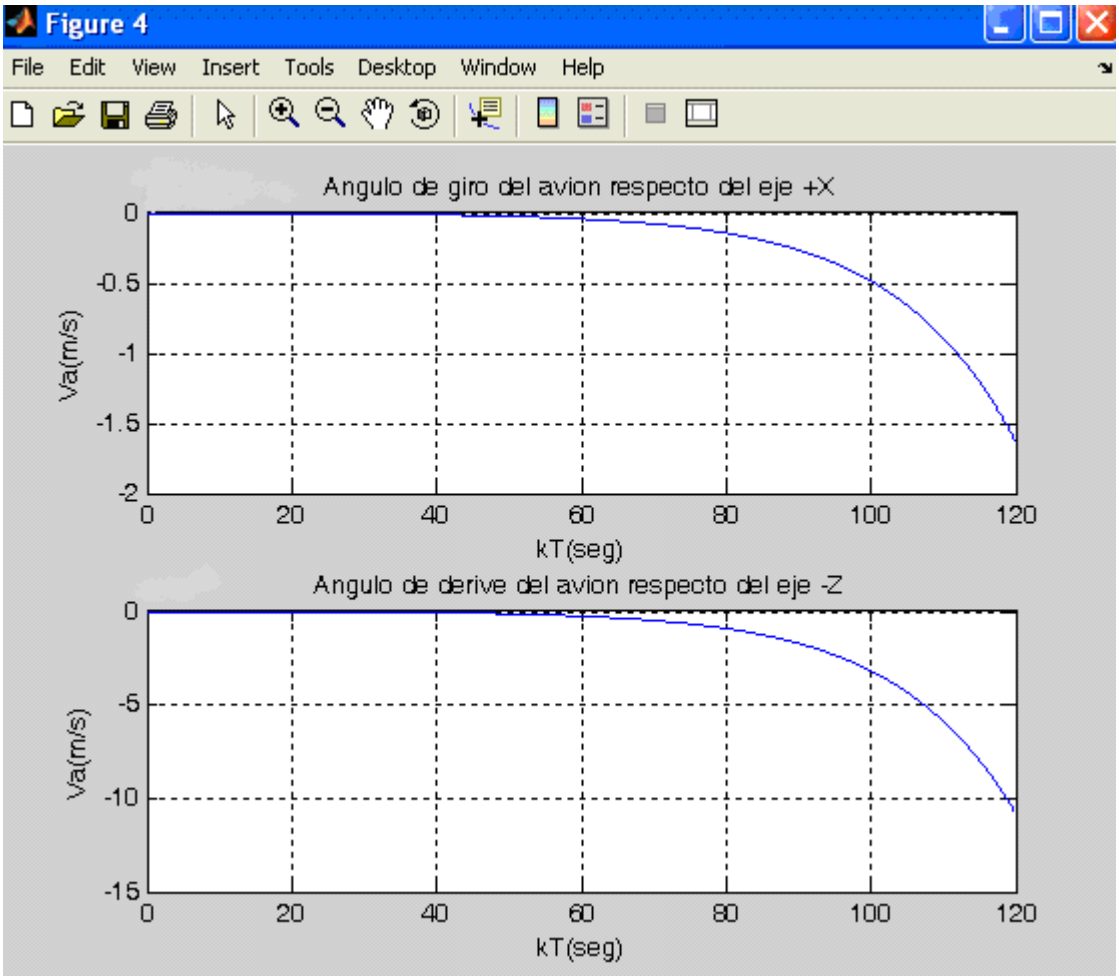
La velocidad del avión en la dirección +x en m/s y subiendo H(m) en la dirección –z.



Modelo discretizado en donde se muestra la velocidad (m/s) y H (m).

FIG. 42

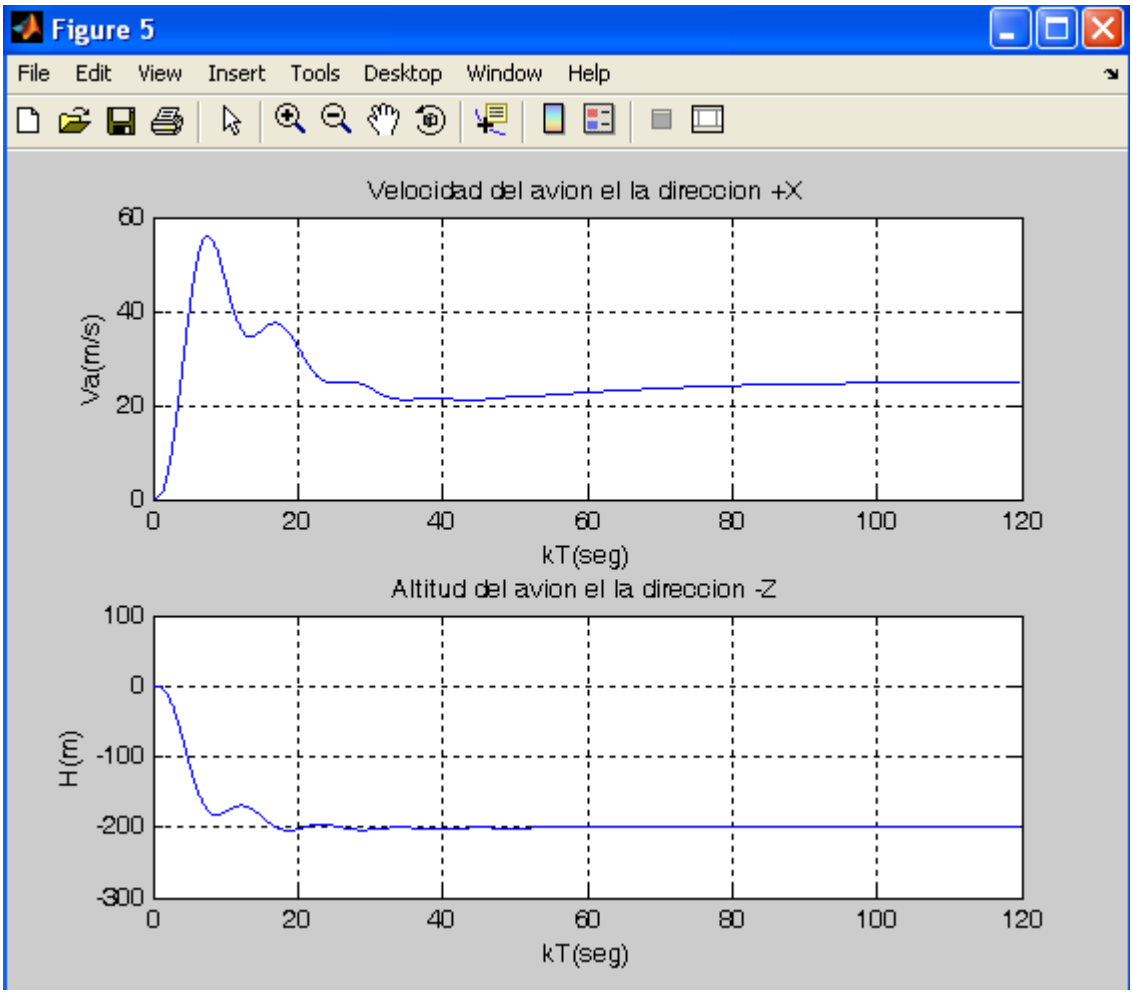
En la siguiente figura se aprecia la simulación lateral discreta en lazo cerrado ante un impulso del escalón unitario. El ángulo de giro del avión respecto al eje +x con un ángulo de deriva respecto al eje –z.



Modelo discretizado de la Velocidad en (m/s) de giro del aeromodelo en el eje X y en el eje Y respectivamente.

FIG. 43

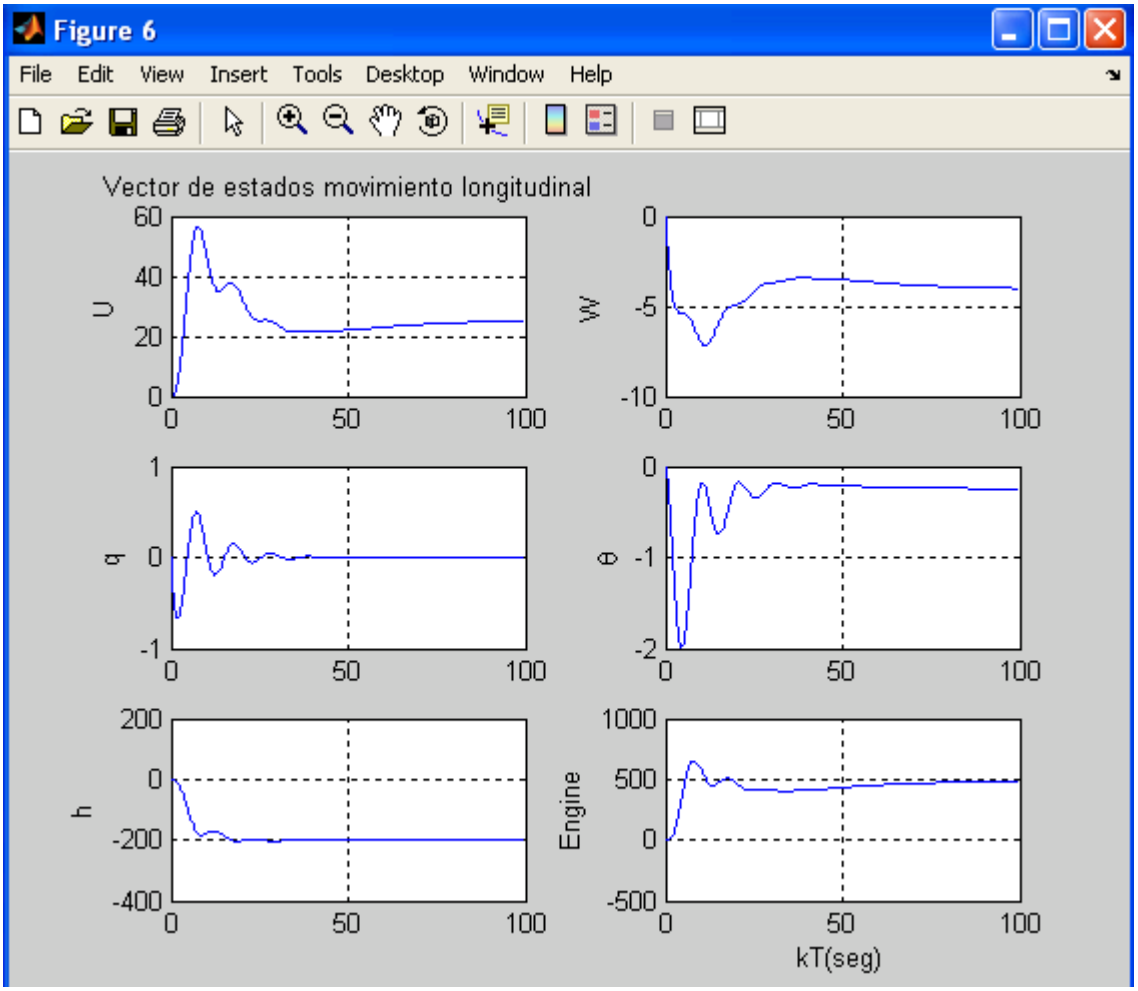
La simulación del movimiento longitudinal del avión utilizando el modelo extendido con integradores se aprecia a continuación. El avión realiza un giro con la dirección del eje +x con una altitud en la dirección –z de 200 metros.



Simulación en lazo cerrado de la velocidad (m/s) y la altura (m).

FIG. 44

Se aprecia los vectores de estado en lazo cerrado.



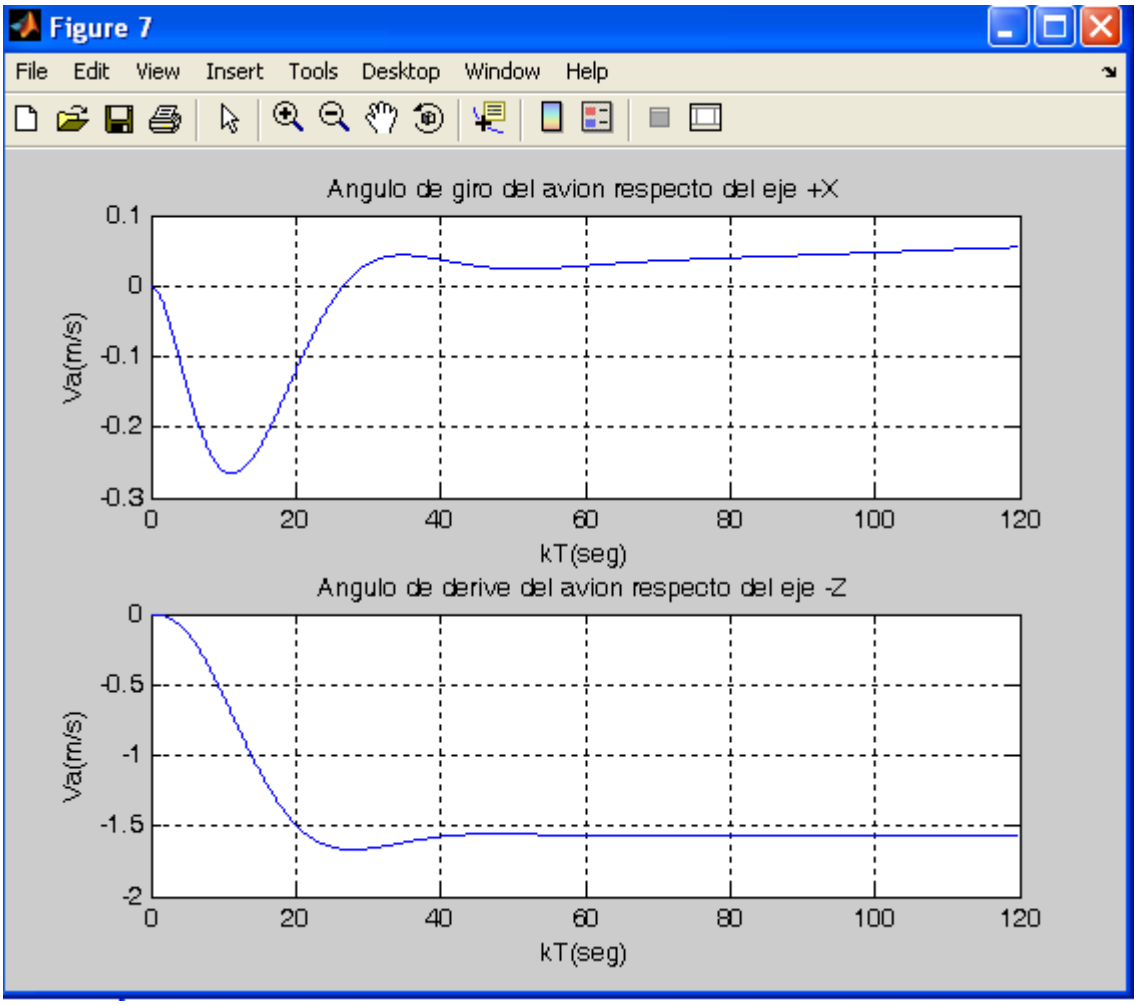
Vector de estados en la dinámica longitudinal.

FIG. 45

Donde:

U = Vector de estados en movimiento longitudinal.
q = Tasa de elevación [rad/s]
h= altura del avión en metros.
W = Velocidad del fuselaje.[m/s]
θ = Ángulo de elevación [rad].

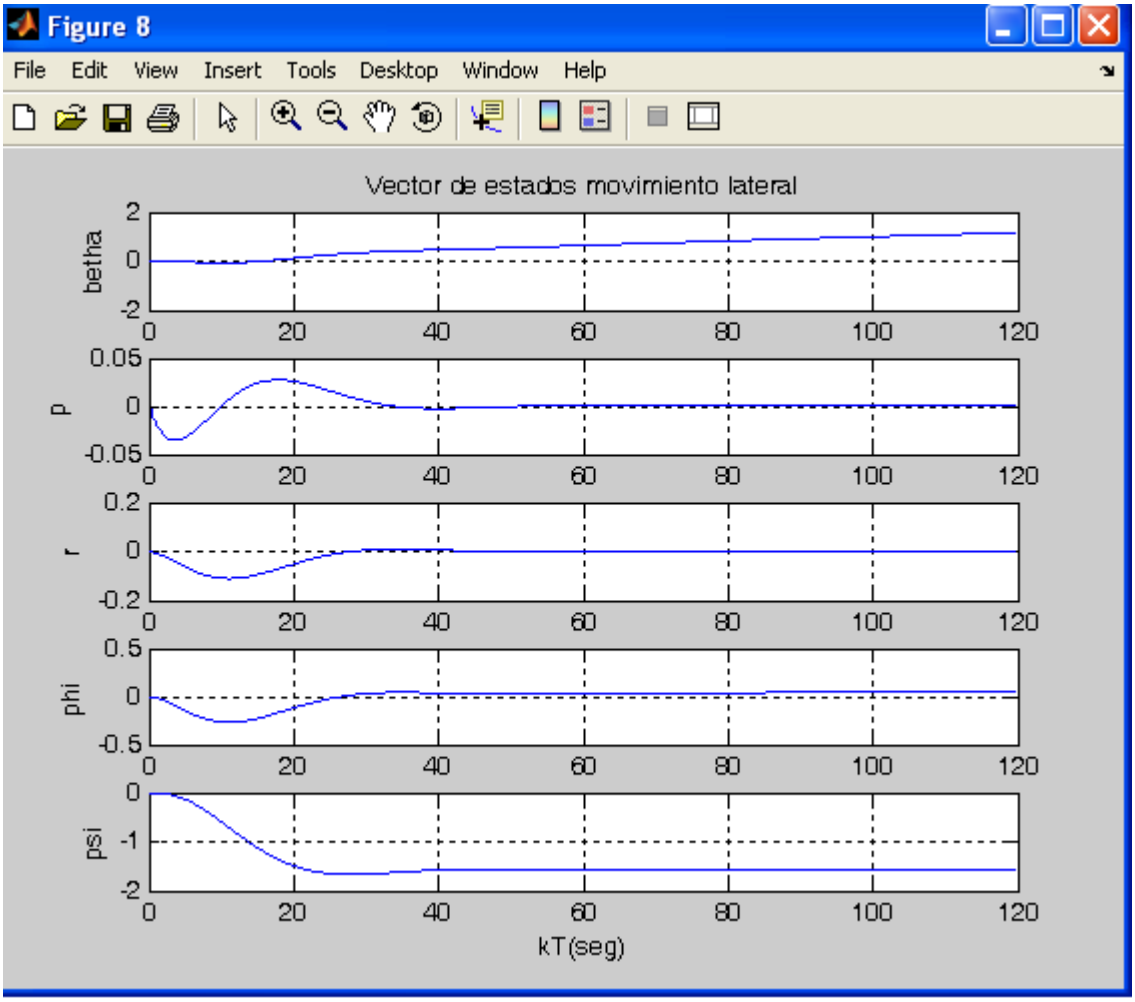
Simulación del movimiento lateral con integradores, se muestra el giro del avión respecto al eje +x con una deriva respecto del eje –z.



Angulo de giro del avión respecto al eje X y el ángulo de la deriva respecto al eje Z.

FIG. 46

La simulación nos muestra el vector de estados del movimiento lateral en lazo cerrado.



Vector de estados en la dinámica del movimiento lateral.

FIG. 47

Donde:

betha = Ángulo de deslizamiento lateral [rad].
p = Tasa de rotación [rad/s].
r= Tasa de giros [rad/s].
phi = Ángulo de rotación del fuselaje[rad].
psi = Ángulo de giro [rad].

4.10.-SIMULACIÓN EN MATLAB DEL MODELO LINEALIZADO, SIGUIENDO DOS TIPOS DE TRAYECTORIAS COMO LA SINUSOIDAL Y LA CUADRADA.

```
home
clear all
close all

 % Dinámica longitudinal del avión en tiempo continuo -  lazo abierto
 % State vector: x = [u w q theta h Omega]
 % Input vector: u = [elevator throttle]
 % Output vector: y = [Va-x  h-z]

 % Matriz de estados de lazo abierto
A1=[
    -0.2197    0.6002   -1.4882   -9.7969   -0.0001    0.0108
   -0.5820   -4.1207   22.4024   -0.6460    0.0009         0
    0.4823   -4.5286   -4.7515         0   -0.0000   -0.0084
         0         0    1.0000         0         0         0
    0.0658   -0.9978         0   22.9997         0         0
   32.1028    2.1170         0         0   -0.0294   -2.7813

    ];

 % Matriz de control

B1=[  0.3246         0
     -2.1521         0
    -29.8230         0
         0           0
         0           0
         0      448.6010  ];

 % Matriz de observación (de salida)

C1=[ 0.9978    0.0658         0         0         0         0
     0         0              0         0         1         0
    ];

% Matriz de transferencia directa
D1=[0 0
    0 0 ];

% Dinámica de dirección lateral del avión en tiempo continuo -  lazo
abierto

% State vector: x = [v p r phi psi]
% Input vector: u = [aileron rudder]
% Output vector: y = [phi-x psi-z]
```

```
 % Matriz de estados

A2=[ -0.6373    1.5134  -22.9499    9.7969         0
   -4.1921  -20.6293    9.9287         0         0
    0.6799   -2.6759   -1.0377         0         0
         0    1.0000    0.0659    0.0000         0
         0         0    1.0022    0.0000         0

    ];


 % Matriz de control

B2=[   -1.2511    3.1933
 -109.8428    1.9764
   -4.3309  -20.1764
         0         0
         0         0

    ];
% Matriz de observación
C2=[      0         0         0    1.0000         0
         0         0         0         0    1.0000
    ];

D2=[0 0
   0 0];

%=======================================
%Discretización de la dinámica del avión
%=======================================
% periodo de muestreo T
T=0.01;
% Discretización longitudinal
[G1,H1,C1,D1]=c2dm(A1,B1,C1,D1,T);
% Discretización lateral
[G2,H2,C2,D2]=c2dm(A2,B2,C2,D2,T);

% Controlabilidad longitudinal
M1=ctrb(G1,H1);
r1=rank(M1)
% Controlabilidad lateral
M2=ctrb(G2,H2);
r2=rank(M2)

%========================
% Movimiento longitudinal
%------------------------
% Modelo entendido del avión-integradores
G1e=[G1, zeros(6,2);-C1*G1 eye(2)];
H1e=[H1;-C1*H1];
% Controlabilidad del sistema
M1e=ctrb(G1e,H1e);
r=rank(M1e)

% matrices de ponderación para la minimización del índice de rendimiento
Q1e=diag([1000000 100000 10000 1000 1000 1000 1000 1000]);
```

```
R1=diag([10^12 10^12]);
% Calculo de la matriz extendida de ganancia de realimentación
[K1e P1e E1e]=dlqr(G1e,H1e,Q1e,R1);
% Matriz de ganancia de realimentación de estados
K1=K1e(:,1:6)
% Matriz de ganancia de integración de error
KI1=-K1e(:,7:8)
% Valor absoluto de los autovalores
aE1e=abs(E1e)

% Matriz de estado ampliado en lazo cerrado
G1ea=G1e-H1e*K1e;
% Matriz de referencia para el control del sistema de lazo cerrado
H1ea=[0 0
      0 0
      0 0
      0 0
      0 0
      0 0
      1 0
      0 1];
% Matriz de salida ampliada en lazo cerrado
C1ea=[C1 zeros(2,2)]
% Matriz de transferencia directa
D1ea=zeros(2,2)

%==============================
%Movimiento longitudinal en lazo cerrado para entradas de referencia
% suaves tipo ondas sinusoidales
%==============================

N=180000; % Indica la cantidad de muestras para un vuelo de duración de
media hora
k=0:N-1;
% Respuesta del sistema para una señal de referencia de velocidad
% relocidad=25*sin(0.005*t)
r1as=25*sin(0.005*k*T);
% Respuesta del sistema para una señal de referencia de altura
% raltura=100*sin(0.01*t)
r2as=-100*sin(0.01*k*T);
Ras1=[r1as;r2as];
[Y1r,X1r]=dlsim(G1ea,H1ea,C1ea,D1ea,Ras1);

figure(1)
subplot(2,1,1)
stairs(k*T,Y1r(:,1))
hold
stairs(k*T,r1as,'r')
title('Velocidad del avión el la dirección +X')
ylabel('Vx(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y1r(:,2))
hold
stairs(k*T,r2as,'r')
title('Altitud del avión el la dirección -Z')
ylabel('H(m)')
xlabel('kT(seg)')
grid
```

```
figure(2)
subplot(3,2,1)
stairs(k*T,X1r(:,1))
ylabel('U')
title('Vector de estados movimiento longitudinal')
grid
subplot(3,2,2)
stairs(k*T,X1r(:,2))
ylabel('W')
grid
subplot(3,2,3)
stairs(k*T,X1r(:,3))
ylabel('q')
grid
subplot(3,2,4)
stairs(k*T,X1r(:,4))
ylabel('\theta')
grid
subplot(3,2,5)
stairs(k*T,X1r(:,5))
ylabel('h')
grid
subplot(3,2,6)
stairs(k*T,X1r(:,6))
ylabel('Engine')
xlabel('kT(seg)')
grid


%=========================
% Movimiento lateral
%-------------------------
% Modelo entendido del avión-integradores

G2e=[G2, zeros(5,2);-C2*G2 eye(2)];
H2e=[H2;-C2*H2];
% Controlabilidad del sistema
M2e=ctrb(G2e,H2e);

r2=rank(M2e)
% matrices de ponderación
Q2e=diag([1000000 100000 1000 1000 100 1000 1000]);
R2=diag([10^12 10^12]);

[K2e P2e E2e]=dlqr(G2e,H2e,Q2e,R2);
K2=K2e(:,1:5)
KI2=-K2e(:,6:7)
aE2e=abs(E2e)
G2ea=G2e-H2e*K2e;
H2ea=[0 0
      0 0
      0 0
      0 0
      0 0
      1 0
      0 1];
C2ea=[C2 zeros(2,2)]
D2ea=zeros(2,2)
```

```
%==============================
%Movimiento lateral en lazo cerrado para entradas de referencia
% suaves tipo ondas sinusoidales
%==============================

N=180000; % Indica la cantidad de muestras para un vuelo de duración de
media hora
k=0:N-1;

%Referencia del ángulo de giro respecto del eje +X
%rangiro=pi/10*sin(0.005*t);
r1bs=pi/10*sin(0.005*k*T);

%Referencia del ángulo dederive respecto del eje +Z
%rangiro=pi/10*sin(0.008*t);
r2bs=pi/2*sin(0.008*k*T);

Ras2=[r1bs;r2bs];
[Y2r,X2r]=dlsim(G2ea,H2ea,C2ea,D2ea,Ras2);

figure(3)
subplot(2,1,1)
stairs(k*T,Y2r(:,1))
title('Angulo de giro del avión respecto del eje +X')
ylabel('\phi (rad)')
xlabel('kT(seg)')
hold
stairs(k*T,r1bs,'r')
grid
subplot(2,1,2)
stairs(k*T,Y2r(:,2))
title('Angulo de derive del avión respecto del eje -Z')
ylabel('\psi (rad)')
xlabel('kT(seg)')
hold
stairs(k*T,r2bs,'r')
grid

figure(4)
subplot(5,1,1)
stairs(k*T,X2r(:,1))
title('Vector de estados movimiento lateral')
ylabel('\betha')
grid
subplot(5,1,2)
stairs(k*T,X2r(:,2))
ylabel('p')
grid
subplot(5,1,3)
stairs(k*T,X2r(:,3))
ylabel('r')
grid
subplot(5,1,4)
stairs(k*T,X2r(:,4))
ylabel('\phi')
grid
subplot(5,1,5)
stairs(k*T,X2r(:,5))
ylabel('\psi')
xlabel('kT(seg)')
grid
```

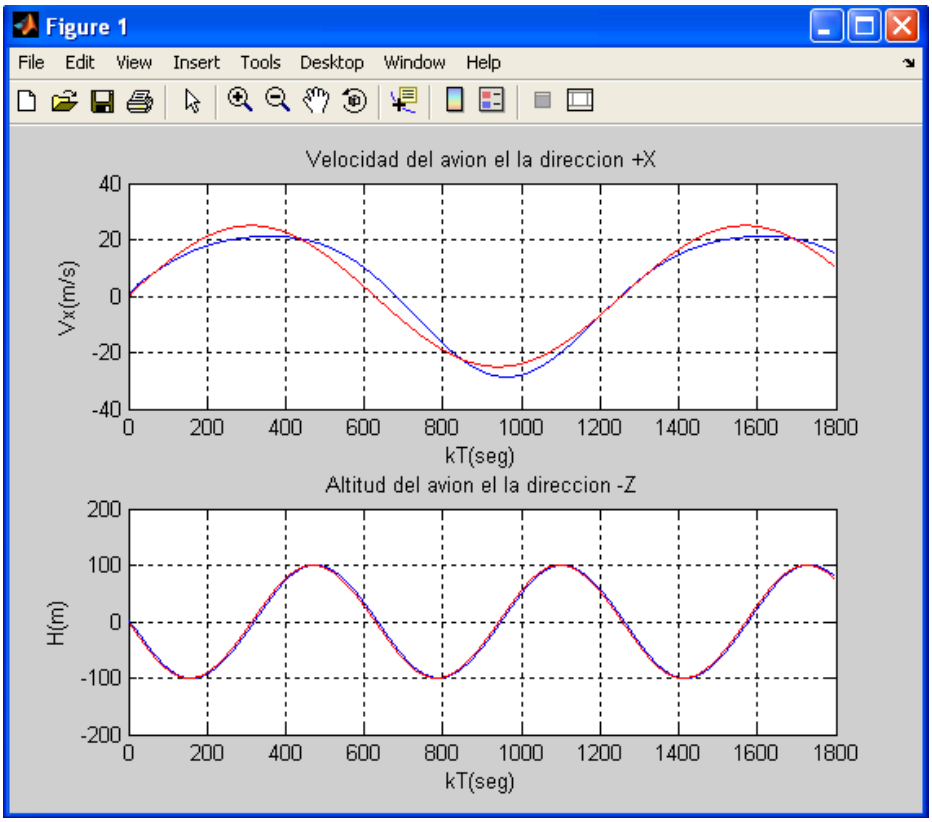
```
%==============================
%Movimiento longitudinal en lazo cerrado para entradas de referencia
% suaves tipo ondas cuadradas
%==============================

N=180000; % Indica la cantidad de muestras para un vuelo de duracion de
media hora
k=0:N-1;
% Respuesta del sistema para una señal de referencia de velocidad
% relocidad==onda cuadrada de 50 pico a pico
r1ac=25*sign(sin(0.005*k*T));
% Respuesta del sistema para una señal de referencia de altura
% raltura=onda cuadrada de 200 pico a pico
r2ac=100*sign(sin(0.01*k*T));
Rac1=[r1ac;r2ac];
[Y1r,X1r]=dlsim(G1ea,H1ea,C1ea,D1ea,Rac1);

figure(5)
subplot(2,1,1)
stairs(k*T,Y1r(:,1))
hold
stairs(k*T,r1ac,'r')
title('Velocidad del avión el la dirección +X')
ylabel('Vx(m/s)')
xlabel('kT(seg)')
grid
subplot(2,1,2)
stairs(k*T,Y1r(:,2))
hold
stairs(k*T,r2ac,'r')
title('Altitud del avión el la dirección -Z')
ylabel('H(m)')
xlabel('kT(seg)')
grid

figure(6)
subplot(3,2,1)
stairs(k*T,X1r(:,1))
ylabel('U')
title('Vector de estados movimiento longitudinal')
grid
subplot(3,2,2)
stairs(k*T,X1r(:,2))
ylabel('W')
grid
subplot(3,2,3)
stairs(k*T,X1r(:,3))
ylabel('q')
grid
subplot(3,2,4)
stairs(k*T,X1r(:,4))
ylabel('\theta')
grid
subplot(3,2,5)
stairs(k*T,X1r(:,5))
ylabel('h')
grid
```

```
subplot(3,2,6)
stairs(k*T,X1r(:,6))
ylabel('Engine')
xlabel('kT(seg)')
grid

%==============================
%Movimiento lateral en lazo cerrado para entradas de referencia
% suaves tipo ondas cuadradas
%==============================

N=180000; % Indica la cantidad de muestras para un vuelo de duracion de
media hora
k=0:N-1;

%Referencia del ángulo de giro respecto del eje +X
%rangiro=onda cuadrada de pi/5 pico a pico
r1bc=pi/10*sign(sin(0.005*k*T));
%Referencia del ángulo dederive respecto del eje +Z
%rangiro=onda cuadrada de pi pico a pico
r2bc=pi/2*sign(sin(0.008*k*T));

Rac2=[r1bc;r2bc];
[Y2r,X2r]=dlsim(G2ea,H2ea,C2ea,D2ea,Rac2);

figure(7)
subplot(2,1,1)
stairs(k*T,Y2r(:,1))
title('Angulo de giro del avión respecto del eje +X')
ylabel('\phi (rad)')
xlabel('kT(seg)')
hold
stairs(k*T,r1bc,'r')
grid
subplot(2,1,2)
stairs(k*T,Y2r(:,2))
title('Angulo de derive del avión respecto del eje -Z')
ylabel('\psi (rad)')
xlabel('kT(seg)')
hold
stairs(k*T,r2bc,'r')
grid

figure(8)
subplot(5,1,1)
stairs(k*T,X2r(:,1))
title('Vector de estados movimiento lateral')
ylabel('\betha')
grid
subplot(5,1,2)
stairs(k*T,X2r(:,2))
ylabel('p')
grid
subplot(5,1,3)
stairs(k*T,X2r(:,3))
ylabel('r')
grid
subplot(5,1,4)
stairs(k*T,X2r(:,4))
ylabel('\phi')
grid
```

```
subplot(5,1,5)
stairs(k*T,X2r(:,5))
ylabel('\psi')
xlabel('kT(seg)')grid
```

En las siguientes figuras se muestra la simulación del modelo matemático siguiendo dos tipos de trayectorias como la sinusoidal y la cuadrada.

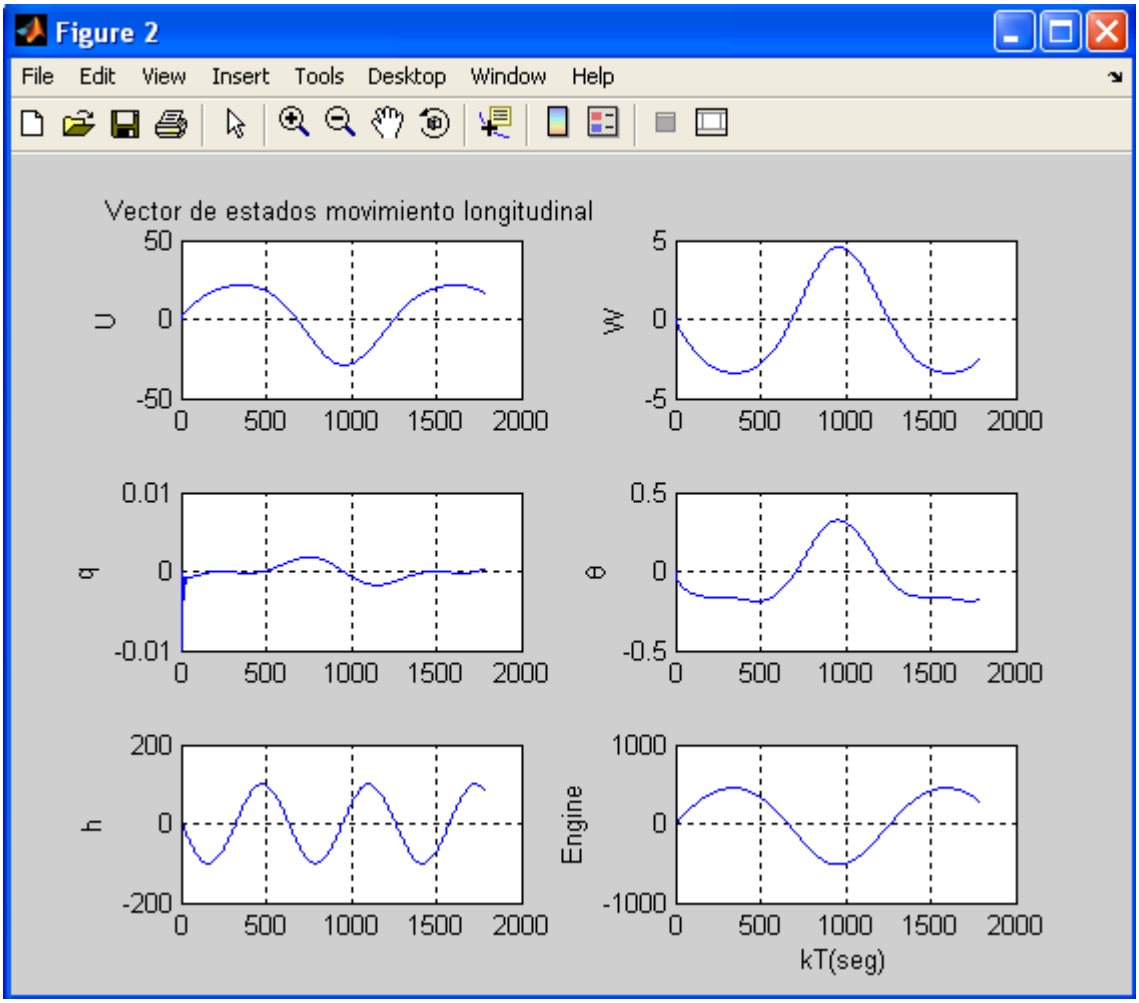
La figura a continuación nos presenta el movimiento longitudinal en lazo cerrado para entradas de referencia suaves tipo ondas sinusoidales. Se grafica la velocidad del avión en la dirección +x en m/s y la altura del avión en la dirección –z en H(m). Hay que hacer notar que la dirección del eje –z indica que el avión se encuentra subiendo. La simulación tiene un tiempo `promedio de media hora. Se debe mencionar el pequeño desfase que se produce en la dirección del eje x debido a que en la práctica para compensar se entran moviendo mecanismos mecánicos que son un poco lentos y que no reaccionan de forma rápida.



Velocidad (m/s) y H (m) del modelo longitudinal en lazo cerrado para entradas de referencia suaves tipo ondas sinusoidales.

FIG. 48

La simulación nos muestra los vectores de la dinámica longitudinal ante una entrada de referencia suave tipo onda sinusoidal.



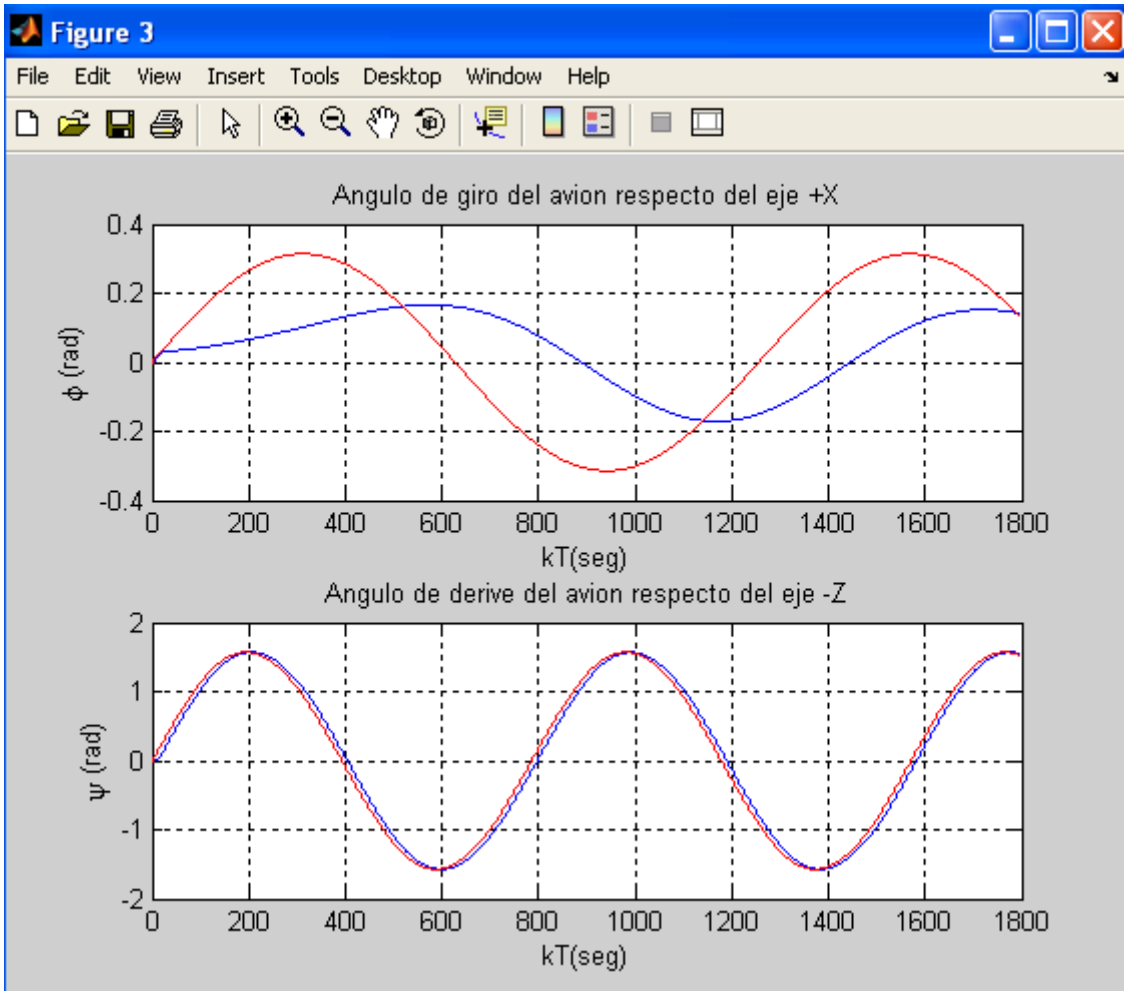
Vector de estados en la dinámica longitudinal.

FIG. 49

Donde:

U = Vector de estados en movimiento longitudinal.
q = Tasa de elevación [rad/s]
h= altura del avión en metros.
W = Velocidad del fuselaje.[m/s]
θ = Ángulo de elevación [rad].

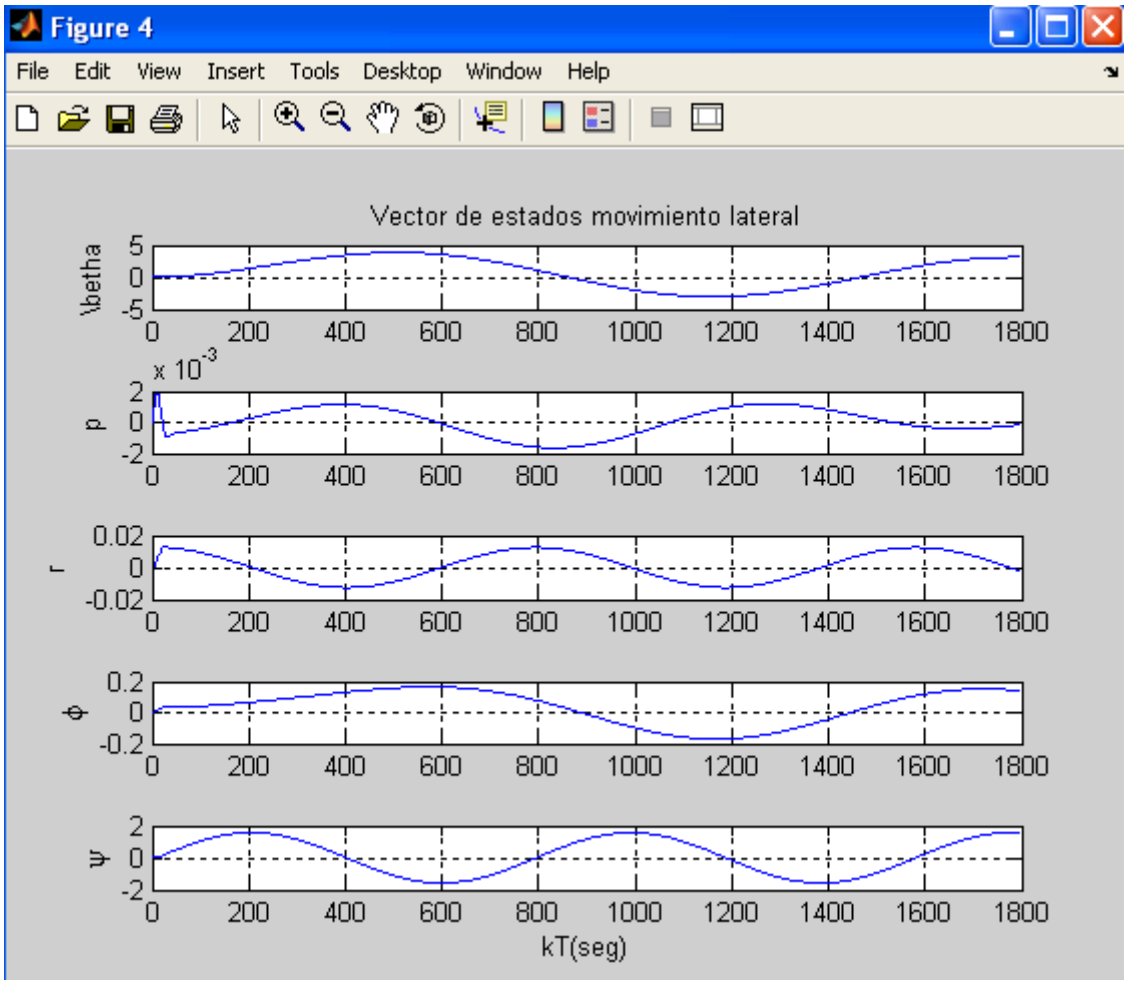
Simulación del movimiento lateral en lazo cerrado para entradas de referencias suaves de tipo ondas sinusoidales. Al que destacar con en la simulación anterior que los mecanismos son lentos y que no hay una respuesta rápida como se quiere.



Movimiento lateral del avión para entrada de referencias suaves.

FIG. 50

Simulación de los vectores de estado en lazo cerrado para entradas de referencia suaves de tipo sinusoidales.



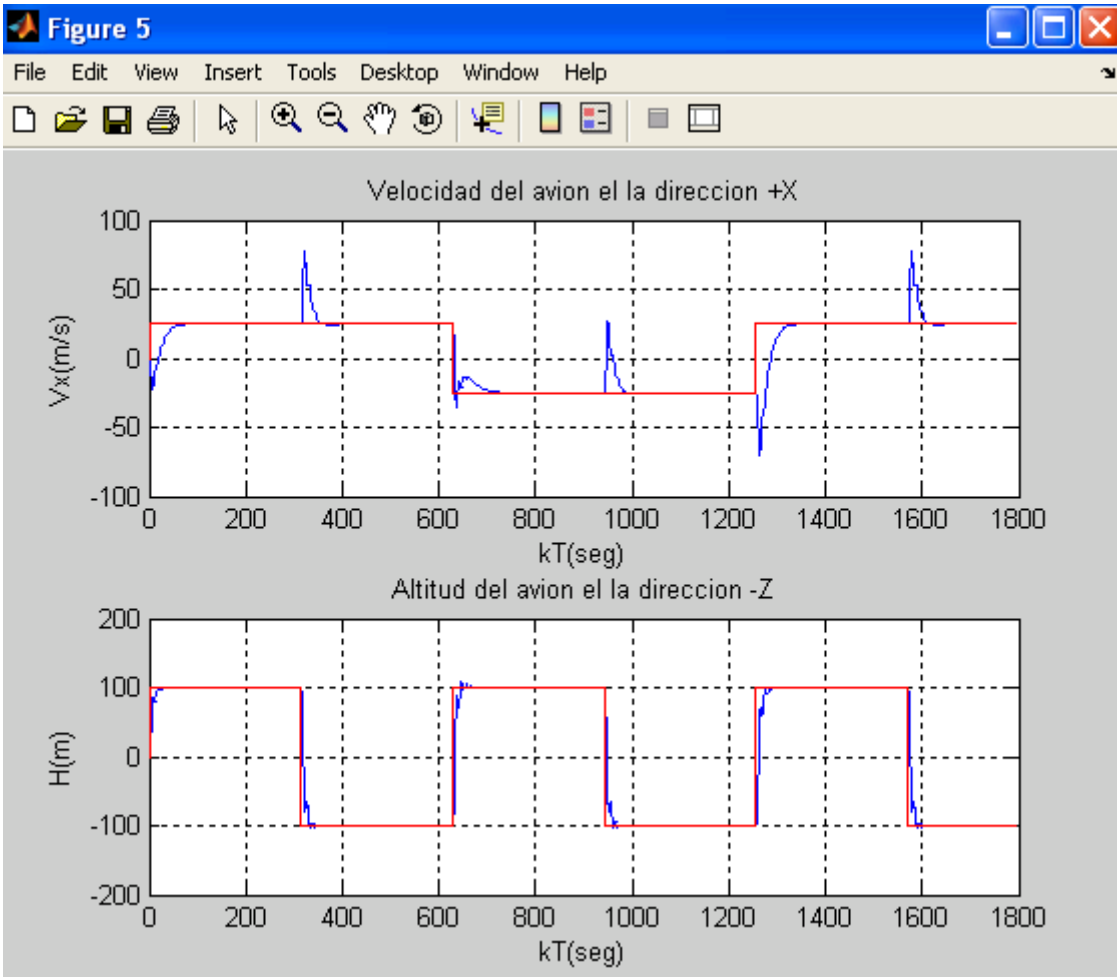
Vectores de estados del movimiento lateral.

FIG. 51

Donde:

betha = Ángulo de deslizamiento lateral [rad].
p = Tasa de rotación [rad/s].
r= Tasa de giros [rad/s].
phi = Ángulo de rotación del fuselaje[rad].
psi = Ángulo de giro [rad].

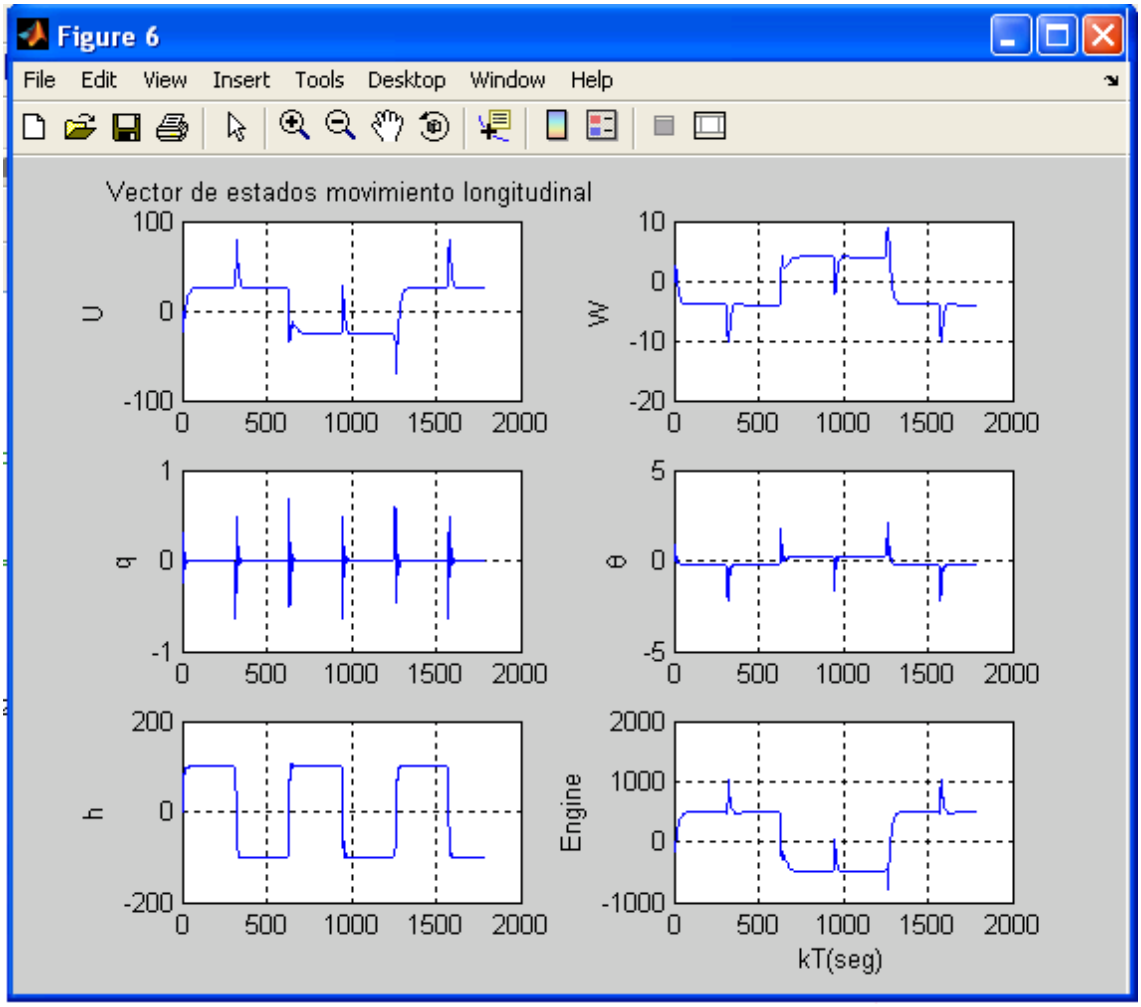
La simulación nos muestra el movimiento longitudinal en lazo cerrado para entradas de referencia suaves tipo ondas cuadradas. Hay que destacar que la trayectoria de un avión no podría ser una forma de tipo cuadrada. Se aprecia unos overshot propios de un sistema de control para trayectoria suaves.



Movimiento longitudinal del avión en la dirección +x y la altitud del avión en la dirección -Z

FIG. 52

La simulación nos muestra los vectores de estado en movimiento longitudinal en lazo cerrado para entradas de referencia suaves de tipo onda cuadrada.



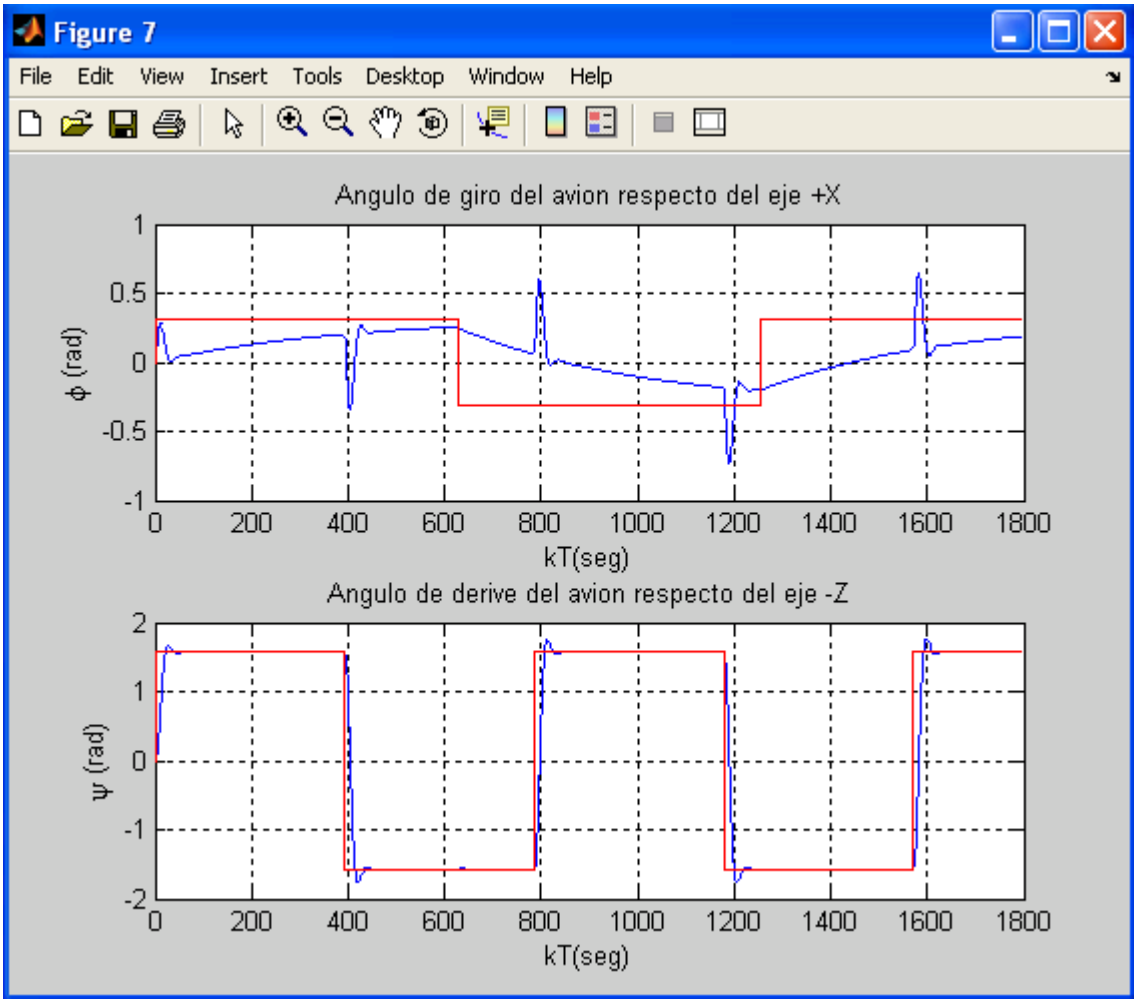
Vector de estados en movimiento longitudinal.

FIG. 53

Donde:

U = Vector de estados en movimiento longitudinal.
q = Tasa de elevación [rad/s]
h= altura del avión en metros.
W = Velocidad del fuselaje.[m/s]
θ = Ángulo de elevación [rad].

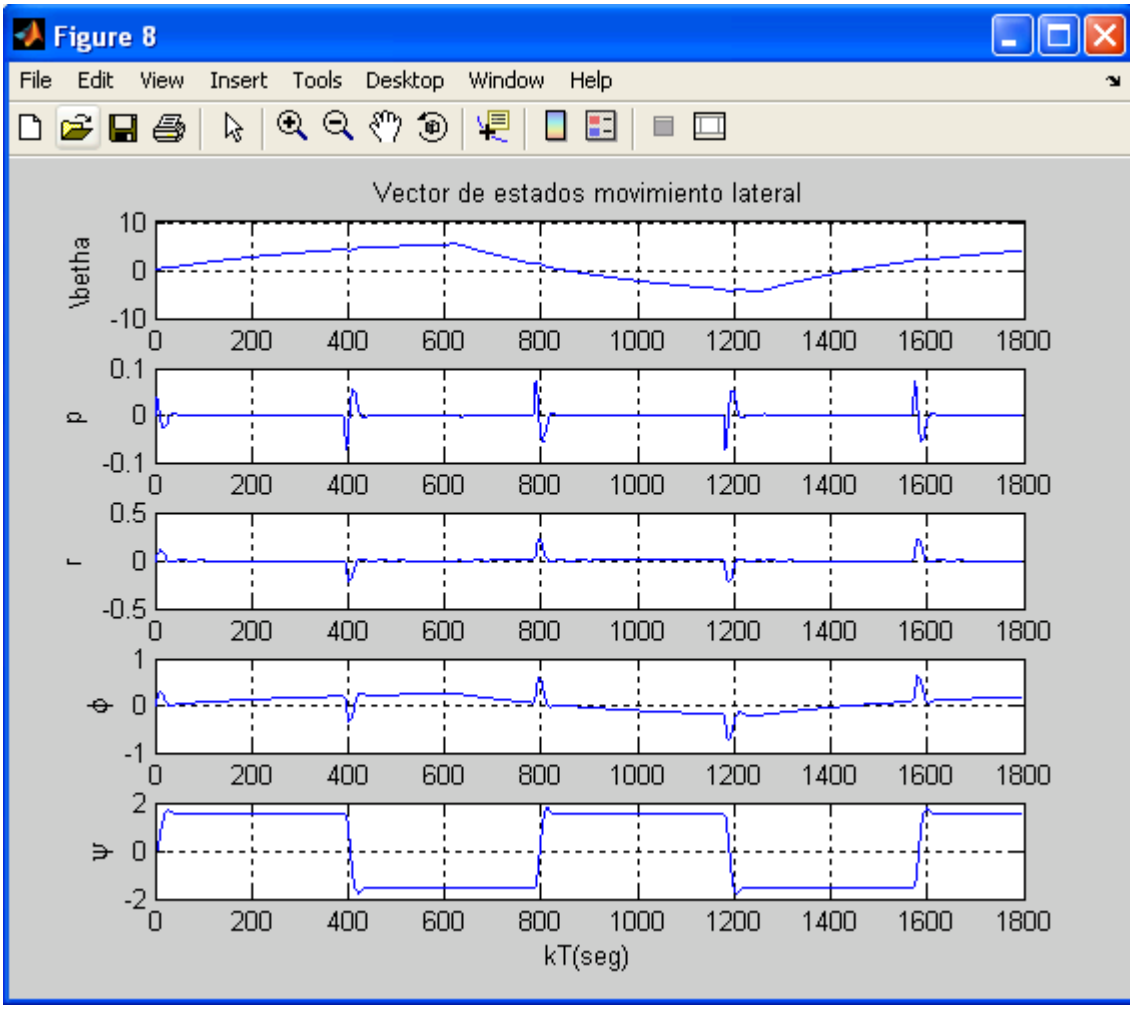
En la figura que a continuación se muestra, el moviendo lateral en lazo cerrado para entradas de referencia suaves tipo ondas cuadradas, para en ángulo respecto al eje +x se aprecian unos overshot producto del cambio brusco de la trayectoria del sistema.



Movimiento lateral en lazo cerrado para entradas de regencia ondas cuadradas.

FIG. 54

Se aprecia los vectores de estado del movimiento lateral en lazo cerrado para entradas de referencia suaves tipo ondas cuadradas.



Vector de estados en movimiento lateral.

FIG. 55

Donde:

betha = Ángulo de deslizamiento lateral [rad].
p = Tasa de rotación [rad/s].
r= Tasa de giros [rad/s].
phi = Ángulo de rotación del fuselaje[rad].
psi = Ángulo de giro [rad].

5.- LÓGICA DIFUSA APLICADO AL CONTROL DEL TIMÓN Y ELEVADOR.

La lógica difusa fue creada para emular la lógica humana y tomar decisiones acertadas a pesar de la información. Es una herramienta flexible que se basa en reglas lingüísticas dictadas por los expertos. La lógica difusa es un conjuntos de principios matemáticos basados en grados de membresía o pertenencia, cuya función es modelar información. Este modelado se hace con base en reglas lingüísticas que aproximan una función mediante la relación de entradas y salidas del sistema (composición). Esta lógica presenta rangos de membresía dentro de un intervalo entre 0 y 1, a diferencia de la lógica convencional, en la que el rango se limita a dos valores el cero y el uno[RNSD01].

Mediante el uso de la lógica difusa se puede representar la forma de la lógica humana, por ejemplo en afirmaciones como “el día es caluroso” en este caso se sabe que hay alta temperatura, pero no se sabe a qué temperatura exactamente se esta refiriendo. Se puede utilizar esta teoría cuando se quiere automatizar un proceso que controla un trabajador, el sistema difuso tendrá la tarea de emular a dicho trabajador. Además, si se toma en cuenta que el trabajador hace juicios con base en su criterio y experiencia, y que estos juicios y decisiones se realizan en forma lingüística, se puede notar que un sistema convencional no maneja este tipo de entradas, mientras que un sistema difuso si lo hace.

Otra ventaja del sistema de control basado en lógica difusa es que no es necesario conocer un modelo matemático del sistema real, pues se puede ver como una caja negra a la cual se le proporcionan entradas y a través del sistema esta planta generará la salida deseada. En el control convencional si es necesario conocer la planta del sistema. Para desarrollar un control con estas características, es necesario un experto, en este caso el trabajador, del cual se tomará un registro de las situaciones que se le presentan, así como de la solución que él les da. Esta experiencia se traduce en reglas que usan variables lingüísticas.

Para hacer este control es necesario tener las entradas del sistema y éstas se van a mapear a variables lingüísticas. A este mapeo se la llama difusificación. Con

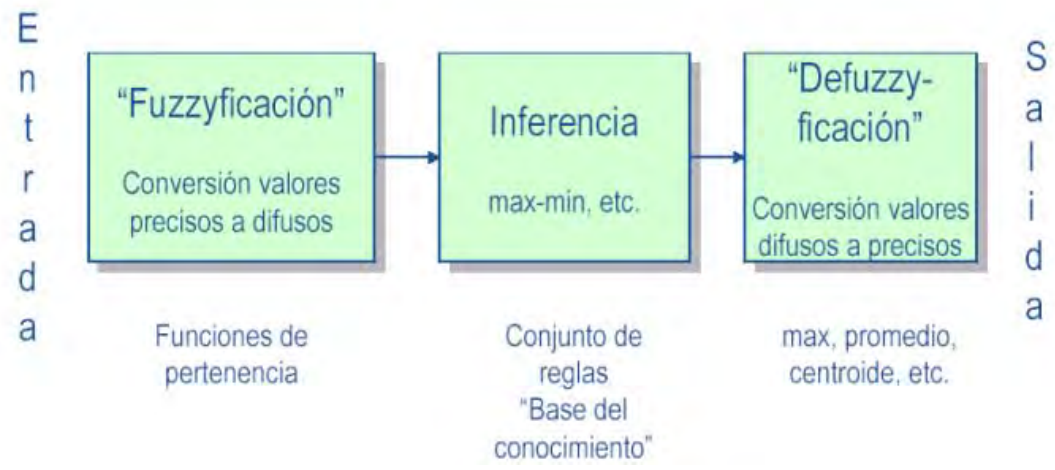
estas variables se forman reglas, las cuales serán las que regirán la acción de control que será la salida del sistema [Springer07].

La forma básica de un controlador difuso consta de tres partes:

REGLAS (INFERENCIA): Estas son reglas que dictan la acción de control que se va a tomar. Éstas se derivan de un experto. Dichas reglas tiene la estructura de relaciones. La lógica difusa se basa en relaciones, las cuales se determinan por medio de cálculo de reglas.

DIFUSIFICADOR (FUZZYFICACIÓN): Es el nexo entre las entradas reales y difusas. Todas las entradas necesitan ser mapeadas a una forma en que las reglas puedan utilizarlas.

DESFUSIFICADOR (DEFUZZYFICACIÓN): Toma el valor difuso de las reglas y genera una salida real.



Esquema del controlador difuso en sus tres etapas.

FIG. 56

En conjuntos difusos la función de pertenencia que se utiliza es la μ. Esta toma los valores entre cero (0) y uno (1), la forma de representación de los conjuntos difusos puede ser de dos maneras: de forma continua o discreta.

Un conjunto difuso se escribe de la siguiente manera: $\tilde{A} = \{a, b, c\ \}$,

En la lógica difusa los conjuntos se pueden reprensar en forma continua o discreta.

Un conjunto difuso discreto se representa: $\tilde{A} = \left\{ \sum_i \frac{\mu_A(X_i)}{X_i} \right\}$

Un conjunto difuso continuo se representa: $\tilde{A} = \left\{ \int \frac{\mu_A(X)}{X} \right\}$

Un conjunto convencional se define por una función característica, que se conoce también como función de pertenencia. El Símbolo de la integral denota unión de elementos del conjunto.

Las operaciones de conjuntos en conjuntos difusos son:

- Intersección: μ (A and B) = min{μ (A), μ (B)}
- Union: μ (A or B) = max{μ (A), μ (B)}
- Complemento: μ (not A) = 1 - μ (A)

Inferencia con reglas difusas: En el caso difuso, si el antecedente tiene cierto grado de pertenencia, entonces el consecuente también lo tiene pero siempre con grado no mayor.

Pasos:

1. Discriminar las variables de los patrones (ej., Temperatura: helado, frio, templado, caliente, hirviendo) y definir sus conjuntos difusos
2. Discriminar las variables de control (e.j., mucho más frio, más frio, no tocar, más caliente, mucho más caliente) y definir sus conjuntos difusos
3. Analizar cada regla a ver si se cumplen sus antecedentes
4. Combinar los valores difusos

5.

Defusificar el resultado

La defusificación artimética se puede hacer:

- Tomando el valor máximo (*maximum method*)
- Tomar el centro del área (*moments method*)

Existen operadores difusos (*hedges*) que pueden aplicarse, aparte de las operaciones básicas de conjuntos. Pueden verse como *modificadores* a los ``términos'' difusos.

- concentración (e.j., alto →muy alto)

$$\text{con } (\mu)(\alpha)=\mu^2(\alpha)$$

- dilatación (e.j., alto →medio alto)

$$\text{dil } (\mu)(\alpha)=\mu^{1/2}(\alpha)$$

- normalización

$$\text{norm } (\mu)(\alpha)=\mu(\alpha)/\max(\mu(\alpha))$$

Si se quiere combinar dos predicados (ejm, alto y pesado), tenemos que definir una relación (matriz).

Las operaciones de conjuntos difusos se pueden extender fácilmente a relaciones [Springer07].

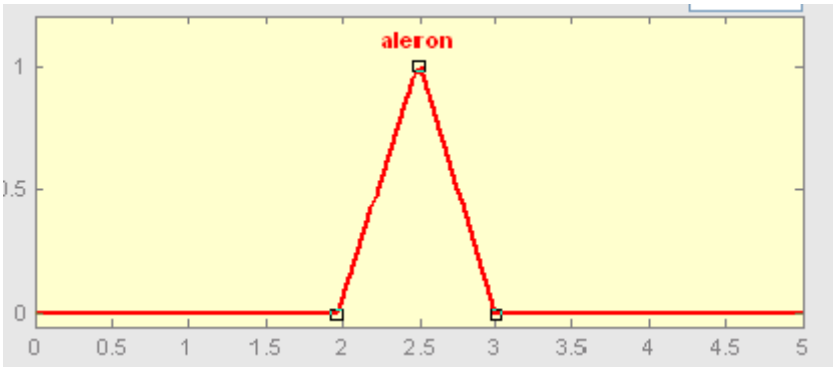
Para hacer la composición de dos relaciones μ:A X B→I y λ:B x C→I

$$\mu(a,b)=\text{MaxMin}\left\{\mu(a,b^I),\lambda(b^I,c)\right\}$$

5.1.-FUNCIÓN TRIANGULAR.

La función triangular consta de una de pendiente positiva constante hasta alcanzar la unidad, y una vez que lo ha logrado desciende de manera uniforme.

La función triangular es muy adecuada para definir situaciones en las que se tiene un valor óptimo central, el cual se va perdiendo conforme uno se aleja de él. Para la elaboración del presente trabajo se ha utilizado esta función tanto en la etapa de Difusificador (entradas) y en la etapa de Desfusificador (salidas) [FUZZY96].

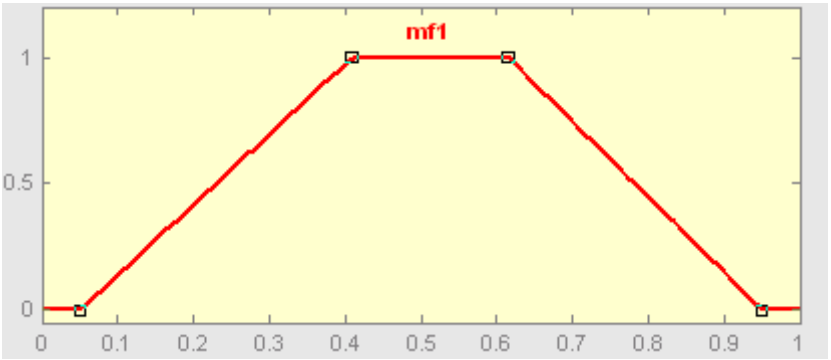


Función triangular.

FIG. 57

5.2-FUNCIÓN TRAPECIO O PI

La función de membresía, no sólo se tiene un valor para el cual la pertenencia es unitaria, sino toda un franja que varía su ancho dependiendo del fenómeno observado. Esta función se emplea cuando hay un rango de valores óptimos, alrededor de los cuales las condiciones no son adecuadas [FUZZY96].

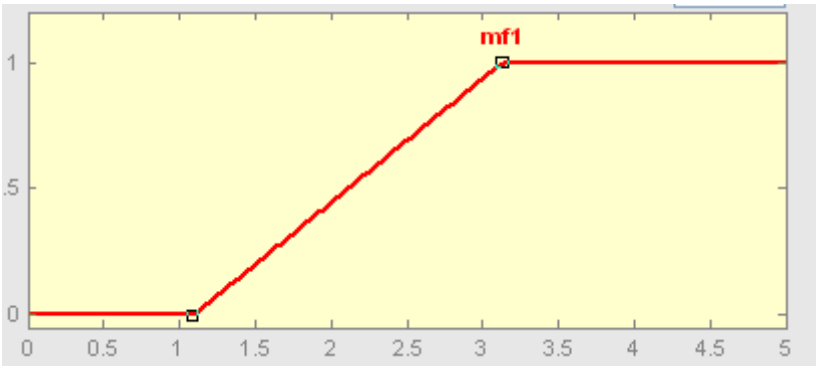


Función Trapecio o Pi.

FIG. 58

5.3.-FUNCIÓN DE SATURACIÓN.

Es la función más sencilla. Tiene un valor de 0 hasta cierto punto y después crece con pendiente constante hasta alcanzar el valor de 1, en donde se estaciona. Este tipo de funciones describe muy bien situaciones en donde se alcanza un nivel máximo a partir de cierto punto [FUZZY96].

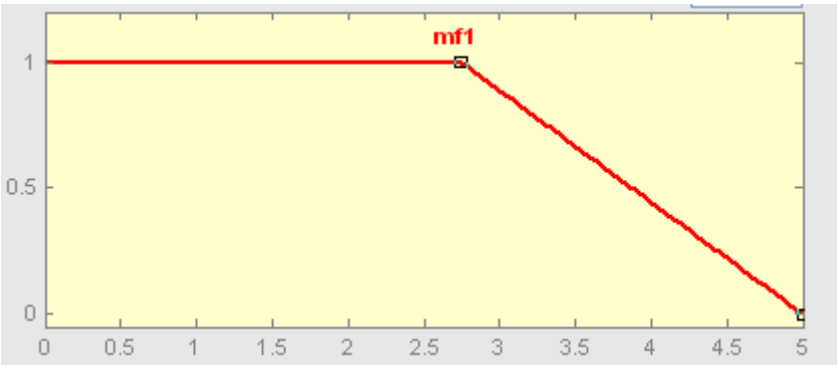


Función de saturación.

FIG. 59

5.4.-FUNCIÓN HOMBRO.

Es la contraparte de la función saturación. Este tipo de funciones se inicia en un valor unitario y de desciende con constante pendiente hasta alcanzar el valor de 0. Este tipo de función es útil cuando el grado de pertenencia es total en valores pequeños y decae conforme el valor de la variable aumente [FUZZY96].



Función hombro.

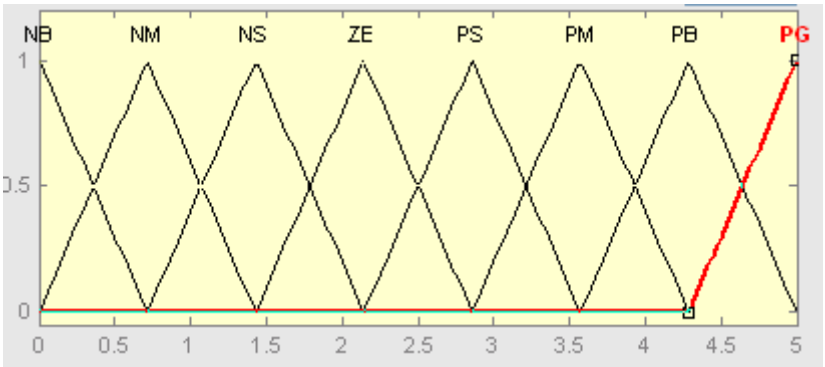
FIG. 60

5.5.-INTERFAZ DE DIFUSIFICACIÓN.

Mide los valores de las variables de entrada para realizar un mapeo a escala que transfiere el rango de valores de las variables a un universo difuso. La difusificación convierte los datos de entrada en valores lingüísticos que son las etiquetas de las funciones de pertenencia.

La presentación de información a través de conjuntos difusos puede realizarse en forma discreta. Al discretizar la información segmenta un universo en un número definido de partes, es posible definir un conjunto difuso asignando un grado de pertenencia a cada elemento genérico del nuevo universo discreto. Los niveles de discretización tienen una gran influencia en la obtención de un control. Para la discretización es necesario realizar un mapeo a escala para transformar valores medidos en las variables a valores del universo discreto, ya sea de manera uniforme o no uniforme o combinaciones de ambos.

Una variable lingüística en general se asocia a un conjunto de términos. Para encontrar cuántos términos son necesarios en un conjunto se emplean particiones difusas. El número de conjuntos difusos determina la complejidad del controlador [FUZZY96].



Particiones difusas con distinto número de términos.

FIG. 61

5.6.-BASE DE CONOCIMIENTOS.

La base del conocimiento contiene toda la información de la aplicación que se va a controlar, así como las metas del controlador. Consta de una base de datos y una base de reglas lingüísticas para controlar la variable. La base de datos proporciona las definiciones para el establecimiento de reglas y la manipulación de datos difusos. La base reglas caracteriza las metas de control y la política que utilizan los expertos para llevar a cabo el control.

Un algoritmo de control difuso debe ser capaz de inferir una acción de control correspondiente para cada estado que va a controlar.

5.7.-LÓGICAS DE DECISIONES.

La lógica utilizada para tomar decisiones dentro de un controlador difuso es el núcleo del mismo. A partir de las misma se simula la lógica que utilizan las personas para tomar decisiones, con base en conceptos difusos y en la inferencia de acciones de control, empleando implicaciones y las reglas establecidas según la base de conocimientos.

5.8.-INTERFAZ DE DESDIFUSIFICACIÓN.

La interfaz de desdifusificación se encarga del mapeo a escala que convierte el rango de valores de las variables de salida. La desdifusificación es la herramienta para obtener la acción de control nítida a partir de una acción de control difusa.

5.9.-MÉTODO DE CENTRO DE ÁREA O GRAVEDAD.

Este método es que se utiliza para encontrar el valor de la salida, en el controlador difuso. La metodología consiste en cortar la función de membresía al grado de la membresía respectiva, segmenta las funciones de membresía, generando a cada función dos áreas. El área inferior que se forma es la que se toma para hacer el cálculo. Se sobreponen todas estas áreas y se saca el centroide de la superposición, el cual nos dice la salida real del sistema [FTB01].

El método del centro de área o centro de gravedad se puede representar en forma discreta:

$$Salida = \frac{\sum_{i} \mu(x).x}{\sum_{x=a}^{b} \mu(x)}$$

ó contínua:

$$Salida = \frac{\int_{a}^{b} \mu(x).xdx}{\int_{a}^{b} \mu(x).dx}$$

Diseño del controlador con base en Mandani.

En el diseño del controlador para el vuelo no tripulado se utilizará el procedimiento de Mandani que comprende el siguiente procedimiento:

1. Siendo el error la diferencia entre el valor deseado y el valor real de la variable que se va a controlar, esto es: $\varepsilon = V_{deseado} - V_{real}$, se seleccionan las funciones de pertenencia que realizarán la difusificación.

2. Se establecen las reglas a partir de proposiciones condicionales, y el dispositivo de inferencia será una composición máx-mín que ha sido previamente definida como:

$$X_T(x,z) = v_{y \epsilon Y}(X_R(x,y) \wedge X_s(y,z))$$

3. Se seleccionan las funciones de pertenencia para la desdifusificación y el método que se va a usar para encontrar el valor nítido de la salida, normalmente correspondiente el método del centroide:

$$\frac{\sum \mu(x).x}{\sum \mu(x)} (discreto) \qquad \frac{\int \mu(x).xdx}{\int \mu(x)dx} (continuo)$$

5.10.-REGLAS BÁSICAS DE CONTROL.

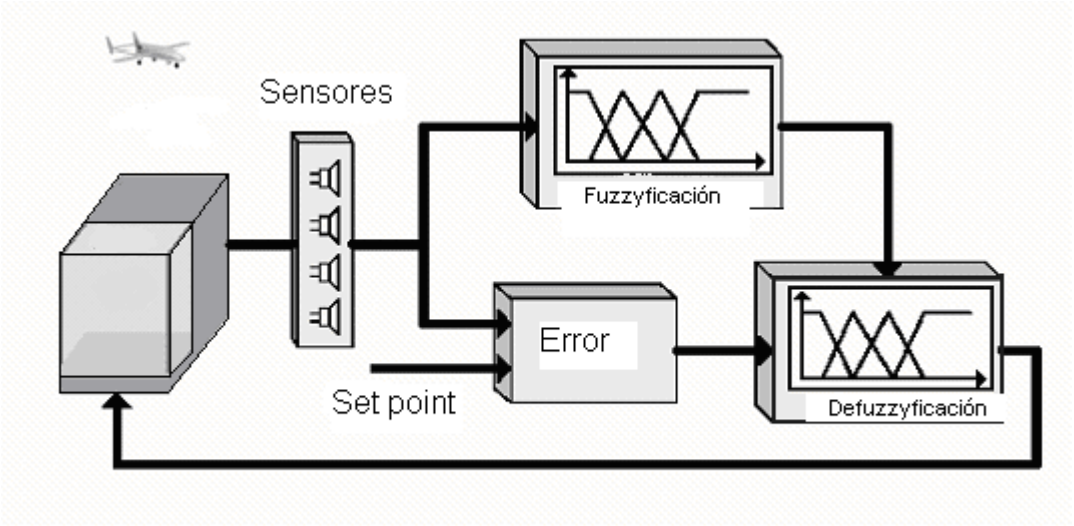
La idea del control por lógica difusa es que el avión sea guiado en forma suave a lo largo de su recorrido de manera de minimizar los efectos que podrían causar los errores de posición.

La lógica difusa es una técnica de control basada en reglas del tipo:

IF Rad_E IS Positivo AND Rad_dE IS Zero THEN dtita Poco_Izquierda
IF Rad_E IS NearZero AND Rad_dE IS Zero THEN dtita Derecho
IF Rad_E IS Negativo AND Rad_dE IS Zero THEN dtita Poco_Derecha

Se puede observar que también entran en juego sustantivos como Rad_E (error de radio) y adjetivos que se refieren a esos sustantivos como son Positivo, Poco_Izquierda, se nota también que puede aplicarse a sistemas de múltiples entradas y salidas.

A diferencia de la lógica convencional, las transiciones entre los estados no son discretas sino que son suaves, evitando de esta manera los movimientos bruscos. Las reglas y los adjetivos deben ser definidos para el proceso que desea controlar, y lo hace de manera que el sistema trate de imitar las acciones de control que tomaría el mismo operador según las condiciones de las entradas y salidas.



Arquitectura básica de la navegación basada en la lógica difusa.

FIG. 62

5.11.-EXPLICACIÓN DEL MODELO

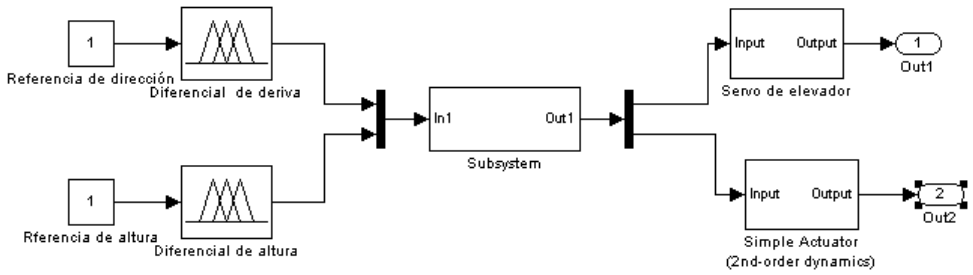


Diagrama de bloques para la estrategia del control difuso del avión [Fuzzy07].

FIG. 63

5.12.-CONTROLADOR DIFUSO PARA LA ALTURA.

Consiste en controlar por medio de las teorías de lógica difusa las variables del error en la altitud y la tasa de ascenso, la idea es generar una deflexión en el elevador para controlar cuanto puede bajar o elevar el avión. Para el ascenso se observan tres zonas, la que se encuentran más cerca de cero se utiliza para aproximarse a la referencia. La zona trapezoidal se utiliza para movimientos entre dos altitudes diferentes, la zona más externa es la que se debe de evitar porque es un zona inestable para el control del avión[Fuzzy07].

Las funciones que controlan el error en la altitud se encuentran simétricamente distribuidas para que el avión pueda aterrizar. El ángulo que el elevador puede subir o bajar es de 10°.

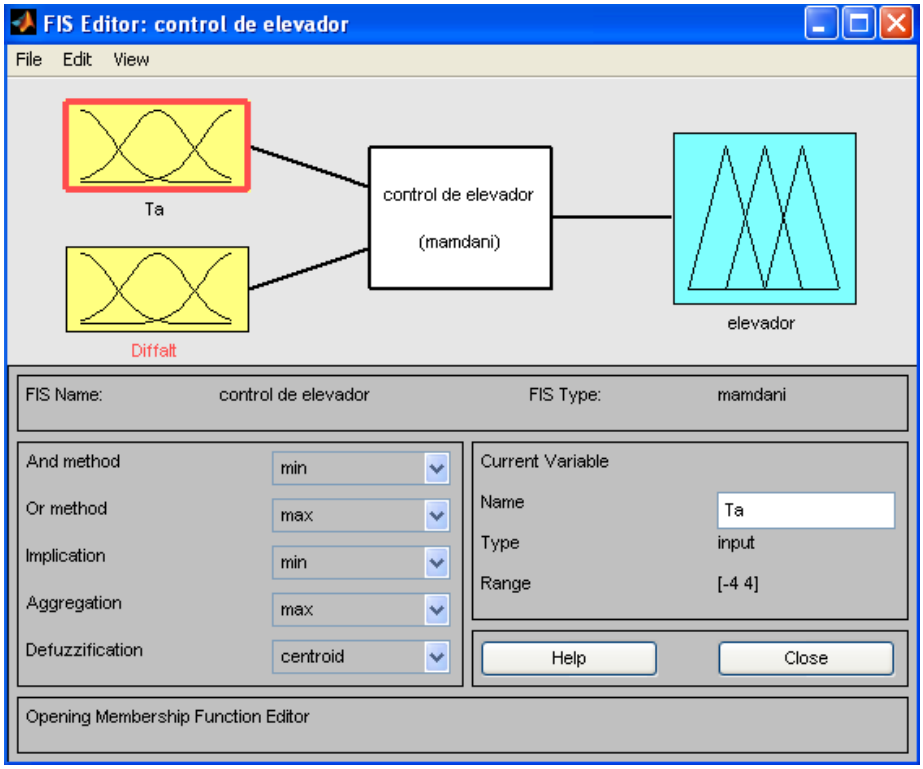
Ta: Tasa de ascenso. Valt en el programa en c.
Diffalt: Es la diferencia entre la altitud medida y su referencia.

Donde:
AN: variación Alta y negativa de altura. (HN en el programa en c.)
BN: variación Baja y negativa de altura. (LH en el programa en c.)
Z: variación pequeña de altura.
BP: variación Baja y positiva de altura. (LP en el programa en c.)
AP: variación Alta y positiva de altura.(HP en el programa en c.)

Las siguientes figuras, son una muestra de la simulación realizada en el programa Matlab con el toolbox de lógica difusa.

Lo que see muestra en la siguiente figura el entorno del toolbox de matlab, en donde se realizó la simulación de la lógica difusa. Se utilizo el entorno referido al de mandani. Aquí se puede apreciar que el sistema tiene dos entradas(Tasa de

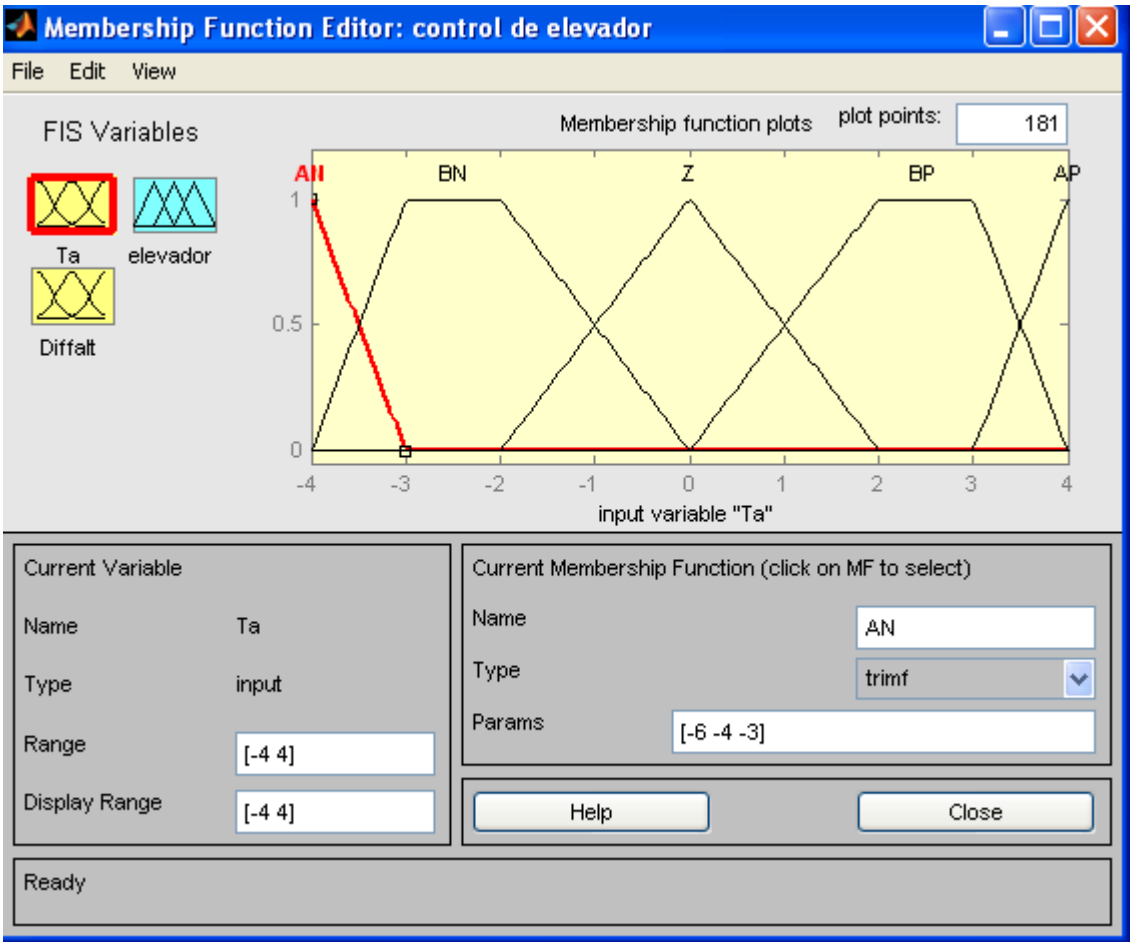
ascenso y la diferencia entre la altitud medida y su referencia) y una salida (El elevador del avión que sólo es posible variar entre los -10 y 10 grados) [Fuzzy07].



Presentación inicial de las variables de entrada(Ta, Diffalt) y la salida (Elevador)

FIG. 64

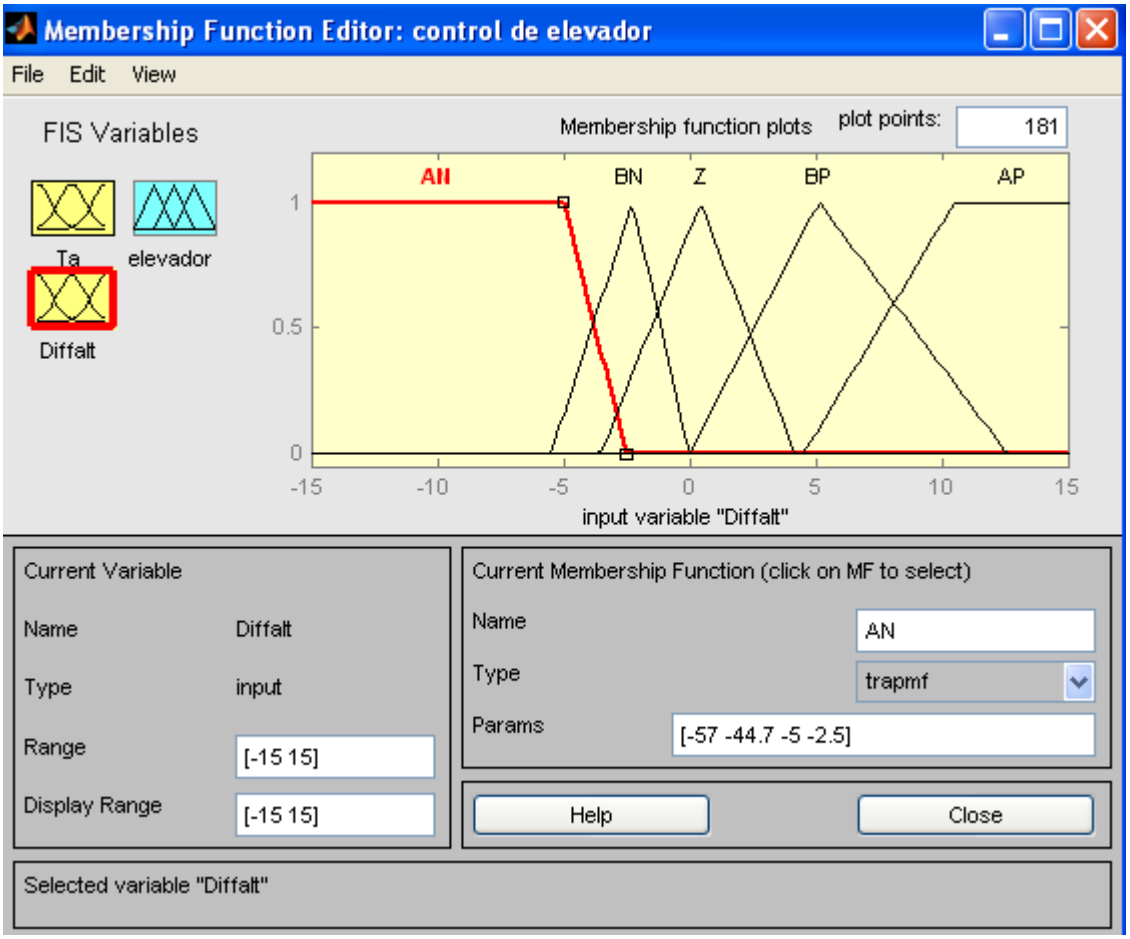
La siguiente simulación nos presenta 5 reglas utilizadas en la variable de entrada Ta (Tasa de ascenso). Dos de ellas corresponden a la función trapecio y se emplean cuando hay un rango de valores óptimos. Una función triangular en la parte central en la que se obtiene un valor óptimo en el centro y se va perdiendo conforme se aleja de él. En los extremos hay dos funciones. Una derecha la de saturación que describe situaciones en donde se alcanza un nivel máximo de 4 (es la rapidez con que sube o baja el avión), y otra a la izquierda llamada función hombro que nos muestra el grado de pertenencia en contraparte a la función de saturación [Fuzzy07].



Variable de entrada correspondiente a la tasa de ascenso (Ta) con el conjunto de funciones.

FIG. 65

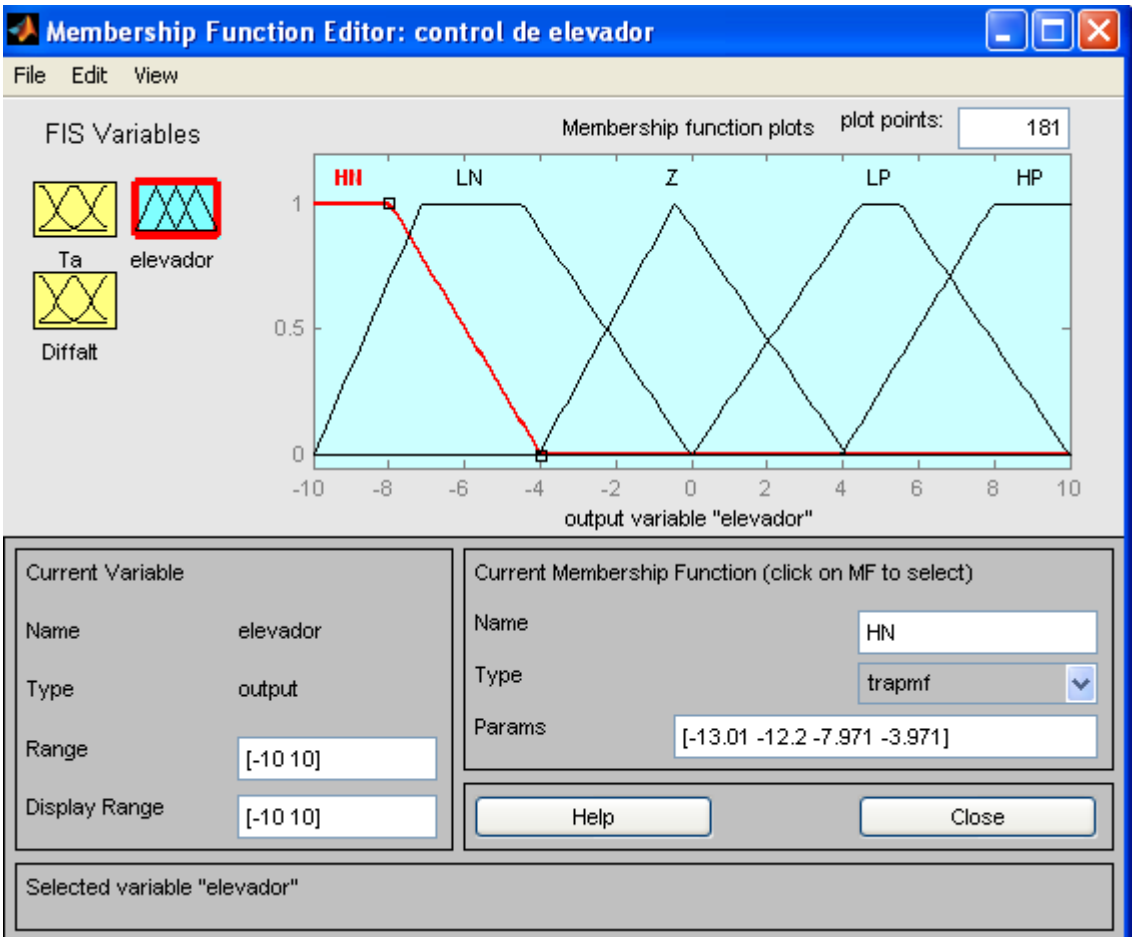
En la variable de entrada correspondiente a la diferencia de altura medida a través de el sensor Zlog y la altura deseada. En esta simulación hay 5 funciones, 3 de ellas son de tipo de triangulares en donde el valor óptimo son los puntos centrales, y una de saturación y otra función hombro en donde se tonaran situaciones en donde se alcanzan valores máximos como por ejemplo -15 y 15 metros de diferencia [Fuzzy07].



Variable de entrada correspondiente a la diferencia de alturas (Diffalt) con el conjunto de funciones.

FIG. 66

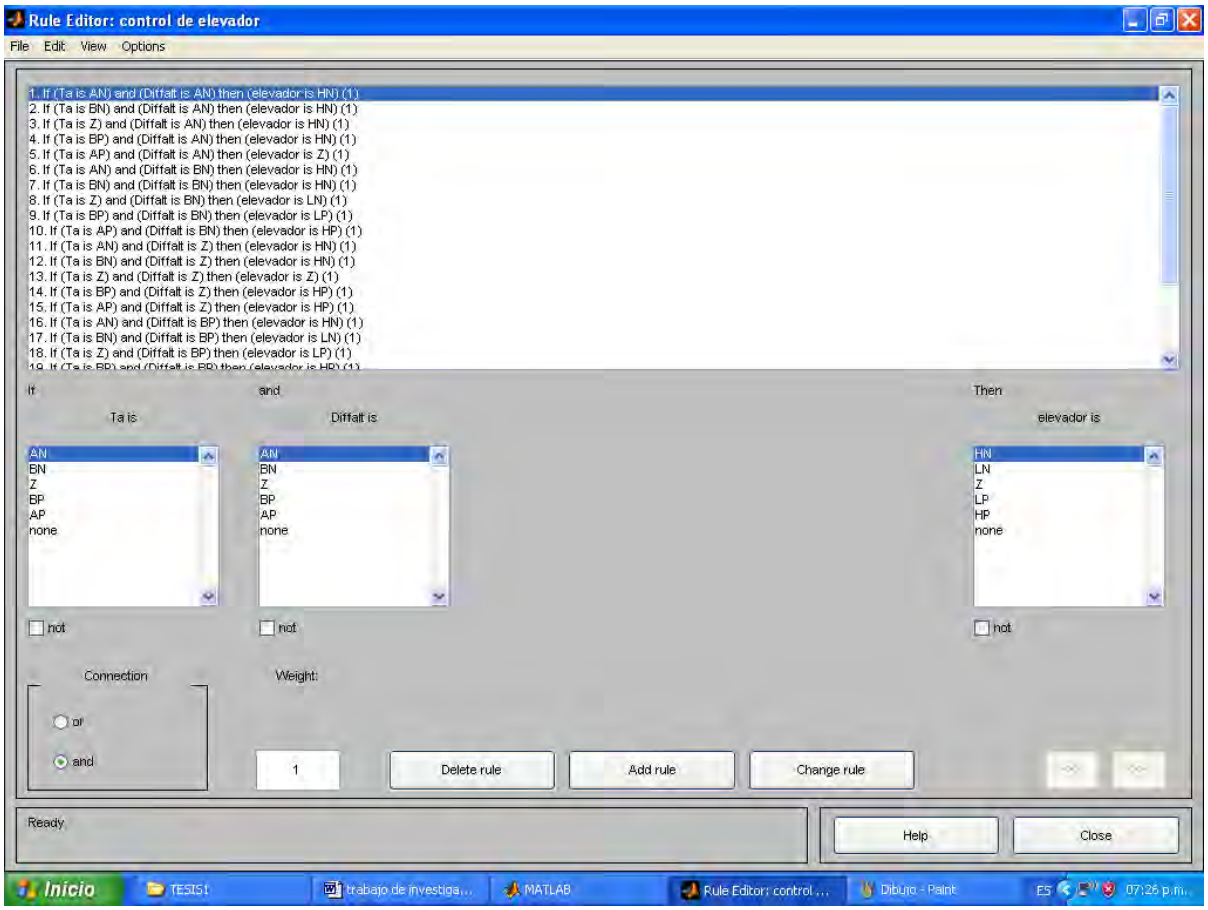
La salida de la desfuzificación es la variable elevador en donde puede fluctuar entre los 10 y -10 grados. Aparentemente esto podría parecer un ángulo pequeño pero debido al tamaño del aeromodelo es más que suficiente para poder subir y bajar sin dificultad. La salida de la desfuzificación es llevada al actuador que es servo motor del aeromodelo[Fuzzy07].



La función de salida correspondiente al elevador.

FIG. 67

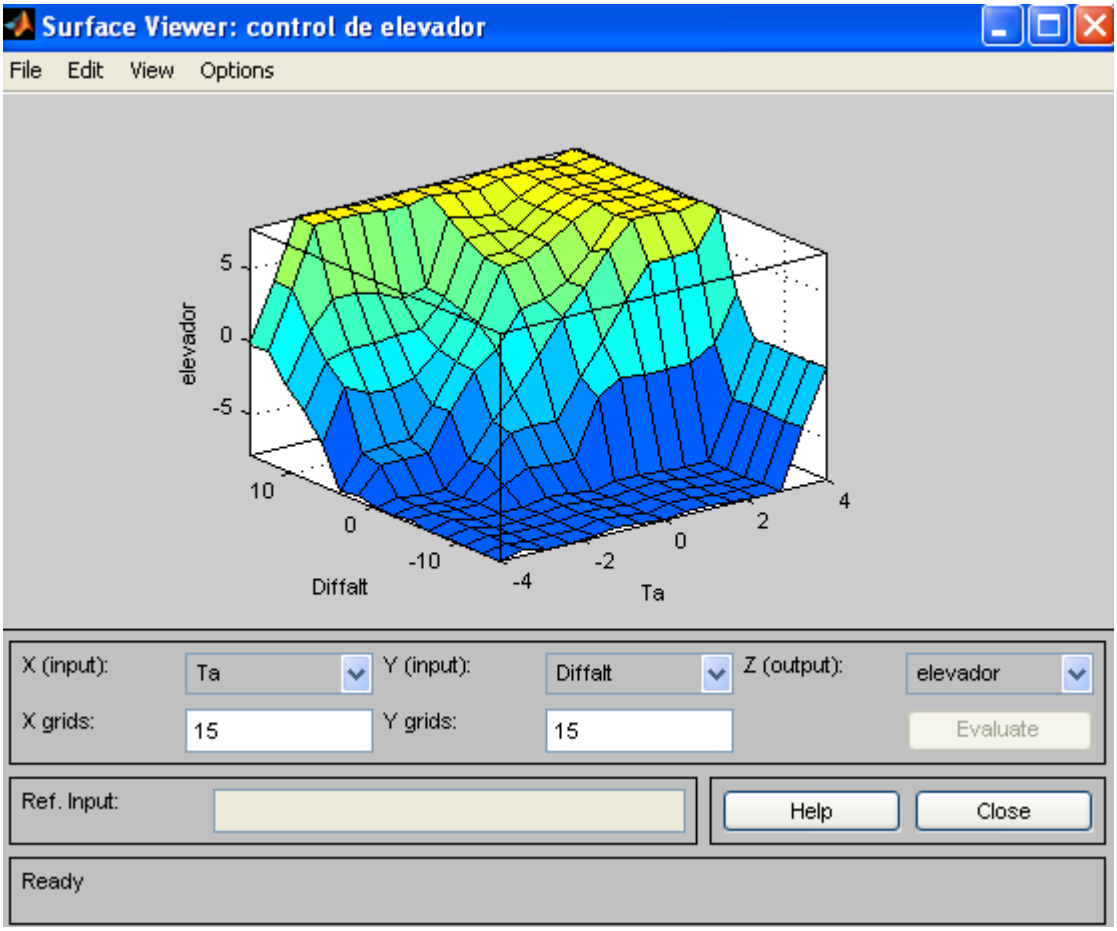
Las reglas de que constituyen la parte de la lógica difusa para el control del alerón son de 25. Estas nos brindan un control absoluto sobre el alerón. En este sentido la experiencia en el aeromodelismo me ha permitido poder plasmar estar reglas de control basadas en la experiencia propia [Fuzzy07].



Editor del matlab, con el conjunto de reglas de lógica difusa.

FIG. 68

En esta simulación se aprecia el acople entre las dos variables de entrada y la salida. Se aprecia la interrelación de las variables a través de las reglas de la lógica difusa y sus 25 reglas[Fuzzy07].



Superficie de la base de las reglas para el control difuso para el elevador del avión.

FIG. 69

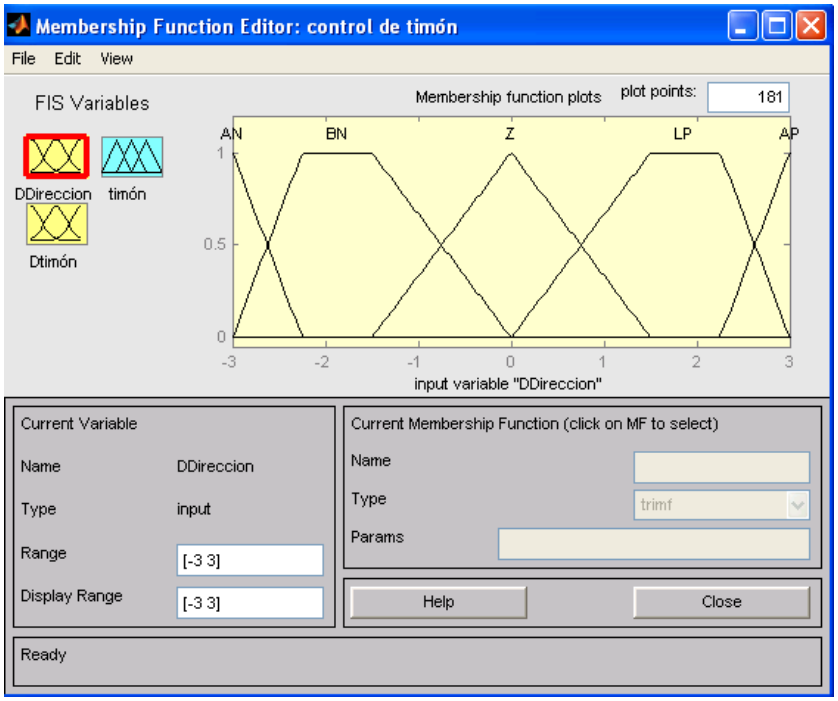
5.13.-CONTROLADOR DIFUSO PARA DIRECCIÓN (COLA).

Para poder controlar la deriva o dirección del avión se ha establecido un conjunto de reglas y condiciones también basados en el control en lógica difusa.

Al que destacar que los ángulos de movimiento del timón de cola son de 8 a -8 grados, con estos ángulos el aeromodelo es capaz de girar de forma completa los 360ºC sobre su eje Z.

Las variables de entrada a este controlador difuso son las siguientes:
DDirección: Razón de cambio de dirección. (Vdir en el programa en c.)
Dtimón: Es la diferencia entre la dirección medida y su la referencia. (Ddir en el programa en c.) Donde:
AN: variación Alta y negativa de timón. (HN en el programa en c.)
BN: variación Baja y negativa de timón. (LH en el programa en c.)
Z: variación pequeña de timón.
BP: variación Baja y positiva de timón. (LP en el programa en c.)
AP: variación Alta y positiva de timón. (HP en el programa en c.)

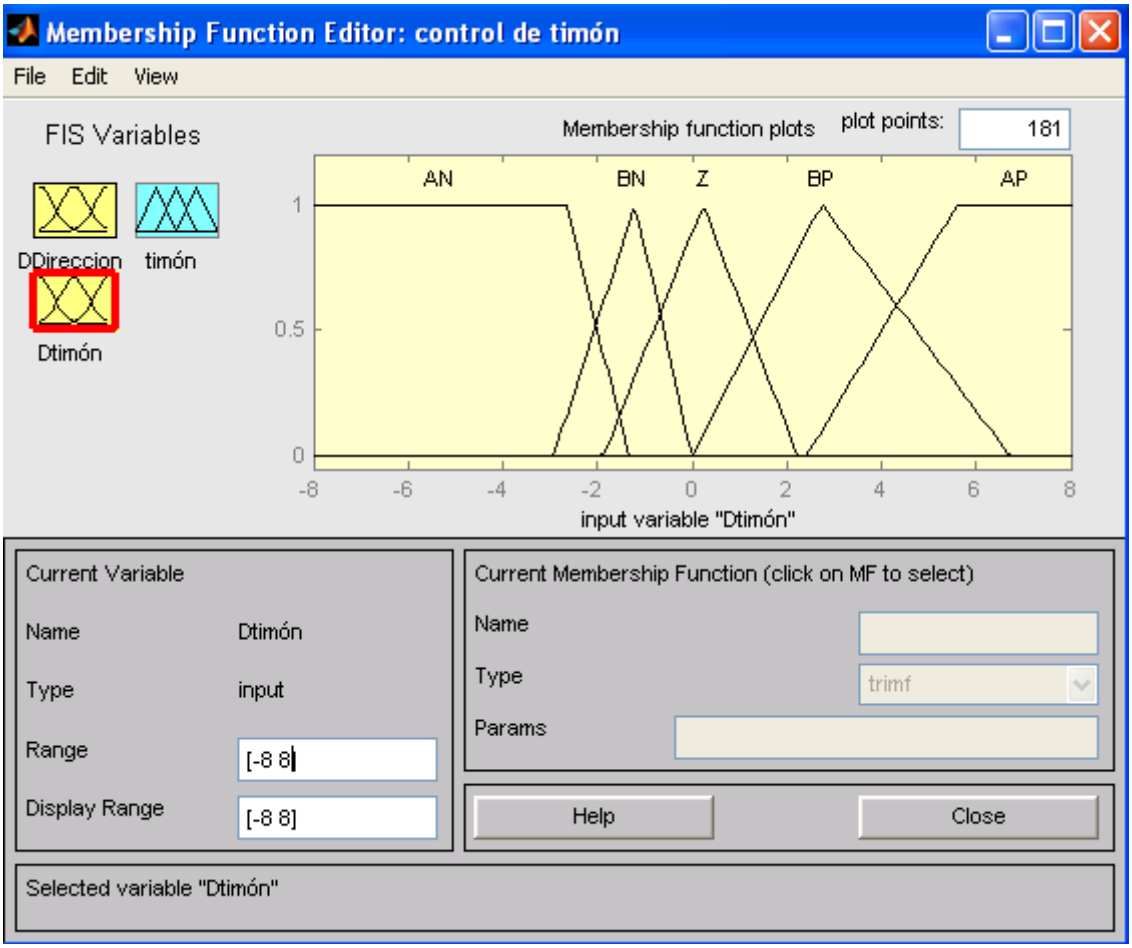
La variable razón de cambio de dirección es alimentada a través de los sensores Compás (valor real) y el GPS (es el valor de consigna). En ella se aprecian 5 funciones. Una triangular, dos trapecios y una función de saturación y otra de hombro [Fuzzy07].



Variable de entrada correspondiente razón de cambio de dirección (Ta) con el conjunto de funciones.

FIG. 70

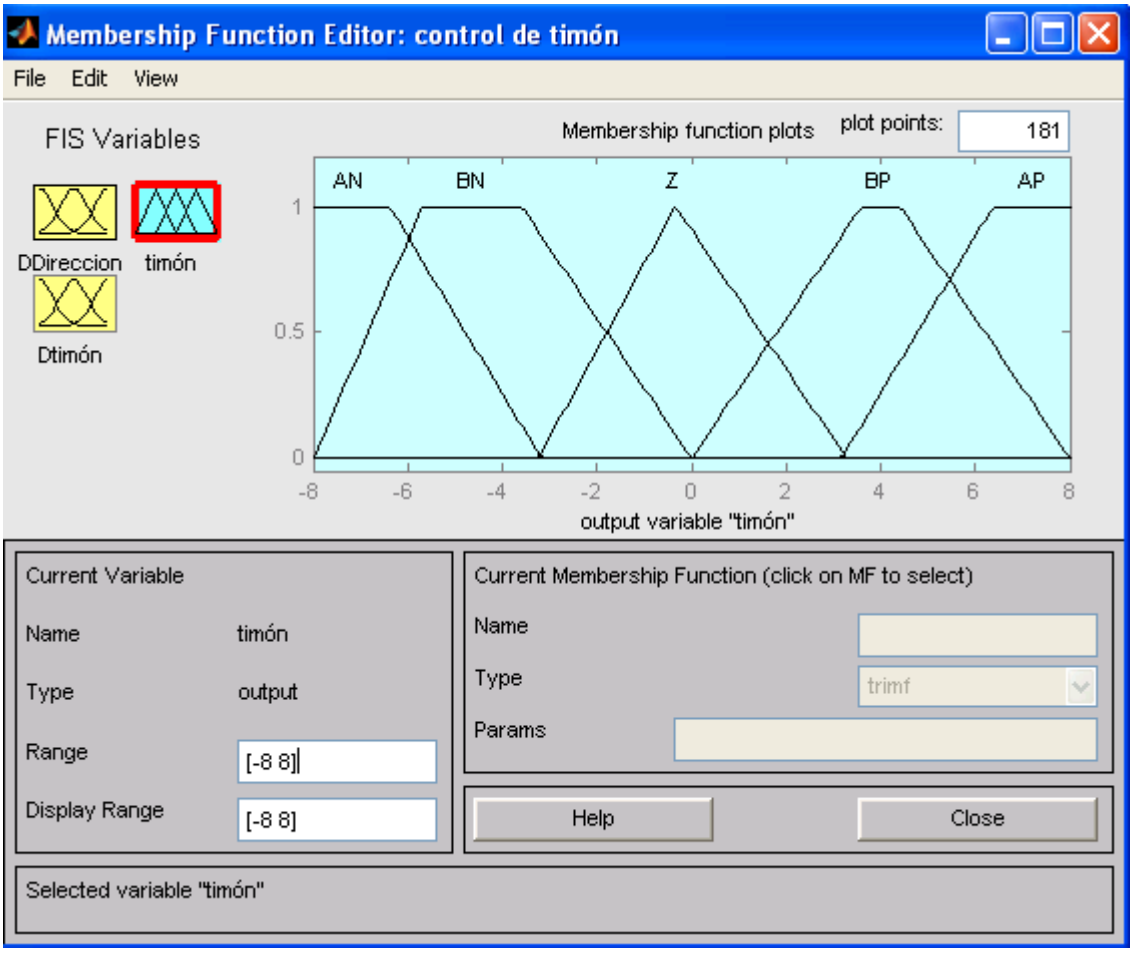
La variable Dtimón es la entrada al DSP que indica la posición del servo en un momento dado. Esta no puede tener más de 8 grados por que el avión podría entrar en una situación fuera de control, primero el avión se frenaría y luego podría perder sustentación[Fuzzy07].



Simulación de la variable diferencia entre la dirección medida y su la referencia (Dtimón).

FIG. 71

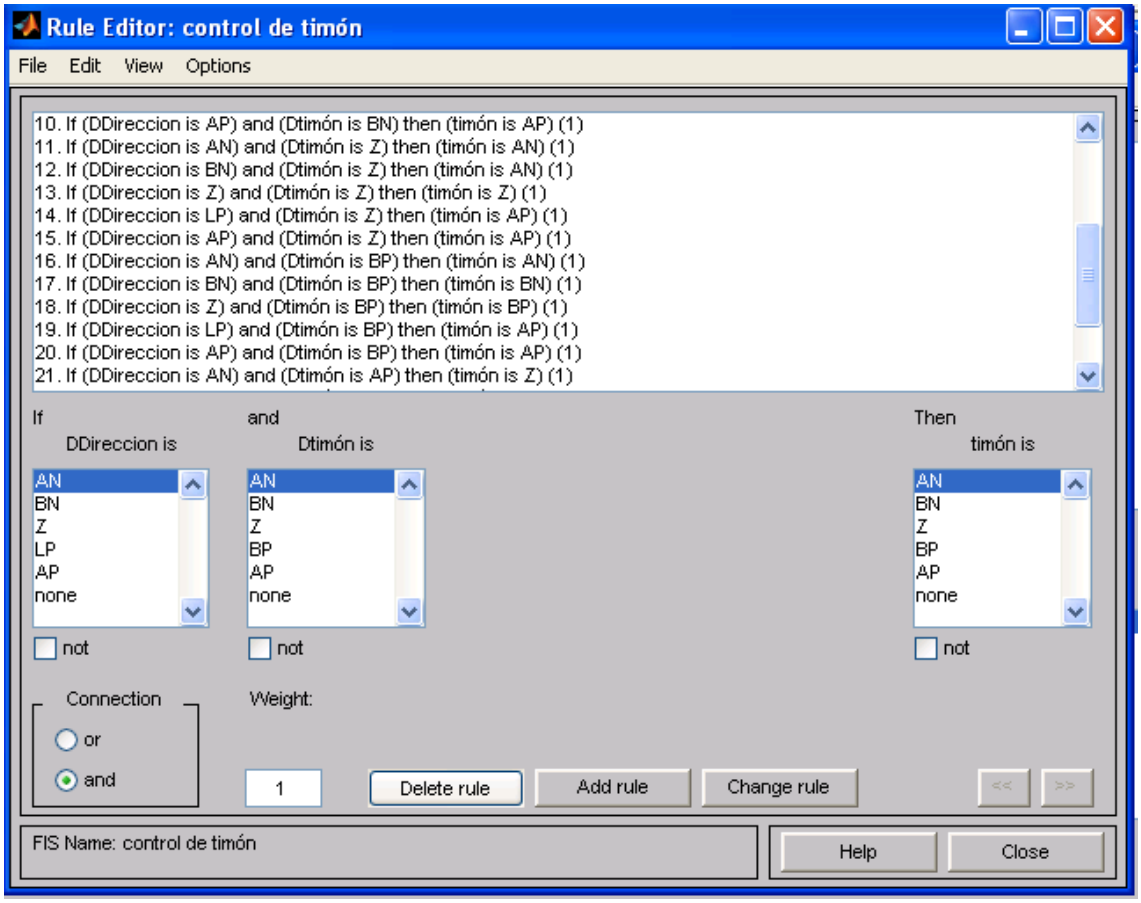
La salida del controlador difuso que es la dirección del timón se refleja en el actuador del timón de cola. Este movimiento es que finalmente nos da el giro del aeromodelo. La forma de las pertenencias del modelo de la variable de salida se aprecia en la siguiente simulación. Cabe mencionar que el ángulo de movimiento del timón de cola es de 10 a -10 grados[Fuzzy07].



La función de salida correspondiente al timón de cola.

FIG.72

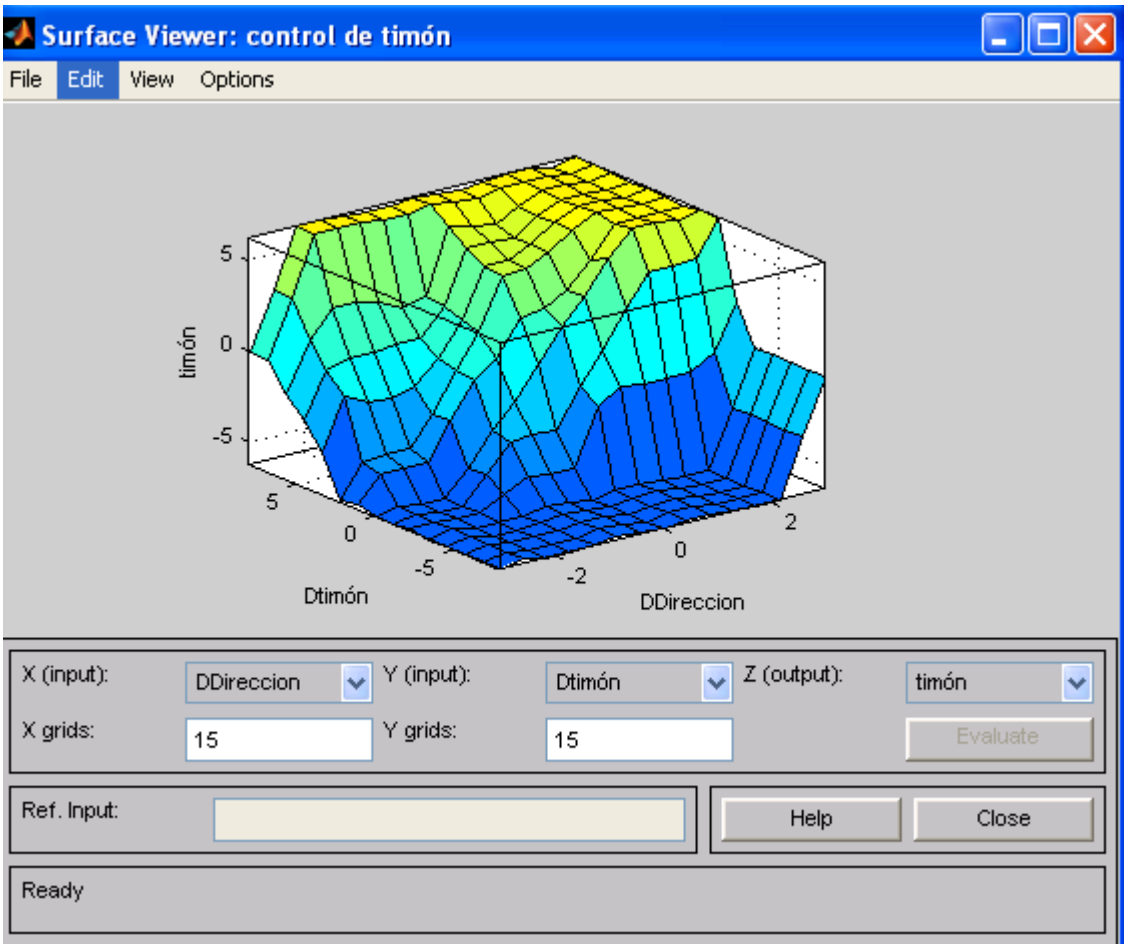
Las reglas que gobiernan el controlador difuso se muestran en la siguiente simulación. Son cerca de 25 reglas y con ellas se procede a realizar el controlador. Para poder procesar estas reglas se debe tener un procesador capaz de procesar una gran cantidad de información en un tiempo muy corto, por uso la utilización de un DSP[Fuzzy07].



Editor del matlab, con el conjunto de reglas de lógica difusa referidas al timón de cola.

FIG. 73

La correlación entre las variables de entrada y las variables de salida se muestran en la siguiente simulación. En ella se puede observar las variables acopladas trabajando en simultáneo y se puede apreciar con claridad el desempeño del controlador difuso[Fuzzy07].



Superficie de la base de las reglas para el control difuso para el timón del avión.

FIG. 74

Para que el sistema de control sea el más apropiado debería ser posible de implementar un control híbrido entre una estrategia de control PID y una estrategia de control de lógica difusa esto dará una mejor precisión y mayor estabilidad y así poder ajustar variaciones mucho más pequeñas. Esto dará a una estrategia cualitativa de control y una capacidad de implementar un control mucho más flexible, esto hará que el sistema pueda ajustarse a situaciones mucho más cambiantes que en algunos casos son imposibles de predecir.

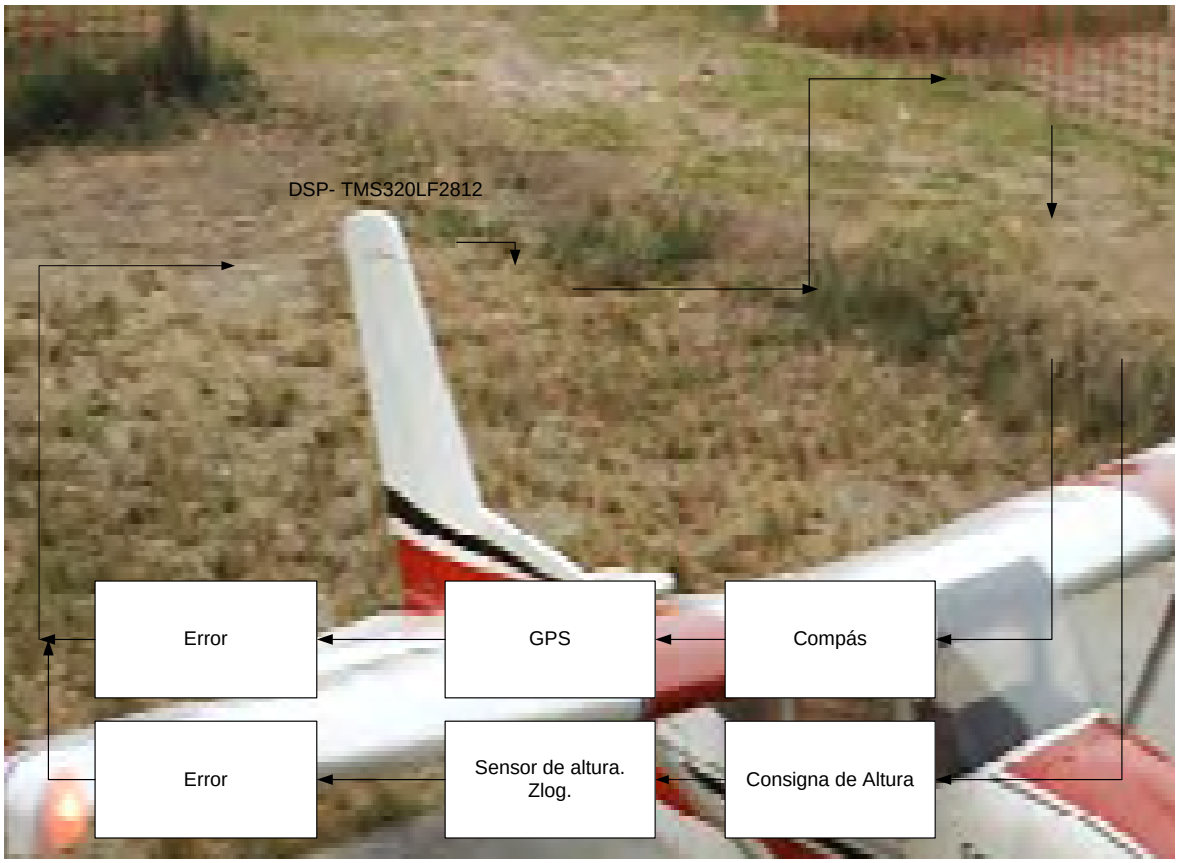
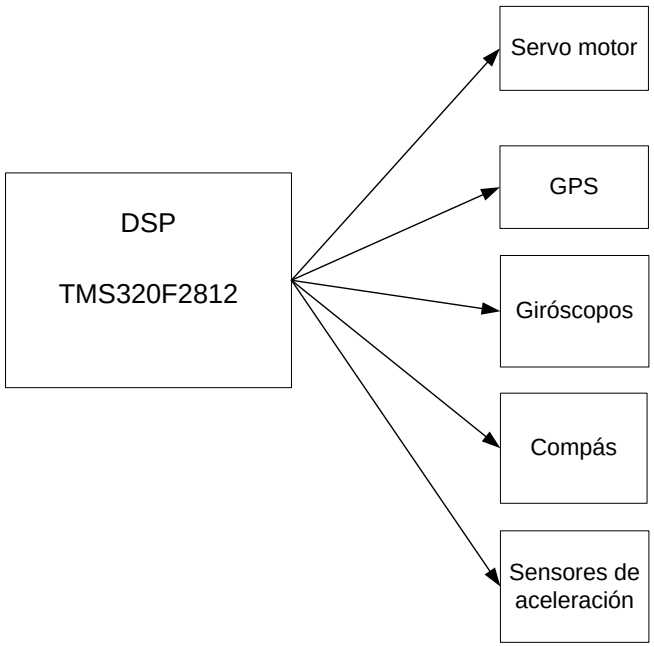


Diagrama de control del Cessna 182 con sus sensores y el sistema de control de lógica difusa.

FIG. 74.b.

6.- INTEGRACIÓN DE LOS DIFERENTES ELEMENTOS.

Para la integración del sistema se ha pensado en utilizar un IMU con un DSP y los diferentes sensores que continuación se muestran.



Esquema de los periféricos que controla el IMU.

FIG. 75

6.1 UNIDAD DE MEDICIÓN UNINVERSAL (IMU).

La unidad de medición inercial es un dispositivo de uso común en instrumentos de aeronáutica, vehículos no tripulados y vehículos de transporte público. La tarjeta es un sistema capaz de manejar un avión sin necesidad de un control remoto, es capaz de reconocer su posición mediante el uso de un GPS y asegurar su plan de vuelo con sensores de aceleración y giróscopos. Puede establecer velocidades, posiciones y aceleraciones en sus 6 grados de libertad. Contiene un reloj de tiempo real, un compás de navegación que usa el campo magnético de la tierra para orientarse y una memoria mini de 2Gb para grabar los datos de vuelo.

Esta tarjeta puede ser usado para aplicaciones de transporte público, como caja negra, para la detección de excesos de velocidad, rutas prohibidas, multas automáticas o investigación de accidentes vehiculares.

El módulo dispone del último modelo de DSP para aplicaciones de control, el TMS320F2812 de la Texas Instruments, con capacidad de procesamiento de 135 y 150 MIPS según el modelo, tiene dos unidades de manejo de relojerías y contadores, 4 timers, 16 canales de conversión AD con secuencia programada, 2 lectores de encoders en cuadratura, perro guardián, varios modos de operación en baja potencia, 15 PWMs, puertos seriales síncronos y asíncronos (2), puerto CAN, memoria externa de 256Kwords, reloj principal, interrupciones, memoria flash entre otros. Puede manejar fácilmente dos motores de inducción AC y/o dos motores DC con técnicas de modulación de ancho de pulso o espacios vectoriales PWM y SVPWM respectivamente, con algoritmos de control adaptivo, fuzzy, neuronal, deslizamiento u otros que requieran un procesamiento en tiempo real. Gracias a sus dos unidades de decodificación en cuadratura, se pueden manejar dos motores de posicionamiento e incluso se puede lograr hasta cuatro posicionamientos con dos de ellos desarrollados por software.

IMU

FIG. 76

6.1.-SENSORES:

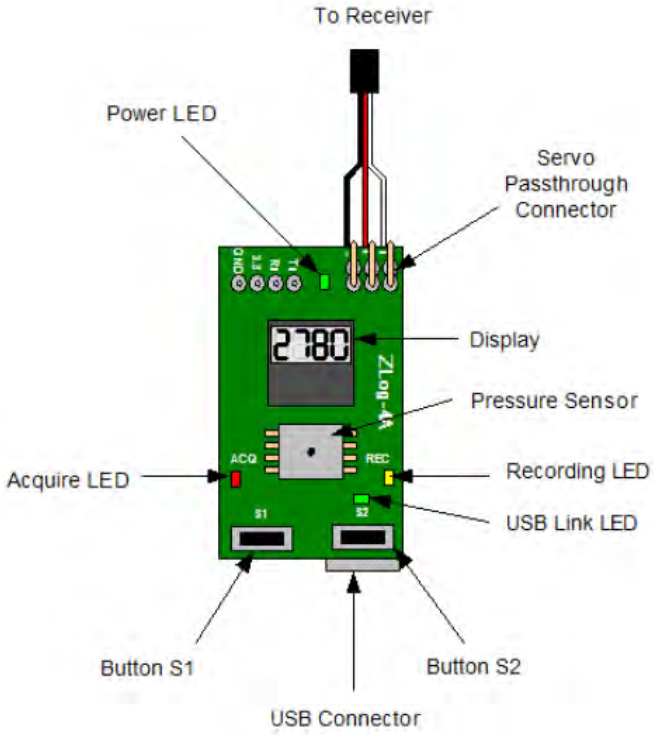
Un sensor es un elemento que genera una señal relacionada con la variable que se pretende medir. La importancia de los sensores en los sistemas de control realimentados es fundamental, debido a que el desempeño de dicho sistema estará restringido a las limitaciones del sensor [RM07].

6.1.2.-SENSOR DE ALTITUD

Este sensor se utilizará en el avión con el objetivo de medir su altitud Zlog estimándola a partir de la presión estática presente en un punto dado del espacio.

Sensor de altura Zlog.

FIG. 77



Sensor de altura Zlog visto a través de sus funcionalidades.

FIG. 78

6.1.3.-GPS GLOBAL POSITIONING SYSTEM

Este modulo nos permite conocer las coordenadas geodésicas para poder trazar una ruta con el aeromodelo.

GPS del sistema Inercial IMU.

FIG. 79

6.2.-DSP- TMS320LF2812

6.2.1-ESPECIFICACIONES TÉCNICAS

El procesador TMS320F2812 es un DSP con una potencia de procesamiento de 150 MIPS, con núcleo de la familia 2800 de Texas Instruments, tiene como característica una gran facilidad para el procesamiento de señales y es usado en la industria para el control de dispositivos periféricos, como motores de inducción trifásicos, monofásicos, adquisición de datos, sistemas de seguridad, entre otros.
CPU de 32 bits con juego de instrucciones RISC, unidad de multiplicación MAC de 32x32, operaciones atómicas, bus con arquitectura Hardvard, Modelo de programación de memoria unificada, código eficiente para C++ y ANSI C y buscador de direcciones de programas y datos lineal de 4M.

Memoria interna

- Flash Devices: 128K x 16 Flash (Cuatro Sectores de 8K x 16 y seis de 16K x 16)
- ROM Devices: Up to 128K x 16 ROM
- 1K x 16 OTP ROM memoria programable una sola vez
- L0 and L1: 2 Bloques de 4K x 16 cada uno. Memoria de simple acceso (SARAM)
- H0: 1 Bloque de 8K x 16 SARAM
- M0 and M1: 2 Bloques de 1K x 16 cada uno

256Kw programa y/o data RAM Memoria Externa
Memoria externa al DSP para la implementación de nuevos programas y datos

12 Canales de PWM. Pulse-Width Modulation
Dos manejadores de eventos para 6 canales PWM cada uno, utilizados en el manejo de llaves MOSFETS o IGBTs en motores industrials

4 Timers de Propósito general con 4 modos de trabajo
Relojes de 16 bits que pueden llegar a una frecuencia de 20MHz

3 Unidades de comparación completa con deadtime

Compara los relojes con un valor de 16 bits y dispara una señal por sus pines o una interrupción.

6 Unidades de Captura con cuadratura para 2 encoders

Capturan señales desde sus pines externos con polaridad programable, también 4 de ellos pueden ser usados para la lectura de dos encoders, para el sensado de posición de ejes de 2 motores o servomecanismos y su utilización en telemetría.

1 Módulo de Conversión Analógico Digital de 16 canales cada uno, total 16ch, hasta 12.5MSPS

Trabajo en modalidad doble o de cascada, a 12 bits, dando un total de 16 canales de conversión analógico digital, a un tiempo de restablecimiento de 80ns, los convertidores pueden ser usados para adquisición de datos, voces, lecturas de sensores analógicos y otros.

49 pines de entradas y salidas I/O Programables

Para el encendido y apagado de interruptores, captura de encendido y apagado.

Unidad de tiempo de interrupción en tiempo real

El DSP tiene una unidad de captura versátil, para el manejo de interrupciones de hardware y de software, por ubicación de programa en su tabla de interrupción.

Unidad de Comunicación Serial asíncrona (SCI)

Para la comunicación con un MODEM, comunicación con una PC u otros DSPs.

Unidad de Comunicación Serial síncrona (SPI)

Diversos periféricos como memorias flash, pantallas LCDs, EEproms pueden ser manejadas con este tipo de comunicación síncrona.

5 Interrupciones externas INT12, Reset, PDPs

Interrupciones externas XINT1, XINT2, pin de Reset, pines de detección de falla de power PDPA y PDPB, además de un detector de instrucciones no legales como NMI.

Flash externo de 4Mbits

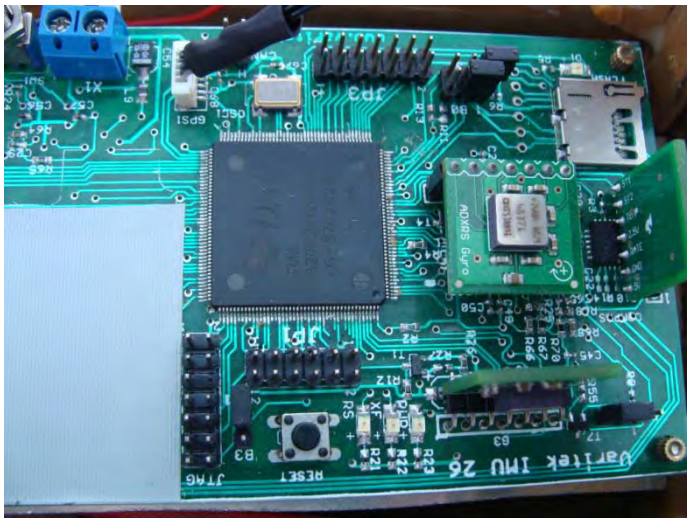
Memoria externa que se comunica con el DSP utilizando su puerto SPI.

7.-RESULTADOS OBTENIDOS.

En la puesta en marcha del UAV a través del aeromodelo se construyó un modelo a escala 1:5 de un avión Cessna 182. Este aeromodelo cuenta con una capacidad de carga de 500 gramos. Hay que hacer notar que se contaba con un avión más pequeño para las pruebas iniciales ya que todo el tiempo se probaban diferentes sensores. Se le ha dedicado mucho tiempo a la preparación, prueba y puesta en marcha del aeromodelo. Se encontraron un sin fin de dificultades y se compro algunos dispositivos que no sirvieron finalmente ya sea por la precisión o por la dificultad de programación o porque la integración no se podía realizar fácilmente. La solución fue, finalmente la de realizar el trabajo con el IMU donde este tiene incorporado un DSP TMS320F2812 y es allí en donde se realizo la programación de la lógica difusa [RM07].

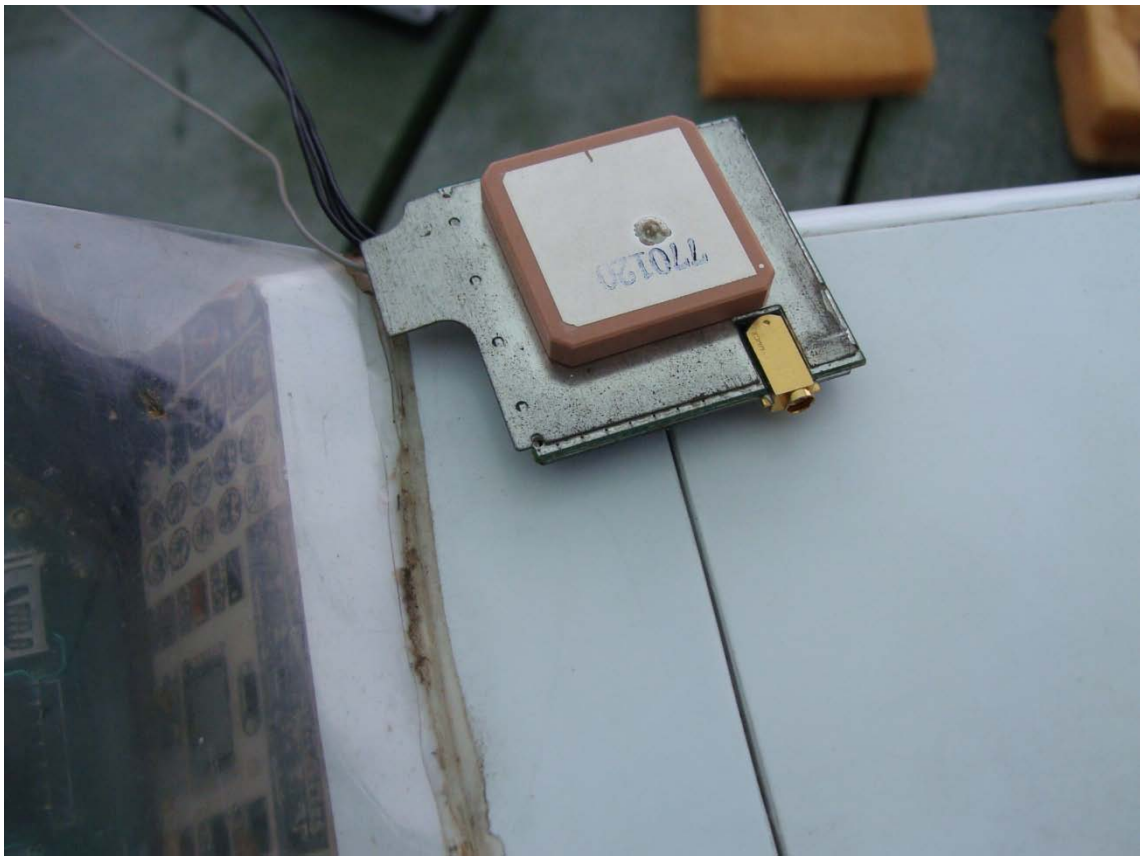
El avión logra volar finalmente y completa una pequeña ruta, estos datos han sido llevados al google, esto nos permite visualizarlo en 3D. Hay que destacar que el aeromodelo es capaz de tomar datos a través de los sensores conectados en el DSP y este es capaz de grabar estos datos en una memoria sd.

A continuación se muestran las fotos del aeromodelo con el IMU dentro del avión antes de una prueba de campo. Esta unidad cuenta con un procesador DSP, los acelerómetros, unidad de grabación en tarjeta SD.



IMU dentro del Cessna 182.

FIG. 80



GPS antes de ser montado en el ala del avión.

FIG. 81

Esta unidad de GPS admemás tiene una antema exterior para ser utilizada fuera del avión. Cuando trabajan varios sistemas junto con la unidad GPS pueden entrar en interferencia. Para esto también se debe tener cuidado de apantallar el resto de unidades con papel aluminio, tener una buena disposición de los equipos dentro del avión para evitar los ruidos que se puedan filtrar en el sistema. Las líneas a tierra en mi caso son redundantes, debido a la gran cantidad de diferentes equipos con diferentes voltajes.

El avión Cessna de escala 1:5.

FIG. 82

El avión Cessna utilizado para las pruebas finales. Hay que destacar de este modelo la forma lenta del vuelo así como su facilidad de recuperarse en caso de una emergencia, este modelo cuenta con flaps para poder tener un vuelo más pausado que los aviones similares de igual longitud.

Para poder tener una mejor idea de la implementación de este UAV, se muestra aquí una misión de vuelo del avión con una pequeña consigna. Al que destacar que el aterrizaje y el despegue se realizaron en forma manual debido a que el proyecto no contempla esas singularidades, además esto podría poner en peligro la vida de todo el proyecto.

Trazado de un vuelo no tripulado.

FIG. 83

En la fig.83 se muestra un trazado de un vuelo realizado de forma manual para poder estabilizar el modelo, luego se le deja al DSP realizar su algoritmo de control.

Las líneas que siguen a continuación son los programas de lógica difusa que se encuentran programados dentro del DSP, el siguiente programa muestra la programación del sensor de altura Zlog.

7.1 PROGRAMA DE LA LÓGICA DIFUSA EN LENGUAJE ANSI C TRABAJANDO EN EL DSP:

```
#include "DSP281x_Device.h"     // DSP281x Headerfile Include File

#include "Fuzzy.h"

struct mu_struct_Dalt mu_Dalt;

struct mu_struct_Valt mu_Valt;

struct mu_struct_Ddir mu_Ddir;

struct mu_struct_Vdir mu_Vdir;

struct VectDefuzzAlt VectoresAlt[25];

struct VectDefuzzDir VectoresDir[25];

long int Altura_Ref=125000;

long int Direccion_Ref=227600; // ir a 227.6 grados

long int Error_dir=0;

long int Error_dir_1=0;

long int Erroraltura=0;

long int Erroraltura_1=0;

long int Vel_dir=0;

int cuentafuzzy=0;

void calc_mu_Dalt(long int altura)

{

Erroraltura=Altura_Ref-altura;

mu_Dalt.mu_Dalt_HN=fp_Dalt_HN(Erroraltura);

mu_Dalt.mu_Dalt_LN=fp_Dalt_LN(Erroraltura);

mu_Dalt.mu_Dalt_Z = fp_Dalt_Z(Erroraltura);

mu_Dalt.mu_Dalt_LP = fp_Dalt_LP(Erroraltura);
```

```
mu_Dalt.mu_Dalt_HP = fp_Dalt_HP(Erroraltura);

}

void calc_mu_Valt(long int altura)

{

long int Valtura=0;

Valtura=(Erroraltura-Erroraltura_1)*F_Tm; //para tiempo de muestreo de la entrada de datos

Erroraltura_1=Erroraltura;

mu_Valt.mu_Valt_HN=fp_Valt_HN(Valtura);

mu_Valt.mu_Valt_LN=fp_Valt_LN(Valtura);

mu_Valt.mu_Valt_Z = fp_Valt_Z(Valtura);

mu_Valt.mu_Valt_LP = fp_Valt_LP(Valtura);

mu_Valt.mu_Valt_HP = fp_Valt_HP(Valtura);

}

void calc_mu_Ddir(long int dir)

{

Error_dir=(Direccion_Ref-dir)/4; //La división entre 4 permite abrir más la sensibilidad del ángulo

mu_Ddir.mu_Ddir_HN=fp_Ddir_HN(Error_dir);

mu_Ddir.mu_Ddir_LN=fp_Ddir_LN(Error_dir);

mu_Ddir.mu_Ddir_Z = fp_Ddir_Z(Error_dir);

mu_Ddir.mu_Ddir_LP = fp_Ddir_LP(Error_dir);

mu_Ddir.mu_Ddir_HP = fp_Ddir_HP(Error_dir);

Error_dir=Error_dir*4;

}

void calc_mu_Vdir(long int dir)
```

```
{

if (cuentafuzzy==F_Tm) // calcula velocidad teniendo en cuenta bajo muestreo <1seg del compas

        { // cada F_Tm pasadas calcula la velocidad

        Vel_dir=(Error_dir-Error_dir_1); //para tiempo de muestreo de la entrada de datos

        cuentafuzzy=0;

        Error_dir_1=Error_dir;

        }

cuentafuzzy++;

Vel_dir=Vel_dir/8; //Le quita sensibilidad a la aceleración del error de dir

mu_Vdir.mu_Vdir_HN=fp_Vdir_HN(Vel_dir);

mu_Vdir.mu_Vdir_LN=fp_Vdir_LN(Vel_dir);

mu_Vdir.mu_Vdir_Z = fp_Vdir_Z(Vel_dir);

mu_Vdir.mu_Vdir_LP = fp_Vdir_LP(Vel_dir);

mu_Vdir.mu_Vdir_HP = fp_Vdir_HP(Vel_dir);

Vel_dir=Vel_dir*8;

}

long int fp_Dalt_HN(long int Ealt)

{

        long int mu_Dalt_HN=0;

if (Ealt<-15000) mu_Dalt_HN = 1000;

if ((Ealt>=-15000)&&(Ealt<-5000)) mu_Dalt_HN = 1000;

if ((Ealt>=-5000)&&(Ealt<-2500)) mu_Dalt_HN = ((-4*Ealt)/10)-1000;

if ((Ealt>=-2500)&&(Ealt<15000)) mu_Dalt_HN = 0;

if (Ealt>=15000) mu_Dalt_HN = 0;
```

```
return mu_Dalt_HN;
}
long int fp_Dalt_LN(long int Ealt)
{
        long int mu_Dalt_LN=0.0;
if (Ealt<-15000) mu_Dalt_LN = 0;
if (Ealt<-5000) mu_Dalt_LN = 0;
if ((Ealt>=-5000)&&(Ealt<-2500)) mu_Dalt_LN = ((4*Ealt)/10)+2000;
if ((Ealt>=-2500)&&(Ealt<0)) mu_Dalt_LN = ((-4*Ealt)/10);
if (Ealt>=0) mu_Dalt_LN = 0;
if (Ealt>=15000) mu_Dalt_LN = 0;
return mu_Dalt_LN;
}
long int fp_Dalt_Z(long int Ealt)
{
        long int mu_Dalt_Z=0;
if (Ealt<-15000) mu_Dalt_Z = 0;
if (Ealt<-2500) mu_Dalt_Z = 0;
if ((Ealt>=-2500)&&(Ealt<0)) mu_Dalt_Z = ((4*Ealt)/10)+1000;
if ((Ealt>=0)&&(Ealt<2500)) mu_Dalt_Z = ((-4*Ealt)/10)+1000;
if (Ealt>=2500) mu_Dalt_Z = 0;
if (Ealt>=15000) mu_Dalt_Z = 0;
return mu_Dalt_Z;
}
long int fp_Dalt_LP(long int Ealt)
```

```
{
        long int mu_Dalt_LP=0;
if (Ealt<-15000) mu_Dalt_LP = 0;
if (Ealt<0) mu_Dalt_LP = 0;
if ((Ealt>=0)&&(Ealt<2500)) mu_Dalt_LP = ((4*Ealt)/10);
if ((Ealt>=2500)&&(Ealt<5000)) mu_Dalt_LP = ((-4*Ealt)/10)+2000;
if (Ealt>=5000) mu_Dalt_LP = 0;
if (Ealt>=15000) mu_Dalt_LP = 0;
return mu_Dalt_LP;
}
long int fp_Dalt_HP(long int Ealt)
{
        long int mu_Dalt_HP=0;
if (Ealt<-15000) mu_Dalt_HP = 0;
if (Ealt<2500) mu_Dalt_HP = 0;
if ((Ealt>=2500)&&(Ealt<5000)) mu_Dalt_HP = ((4*Ealt)/10)-1000;
if ((Ealt>=5000)&&(Ealt<15000)) mu_Dalt_HP = 1000;
if (Ealt>=15000) mu_Dalt_HP = 1000;
return mu_Dalt_HP;
}
long int fp_Valt_HN(long int Valt)
{
        long int mu_Valt_HN=0;
if (Valt<-4000) mu_Valt_HN = 1000;
if ((Valt>=-4000)&&(Valt<-3000)) mu_Valt_HN = -Valt-3000;
```

```
if ((Valt>=-3000)&&(Valt<4000)) mu_Valt_HN = 0;
if (Valt>=4000) mu_Valt_HN = 0;
return mu_Valt_HN;
}
long int fp_Valt_LN(long int Valt)
{
        long int mu_Valt_LN=0;
if (Valt<-4000) mu_Valt_LN = 0;
if ((Valt>=-4000)&&(Valt<-3000)) mu_Valt_LN = Valt+4000;
if ((Valt>=-3000)&&(Valt<-2000)) mu_Valt_LN = 1000;
if ((Valt>=-2000)&&(Valt<0)) mu_Valt_LN = -Valt/2;
if ((Valt>=0)&&(Valt<4000)) mu_Valt_LN = 0;
if (Valt>=4000) mu_Valt_LN = 0;
return mu_Valt_LN;
}
long int fp_Valt_Z(long int Valt)
{
        long int mu_Valt_Z=0;
if (Valt<-4000) mu_Valt_Z = 0;
if (Valt<-2000) mu_Valt_Z = 0;
if ((Valt>=-2000)&&(Valt<0)) mu_Valt_Z = Valt/2+1000;
if ((Valt>=0)&&(Valt<2000)) mu_Valt_Z = -Valt/2+1000;
if (Valt>=2000) mu_Valt_Z = 0;
if (Valt>=4000) mu_Valt_Z = 0;
return mu_Valt_Z;
```

```
}
long int fp_Valt_LP(long int Valt)
{
        long int mu_Valt_LP=0;
if (Valt<-4000) mu_Valt_LP = 0;
if (Valt<0) mu_Valt_LP = 0;
if ((Valt>=0)&&(Valt<2000)) mu_Valt_LP = Valt/2;
if ((Valt>=2000)&&(Valt<3000)) mu_Valt_LP = 1000;
if ((Valt>=3000)&&(Valt<4000)) mu_Valt_LP = -Valt+4000;
if (Valt>=4000) mu_Valt_LP = 0;
return mu_Valt_LP;
}
long int fp_Valt_HP(long int Valt)
{
        long int mu_Valt_HP=0;
if (Valt<-4000) mu_Valt_HP = 0;
if (Valt<3000) mu_Valt_HP = 0;
if ((Valt>=3000)&&(Valt<4000)) mu_Valt_HP = Valt-3000;
if (Valt>=4000) mu_Valt_HP = 1000;
return mu_Valt_HP;
}
//Dirección funciones de pertenencia que devuelven los mu de Dirección
long int fp_Ddir_HN(long int Edir)
{
        long int mu_Ddir_HN=0;
```

```
//if (Edir<-15000) mu_Ddir_HN = 1000;
if (Edir<-5000) mu_Ddir_HN = 1000;
if ((Edir>=-5000)&&(Edir<-2500)) mu_Ddir_HN = ((-4*Edir)/10)-1000;
if (Edir>=-2500) mu_Ddir_HN = 0;
//if (Edir>=15000) mu_Ddir_HN = 0;
return mu_Ddir_HN;
}
long int fp_Ddir_LN(long int Edir)
{
	long int mu_Ddir_LN=0;
//if (Edir<-15000) mu_Ddir_LN = 0;
if (Edir<-5000) mu_Ddir_LN = 0;
if ((Edir>=-5000)&&(Edir<-2500)) mu_Ddir_LN = ((4*Edir)/10)+2000;
if ((Edir>=-2500)&&(Edir<0)) mu_Ddir_LN = ((-4*Edir)/10);
if (Edir>=0) mu_Ddir_LN = 0;
//if (Edir>=15000) mu_Ddir_LN = 0;
return mu_Ddir_LN;
}
long int fp_Ddir_Z(long int Edir)
{
	long int mu_Ddir_Z=0;
//if (Edir<-15000) mu_Ddir_Z = 0;
if (Edir<-2500) mu_Ddir_Z = 0;
if ((Edir>=-2500)&&(Edir<0)) mu_Ddir_Z = ((4*Edir)/10)+1000;
if ((Edir>=0)&&(Edir<2500)) mu_Ddir_Z = ((-4*Edir)/10)+1000;
```

```
if (Edir>=2500) mu_Ddir_Z = 0;
//if (Edir>=15000) mu_Ddir_Z = 0;
return mu_Ddir_Z;
}
long int fp_Ddir_LP(long int Edir)
{
        long int mu_Ddir_LP=0;
//if (Edir<-15000) mu_Ddir_LP = 0;
if (Edir<0) mu_Ddir_LP = 0;
if ((Edir>=0)&&(Edir<2500)) mu_Ddir_LP = ((4*Edir)/10);
if ((Edir>=2500)&&(Edir<5000)) mu_Ddir_LP = ((-4*Edir)/10)+2000;
if (Edir>=5000) mu_Ddir_LP = 0;
//if (Edir>=15000) mu_Ddir_LP = 0;
return mu_Ddir_LP;
}
long int fp_Ddir_HP(long int Edir)
{
        long int mu_Ddir_HP=0;
//if (Edir<-15000) mu_Ddir_HP = 0;
if (Edir<2500) mu_Ddir_HP = 0;
if ((Edir>=2500)&&(Edir<5000)) mu_Ddir_HP = ((4*Edir)/10)-1000;
if (Edir>=5000) mu_Ddir_HP = 1000;
//if (Edir>=15000) mu_Ddir_HP = 1000;
return mu_Ddir_HP;
}
```

```
long int fp_Vdir_HN(long int Vdir)
{
	long int mu_Vdir_HN=0;
if (Vdir<-4000) mu_Vdir_HN = 1000;
if ((Vdir>=-4000)&&(Vdir<-3000)) mu_Vdir_HN = -Vdir-3000;
if ((Vdir>=-3000)&&(Vdir<4000)) mu_Vdir_HN = 0;
if (Vdir>=4000) mu_Vdir_HN = 0;
return mu_Vdir_HN;
}
long int fp_Vdir_LN(long int Vdir)
{
	long int mu_Vdir_LN=0;
if (Vdir<-4000) mu_Vdir_LN = 0;
if ((Vdir>=-4000)&&(Vdir<-3000)) mu_Vdir_LN = Vdir+4000;
if ((Vdir>=-3000)&&(Vdir<-2000)) mu_Vdir_LN = 1000;
if ((Vdir>=-2000)&&(Vdir<0)) mu_Vdir_LN = -Vdir/2;
if ((Vdir>=0)&&(Vdir<4000)) mu_Vdir_LN = 0;
if (Vdir>=4000) mu_Vdir_LN = 0;
return mu_Vdir_LN;
}
long int fp_Vdir_Z(long int Vdir)
{
	long int mu_Vdir_Z=0;
if (Vdir<-4000) mu_Vdir_Z = 0;
if (Vdir<-2000) mu_Vdir_Z = 0;
```

```
if ((Vdir>=-2000)&&(Vdir<0)) mu_Vdir_Z = Vdir/2+1000;
if ((Vdir>=0)&&(Vdir<2000)) mu_Vdir_Z = -Vdir/2+1000;
if (Vdir>=2000) mu_Vdir_Z = 0;
if (Vdir>=4000) mu_Vdir_Z = 0;
return mu_Vdir_Z;
}
long int fp_Vdir_LP(long int Vdir)
{
        long int mu_Vdir_LP=0;
if (Vdir<-4000) mu_Vdir_LP = 0;
if (Vdir<0) mu_Vdir_LP = 0;
if ((Vdir>=0)&&(Vdir<2000)) mu_Vdir_LP = Vdir/2;
if ((Vdir>=2000)&&(Vdir<3000)) mu_Vdir_LP = 1000;
if ((Vdir>=3000)&&(Vdir<4000)) mu_Vdir_LP = -Vdir+4000;
if (Vdir>=4000) mu_Vdir_LP = 0;
return mu_Vdir_LP;
}
long int fp_Vdir_HP(long int Vdir)
{
        long int mu_Vdir_HP=0;
if (Vdir<-4000) mu_Vdir_HP = 0;
if (Vdir<3000) mu_Vdir_HP = 0;
if ((Vdir>=3000)&&(Vdir<4000)) mu_Vdir_HP = Vdir-3000;
if (Vdir>=4000) mu_Vdir_HP = 1000;
return mu_Vdir_HP;
```

```
}

long int calc_reglas_Alt(void)

{

VectoresAlt[0].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HN
,mu_Dalt.mu_Dalt_HN);//Valt HN Dalt HN

VectoresAlt[0].tipoElevador = HN;

VectoresAlt[1].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HN
,mu_Dalt.mu_Dalt_LN);//Valt HN Dalt LN

VectoresAlt[1].tipoElevador = HN;

VectoresAlt[2].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HN
,mu_Dalt.mu_Dalt_Z);//Valt HN Dalt Z

VectoresAlt[2].tipoElevador = HN;

VectoresAlt[3].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HN
,mu_Dalt.mu_Dalt_LP);//Valt HN Dalt LP

VectoresAlt[3].tipoElevador = HN;

VectoresAlt[4].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HN
,mu_Dalt.mu_Dalt_HP);//Valt HN Dalt HP

VectoresAlt[4].tipoElevador = Z;



VectoresAlt[5].mu_defuzz=                     FuzzyAND(mu_Valt.mu_Valt_LN
,mu_Dalt.mu_Dalt_HN);//Valt LN Dalt HN

VectoresAlt[5].tipoElevador = HN;

VectoresAlt[6].mu_defuzz=                     FuzzyAND(mu_Valt.mu_Valt_LN
,mu_Dalt.mu_Dalt_LN);//Valt LN Dalt LN

VectoresAlt[6].tipoElevador = HN;

VectoresAlt[7].mu_defuzz=                     FuzzyAND(mu_Valt.mu_Valt_LN
,mu_Dalt.mu_Dalt_Z);//Valt LN Dalt Z

VectoresAlt[7].tipoElevador = HN;

VectoresAlt[8].mu_defuzz=                     FuzzyAND(mu_Valt.mu_Valt_LN
,mu_Dalt.mu_Dalt_LP);//Valt LN Dalt LP
```

```
VectoresAlt[8].tipoElevador = LN;

VectoresAlt[9].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_LN
,mu_Dalt.mu_Dalt_HP);//Valt LN Dalt HP

VectoresAlt[9].tipoElevador = HP;


VectoresAlt[10].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_Z
,mu_Dalt.mu_Dalt_HN);//Valt Z Dalt HN

VectoresAlt[10].tipoElevador = HN;

VectoresAlt[11].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_Z
,mu_Dalt.mu_Dalt_LN);//Valt Z Dalt LN

VectoresAlt[11].tipoElevador = LN;

VectoresAlt[12].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_Z
,mu_Dalt.mu_Dalt_Z);//Valt Z Dalt Z

VectoresAlt[12].tipoElevador = Z;

VectoresAlt[13].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_Z
,mu_Dalt.mu_Dalt_LP);//Valt Z Dalt LP

VectoresAlt[13].tipoElevador = LP;

VectoresAlt[14].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_Z
,mu_Dalt.mu_Dalt_HP);//Valt Z Dalt HP

VectoresAlt[14].tipoElevador = HP;


VectoresAlt[15].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_LP
,mu_Dalt.mu_Dalt_HN);//Valt LP Dalt HN

VectoresAlt[15].tipoElevador = HN;

VectoresAlt[16].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_LP
,mu_Dalt.mu_Dalt_LN);//Valt LP Dalt LN

VectoresAlt[16].tipoElevador = LP;

VectoresAlt[17].mu_defuzz=                   FuzzyAND(mu_Valt.mu_Valt_LP
,mu_Dalt.mu_Dalt_Z);//Valt LP Dalt Z

VectoresAlt[17].tipoElevador = HP;
```

```
VectoresAlt[18].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_LP
,mu_Dalt.mu_Dalt_LP);//Valt LP Dalt LP

VectoresAlt[18].tipoElevador = HP;

VectoresAlt[19].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_LP
,mu_Dalt.mu_Dalt_HP);//Valt LP Dalt HP

VectoresAlt[19].tipoElevador = HP;


VectoresAlt[20].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HP
,mu_Dalt.mu_Dalt_HN);//Valt HP Dalt HN

VectoresAlt[20].tipoElevador = Z;

VectoresAlt[21].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HP
,mu_Dalt.mu_Dalt_LN);//Valt HP Dalt LN

VectoresAlt[21].tipoElevador = HP;

VectoresAlt[22].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HP
,mu_Dalt.mu_Dalt_Z);//Valt HP Dalt Z

VectoresAlt[22].tipoElevador = HP;

VectoresAlt[23].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HP
,mu_Dalt.mu_Dalt_LP);//Valt HP Dalt LP

VectoresAlt[23].tipoElevador = HP;

VectoresAlt[24].mu_defuzz=                    FuzzyAND(mu_Valt.mu_Valt_HP
,mu_Dalt.mu_Dalt_HP);//Valt HP Dalt HP

VectoresAlt[24].tipoElevador = HP;

return 0.0;

}

long int calc_reglas_Dir(void)

{

VectoresDir[0].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HN
,mu_Ddir.mu_Ddir_HN);//Vdir HN Ddir HN

VectoresDir[0].tipoTimon = HN;
```

```
VectoresDir[1].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_HN
,mu_Ddir.mu_Ddir_LN);//Vdir HN Ddir LN

VectoresDir[1].tipoTimon = HN;

VectoresDir[2].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_HN
,mu_Ddir.mu_Ddir_Z);//Vdir HN Ddir Z

VectoresDir[2].tipoTimon = HN;

VectoresDir[3].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_HN
,mu_Ddir.mu_Ddir_LP);//Vdir HN Ddir LP

VectoresDir[3].tipoTimon = HN;

VectoresDir[4].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_HN
,mu_Ddir.mu_Ddir_HP);//Vdir HN Ddir HP

VectoresDir[4].tipoTimon = Z;


VectoresDir[5].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_LN
,mu_Ddir.mu_Ddir_HN);//Vdir LN Ddir HN

VectoresDir[5].tipoTimon = HN;

VectoresDir[6].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_LN
,mu_Ddir.mu_Ddir_LN);//Vdir LN Ddir LN

VectoresDir[6].tipoTimon = HN;

VectoresDir[7].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_LN
,mu_Ddir.mu_Ddir_Z);//Vdir LN Ddir Z

VectoresDir[7].tipoTimon = HN;

VectoresDir[8].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_LN
,mu_Ddir.mu_Ddir_LP);//Vdir LN Ddir LP

VectoresDir[8].tipoTimon = LN;

VectoresDir[9].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_LN
,mu_Ddir.mu_Ddir_HP);//Vdir LN Ddir HP

VectoresDir[9].tipoTimon = HP;


VectoresDir[10].mu_defuzz= FuzzyAND(mu_Vdir.mu_Vdir_Z
,mu_Ddir.mu_Ddir_HN);//Vdir Z Ddir HN
```

```
VectoresDir[10].tipoTimon = HN;

VectoresDir[11].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_Z
,mu_Ddir.mu_Ddir_LN);//Vdir Z Ddir LN

VectoresDir[11].tipoTimon = LN;

VectoresDir[12].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_Z
,mu_Ddir.mu_Ddir_Z);//Vdir Z Ddir Z

VectoresDir[12].tipoTimon = Z;

VectoresDir[13].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_Z
,mu_Ddir.mu_Ddir_LP);//Vdir Z Ddir LP

VectoresDir[13].tipoTimon = LP;

VectoresDir[14].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_Z
,mu_Ddir.mu_Ddir_HP);//Vdir Z Ddir HP

VectoresDir[14].tipoTimon = HP;


VectoresDir[15].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_LP
,mu_Ddir.mu_Ddir_HN);//Vdir LP Ddir HN

VectoresDir[15].tipoTimon = HN;

VectoresDir[16].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_LP
,mu_Ddir.mu_Ddir_LN);//Vdir LP Ddir LN

VectoresDir[16].tipoTimon = LP;

VectoresDir[17].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_LP
,mu_Ddir.mu_Ddir_Z);//Vdir LP Ddir Z

VectoresDir[17].tipoTimon = HP;

VectoresDir[18].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_LP
,mu_Ddir.mu_Ddir_LP);//Vdir LP Ddir LP

VectoresDir[18].tipoTimon = HP;

VectoresDir[19].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_LP
,mu_Ddir.mu_Ddir_HP);//Vdir LP Ddir HP

VectoresDir[19].tipoTimon = HP;
```

```
VectoresDir[20].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HP
,mu_Ddir.mu_Ddir_HN);//Vdir HP Ddir HN

VectoresDir[20].tipoTimon = Z;

VectoresDir[21].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HP
,mu_Ddir.mu_Ddir_LN);//Vdir HP Ddir LN

VectoresDir[21].tipoTimon = HP;

VectoresDir[22].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HP
,mu_Ddir.mu_Ddir_Z);//Vdir HP Ddir Z

VectoresDir[22].tipoTimon = HP;

VectoresDir[23].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HP
,mu_Ddir.mu_Ddir_LP);//Vdir HP Ddir LP

VectoresDir[23].tipoTimon = HP;

VectoresDir[24].mu_defuzz=                    FuzzyAND(mu_Vdir.mu_Vdir_HP
,mu_Ddir.mu_Ddir_HP);//Vdir HP Ddir HP

VectoresDir[24].tipoTimon = HP;

return 0.0;

}

long int FuzzyAND(long int xx, long int yy)

{

if (xx<yy) return xx;

else return yy;

}

long int FuzzyOR(long int xx, long int yy)

{

if (xx<yy) return yy;

else return xx;

}
```

```
float DefzzAlt()
{
	int jk=0;
	float CX=0.0;
	float xtemp=0.0;
	float mutotaltemp=0.0;
for(jk=0;jk<25;jk++)
{
		if (VectoresAlt[jk].tipoElevador ==HN) xtemp=-8.0;
		if (VectoresAlt[jk].tipoElevador ==LN) xtemp=-6.0;
		if (VectoresAlt[jk].tipoElevador ==Z) xtemp=0.0;
		if (VectoresAlt[jk].tipoElevador ==LP) xtemp=6.0;
		if (VectoresAlt[jk].tipoElevador ==HP) xtemp=8.0;

	CX=CX+VectoresAlt[jk].mu_defuzz*xtemp;
	mutotaltemp=mutotaltemp+VectoresAlt[jk].mu_defuzz;
};
CX=CX/mutotaltemp;
return CX;
}
float DefzzDir()
{
	int jk=0;
	float CX=0.0;
	float xtemp=0.0;
```

```
        float mutotaltemp=0.0;
for(jk=0;jk<25;jk++)
{
                if (VectoresDir[jk].tipoTimon ==HN) xtemp=-8.0;
                if (VectoresDir[jk].tipoTimon ==LN) xtemp=-6.0;
                if (VectoresDir[jk].tipoTimon ==Z) xtemp=0.0;
                if (VectoresDir[jk].tipoTimon ==LP) xtemp=6.0;
                if (VectoresDir[jk].tipoTimon ==HP) xtemp=8.0;

        CX=CX+((float)VectoresDir[jk].mu_defuzz)*xtemp;
        mutotaltemp=mutotaltemp+((float)VectoresDir[jk].mu_defuzz);
};
CX=CX/mutotaltemp;
return CX;
}
float FuzzificarAlt(float FuzziAltura)
{
long int Faltura;
Faltura=(long int)(FuzziAltura*1000.0);
calc_mu_Dalt(Faltura);
calc_mu_Valt(Faltura);
calc_reglas_Alt();
return DefzzAlt()/1000.0;
}
float FuzzificarDir(long int FuzziDireccion)
```

```
{
long int Fdireccion;
Fdireccion=(FuzziDireccion*100); //Heading es entero y ya fué multiplicado por 10
calc_mu_Ddir(Fdireccion);
calc_mu_Vdir(Fdireccion);
calc_reglas_Dir();
return DefzzDir();
}
```

7.2.-PROGRAMA EN LENGUAJE ANSI C DEL SENSOR DE ALTURA.

```
#include "stdlib.h"
#include "zlog.h"
#include "DSP281x_Device.h"
#include "DSP281x_Examples.h"
void scia_init(void);
void scia_fifo_init(void);
void scia_xmit(int a);
extern unsigned long int retardar(unsigned long int retardo);
//static char HexChars[] = "0123456789ABCDEF";
interrupt void zRX_isr(void);
int cuenta_fifoRX=0;
int alturaF=0;
long alturaM100=0;
int zlogtest=0;
int erroraltura=0;
void ZlogIniciarSerial(void)
{

   scia_init();  // Inicializa el lazo de iteración
   scia_fifo_init();       // Inicializa el FIFO del SCI
   SciaRegs.SCIRXBUF.all;
}
// Test 1,SCIA  DLB, 8-bit word, baud rate 0x000F, default, 1 STOP bit, no parity
void scia_init()
{
        SciaRegs.SCICCR.all =0x0007;   // 1 stop bit,  No loopback
                        // No parity,8 char bits,
                        // modo asíncrono, protocolo idle-line línea en espera
        SciaRegs.SCICTL1.all =0x0003;  // Habilitar TX, RX, SCICLK interno,
                        // Deshabilitar RX ERR, SLEEP, TXWAKE
        SciaRegs.SCICTL2.all =0x0000;
   SciaRegs.SCIHBAUD    =0x000;
   SciaRegs.SCILBAUD       =0x028; //150MHz y losclk =75/2 Mhz el 8 y final
115200baud
}
// Transmit a character from the SCI'
void scia_xmit(int a)
{
   SciaRegs.SCITXBUF=a;
   retardar(1);
}
// Initalize the SCI FIFO
void scia_fifo_init()
{
   SciaRegs.SCIFFTX.all=0x4000; //FIFOS prendidos y reset no hay int de tx
   SciaRegs.SCIFFRX.all=0x0062; //16 niv fifo rx con int y espera dos words para
int
   asm("                nop");
   asm("                nop");
```

```
   SciaRegs.SCIFFCT.all=0x0;
  SciaRegs.SCICTL1.all =0x0023;     // Relinquish SCI from Reset
  SciaRegs.SCIFFTX.bit.TXFIFOXRESET=1;
  SciaRegs.SCIFFRX.bit.RXFIFORESET=1;
  SciaRegs.SCIFFTX.all=0xE000; //FIFOS prendidos y reset a su sitio
}
interrupt void zRX_isr(void)
{
   Uint16 rxi;
   cuenta_fifoRX=SciaRegs.SCIFFRX.bit.RXFIFST;
   if (cuenta_fifoRX>4) erroraltura=1; // condición extrema de lectura, se pasó
        zlogtest++;
          rxi=SciaRegs.SCIRXBUF.all;     // Read data
                    if (cuenta_fifoRX==2)
                          {
                                alturaF=rxi<<8;
                             rxi=SciaRegs.SCIRXBUF.all;    // Read data
                                alturaF|=rxi;
                                cuenta_fifoRX=0;
                                alturaM100=(int)(alturaF*0.3048);    //cambio    de
pies a altura metros por 100
                          };
        SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1; // resetea flag de interrupt
        PieCtrlRegs.PIEACK.all|=PIEACK_GROUP9;           // Issue PIE ack
}

void ZlogEnviarL(void)
{
              while(SciaRegs.SCICTL2.bit.TXRDY == 0) { }; // esperar a TRDY =1
estado listo
        scia_xmit('l');

        cuenta_fifoRX=0;
}
```

7.3.-PROGRAMA QUE CONTROLA LOS SERVOS DEL AVIÓN EN ANSI C.

```
   #include "DSP281x_Device.h"
#include "DSP281x_Examples.h"
#include "Varitek26.h"
#include "Sd.h"
#include "Sd_SPI.h"
#include "CPU_Timers.h"
#include "GrabaSD.h"
#include "Compas.h"
#include "Spi.h"
#include "gps.h"
#include "printing.h"
#include "FAT32_Access.h"
#include "kalman.h"
#include "zlog.h"
#include "math.h"
#include "Fuzzy.h"
#include <math.h>
float gyro_x;
float gyro_y;
float accel_x;
float accel_y;
float accel_z;
float roll;
float pitch;
```

```
float roll_rate;
float pitch_rate;
float roll_est;
float pitch_est;
// Kalman state structures.
kalman roll_kalman_state;
kalman pitch_kalman_state;
//#include "SD_File.h"

//#include "FAT32_Base.h"
//#include "FAT32_Access.h"

unsigned int ResultadoADC[16];
unsigned int *Source;
unsigned int *Dest;
Uint32 i;
Uint32 jsd;
void ini_eva_timer1(void);
void ini_eva_timer2(void);
void ini_eva_timer3(void);
void ini_eva_timer4(void);
interrupt void T1PER_isr(void);
interrupt void T1CMP_isr(void);
interrupt void T3PER_isr(void);
interrupt void T3CMP_isr(void);
```

```
interrupt void T2_CPU_isr(void);
void init_pwm_eva(void);
void init_pwm_evb(void);
void initADC(void);
int iniciarSD(void);
int SD_ok = 0;
int SD_enslot=0;
int errorSD=0;
extern sd_context_t sdc;
        FileLib_File * files[5]; //Solo son punteros, en FileLib están los reales
        FileLib_File *readFile;
        char filenames[5][260];
        BYTE fileData[5][10000];
        BYTE readBuffer[10000];
        int fileLengths[5];

union MP mprocesos;
void ManejaJobs(void);
extern interrupt void RX_isr(void);
extern interrupt void zRX_isr(void);
int testV26=0;
void imprimirChar(unsigned char c);
#pragma DATA_SECTION(Hoja,"Datos");
int Hoja[8192];
int indiceHoja=0;
```

```
void IniciarGrabaSD(void);
long int secundero=0;
long int decimosegundo=0;
int alternaXF=0;
int prenderXF=0;
extern int GPSnoData;
long int Fase_servoPWM[6]={14,14,14,14,14,14};
int cuenta_servoPWM=0;
float elevador=0.0;
float timon=0.0;
extern int alturaF;
extern int Heading;
extern float Altura_Ref;
extern float Direccion_Ref;
int controlRemoto = 0; //Control remoto ó UAV maneja 0=UAV 1=Control
int dataCH[3]; // lee estado del canal de control remoto
void main(void)
{
DINT;
DRTM;
        testV26=1;
InitSysCtrl();
EALLOW;
  SysCtrlRegs.PCLKCR.bit.MCBSPENCLK=0;
  SysCtrlRegs.PCLKCR.bit.ECANENCLK=0;
```

```
EDIS;

InitPieCtrl();

IER = 0x0000;

IFR = 0x0000;

InitPieVectTable();

InitXintf();

// Inicialización de Pines GPIOs

  EALLOW;

  GpioMuxRegs.GPAMUX.all = 0x07C0; // PWM1_6s(IOS) T1PWM T2PWM

   GpioMuxRegs.GPADIR.bit.GPIOA0 = 1; //PWM1 salida IO

        GpioMuxRegs.GPADIR.bit.GPIOA1 = 1; //PWM2 salida IO

        GpioMuxRegs.GPADIR.bit.GPIOA2 = 1; //PWM3 salida IO

        GpioMuxRegs.GPADIR.bit.GPIOA3 = 1; //PWM4 salida IO

        GpioMuxRegs.GPADIR.bit.GPIOA4 = 1; //PWM5 salida IO

        GpioMuxRegs.GPADIR.bit.GPIOA5 = 1; //PWM6 salida IO

  GpioMuxRegs.GPAMUX.bit.TCLKINA_GPIOA12 = 0; // Manejo de la fuente
VCC3V3_SW

  GpioMuxRegs.GPAMUX.bit.TDIRA_GPIOA11 = 0; // Manejo del RS del CAN

  GpioMuxRegs.GPAMUX.bit.C2TRIP_GPIOA14 = 0; // Manejo del selector de
gravedad G1 MMA7260

  GpioMuxRegs.GPAMUX.bit.C1TRIP_GPIOA13 = 0; // Manejo del selector de
gravedad G2 MMA7260

  GpioMuxRegs.GPADIR.bit.GPIOA12 = 1; // Es salida fuente VCC3V3_SW

  GpioMuxRegs.GPADIR.bit.GPIOA11 = 1;// Es salida RS del CAN

  GpioMuxRegs.GPADIR.bit.GPIOA14 = 1; // Es salida selector de gravedad G1
MMA7260

   GpioMuxRegs.GPADIR.bit.GPIOA13 = 1;  // Es salida selector de gravedad G2
MMA7260
```

```
        testV26=15;

        GpioMuxRegs.GPBMUX.all = 0x07C0; // PWM7_12s (IOS)T3PWM T4PWM

        GpioMuxRegs.GPBDIR.bit.GPIOB0 = 0; //PWM7 salida IO

        GpioMuxRegs.GPBDIR.bit.GPIOB1 = 1; //PWM8 salida IO

        GpioMuxRegs.GPBDIR.bit.GPIOB2 = 1; //PWM9 salida IO

        GpioMuxRegs.GPBDIR.bit.GPIOB3 = 0; //PWM10 salida IO

        GpioMuxRegs.GPBDIR.bit.GPIOB4 = 1; //PWM11 salida IO

        GpioMuxRegs.GPBDIR.bit.GPIOB5 = 1; //PWM12 salida IO

        GpioMuxRegs.GPBMUX.bit.CAP4Q1_GPIOB8 = 0; //IO Debug
oscilloscopio

        GpioMuxRegs.GPBMUX.bit.C4TRIP_GPIOB13 = 0;  // Manejo de la fuente
del SD y Led

        GpioMuxRegs.GPBMUX.bit.C5TRIP_GPIOB14 = 0; // Manejo de ST1_A del
Giro A G1

        GpioMuxRegs.GPBMUX.bit.C6TRIP_GPIOB15 = 0; // Manejo de ST2_A del
Giro A G1

        GpioMuxRegs.GPBDIR.bit.GPIOB13 = 1;       // Es salida fuente del SD y
Led SD

        GpioMuxRegs.GPBDIR.bit.GPIOB14 = 1; // Es salida ST1_A del Giro A G1

        GpioMuxRegs.GPBDIR.bit.GPIOB15 = 1;// Es salida ST2_A del Giro A G1

        GpioMuxRegs.GPBDIR.bit.GPIOB8 = 1;//Es salida debug oscilloscopio

        testV26=16;

        GpioMuxRegs.GPDMUX.bit.T2CTRIP_SOCA_GPIOD1 = 0; // Manejo de la
fuente VCC5V

        GpioMuxRegs.GPDMUX.bit.T3CTRIP_PDPB_GPIOD5 = 0; // Debug
oscilloscopio

        GpioMuxRegs.GPDMUX.bit.T4CTRIP_SOCB_GPIOD6 = 0; // Manejo del
Enable del GPS

    GpioMuxRegs.GPDDIR.bit.GPIOD1 = 1; // Es salida fuente VCC5V

    GpioMuxRegs.GPDDIR.bit.GPIOD5 = 1; //Es salida debug oscilloscopio
```

```
    GpioMuxRegs.GPDDIR.bit.GPIOD6 = 1; // Es salida Enable GPS

     GpioMuxRegs.GPFMUX.all = 0x00ff;// spi sciA can

     GpioMuxRegs.GPFMUX.bit.XF_GPIOF14 = 1; // Bandera XF

    GpioMuxRegs.GPFMUX.bit.MFSXA_GPIOF10 = 0;// Manejo de ST1_B del Giro B G2

    GpioMuxRegs.GPFMUX.bit.MCLKRA_GPIOF9 = 0;// Manejo de ST2_B del Giro B G2

    GpioMuxRegs.GPFMUX.bit.MDXA_GPIOF12 = 0;// Manejo de ST1_C del Giro C G3

    GpioMuxRegs.GPFMUX.bit.MDRA_GPIOF13 = 0;// Manejo de ST2_C del Giro C G3

    GpioMuxRegs.GPFMUX.bit.MFSRA_GPIOF11 = 0; // Manejo del SDI del Compas I2C

    GpioMuxRegs.GPFMUX.bit.MCLKXA_GPIOF8 = 0; // Manejo del SCL del Compas I2C

    GpioMuxRegs.GPFMUX.bit.SPISTEA_GPIOF3 = 0; // Manejo del CE del SD card

    GpioMuxRegs.GPFDIR.bit.GPIOF10 = 1;// Es salida ST1_B del Giro B G2

    GpioMuxRegs.GPFDIR.bit.GPIOF9 = 1;// Es salida ST2_B del Giro B G2

     GpioMuxRegs.GPFDIR.bit.GPIOF12 = 1;// Es salida ST1_C del Giro C G3

    GpioMuxRegs.GPFDIR.bit.GPIOF13 = 1;// Es salida ST2_C del Giro C G3

    GpioMuxRegs.GPFDIR.bit.GPIOF11 = 1; // Es salida inicio Master I2C SDI del Compas

    GpioMuxRegs.GPFDIR.bit.GPIOF8 = 1; // Es salida inicio Master I2C SCL del Compas

    GpioMuxRegs.GPFDIR.bit.GPIOF3 = 0; //Es salida el CS de la SD card UP

                                                    // Primero es entrada para chequear SD presente

                                                    // Ojo verificar no jumper en B1 (tampoco B0)

   GpioMuxRegs.GPGMUX.all = 0x30;  //SCIB para el GPS
```

```
        testV26=2;

  PieVectTable.T1PINT = &T1PER_isr;

  PieVectTable.T1CINT = &T1CMP_isr;

  PieVectTable.T3PINT = &T3PER_isr;

  PieVectTable.T3CINT = &T3CMP_isr;

   PieVectTable.RXBINT = &RX_isr;

   PieVectTable.RXAINT = &zRX_isr;

   EvaRegs.EVAIMRA.bit.PDPINTA=0; //OJO importantisimo TRUCO para que no se caiga el pwm1-6 t1pwm

                                                    //ponerlo despues de dar clock al periferico eva

        EvbRegs.EVBIMRA.bit.PDPINTB=0;

  PieVectTable.TINT2 = &T2_CPU_isr;



  EDIS;

  GpioDataRegs.GPADAT.bit.GPIOA12 = 0; // Encender fuente VCC3V3_SW 0 es encender 1-apagar

  retardar(100);



  GpioDataRegs.GPADAT.bit.GPIOA11 = 0; // En off enciende el slew rate RS del CAN

  GpioDataRegs.GPADAT.bit.GPIOA14 = 0; // Selector de gravedad G1G2 en 00 1.5g MMA7260

  GpioDataRegs.GPADAT.bit.GPIOA13 = 0; // Selector de gravedad G1G2 en 00 1.5g MMA7260

  testV26=18;

  GpioDataRegs.GPDDAT.bit.GPIOD1 = 1; // Encender fuente de VCC5V 0-apagar 1 encender

  retardar(1000);
```

```
  testV26=19;

  GpioDataRegs.GPDDAT.bit.GPIOD6 = 0; // Disable GPS

  retardar(100);

  GpioDataRegs.GPDDAT.bit.GPIOD5 = 0;// Apagado depurador oscilloscopio

  GpioDataRegs.GPBDAT.bit.GPIOB8 = 0;// Apagado depurador oscilloscopio

  GpioDataRegs.GPDDAT.bit.GPIOD6 = 1; // Enable GPS

  retardar(10);

  /*EALLOW;

    GpioMuxRegs.GPDDIR.bit.GPIOD6 = 0; // Es entrada alta impedancia Enable GPS

  EDIS;  */

  GpioDataRegs.GPBDAT.bit.GPIOB13 = 1; // Encendido Led y alimentacióna al SD 1:OFF 0:ON

                                                                    //Aún no, hay que esperar al inicializar

      GpioDataRegs.GPBDAT.bit.GPIOB14 = 0; // Prueba -270mV ST1_A del Giro A G1

      GpioDataRegs.GPBDAT.bit.GPIOB15  = 0;// Prueba +270mV ST2_A del Giro A G1

      GpioDataRegs.GPFDAT.bit.GPIOF10 = 0;// Prueba -270mV ST1_B del Giro B G2

      GpioDataRegs.GPFDAT.bit.GPIOF9 = 0;// Prueba +270mV ST2_B del Giro B G2

      GpioDataRegs.GPFDAT.bit.GPIOF12 = 0;// Prueba -270mV ST1_C del Giro C G3

      GpioDataRegs.GPFDAT.bit.GPIOF13 = 0;// Prueba +270mV ST2_C del Giro C G3

      GpioDataRegs.GPFDAT.bit.GPIOF11 = 1;// SDI del I2C arriba '1'

      GpioDataRegs.GPFDAT.bit.GPIOF8 = 1; // SCL del I2C arriba '1'

      GpioDataRegs.GPFDAT.bit.GPIOF3 = 1; // CS de la SD card UP abajo para no alimentar por 50Kohm del sd
```

```
        testV26=3;
PieCtrlRegs.PIEIER2.all = M_INT4 | M_INT5; //Grupo 2 T1PER y T1CMP
PieCtrlRegs.PIEIER4.all = M_INT4 | M_INT5; //Grupo 4 T3PER y T3CMP
PieCtrlRegs.PIEIER9.all = M_INT3; // serial A y B
IER = M_INT2 | M_INT4 | M_INT9 | M_INT14; // I2 T1 periodo y comparador, I9 Serial GPS, I14 TimerCPU_2
// I4 T3 periodo y comparador
initADC();
ini_eva_timer1();
ini_eva_timer3();
//ini_eva_timer2();
//ini_eva_timer4();
//init_pwm_eva();
//init_pwm_evb();

/* Set some reasonable values for the timeouts. */
sdc.timeout_write = (8/38.25)*PERIPH_CLOCKRATE/8;
sdc.timeout_read = (8/38.25)*PERIPH_CLOCKRATE/20;
sdc.busyflag = 0;
testV26=1;
        printfInit(imprimirChar); //inicializa el puntero a la funcion de impresion
testV26=21;
    spi_init();             // init SPI
    testV26=4;
        CompasInit(); //Lo pone en Query mode y envìa primer "A"
        testV26=5;
```

```
        //eCanInit();
        testV26=6;
        ZlogIniciarSerial();
        testV26=7;
        GPSIniciarSerial();
        testV26=8;
if(sd_card_present()!=1)
        {
        //GpioDataRegs.GPBDAT.bit.GPIOB13 = 1; // Apaga-NO HAY Led y alimentacióna al SD
        SD_enslot=0;
        }
else
        {
        SD_enslot=1;
        GpioDataRegs.GPBDAT.bit.GPIOB13 = 0; // Encender Led y alimentacióna al SD
        retardar(100);
          EALLOW;
          GpioMuxRegs.GPFDIR.bit.GPIOF3 = 1; // Ahora el CS de la SD es salida y era '1'
          EDIS;
        }
if(SD_enslot==1) //si hay card
{
        for (jsd=0; (jsd<SD_INIT_TRY) && (SD_ok != 1); jsd++)
        {
```

```
                SD_ok = iniciarSD();
        };
testV26=2;
        if(jsd==SD_INIT_TRY)
                {asm("          setc xf");//problema en sd
                errorSD = 1;
                }
         else{
                if(SD_ok==1)
                        {
                        testV26=9;
                        SD_ok=FAT32_InitFAT();
                        testV26=10;
                        if (SD_ok == 1)
                                {
                                FileLib_Init();  // libera los files 0 -4 hasta
maxopnefiles=4
                                IniciarGrabaSD();
                                testV26=5;
                                //files[0]=(FileLib_File*)FindSpareFile(); //consigue lugar
para filelibs
                                }
                        else {
                                 errorSD=2;
                                 }; // fin SD_ok=1
                        };// fin SD_ok=1 desde else
```

```
                    };// fin jsd
};  //fin si hay o no card
testV26=11;
mprocesos.bit.zlog = 1;
mprocesos.bit.GPSserial=1;
mprocesos.bit.Compas=1; //Para activarlo que este en 1
mprocesos.bit.CalibrarCompas=0;// debe desactivar Compas si 1
mprocesos.bit.GrabaSD=0; //0 no graba CAMBIAR a 1 PARA GRABAR, OJO el Timer2 lo enciende autom
InitCpuTimers();   // Inicio de CPU Timers
ConfigCpuTimer(&CpuTimer2, 150, 100000); //150MHz y 0.1 segundo=100000uS
StartCpuTimer2();
   // Initialize the Kalman states for pitch and roll.  The measurement noise
   // covariance (Sz_00) is the noise in the accelerometer measurements.  Data
   // read from 1000 samples of the accelerometer had a variance of 0.00002.
   // The process noise covariance (Sw_00, Sw_11) values below were pulled from
   // the autopilot site.  The proper settings for these values needs to be
   // explored further, but apparently Sw_00 is how much we trust the accelerometer
   // and Sw_11 is how much we trust the gyro.
   kalman_init(&roll_kalman_state, 0.0000998, 0.00002, 0.001, 0.003);
   kalman_init(&pitch_kalman_state, 0.0000998, 0.00002, 0.001, 0.003);
Fase_servoPWM[0] = 15;
Fase_servoPWM[1] = 15;
Fase_servoPWM[2] = 15;
Fase_servoPWM[3] = 21;
```

```
Fase_servoPWM[4] = 23;
Fase_servoPWM[5] = 23;
EINT;
ERTM;
ManejaJobs();
}
void IniciarGrabaSD(void)
{
files[0]=PreparaFile();
if (files[0]==NULL)
        {
        asm("           setc xf"); //problema en File ;
        errorSD = 3;
        mprocesos.bit.CierraGrabaSD=1;
        }
else
        {
        mprocesos.bit.CierraGrabaSD=0;
        }
}
void ManejaJobs(void)
{
while(TRUE) //Ojo así es sin el ; al final
        {
testV26=12;
```

```
GpioDataRegs.GPBDAT.bit.GPIOB8 = 1; //DEPURA CAP4
        if ((mprocesos.bit.Compas == 0)&& (mprocesos.bit.CalibrarCompas==1))
                {long int esperainicial=0;
                        asm("           setc XF");
                        esperainicial =secundero;
                        while((secundero- esperainicial)< 30) {}; //Esperar 60 segundos para preparar calibrador
                        asm("           clrc XF");
                        CalibrarCompas();
                        esperainicial =secundero;
                        while((secundero- esperainicial)< 150) {}; // Calibrar durante 150 segundos
                        NoCalibrarCompas(); // terminar de calibrar
                        asm("           setc XF");
                        mprocesos.bit.CalibrarCompas=0; // deshabilitar calibración
                };
        if(GPSnoData==1)
                if (prenderXF==1)
                        asm("           setc XF");
                else
                        asm("           clrc XF");
        else
                asm("           clrc XF");
        testV26=8;
        if (mprocesos.bit.GPSserial==1) GPSIteraSerial();
        GpioDataRegs.GPBDAT.bit.GPIOB8 = 0;   //DEPURA CAP4
```

```
        if (mprocesos.bit.CierraGrabaSD ==1) CierraGrabaSD(files[0]);
GpioDataRegs.GPDDAT.bit.GPIOD5 = 1;//DEPURA T3TRIP


        if ((mprocesos.bit.GrabaSD ==1) && (mprocesos.bit.CierraGrabaSD==0))
                if (GrabaSD(files[0])==0) //graba sd cada segundo y prueba si hubo error
                        {
                                errorSD=4; //error cancelar y cerrar
                                mprocesos.bit.CierraGrabaSD =1; //si hubo error cerrar SD prox.
                                testV26=91;
                        }
                else
                        mprocesos.bit.GrabaSD =0; //todo bien restaurar bandera
GpioDataRegs.GPDDAT.bit.GPIOD5 = 0;   //DEPURA T3TRIP
        if ((mprocesos.bit.Compas == 1)&& (mprocesos.bit.CalibrarCompas==0))
CompasLeer();



        testV26=13;
        };
}
void ini_eva_timer1(void)
{
   EvaRegs.T1PR = 46874;//0x0752;  // 100hz
   EvaRegs.T1CMPR = 11000;
   EvaRegs.EVAIMRA.bit.T1PINT = 1; // Habilitar l aint Perioto del T1
```

```
    EvaRegs.EVAIFRA.bit.T1PINT = 1; // Limpiando la bandera

    EvaRegs.EVAIMRA.bit.T1CINT = 1;

    EvaRegs.EVAIFRA.bit.T1CINT = 1;

    EvaRegs.T1CNT = 0x0000;

    EvaRegs.T1CON.all = 0x1402; //(75MHz/16)

    asm("                nop");

    asm("                nop");

    EvaRegs.T1CON.all = 0x1442;//(75MHz/16)

    EvaRegs.GPTCONA.bit.T1TOADC = 2; // interrupt del T1PER

    EvaRegs.GPTCONA.bit.T1PIN =2; // activo arriba

    EvaRegs.GPTCONA.bit.T1CMPOE =1; // habilitar las comparaciones

    EvaRegs.GPTCONA.bit.TCMPOE =1; // habilitar ambos comparadores sistema
antiguo

    }

void ini_eva_timer2(void)

{

    EvaRegs.T2PR = 31874;//0x0752;  // 10Khz

    EvaRegs.T2CMPR = 18000;

    EvaRegs.T2CNT = 0x0000;

    EvaRegs.T2CON.all = 0x1202; //(75MHz/4)

    asm("                nop");

    asm("                nop");

    EvaRegs.T2CON.all = 0x1242;

    EvaRegs.GPTCONA.bit.T2PIN =2; // activo arriba

    EvaRegs.GPTCONA.bit.T2CMPOE =1; // habilitar las comparaciones

    }
```

```
void ini_eva_timer3(void)
{
    EvbRegs.T3PR = 1874; //1khz 1874;//  10Khz
    EvbRegs.T3CMPR = 937;
    EvbRegs.EVBIMRA.bit.T3PINT = 1; // Habilitar l aint Perioto del T1
    EvbRegs.EVBIFRA.bit.T3PINT = 1; // Limpiando la bandera
    EvbRegs.EVBIMRA.bit.T3CINT = 1;
    EvbRegs.EVBIFRA.bit.T3CINT = 1;
    EvbRegs.T3CNT = 0x0000;
    EvbRegs.T3CON.all = 0x1202; //(75MHz/4)
    asm("                nop");
    asm("                nop");
    EvbRegs.T3CON.all = 0x1242;
    EvbRegs.GPTCONB.bit.T3TOADC = 2; // interrupt del T1PER
    EvbRegs.GPTCONB.bit.T3PIN =2; // activo arriba
    EvbRegs.GPTCONB.bit.T3CMPOE =1; // habilitar las comparaciones
    EvbRegs.GPTCONB.bit.TCMPOE =1; // habilitar ambos comparadores sistema
antiguo
    }
void ini_eva_timer4(void)
{
    EvbRegs.T4PR = 31874;//0x0752;  // 10Khz
    EvbRegs.T4CMPR = 18000;
    EvbRegs.T4CNT = 0x0000;
    EvbRegs.T4CON.all = 0x1202; //(75MHz/4)
    asm("                nop");
```

```
    asm("                nop");
    EvbRegs.T4CON.all = 0x1242;
    EvbRegs.GPTCONB.bit.T4PIN =2; // activo arriba
    EvbRegs.GPTCONB.bit.T4CMPOE =1; // habilitar las comparaciones
    }
interrupt void T1PER_isr(void)
{
EINT;
ERTM;
        controlRemoto=GpioDataRegs.GPBDAT.bit.GPIOB3;
   // Pass the measured pitch and roll values and rates through the Kalman filter to
   // determine the estimated pitch and roll values in radians.
   roll_est = kalman_update(&roll_kalman_state, roll_rate, roll);
   //pitch_est = kalman_update(&pitch_kalman_state, pitch_rate, pitch);

   EvaRegs.EVAIMRA.bit.T1PINT = 1;
   EvaRegs.EVAIFRA.all = BIT7;
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP2;
}
interrupt void T1CMP_isr(void)
{
EINT;
ERTM;
        Source= (void *)&AdcRegs.ADCRESULT0;
        Dest = ResultadoADC;
```

```
                for(i=0; i < 16; i++)
                *Dest++ = *Source++;
    // Get the x and y gyro values.
    gyro_x = (float) (ResultadoADC[2]>>4);
    gyro_y = (float) (ResultadoADC[15]>>4);
        // Get the x, y and z accelerometer values.
    accel_x = (float) (ResultadoADC[8]>>4);
    accel_y = (float) (ResultadoADC[9]>>4);
    accel_z = (float) (ResultadoADC[10]>>4);
        // Zero adjust the gyro values.  A better way of dynamically determining
        // these values must be found rather than using hard coded constants.
        gyro_x -= 2027.5;
        gyro_y -= 2028.5;

        // Zero adjust the accelerometer values.  A better way of dynamically determining
        // these values must be found rather than using hard coded constants.
        accel_x -= 2006.0;
        accel_y -= 2026.0;
        accel_z -= 1800.0;
        accel_z = -accel_z;

        // Determine the pitch and roll in radians.  X and Z acceleration determines the
        // pitch and Y and Z acceleration determines roll.  Note the accelerometer vectors
```

```
        // that are perpendicular to the rotation of the axis are used.  We can compute the

        // angle for the full 360 degree rotation with no linearization errors by using the

        // arctangent of the two accelerometer readings.  The accelerometer values do not

        // need to be scaled into actual units, but must be zeroed and have the same scale.

        // Note that we manipulate the sign of the acceleration so the sign of the accelerometer

        // derived angles match the gyro rates.

        roll = atan2(accel_y, accel_z);

        pitch = atan2(-accel_x, accel_z);



        // Determine gyro angular rate from raw analog values.

        // Each ADC unit: 3300 / 1024 = 3.222656 mV

        // Gyro measures rate: 2.0 mV/degrees/second

        // Gyro measures rate: 114.591559 mV/radians/second

        // Each ADC unit equals: 3.222656 / 2.0 = 1.611328 degrees/sec

        // Each ADC unit equals: 3.222656 / 114.591559 = 0.0281230 radians/sec

        // Gyro rate: adc * 0.0281230 radians/sec

        roll_rate = gyro_x * 0.0025566;

        pitch_rate = gyro_y * 0.0025566;
  elevador=FuzzificarAlt(alturaF); //alturaF proviene del sensor de altura en zlog.c

  timon=FuzzificarDir((long int)Heading); // Heading proviene del compás del IMU



  EvaRegs.EVAIMRA.bit.T1CINT = 1;

  EvaRegs.EVAIFRA.all = BIT8;
```

```
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP2;
}
interrupt void T3PER_isr(void)
{
        if(controlRemoto==0)
        {
           if(Fase_servoPWM[0]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA0=0;
                if(Fase_servoPWM[1]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA1=0;
                if(Fase_servoPWM[2]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA2=0;
                if(Fase_servoPWM[3]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA3=0;
                if(Fase_servoPWM[4]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA4=0;
                if(Fase_servoPWM[5]==cuenta_servoPWM)
                                GpioDataRegs.GPADAT.bit.GPIOA5=0;
                if(cuenta_servoPWM==198)// 9)  //equivale a 20mS
                        {cuenta_servoPWM=0;
                GpioDataRegs.GPADAT.bit.GPIOA0=1; //OJO la subida de línea
puede demorar 20-30useg
                GpioDataRegs.GPADAT.bit.GPIOA1=1;
                GpioDataRegs.GPADAT.bit.GPIOA2=1;
                GpioDataRegs.GPADAT.bit.GPIOA3=1;
                GpioDataRegs.GPADAT.bit.GPIOA4=1;
```

```
                GpioDataRegs.GPADAT.bit.GPIOA5=1;
                        };
                cuenta_servoPWM++;
        }
        else
        {
        dataCH[0] = GpioDataRegs.GPBDAT.bit.GPIOB0;
        //GpioDataRegs.GPBDAT.bit.GPIOB3 = dataCH[0];
        GpioDataRegs.GPADAT.bit.GPIOA0 = dataCH[0];
        dataCH[1] = GpioDataRegs.GPBDAT.bit.GPIOB1;
        GpioDataRegs.GPBDAT.bit.GPIOB4 = dataCH[1];
        dataCH[2] = GpioDataRegs.GPBDAT.bit.GPIOB2;
        GpioDataRegs.GPBDAT.bit.GPIOB5 = dataCH[1];
        };

        Fase_servoPWM[0]=14+(int)timon;
        Fase_servoPWM[2]=16+(int)(elevador/2.0);
   EvbRegs.EVBIMRA.bit.T3PINT = 1;
   EvbRegs.EVBIFRA.all = BIT7;
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
}
interrupt void T3CMP_isr(void)
{

   EvbRegs.EVBIMRA.bit.T3CINT = 1;
```

```
   EvbRegs.EVBIFRA.all = BIT8;
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
}
interrupt void T2_CPU_isr(void)
{
        decimosegundo++;
        if(decimosegundo == 10) //El timer esta a 0.1 segundos por interrupción
        {
        decimosegundo=0;
        secundero++;
                if (errorSD==0)
                {
                        if((alternaXF ^= 0x1)==1)
                                prenderXF=1;
                        else
                                prenderXF=0;
                };
        };
        if (secundero<216000) mprocesos.bit.GrabaSD=1; //permite grabar el SD cada 1 segundo
        else mprocesos.bit.CierraGrabaSD=1;  //Ojo con el límite que le impone a la memoria flash
}
void initADC(void)
{
   // Configuración del ADC:
```

```
   AdcRegs.ADCTRL1.bit.RESET = 1;

         asm(" NOP ");

         asm(" NOP ");

   AdcRegs.ADCTRL1.bit.ACQ_PS = 15;

   AdcRegs.ADCTRL3.bit.ADCCLKPS = 1;

   // Arranque de las fuentes

   AdcRegs.ADCTRL3.bit.ADCBGRFDN = 3;

   DSP28x_usDelay(((((long double) 5000L * 1000.0L) / (long double)CPU_RATE)
- 9.0L) / 5.0L);

   AdcRegs.ADCTRL3.bit.ADCPWDN = 1;

   DSP28x_usDelay(((((long double) 5000L * 1000.0L) / (long double)CPU_RATE)
- 9.0L) / 5.0L);

   AdcRegs.ADCTRL1.bit.RESET = 0;

   AdcRegs.ADCTRL1.bit.SEQ_CASC = 1;        // 1  Cascaded mode

   AdcRegs.ADCCHSELSEQ1.all = 0x3210;

      AdcRegs.ADCCHSELSEQ2.all = 0x7654;

         AdcRegs.ADCCHSELSEQ3.all = 0xBA98;

            AdcRegs.ADCCHSELSEQ4.all = 0xFEDC;

   AdcRegs.ADCMAXCONV.bit.MAX_CONV1 = 15;

   AdcRegs.ADCTRL1.bit.CONT_RUN = 0;       // Modo no continuo

   AdcRegs.ADCTRL2.bit.EVA_SOC_SEQ1 = 1; // SOC desde EVA

   AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;  //Sugerencia de Kick Off

         while(AdcRegs.ADCST.bit.SEQ1_BSY ==1) {};

         Source= (void *)&AdcRegs.ADCRESULT0;

         Dest = ResultadoADC;

                  for(i=0; i < 16; i++)
```

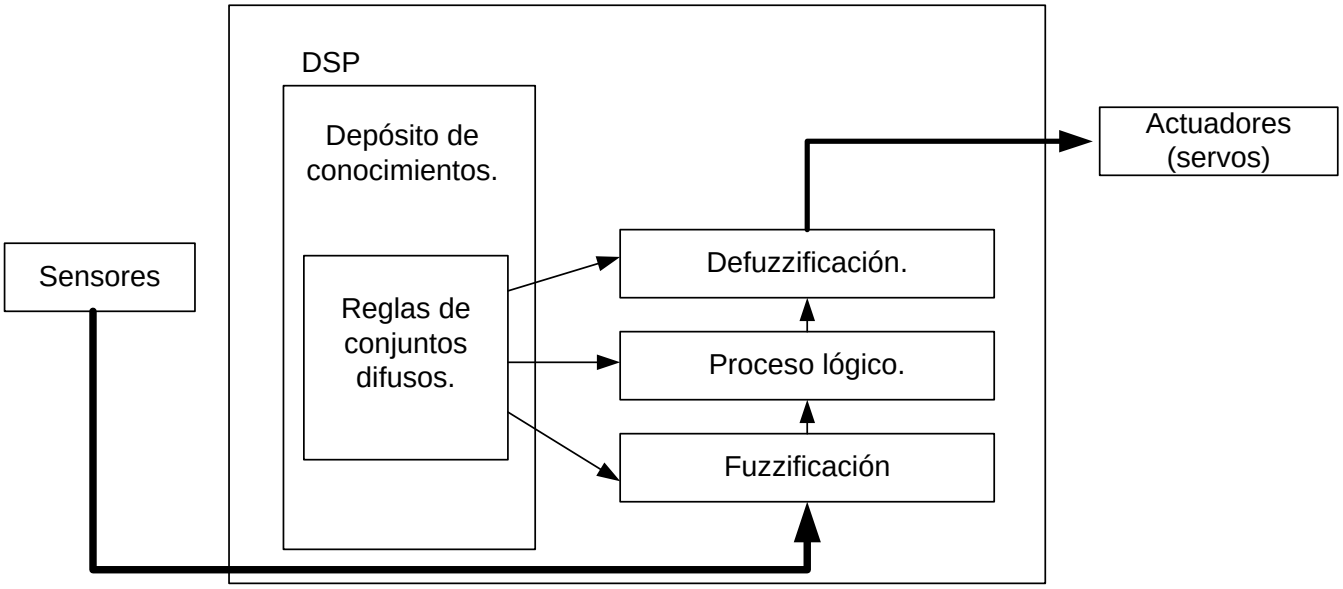
```
                *Dest++ = *Source++;
}
void init_pwm_eva()
{
  EvaRegs.GPTCONA.bit.T1CMPOE = 1; // Habilitar comparaciones
    EvaRegs.GPTCONA.bit.T2CMPOE = 1;
          EvaRegs.GPTCONA.bit.TCMPOE =1;
  EvaRegs.GPTCONA.bit.T1PIN = 1;// Polaridad del pin T1PWM Active low
  EvaRegs.GPTCONA.bit.T2PIN = 2;// Polaridad del pin T2PWM Active high
  EvaRegs.CMPR1 = 0x1600;// Fase de comparación for PWM1-PWM6
  EvaRegs.CMPR2 = 0x1300;
  EvaRegs.CMPR3 = 0x1500;
  EvaRegs.ACTRA.all = 0x0AAA; // Polaridad de los PWM1-PWM6
    EvaRegs.DBTCONA.bit.DBT = 15; //Tiempo muerto
    EvaRegs.DBTCONA.bit.EDBT1 = 1;
    EvaRegs.DBTCONA.bit.EDBT2 = 1;
    EvaRegs.DBTCONA.bit.EDBT3 = 1;
    EvaRegs.DBTCONA.bit.DBTPS= 0x4;
  EvaRegs.COMCONA.all = 0xA6E0;     // Control de PWMs
}
void init_pwm_evb()
{
  EvbRegs.GPTCONB.bit.T3CMPOE     = 1;
    EvbRegs.GPTCONB.bit.T4CMPOE   = 1;
          EvbRegs.GPTCONB.bit.TCMPOE=1;
```

```
  EvbRegs.GPTCONB.bit.T3PIN = 1;
  EvbRegs.GPTCONB.bit.T4PIN = 2;
  EvbRegs.CMPR4 = 0x2000;
  EvbRegs.CMPR5 = 0x1200;
  EvbRegs.CMPR6 = 0x1b00;
  EvbRegs.ACTRB.all = 0x0666;
    EvbRegs.DBTCONB.bit.DBT = 15;
    EvbRegs.DBTCONB.bit.EDBT1 = 1;
    EvbRegs.DBTCONB.bit.EDBT2 = 1;
    EvbRegs.DBTCONB.bit.EDBT3 = 1;
    EvbRegs.DBTCONB.bit.DBTPS= 0x4;
  EvbRegs.COMCONB.all = 0xA6E0;
}
int iniciarSD(void)
{
/* Initialize the SPI controller */
// spi_initialize(); ya no es necesario
/* Start out with a slow SPI clock, 400kHz, as required by the SD spec
(for MMC compatibility). */
spi_set_divisor(PERIPH_CLOCKRATE/400000);
/* Initialization OK? */
testV26=10;
if (sd_initialize() != 1)
return 0;
testV26=14;
```

```
/* Set the maximum SPI clock rate possible */
spi_set_divisor(2);


return 1;
}
void imprimirChar(unsigned char c)
{
Hoja[indiceHoja]=c;
indiceHoja++;
if (indiceHoja>         8000) indiceHoja=0;
}
unsigned long int retardar(unsigned long int retardo)
{
int r=0;
do{
                    for(r=0;r<12000;r++){asm("                    nop");};
                    retardo--;
          }while (retardo!=0);
return retardo;
}
```

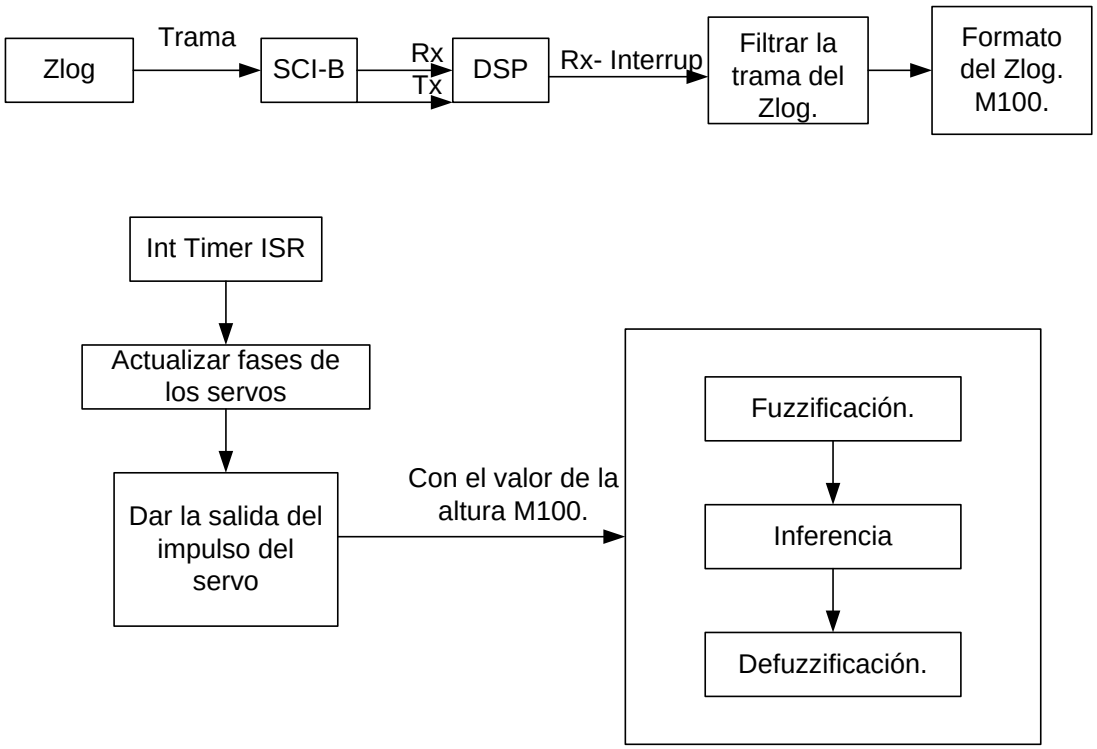
7.4.-DIAGRAMA DE BLOQUES DEL FLUJO DEL PROGRAMA DE CONTROL.

Diagrama de bloques del flujo del programa de control implementado en el DSP.



7.5.-DIAGRAMA DE FLUJO DEL PROGRAMA DE CONTROL.

Diagrama de flujo con respecto al sensor de Altura (Zlog).



Fuzzificación de la variable M100 del Zlog. Variable de altura.

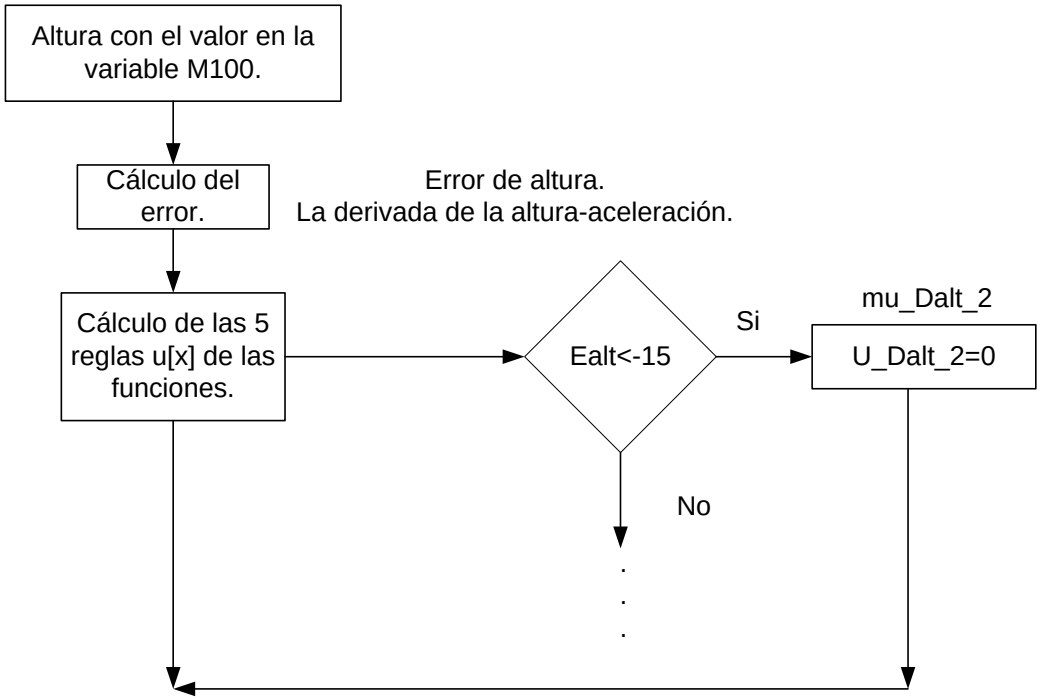


Diagrama de bloques de las reglas de pertenencia del sensor de altura (Zlog).

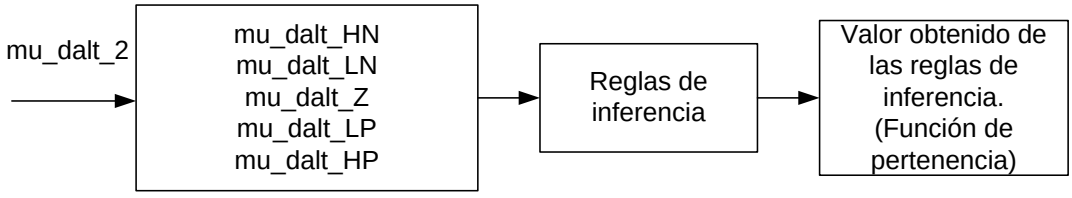


Diagrama de bloques del Defuzzicador del sensor de altura (Zlog).

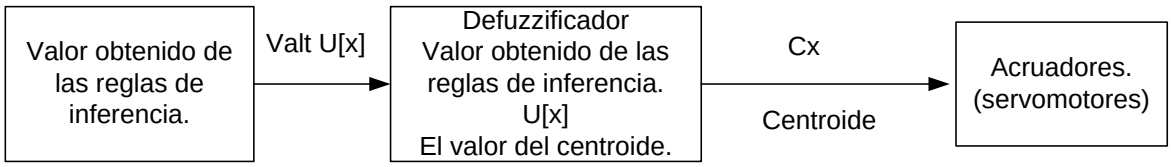
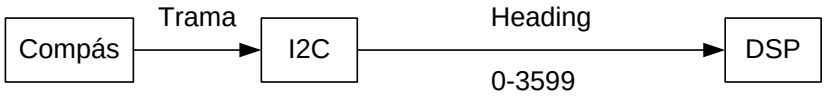
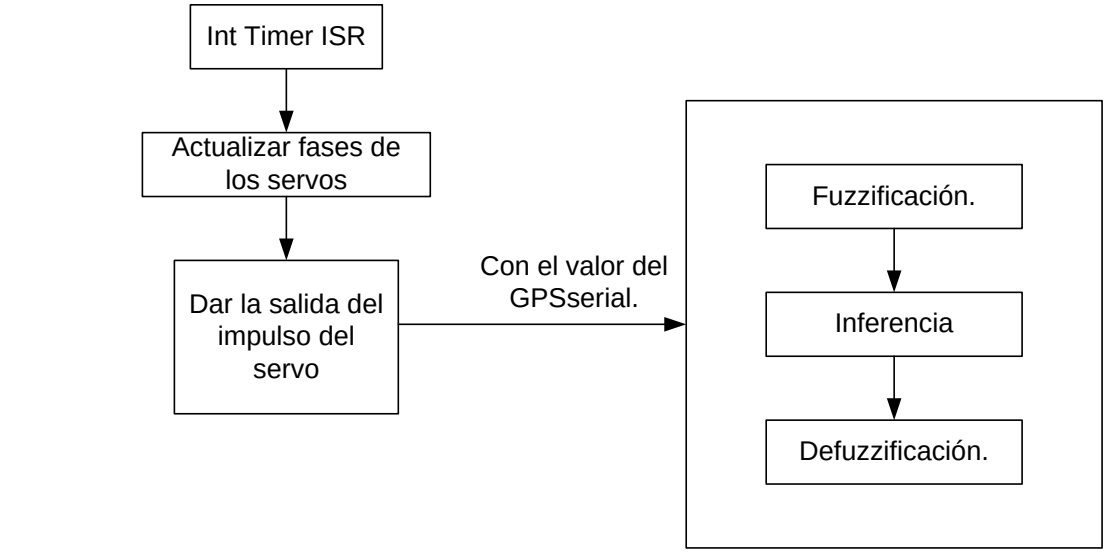


Diagrama de flujo con respecto a la dirección. (Compás).



Rutina de proceso.



Fuzzificación de la variable dirección. Sensor Compás.

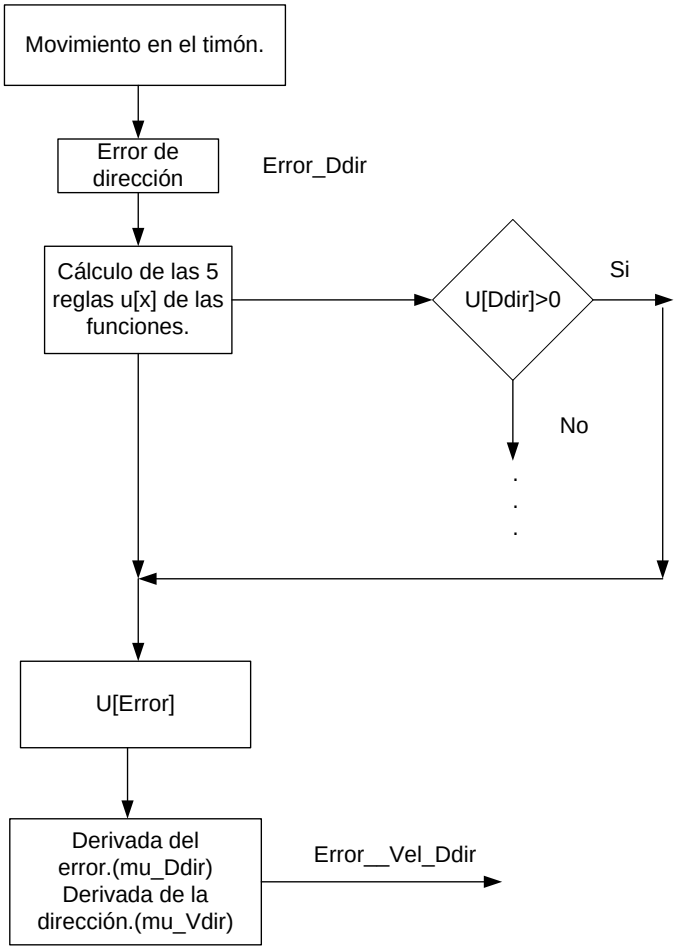


Diagrama de bloques de las reglas de pertenencia de la dirección (Compás).

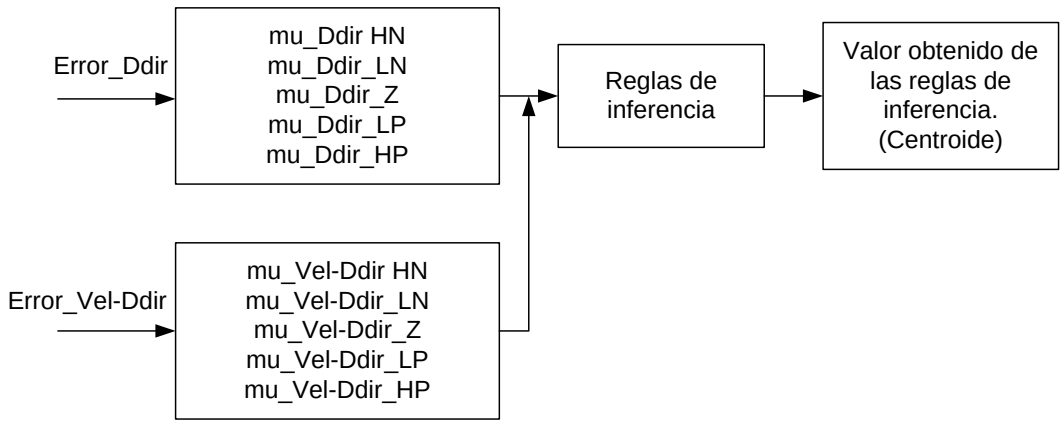
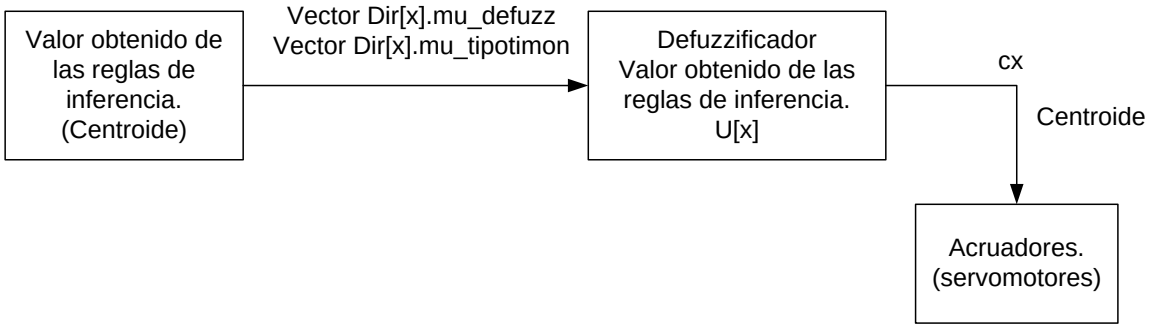


Diagrama de bloques del Defuzzicador de la dirección (Compás).



8.-Conclusiones.

- El uso de los UAV reemplazaría los altos costos de combustible en misiones militares o civiles en donde hay riesgos para los pilotos y las máquinas.
- Los UAV pueden darnos información en tiempo real a través de sus sensores o utilizando cámaras como aviones observadores en donde puedan ser utilizados como aviones espías o monitoriando el tráfico de una vía.
- Los coeficientes dinámicos como el de la sustentación (lift), arrastre (drag) y las fuerzas laterales como el giro en eje Y (pich), giro en eje X (roll) , giro en eje Z (yaw) fueron tomadas de modelos iguales desarrollados en otras tesis y de tablas aerodinámicas.
- El vuelo a través de la lógica difusa si es posible, porque el procesador que se requiere es de menor capacidad en contraposición a la implementación del control de tipo óptimo en el DSP.
- Para que la lógica difusa funcione en un aeromodelo, la persona que implementa el controlador difuso debe de tener en cuenta el comportamiento de la aeronave.
- Las reglas en el controlador difuso fueron probados en el aeromodelo en varios vuelos para obtener un óptimo de resultado.
- A pesar que las matrices de estado fueron obtenidas de forma aproximada, describen al sistema bastante bien respecto al modelo real del avión.
- La simulación con las matrices de estado a través del control óptimo nos muestran la controlabilidad del sistema ante diferentes situaciones.
- Las características inerciales (masa, momento de inercia) del avión cambian conforme el combustible se consume, esto en un modelo matemático debe de tomarse en cuenta.El uso de sensores en el IMU tales como: Giróscopos, acelerómetros permiten controlar mejor el avión en el estado estacionario.
- Al poseer el IMU un GPS permite conocer la posición, la dirección, altitud permite controlar de forma más exacta la dinámica lateral del avión.

- El viento es uno de los factores negativos en la experimentación. Se esperaba que esta variable pudiera cambiar de dirección al avión en su trayectoria, afortunadamente no hubo mayores contratiempos.
- El presente trabajo abré una puerta para nuevos estudiantes de pre-grado y maestría para el desarrollo futuro de plataformas mucho más complejas.

9.-BIBLIOGRAFÍA

[Fuzzy07] 1 .Fuzzy logic Toolbox- User´s Guide. Matlab 2007.pp. 2-4, 2-132.

[RNSD01] 2. Redes Neuronales y Sistemas Difusos- Alfaomega 2001.Segunda Edición. Autores: Bonifacio Martín del Bríο. Alfredo Sanz Molina.pp.283-334.

[Springer07] 3. Introduction to Fuzzy Logic using MATLAB -Springer. by S.N. Sivanandam (Author), S. Sumathi (Author), S. N. Deepa (Author) 2007. pp.38-209.

[Springer01] 4. Fuzzy Logic Techniques for Autonomous Vehicle Navigation-Springer 2001. Autores: Dimiter Driankow, Alessandro Saffiotti.pp 25-46 y 51-72.

[AFD98] 5. Introduction to Aircraft Flight Dynamics (AIAA. Education Series) L.Stevens and Louis V. Schmidt 1998. pp.2-163

[ACS03] 6. Aircraft Control and Simulation Autores: Brian L. Stevens (Author), Frank L. Lewis (Author) –Wiley 2003. pp.254-301.

[FUZZY96] 7. Fuzzy Logic- Implementation and applications-Wiley 1996. pp.117-134.

[FTB01] 8. The Fuzzy Future.-Bart Kosko 2001.pp.158-169.

[RM07] 9. Robótica manipuladores y robot móviles. 2007. Anibal Ollero Baturone.pp-303-320.

[CA01] 10. Control Avanzado. Diseño y aplicaciones en tiempo real. Arturo Rojas Moreno.2001.pp123-187.

[Airfoil] 11. Airfoil. Software de simulación de perfiles. V.2.2

[WingS] 12. Theory of wing Section Whit summary airfoil data.Abbot and Doenhoff.pp-5-111.

[Airlow] 13. Airfoils at low speeds. Michael S.Selig, John Donovan and B. Fraser. Soartech 8. 1989. pp.2-129

Dedicada a mis Padres, a mi
esposa y a mi hijo Esteban.

El autor.

Un agradecimiento muy especial al asesor de mi tesis MSc. Víctor Sotelo por sus consejos y el apoyo brindado, los cuáles fueron muy útiles para llevar a cabo la programación del DSP. Igualmente agradezco a los revisores de la tesis: Dr Aurelio Albildo López y al Dr Antonio Morán, quienes han colaborado con sus comentarios a darle claridad a este trabajo.

El autor.