RAE

1. **TIPO DE DOCUMENTO:** Trabajo de grado para optar por el titulo de INGENIERO AERONÁUTICO
2. **TÍTULO:** DESARROLLO DE LA SIMULACIÓN DE UN SISTEMA DE CONTROL QUE MANTENGA LA TRAYECTORIA DE UN COHETE BALÍSTICO ATMOSFÉRICO, NO TELEDIRIGIDO
3. **AUTOR (ES):** Angélica María Rincón Cardozo , Yesid Camilo Parra Rojas
4. **LUGAR:** Bogotá D.C.
5. **FECHA:** Octubre de 2016
6. **PALABRAS CLAVE:** Cohete atmosférico, función de transferencia, controlador, análisis dinámico, PID.
7. **DESCRPCIÓN DEL TRABAJO:** Este proyecto presenta el diseño de un sistema de control capaz de mantener una trayectoria recta modificando la posición de cabeceo en un cohete atmosférico. Esto se hizo bajo las especificaciones de diseño del cohete Ainkka V construido en la Universidad de San Buenaventura, sede Bogotá. El sistema se basó en el cálculo de la función de transferencia que relaciona el ángulo de salida (que en este caso es el de cabeceo) con la deflexión de las aletas. Posteriormente se implementó un controlador que por medio de una señal retroalimentada es capaz de estabilizar el sistema, para obtener el resultado deseado.
8. **LÍNEA DE INVESTIGACIÓN:** La línea de investigación a la que pertenece este proyecto, es vehículos aeroespaciales y el campo de investigación es diseño de sistemas de control para vehículos tipo cohete.
9. **METODOLOGÍA:** Pertenece a un enfoque empírico-analítico, ya que a partir de los datos obtenidos con el cohete Ainkka V desarrollado anteriormente, se busca analizar, comprender y controlar variables del sistema, las cuales presentan comportamientos frente a condiciones de vuelo vertical, esto por medio del diseño de un sistema de control que asegure una trayectoria recta del cohete, que será implementado en las aletas del mismo.
10. **CONCLUSIONES:** Por medio de Rocksim V9.1.3® se simularon los parámetros del vuelo del cohete Ainkka V, los cuales se utilizaron para el desarrollo del sistema de control y por medio de análisis dinámicos se pudo generar un modelo capaz de estabilizar el cohete en su ángulo de cabeceo, el cual implementa un control PID, que llega a estabilizar el sistema conforme a un valor de entrada deseado (ángulo de cabeceo). Además de esto, se pretende que este sistema de control sea implementado físicamente en futuros proyectos.

DESARROLLO DE LA SIMULACIÓN DE UN SISTEMA DE CONTROL QUE MANTENGA LA TRAYECTORIA DE UN COHETE BALÍSTICO ATMOSFÉRICO, NO TELEDIRIGIDO

INTEGRANTES:

ANGÉLICA MARÍA RINCÓN CARDOZO

YESID CAMILO PARRA ROJAS

UNIVERSIDAD DE SAN BUENAVENTURA

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

INGENIERÍA AERONÁUTICA

BOGOTÁ D.C.

2016

DESARROLLO DE LA SIMULACIÓN DE UN SISTEMA DE CONTROL QUE MANTENGA LA TRAYECTORIA DE UN COHETE BALÍSTICO ATMOSFÉRICO, NO TELEDIRIGIDO

INTEGRANTES:

ANGÉLICA MARÍA RINCÓN CARDOZO

Cód. 20111231074

YESID CAMILO PARRA ROJAS

Cód. 20111231064

Trabajo de grado para optar el título de Ingeniero Aeronáutico

DIRECTOR:

JOSÉ ALEJANDRO URREGO

M.Sc Ingeniería Mecánica

UNIVERSIDAD DE SAN BUENAVENTURA

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

INGENIERÍA AERONÁUTICA

BOGOTÁ D.C.

2016

Nota de aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C. 2016

DEDICATORIAS

Durante la existencia del ser humano llegan momentos que hacen que venir al mundo valga la pena, es por eso que con este proyecto hecho realidad me doy cuenta que se lo debo a Dios, ya que el saber que culmina esta etapa de mi vida y que siguen nuevos proyectos hacen que la vida tenga una razón de ser y por la cual se debe luchar hasta conseguir lo que se quiere.

Para conseguir esos logros cuento con mis padres Miguel y Esperanza quienes me apoyaron en la iniciativa de estudiar Ingenieria Aeoronáutica y así mismo contribuyeron durante toda mi carrera, gracias a sus esfuerzos que se reflejaban día a día para que nunca me rindiera y asi pudiese lograr mi objetivo.

A mis familiares en especial a mi tia Gloria y mi primo William, quienes pusieron toda su confianza en mí y me ayudaron a salir adelante con la carrera.

A mi amigo y colega Yesid con quien vivimos momentos de peleas, mal entendidos, pero al mismo tiempo supimos sobrellevar los obstáculos que de alguna forma se nos presentaban en el desarrollo de nuestra tesis.

Angélica María Rincón Cardozo

DEDICATORIAS

Este proyecto de grado, que culmina con mi formación profesional como Ingeniero Aeronáutico quiero dedicárselo a mi mamá, papá y mis hermanas que estuvieron acompañándome durante todo este proceso que inicié en la universidad.

Yesid Camilo Parra Rojas

AGRADECIMIENTOS

Agradecemos al docente de la Facultad de Mecatrónica el Ingeniero Julian Severiano Acevedo, por habernos colaborado de manera cordial en el desarrollo de este proyecto de grado, al Ingeniero Victor Cruz por darnos a conocer sus conocimientos, los cuales ayudaron en el desarrollo del proyecto y a nuestro tutor el Ingeniero Alejandro Urrego por encomendarnos la tarea de desarrollar este proyecto, el cual fortalecio nuestros conocimientos.

LISTA DE FIGURAS

LISTA DE GRÁFICAS

LISTA DE TABLAS

LISTA DE ANEXOS

NOMENCLATURA

$A = relacion\ de\ aspecto\ del\ ala.$

$a = area\ de\ tobera.$

$al = ancho\ de\ la\ aleta.$

$b = envergadura.$

$\bar{c} = cuerda\ media.$

$C_{Fxa} = Coeficiente\ de\ fuerza\ en\ la\ dirección\ X\ por\ deflexión\ del\ elevador.$

$C_{Fza} = Coeficiente\ de\ fuerza\ en\ la\ dirección\ Z\ por\ deflexión\ del\ elevador.$

$C_{L\alpha} = pendiente\ de\ la\ curva\ de\ sustentación\ del\ ala.$

$C_{lq} = coeficiente\ de\ la\ derivada\ de\ cabeceo.$

$C_{ma} = Coeficiente\ momento\ de\ cabeceo, por\ movimiento\ de\ superficies\ de\ control.$

$C_{mq} = coeficiente\ de\ amortiguación\ en\ el\ eje\ de\ cabeceo.$

$C_{mu} = efecto\ del\ flujo\ de\ aire\ que\ se\ desplaza.$

$C_{m\alpha} = coeficiente\ del\ momento\ de\ cabeceo.$

$C_{m\delta e} = coeficiente\ de\ momento\ de\ deflexión\ del\ elevador.$

$C_r = Cuerda\ de\ raiz.$

$C_t = Cuerda\ de\ punta.$

$C_{xq} = efecto\ de\ la\ tasa\ de\ cabeceo\ sobre\ el\ arrastre.$

$C_{xu} = variación\ del\ arrastre\ y\ en\ empuje\ respecto\ a\ u.$

$C_{x\alpha} = variación\ de\ sustentación\ y\ arrastre\ a\ lo\ largo\ del\ eje\ x.$

$C_{x\dot{\alpha}} = flujo\ de\ aire\ deflectado\ hacia\ abajo\ del\ arrastre.$

$C_w = coeficiente\ gravitacional.$

$C_{zu} = variación\ de\ la\ fuerza\ normal\ respecto\ a\ u.$

$C_{zq} = efecto\ de\ la\ tasa\ de\ cabeceo\ sobre\ la\ sustentación\ .$

$C_{z\alpha} = coeficiente\ de\ la\ pendiente\ de\ la\ curva\ de\ la\ fuerza\ normal\ .$

$C_{z\dot{\alpha}} = flujo\ de\ aire\ deflectado\ en\ la\ sustentación\ de\ la\ aleta.$

$C_{z\delta e} = variación\ de\ la\ fuerza\ normal\ con\ respecto\ a\ la\ deflexión\ del\ elevador.$

$d = diámetro\ del\ cohete.$

$I_x = momento\ de\ inercia\ en\ el\ eje\ x.$

$I_y = momento\ de\ inercia\ en\ el\ eje\ y.$

$I_z = momento\ de\ inercia\ en\ el\ eje\ z.$

$J_{xz} = producto\ de\ inercia.$

$K_p = ganancia\ proporcional\ ajustable.$

$K_i = ganancia\ integral.$

$K_d = ganancia\ derivativa.$

$l = longitud\ del\ cohete.$

$\dot{m} = tasa\ de\ flujo.$

$n = distancia\ entre\ el\ eje\ del\ ala\ al\ centro\ de\ la\ cuerda\ aerodinámica.$

$n_{zc} = factor\ de\ carga\ en\ el\ plano\ de\ cabeceo.$

$P = velocidad\ angular\ en\ el\ eje\ x.$

$P_e = presión\ de\ salida.$

$P_a = presión\ de\ ambiente.$

$Q = velocidad\ angular\ respecto\ al\ eje\ y.$

$q = presión\ dinámica.$

$r = radio\ del\ fuselaje\ del\ cohete.$

$R = velocidad\ angular\ respecto\ al\ eje\ z.$

$S = área\ de\ la\ aleta.$

$T = empuje\ .$

$ts = tiempo\ de\ estabilización.$

$U = velocidad\ lineal\ respecto\ al\ eje\ x.$

$\dot{U} = tasa\ de\ velocidad\ lineal\ respecto\ al\ eje\ x.$

$'_u = variación\ del\ ángulo\ de\ ataque\ desde\ el\ equilibrio$

$'_{\dot{u}} = tasa\ de\ variación\ del\ ángulo\ de\ ataque\ desde\ el\ equilibrio$

$'_\alpha = variación\ del\ ángulo\ de\ guiñada\ desde\ el\ equilibrio$

$'_{\dot{\alpha}} = tasa\ de\ variación\ del\ ángulo\ de\ guiñada\ desde\ el\ equilibrio$

$v = velocidad\ del\ cohete.$

$V = velocidad\ lineal\ respecto\ al\ eje\ y.$

$\dot{V} = tasa\ de\ velocidad\ lineal\ respecto\ al\ eje\ y.$

$\omega = frecuencia.$

$w_n = frecuancia\ natural.$

$W = velocidad\ lineal\ respecto\ al\ eje\ z.$

$\dot{W} = tasa\ de\ velocidad\ lineal\ respecto\ al\ eje\ z.$

$X_{a.c.} = distancia\ del\ centro\ aerodinámico.$

$X_{cg} = centro\ de\ gravedad\ del\ cohete.$

$y_{mac} = ubicación\ de\ la\ envergadura\ de\ la\ cuerda\ media\ aerodinámica.$

$\lambda = relación\ de\ taperado.$

$\tau = constante\ del\ tiempo.$

$\tau s = operador\ de\ Laplace.$

$\zeta = factor\ de\ amortiguamiento.$

$\Lambda_{LE} = borde\ de\ ataque\ del\ ala.$

$\Lambda_{c/4} = el\ 25\%\ de\ la\ cuerda\ media\ de\ la\ aleta.$

$\Theta = ángulo\ de\ cabeceo.$

$\theta = ángulo\ de\ cabeceo\ respecto\ al\ eje\ de\ equilibrio.$

ÍNDICE

INTRODUCCIÓN

Tras la fase de impulso, un cohete sigue una trayectoria determinada de acuerdo a la misión con la cual fue diseñado. Ahora, en el espacio aéreo en el cual un cohete está volando, no siempre se encontrarán las condiciones esperadas para un vuelo rectilíneo no perturbado, es decir, se producirán condiciones del medio ambiente adversas. Estas condiciones están ligadas a fenómenos tales como ráfagas o vientos cruzados que pueden causar el desvió de la trayectoria o el rumbo del cohete. Esta desviación se puede evidenciar en grados de rotación respecto al eje longitudinal, por ejemplo, pueden existir situaciones en las que estas desviaciones sean mayores, por ejemplo superando aproximadamente los 10 grados con respecto al eje de la trayectoria [1], si es así, el cohete perderá el control, causando atrasos en la misión o el incumplimiento de la misma.

Es por lo anterior, que el proyecto hace referencia a la simulación de un sistema de control para un cohete atmosférico no- teledirigido, con el fin de estabilizarlo y así cumplir con la misión.

De acuerdo a las características de los cohetes, estos poseen 3 ejes de movimiento: guiñada, alabeo y cabeceo. En este proyecto se escogió el eje encargado del movimiento de cabeceo del cohete. Como resultado de la simetría en el plano YZ, tal como se aprecia en la figura 1, viendo el cohete desde el eje longitudinal, la función de transferencia que se va a realizar para el control del eje de cabeceo también puede ser usada para el control del eje de guiñada.

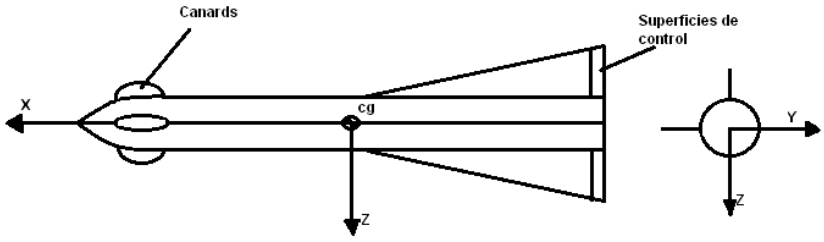


Figura 1. Esquema de un cohete aerodinámico y sus ejes

Se modeló el sistema de control activo a partir de ecuaciones diferenciales lineales e invariantes en el tiempo. Con estas se generó una función de transferencia que se modeló y se simuló por medio de Simulink.

El proceso de simulación en Simulink® posee dos fases, la primera se refiere al diseño del modelo de acuerdo a las especificaciones de la planta y la segunda fase es el análisis del modelo junto con el controlador que estabilizará el sistema en su eje de cabeceo.

Conforme a los resultados de la simulación se obtuvieron diagramas en donde se puede observar la respuesta del sistema respecto al tiempo.

1. PLANTEAMIENTO DEL PROBLEMA

1.1. Antecedentes

A nivel internacional se han desarrollado proyectos encaminados al sistema de control, simulación y parametrización de aeronaves no tripuladas. En 1999 un estudiante de Ingeniería Aeronáutica en el College of Engineering and Computer Design, como trabajo de maestría realizó el diseño de un piloto automático para vehículos de lanzamiento desechables, en este trabajo se desarrolló un modelo para los misiles, con ayuda de la teoría de dinámica de vuelo aplicada en misiles, modelado del control automático y simulación de este.

En el año 2004, el estudiante Reed Siefert Christiansen, de Brigham Young University, realizo el trabajo de maestría 'Diseño de un piloto automático para vehículos aéreos no tripulados', mostrando paso a paso los diferentes tipos de control y modelos empleados para los controladores automáticos, teniendo en cuentas los programas Simulink® y Flight Gear [2].

Para el año 2005 Honeywell Space System genera un diagnóstico integrado para el control de vuelo de un cohete, por medio de algoritmos de VHM (Vehicle Health Management) que incluye actuadores de control de vuelo, sensores, navegación y sistema de propulsión; en este mismo año, el UAV 'LAIMA' fue la primera aeronave de este tipo en cruzar completamente el atlántico exitosamente, todo esto por medio de un piloto automático, es decir, el vuelo se realizó de manera autónoma [3]. En el 2007, los estudiantes Florian Mutter y Stefanie Gareis realizaron la publicación "Model Driven In the loop Validation: Simulation based Testing of UAV software using Virtual environments", en el cual hablan de las ventajas de las simulaciones virtuales de modelos de control automáticos para los UAV's que necesiten entrar a terrenos poco explorados, y sobre las bases fundamentales para este tipo de control [4].

En el año 2011, en la tercera conferencia internacional de Nuevas tecnologías y automatización mecatrónica se expuso la investigación “Diseño de un piloto automático para vehículos superficiales no tripulados”, en este trabajo se expuso un modelo de simulación para vehículos como los barcos, ya que el control de estos es complejo por condiciones ambientales, y por último se realizaron las simulaciones experimentales usando Matlab®.

En la octava Conferencia Internacional de Ingeniera Aeronáutica se expuso la investigación “Educación en el control automático usando Flight Gear y Matlab® como laboratorios virtuales”, en el cual se establecieron varios experimentos en estas plataformas virtuales en los cuales se hace un enfoque en el entendimiento de conceptos fundamentales en el control clásico, dirigido a estudiantes de pregrado de áreas de interés.

En este mismo año en el “Chinese Journal of Aeronautics” se publicó una investigación “Robust Hibrid Control for Ballistic Missile Longitudinal Autopilot”, el cual investiga la fase de arranque del piloto automático longitudinal para un misil balístico equipado con un control vectorial del empuje.

En el año 2011 los estudiantes Lizeth Buendía, Rodrigo Lezama y Daniel Delgado de la Universidad de Concordia realizaron el proyecto de grado “Diseño del piloto automático para el F-16”, en este trabajo muestran de una manera clara el proceso para este diseño, abarcando desde la teoría, el diseño del modelo y el diseño del controlador en Matlab® [5]. En febrero de ese mismo año se publicó el artículo de investigación “Diseño del piloto automático para misiles tácticos” por parte del Ingeniero D. Viswanath, en el cual hace énfasis en los modelos matemáticos aplicados a los diferente tipos de control para las superficies aerodinámicas.

En el 2012 la Estudiante Ingrid Hagen Johansen realizó como proyecto de Maestría en la Universidad de Ciencia y Tecnología de Noruega un piloto automático para vehículos aéreos no tripulados, donde escoge un UAV para generar el modelo, diseñar los sistemas de piloto

automático para despegue, control de altitud y giros y generando una visualización en X-plane [6].

En la Conferencia Internacional de Computación y Aplicación de la información, se desarrolló un modelo basado en el piloto automático de un UAV por medio de Software, para el desarrollo de la actitud, altitud y aceleraciones de los controles implementados en vuelo utilizaron Simulink® y Stateflow. A partir de esas herramientas generaron un modelo base iniciando con la construcción de un modelo algorítmico, luego una simulación numérica, un código de generación y finalizando con la simulación del modelo. El Stateflow se deriva de Simulink®, siendo esta una herramienta grafica basada en la teoría mecánica, con el block FlightManagement de Stateflow se implementa una lógica de vuelo de control, como resultado obtuvieron una trayectoria de la aeronave de acuerdo a las condiciones especificadas.

En el año 2013 el estudiante Siddharth Goyal, del Punjab Engineering College, como trabajo de grado realizó la publicación 'Study of a Longitudinal Autopilot for different Aircrafts', en el cual realizó la comparativa del desempeño del piloto automático de cuatro diferentes aeronaves, teniendo en cuenta que este es el mismo para estas, concluyendo que a pesar que el piloto automático es el mismo, el modelo de este, varía debido a las características y dinámica de las aeronaves son diferentes y eso afecta el modelo [7].

1.2. Descripción del problema

Durante el vuelo del cohete Ainkka V, que es un cohete balístico atmosférico de corto alcance (SRBM), no teledirigido, existe la posibilidad que en su trayectoria se presenten condiciones adversas a la misión, como vientos cruzados o ráfagas de viento causando que el

cohete se desvié de su trayectoria, por eso se quiere diseñar un sistema de control que mantenga la trayectoria recta del cohete.

Este sistema estará implementado en las aletas ubicadas en la parte inferior del cohete, las cuales podrán desviar el flujo de aire cuando se muevan, de esta manera aparecerá una fuerza de reacción que actúa sobre estas, la cual generará momentos aerodinámicos que modificarán la posición del cohete.

1.2.1. Formulación de la pregunta de investigación

Por lo anteriormente planteado se presenta la siguiente pregunta de investigación, la cual va enfocada al sistema de control activo del cohete ¿Cuál es el planteamiento algorítmico para la simulación de un sistema de control que pueda mantener una trayectoria para un cohete balístico atmosférico, no teledirigido?

1.3. Justificación

Este proyecto busca optimizar futuras misiones de cohetes, ya que gracias a este sistema de control las desviaciones que se puedan presentar debido a condiciones del ambiente, como ráfagas de viento se pueden evitar, haciendo que siga una trayectoria recta.

1.4. Objetivos de la investigación

1.4.1. Objetivo general

Desarrollar la simulación de un sistema de control PID que mantenga la trayectoria de un cohete balístico atmosférico, no teledirigido.

1.4.2. Objetivos específicos

1. Simular los parámetros de velocidad, carga paga, momentos aerodinámicos y cambios de altitud que afectan al sistema de control dinámico usando Open Rocket® o similar, Simulink® y/o Xplane.
2. Modelar un sistema de control activo que actúe sobre el cohete en su eje de cabeceo, desde la etapa de despegue hasta el apogeo usando Simulink®.
3. Generar un documento científico publicable referente al proyecto, utilizando las licencias de software aprobadas por la Universidad.

1.5. Alcances y limitaciones

1.5.1. Alcances

- ✓ Se diseñará una simulación de control activo para cohetes balísticos atmosféricos no teledirigidos, en toda su etapa de ascenso.
- ✓ La aplicación del diseño se hará con base en el eje de cabeceo de un cohete atmosférico no teledirigido.

- ✓ Se generará un modelo de control activo para superficies aerodinámicas utilizando herramientas computacionales como Matlab® y Simulink®.

1.5.2. Limitaciones

- Para el desarrollo de las ecuaciones de movimiento se asume el modelo de aproximacioon de periodo corto, asumiendo una velocidad vertical constante.
- No se construirá un cohete funcional (sistema propulsivo, sistema aerodinámico, sistema electrónico de despegue, adquisición de datos de vuelo o sistema de recuperación).
- Las herramientas computacionales utilizadas en este proyecto contarán con la licencia académica.
- No se construirán tarjetas ni se diseñaran circuitos electrónicos; se utilizarán plataformas electrónicas y software de acceso libre disponibles en el mercado para la implementación del proyecto (Arduino, Rasspberry pi).
- Se tendrá en cuenta solo un eje del cohete (pitch), en el proceso de simulación, la misión que se trabajará será la tercera etapa del cohete Ainkaa V construído en la Universidad de San Buenaventura.

2. MARCO DE REFERENCIA

2.1. Marco teórico y conceptual

Para el desarrollo de la simulación de un sistema de control que mantenga la trayectoria de un cohete balístico atmosférico no teledirigido, es fundamental tener claro el funcionamiento de los sistemas de control, además de conococimientos en programación y conceptos que intervienen en el proceso de una simulación.

2.1.1. Sistemas de control

Los sistemas de control son una reunión de dispositivos que unidos actúan para obtener un objetivo para determinados controles. El sistema se desarrolla bajo ciertos criterios que forman una estructura, inicialmente se tiene un objetivo de control, luego se forma un sistema de control donde existe una entrada, una perturbación y finalmente una salida.

Existen también dos clases de sistemas, está el sistema de control en lazo abierto y en lazo cerrado, para el primer caso la variable que se tiene en la salida no va a tener efectos en la variable de control, mientras que para el sistema de lazo cerrado, la señal de salida del sistema es la variable controlada que tiene acciones directas sobre la variable de control. [8]

2.1.2. Control PID (Proportional Integral Derivate)

El PID es un control con capacidad de retroalimentación de datos, su funcionamiento tiene en cuenta un error a partir de un valor de salida deseado menos el obtenido, este error ingresa de nuevo al control PID minimizándolo para la entrada del sistema. Para esto el sistema

utiliza tres parámetros, el proporcional, integral y un sistema derivativo, esto dependiendo de la modalidad que tenga el controlador.

La estructura del PID se puede explicar en la figura 2:

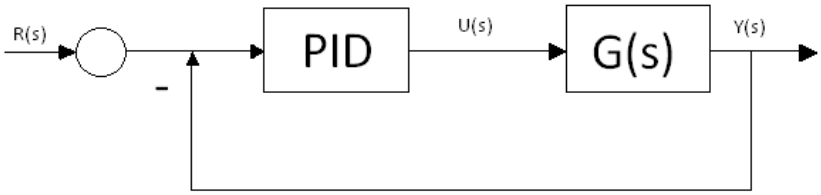


Figura 2. Estructura en bloques para un PID

En el caso del control proporcional, la acción del control es proporcional al error de control.

$$u(t) = K_e(t) + u_b \quad (1)$$

Donde la variable u_b es una señal de polarización.

Este controlador puede llegar a tener un desempeño limitado y algunas veces un error permanente.

La acción integral tiene como función asegurar que la salida del proceso concuerde con la referencia en su estado estacionario. Con esta acción, un error positivo producirá un incremento en la señal de control y un error negativo producirá una señal decreciente sin importar lo pequeño que sea el error.

Esta señal de control se define como:

$$u(t) = K_i \int_0^t e(\tau) d_\tau \quad (2)$$

Para esta ecuación $u(t)$ tiende a tener un valor diferente de cero cuando $e(\tau)$ que es la señal de error, es cero, y su función de transferencia resulta ser:

$$C_i(s) = \frac{K_i}{s} \quad (3)$$

Para la acción proporcional-derivativa la función se define como:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} \quad (4)$$

Donde existe la constante T_d, que es el tiempo derivativo, esto genera una respuesta rápida del control, pero solo resulta ser eficaz cuando existen periodos transitorios. Su función de transferencia es la siguiente:

$$C_{PD}(s) = K_p + sK_p T_d \quad (5)$$

Al combinar una acción de control derivativa con un controlador proporcional, resulta un controlador que responde a la velocidad del cambio que tenga el error y así mismo genera una corrección antes de que el error sea más grande.

Al combinar estas tres acciones proporcional, integral y derivativa se obtiene el controlador PID, donde su ecuación combinada resulta ser:

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(\tau) d\tau + K_p T_d \frac{de(t)}{dt} \quad (6)$$

Y la función de transferencia es:

$$C_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \quad (7)$$

Este mecanismo de control resulta ser el más utilizado a nivel industrial y posee las ventajas de las tres acciones. [9][10]

2.1.3. Señales

La señal es un fenómeno que está constituido por una función continua y discreta.

Una señal en tiempo continuo es aquella que está definida sobre un intervalo continuo de tiempo, la amplitud para este caso tiene un intervalo continuo de valores o valores distintos. También se presenta el proceso de cuantificación, el cual consiste en representar una variable por

medio de un conjunto de valores distintos, la resultante de estos valores se denominan valores cuantificados.

Una señal en tiempo discreto está definida en valores discretos de tiempo, esto quiere decir que la variable independiente t está cuantificada. Una señal digital es una señal en tiempo discreto con amplitud cuantificada.

En ingeniería de control, se le llama planta o proceso al objeto controlado, generalmente las plantas involucran señales en tiempo continuo, es por eso que si el sistema de control posee un controlador digital, se debe realizar la conversión de señal análoga a señal digital o si es el caso de señal digital a análoga. En los sistemas de control en tiempo discreto pueden existir una o más variables que pueden cambiar solo en valores discretos de tiempo. Estos instantes se denotan como kT, los cuales especifican los tiempos en los que se extraen los datos de la memoria de una computadora digital. Cuando se emplea un controlador digital, existe un proceso de muestreo, donde las señales en tiempo continuo reemplazan la señal por una secuencia de valores en puntos discretos de tiempo. Este proceso va acompañado por una cuantificación, en la cual la amplitud analógica muestreada se reemplaza por una amplitud digital [15].

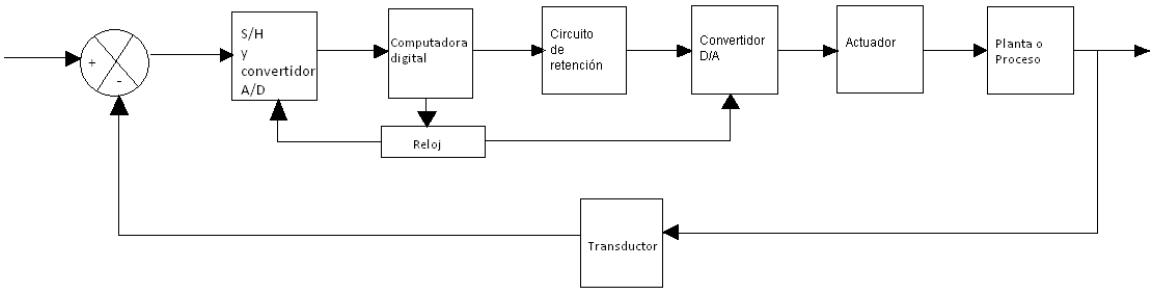


Figura 3. Bloques del sistema de control digital, que muestra las señales en forma binaria y gráfica.

En la figura 3 se muestra que en algunos puntos del sistema pasan señales de amplitud variable, en tiempo continuo o discreto, además de la forma numérica, la salida de la planta está dada en tiempo continuo, la señal de error se convierte a digital gracias al circuito de muestreo y retención y al convertidor analógico-digital. Luego la computadora digital procesa la secuencia de números a través de un algoritmo produciendo nuevas secuencias. El convertidor digital-análogo junto con el circuito de retención genera una secuencia de números en una señal continua por secciones y el reloj de la computadora se encarga de sincronizar los eventos, en la salida del circuito de retención se alimenta la planta, a través de un actuador para controlar su dinámica.

2.1.4. Convertidor Analógico – Digital (A/D)

Un conversor analógico-digital es un dispositivo que se encarga de convertir una señal analógica en un valor binario, este establece una relación entre su entrada que es una señal analógica y su salida que es una señal digital, dependiendo de su resolución, la cual determina la precisión con la que se reproduce la señal original [41].

2.1.5. Función de transferencia

Para trabajar con Matlab® es necesario saber utilizar las funciones de transferencia, las cuales relacionan el sistema de entrada y salida de la transformada de Laplace, junto con un sistema de ecuaciones diferenciales con condiciones iniciales en la entrada.

La función de transferencia es un modelo de caracterización de sistemas, esta se representa con un cociente, el cual relaciona la respuesta del sistema con una señal de entrada deseada. Para el caso del cohete, la respuesta del sistema está representada en el ángulo de cabeceo, y la señal de entrada en la deflexión de las aletas.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_1 s + a_0}{s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0} \quad (8)$$

Para el caso del cohete se maneja un modelo de 6 grados de libertad para implementar una respuesta de control con una función de transferencia estándar o un filtro, donde las constantes de entrada pueden ser seleccionadas por el diseñor para que coincidan con mayor precisión al actual control de respuesta del cohete. [8]

Estas funciones de transferencia son:

Para el alabeo

$$\frac{P_{stab}(s)}{P_{stabcmd}(s)} = \frac{1}{\tau s + 1} \quad (9)$$

Para el cabeceo

$$\frac{n_z(s)}{n_{zcmd}(s)} = \frac{\omega^2(\tau s + 1)}{s^2 + 2\zeta\omega s + \omega^2} \quad (10)$$

Para la guiñada

$$\frac{n_y(s)}{n_{ycmd}(s)} = \frac{\omega^2}{s^2+2\zeta\omega s+\omega^2} \quad (11)$$

Donde τ es la contante de tiempo, s es el operador de Laplace y ω es la frecuencia.

2.1.6. Fuerzas aerodinámicas en un cohete

Para poder determinar las fuerzas aerodinámicas que actúan en un cohete, se debe tener en cuenta la función del número de Mach, ya que este varia a lo largo del lanzamiento del cohete y hace que las características aerodinámicas varíen; además de esto se debe asumir el ángulo de incidencia del flujo, al momento de realizar una simulación en la que exista fuertes efectos en el viento. Para esto se hace referencia a los seis grados de libertad que tiene un cohete, así como sus componentes de movimiento.

En la figura 4 se hace referencia a los ejes como X, Y y Z, y a los movimientos del cohete como P, Q y R que son el movimiento de balanceo, de cabeceo y de guiñada, respectivamente.

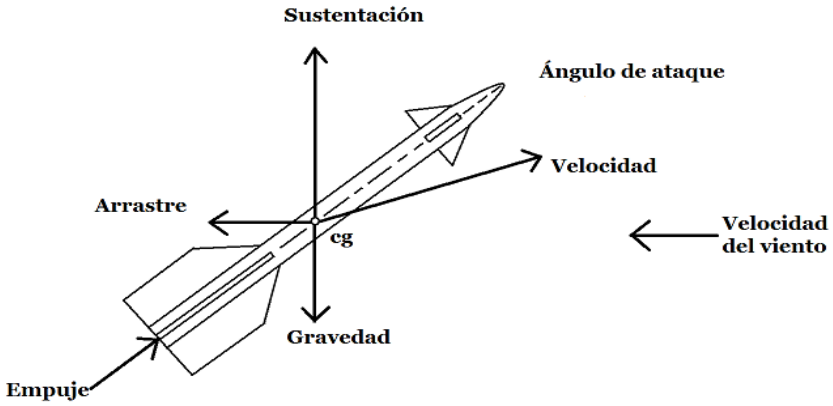


Figura 4. Fuerzas aerodinámicas que actúan sobre el cohete

2.1.7. Cohete tipo balístico

Es un modelo de cohete impulsado por un motor dotado de elementos que permiten el vuelo y su recuperación de forma segura. La trayectoria de este tipo de vehículos es una parábola, sin embargo la presencia de otras fuerzas, como la resistencia aerodinámica, la fuerza de sustentación, entre otras, hacen que la trayectoria sea diferente a una parábola. La propulsión de estos cohetes se desarrolla en la fase inicial del lanzamiento o fase caliente y en su fase de caída carecen de autopropulsión. [11]

2.1.8. Función de transferencia para un cohete tipo balístico

Al derivar la función de transferencia de un cohete tipo balístico, es necesario orientar el cohete en un sistema de ejes. La trayectoria se planea para mantener el cohete a un ángulo de ataque cero. Para tal fin se debe realizar una programación de la actitud de cabeceo y la velocidad de cabeceo, con el fin de mantener una trayectoria recta. Esto se asume mediante un perfil de velocidades, donde puede o no ocurrir variación en el flujo de combustible, esta condición y las ráfagas de viento hacen que no se mantenga en una trayectoria rectilínea.

Para el periodo de análisis se asume que el cohete balístico tiene una masa constante y es considerado como un cuerpo rígido, en la figura 5 se puede apreciar un plano de coordenadas X y Y, en donde el cohete se encuentra posicionado horizontalmente ubicándose su centro de gravedad Xcg y centro de presión Xcp que son tomados desde la punta de la ovija cónica, además de esto se representa el ángulo de deflexión δ_z que se presenta en las aletas y la resultante de la velocidad V_m .

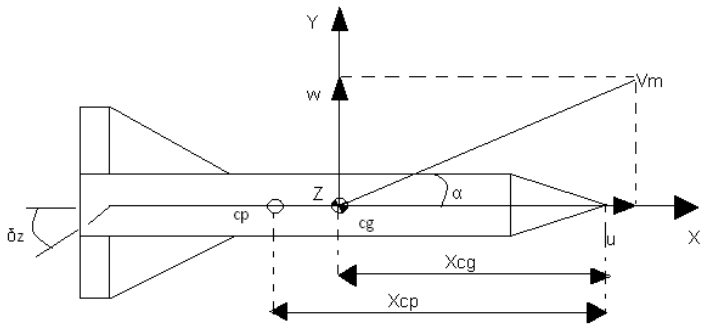


Figura 5. Representación del cohete en el plano de coordenadas

2.1.9. Diagrama de Root Locus

El diagrama de Root Locus es un método gráfico que proporciona información sobre la estabilidad de un sistema dinámico, el cual en un plano geométrico se ubican los polos de una función de transferencia en lazo abierto [12].

Con el software MATLAB® se puede obtener el diagrama de Root Locus, esto a través de los comandos “rlocus” y “sisotool”, este método determina la posición de los polos de la función de transferencia variando la ganancia K. Partiendo del centro del plano cartesiano, cuando la posición de los polos es sobre la parte izquierda se considera que el sistema es estable, por otro lado, si la posición de los polos es sobre la parte derecha se considera que el sistema es inestable.

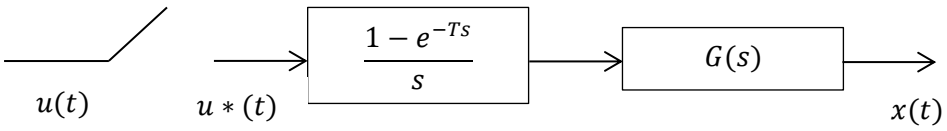
2.1.10. Discretización

El control de sistemas reales tiende a la utilización de computadoras digitales, ya que resultan ser mas eficientes y de bajo costo, se obtienen mejoras en la sensibilidad de las medidas, se pueden utilizar sensores, transductores y calculadoras digitales. Es por eso que si se tiene un sistema en modelo continuo y se desea pasar a un modelo discreto se utiliza el proceso de discretización y así obtener ciertas ventajas. Por medio de Matlab® se puede realizar este proceso

mediante el comando de [Nz,Dz] = c2dm (N,D,Ts,'metodo'), donde se toma en cuenta la función de transferencia del modelo continuo, la cual viene dada por los polinomios numerador y denominador (N y D), además de esto existe el tiempo de muestreo (Ts) y por último se especifica el carácter con el cual se ejecuta la discretizacion (método), dentro de estos métodos se tienen:

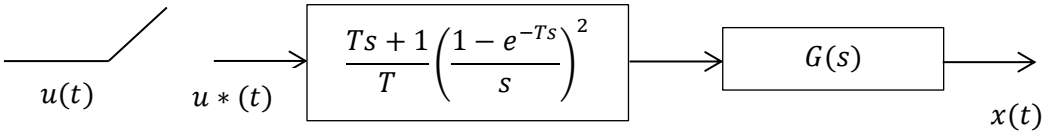
- 'zoh': Discretización utilizando retenedor de orden cero.

El retenedor de orden cero permite conservar la respuesta al escalón de su equivalente analógico.



Algunas de sus características es que conserva la ganancia estatica, preserva la respuesta al escalon y debido a su simplicidad, el retenedor de orden cero se emplea por lo regular en sistemas de control digital.

- 'foh': Discretización utilizando retenedor de primer orden.



En general los circuitos de retención de orden superior recontruirian una señal mas exacta que los retenedores de orden cero, pero desde el punto de vista de la estabilidad de los sistemas en lazo cerrado presentan un retraso en el periodo de muestreo y de este modo este retenedor no se emplea en aplicaciones de sistemas de control.

- ’tustin’: Discretización mediante aproximación trapezoidal.

Algunas características de este método es que los sistemas analógicos estables se conrtirán en equivalentes digitales estables, no requieres factorización de la función de transferencia y preserva la respuesta al impulso como la respuesta en frecuencia [38].

Con el comando dstep(Nz,Dz), se calcula la respuesta temporal de un sistema discreto a una secuencia de escalón, en donde se muestran múltiplos del periodo de muestreo.

La dicretización de los valores se utiliza para que la señal sea compatible con el formato de representación de los computadores.

En algunos casos la práctica de un convertidor análogo – digital no puede adquirir la señal de forma instantánea, lo cual lleva a que se produzca una diferencia entre la discretizacion ideal y la real [12].

2.1.11. Interfaz gráfica (GUIDE)

GUIDE es un entorno de programación disponible en Matlab®, que sirve para realizar y ejecutar programas con el ingreso de datos. Guide proporciona un conjunto de herramientas para crear la interfaz de usuario que se desee y presenta las siguientes opciones:

- Blank GUI: esta opción de interfaz gráfica de usuario blanco, permite generar un formulario nuevo, en donde se diseña el programa.
- GUI con Uicontroles: esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en los dos sistemas de unidades.
- GUI con ejes y menú: esta opción es otro ejemplo el cual contiene las opciones Open, Print y Close. En este formulario tiene un menú *Popup, un push button y un*

objeto axes, con la posibilidad de escoger alguna de las opciones que se encuentran en el menú.

3. METODOLOGÍA PROPUESTA

Para este proyecto se busca realizar una metodología capaz de generar resultados con la ayuda de ciertos procedimientos, es por eso que inicialmente se debe hacer una búsqueda bibliográfica acerca de documentos, artículos o tesis en los cuales se haya trabajado con cohetes de aletas móviles, o sistemas de control activo con lo cual se desarrollará un estado del arte que nos indicará lo que ya se ha hecho a este nivel.

A partir de esto, se generarán las variables dependientes e independientes las cuales servirán para plantear ecuaciones diferenciales lineales e invariantes en el tiempo, al cual se adaptarán las condiciones de operación de la tercera etapa de la misión Ainkka V. Al obtener el análisis dinámico del sistema, se evaluarán las condiciones de vuelo mediante Rocksim V9.1.3® usando la versión de prueba gratuita de 30 dias, donde a partir de ciertos datos se realizarán las estimaciones para la simulación, obteniendo la función de transferencia, con esta se llega a plantear un modelo por medio de Simulink®, el cual es una plataforma de Matlab® con licencia de la Universidad de San Buenaventura, y después de esto se escoge un método de sintonización. En la implementación del sistema de control se diseñará un sistema en lazo cerrado que incluirá el controlador PID y mediante ensayo y error se obtendrán las ganancias del controlador que serán las encargadas de estabilizar la señal del sistema.

En caso de que el sistema de control no llegue a estabilizar la señal, se debe replantear el modelo de control, si por el contrario el sistema de control llega a estabilizar la señal se realiza el

análisis de resultados el cual se hará por medio del diseño de una interfaz gráfica, la cual permite el ingreso de datos por parte de un usuario, para que se generen resultados según el tipo de análisis que se desee realizar, ya sea un análisis preliminar del comportamiento del cohete diseñado o el análisis dinámico y desarrollo de la función de transferencia generándose un sistema estable según lo requiera el usuario, sabiendo así con exactitud si el sistema propuesto anteriormente resulta ser eficaz.

Finiquitando el proceso anteriormente descrito, se redactará el documento final y el artículo para su publicación.

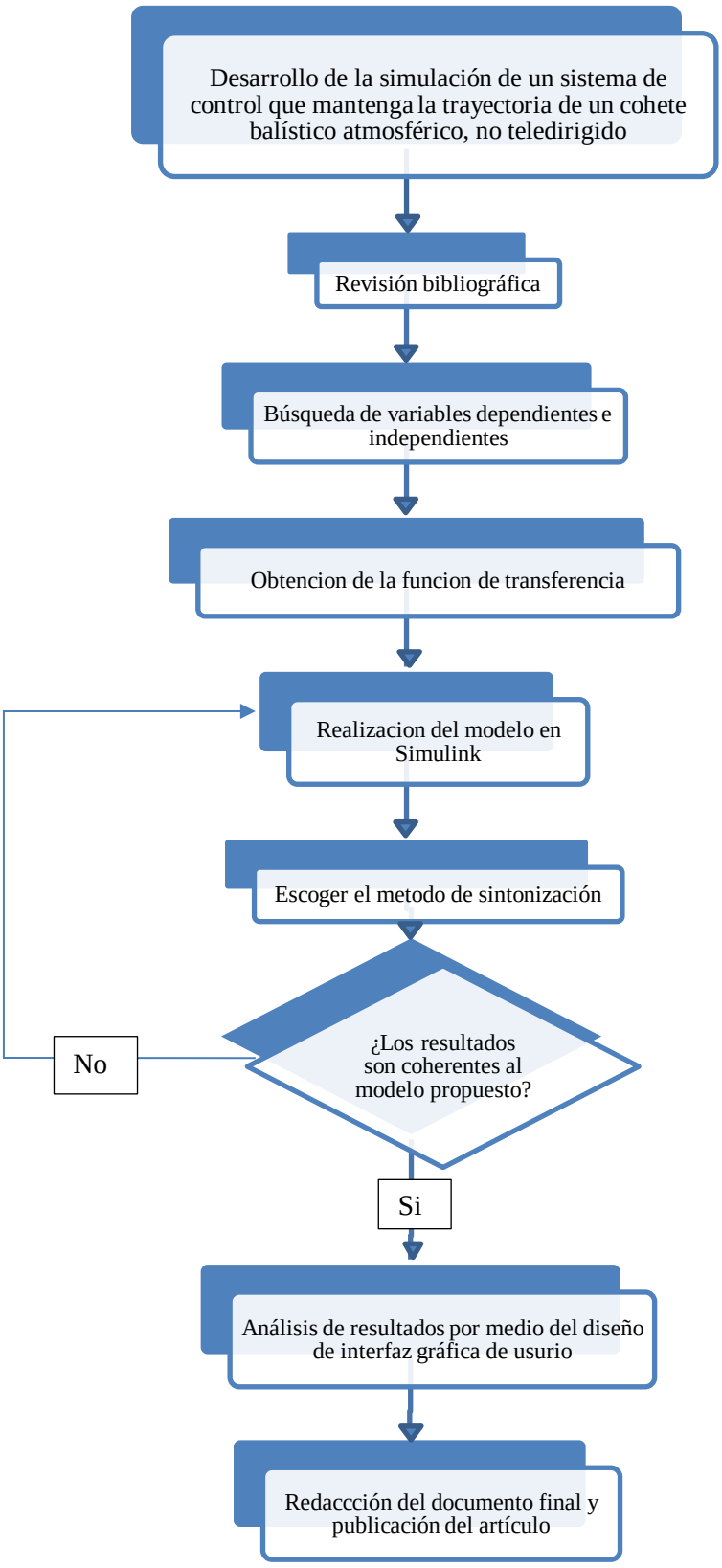


Figura 6. Diagrama de la metodología propuesta

3.1.Enfoque de la investigación

De acuerdo al problema de investigación, este pertenece a un enfoque empírico-analítico, ya que a partir de los datos obtenidos con el cohete Ainkka V desarrollado anteriormente, se busca analizar, comprender y controlar variables del sistema, las cuales presentan comportamientos frente a condiciones de vuelo vertical, esto por medio del diseño de un sistema de control que asegure una trayectoria recta del cohete, que será implementado en las aletas del mismo.

3.2. Línea de investigación

La línea de investigación a la que pertenece este proyecto, es la de vehículos aeroespaciales y el campo de investigación diseño de sistemas de control para vehículos tipo cohete.

4. DESARROLLO DE INGENIERÍA

4.1. Análisis dinámico del sistema

Existen cuatro fuerzas que actúan en un cohete durante una fase de vuelo las cuales se pueden apreciar en la figura 7: fuerza de sustentación, que actúa perpendicular a la dirección de movimiento, fuerza de arrastre, la cual es opuesta a la dirección de movimiento, estas dos fuerzas actúan en el centro de presiones, punto donde actúan las fuerzas aerodinámicas de un objeto, la fuerza gravitacional y la fuerza de empuje, la cual es generada por el motor, esta tiene que ser mayor a la fuerza gravitacional para que el cohete pueda elevarse.

En la figura 4 se pueden apreciar los tres eje de movimiento de un cohete, el eje longitudinal (X), el cual se ubica a lo largo de todo el cohete, el movimiento que realiza el cohete alrededor de este eje se denomina balanceo, el eje lateral (Y), es el eje que se extiende de punta a punta de las aletas, el movimiento que se realiza alrededor de este eje se denomina cabeceo, y el eje vertical (Z), el cual pasa por el centro de gravedad del cohete y es perpendicular al eje longitudinal y lateral, el movimiento que se realiza en este eje se denomina guiñada.

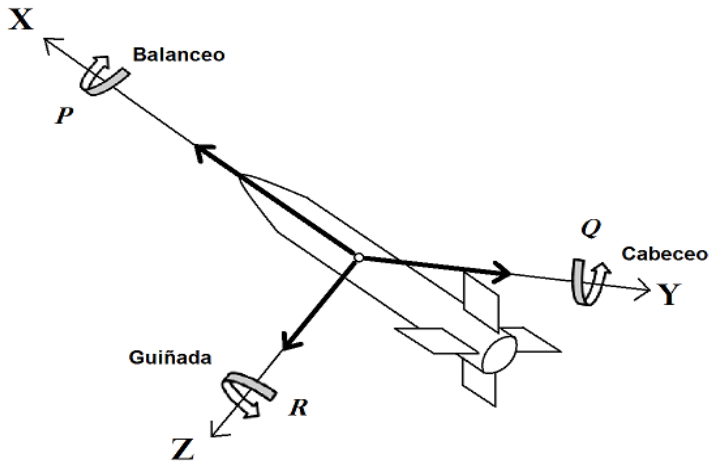


Figura 7. Grados de libertad del cohete

Para el periodo de análisis se asume que el cohete balístico tiene una masa constante y es considerado como un cuerpo rígido.

4.2. Desarrollo de las ecuaciones de movimiento

El movimiento del cohete se rige bajo la segunda ley de Newton, la cual declara que las suma de todas las fuerzas externas que actúan en un cuerpo son directamente proporcional al cambio de movimiento de este, ecuación (14) y la suma de los momentos externos que actúan en un cuerpo deben ser igual a la tasa de cambio de su momentum angular como se observa en la ecuación 15. Las tasas de cambio son tomadas respecto a un espacio inercial [12].

$$\sum F = \frac{d}{dt}(mV_T)\Big]_I \quad (14)$$

$$\sum M = \frac{dH}{dt}\Big]_I \quad (15)$$

Para el análisis se debe asumir lo siguiente:

La masa del cohete permanece constante durante el vuelo; este es un cuerpo rígido, y la tierra se toma como referencia inercial.

De acuerdo a la ecuación (14), se derivan las siguientes tres ecuaciones que corresponden a las ecuaciones del movimiento lineal, en los ejes X, Y y Z:

$$\sum \Delta F_x = m(\dot{U} + WQ - VR) \quad (16)$$
$$\sum \Delta F_Y = m(\dot{V} + UR - WP) \quad (17)$$
$$\sum \Delta F_z = m(\dot{W} + VP - UQ) \quad (18)$$

Donde V y W son las componentes de la velocidad lineal y P, Q y R son los componentes de la velocidad angular. Estos son tomados respecto al eje de referencia que es la tierra.

Las ecuaciones de movimiento angular se derivan de la ecuación (15) y son presentadas a continuación:

$$\sum \Delta \ell = \dot{P} I_x - \dot{R} J_{xz} + QR(I_z - I_y) - PQ J_{xz} \quad (19)$$
$$\sum \Delta \mathcal{M} = \dot{Q} I_y + PR(I_x - I_z) - (P^2 - R^2) J_{xz} \quad (20)$$
$$\sum \Delta \aleph = \dot{P} I_z - \dot{P} J_{xz} + PQ(I_y - I_x) + QR J_{xz} \quad (21)$$

De acuerdo a lo anterior, es necesario determinar seis ecuaciones de movimiento simultáneas para describir el comportamiento del cohete completamente. En algunos casos se pueden hacer ciertas suposiciones para dividir estas en dos grupos de tres ecuaciones simultáneas.

Para estos análisis se debe considerar el cohete en un vuelo recto y sin aceleración, para luego ser perturbado por la deflexión del elevador.

De acuerdo a la figura 8, la deflexión del elevador en el eje OY genera un momento de cabeceo en este mismo eje y a causa de este, se generan cambios en las fuerzas F_x y F_z. Esto no genera momento de rotación ni de guiñada, por lo tanto no hay cambios en F_y, por ende, P=R=V=0 y $\sum \Delta F_Y$, $\sum \Delta \ell$, y $\sum \Delta$א son descartados.

$$\sum \Delta F_x = m(\dot{U} + WQ) \qquad (22)$$
$$\sum \Delta F_z = m(\dot{W} - UQ) \qquad (23)$$
$$\sum \Delta \mathcal{M} = m(\dot{W} - UQ) \qquad (24)$$

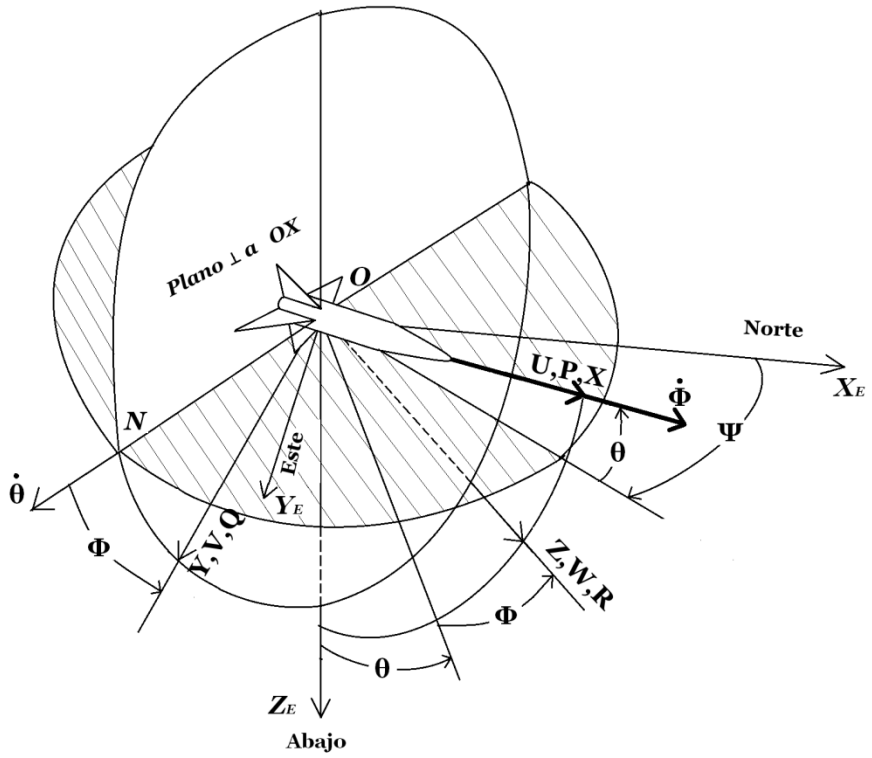


Figura 8. Diagrama de ejes para un cohete.

4.3. Ecuaciones de movimiento longitudinal

Es necesario expandir las ecuaciones de fuerzas y momentos aplicados, y expresarlos en los cambios de estas mismas a causa de perturbaciones. [12]

$$\left(\frac{mU}{sq}{}'_{\dot{u}} - C_{xu}{}'_{u}\right) + \left(-\frac{c}{2U}C_{x\dot{\alpha}}{}'_{\dot{\alpha}} - C_{x\alpha}{}'_{\alpha}\right) + \left[-\frac{c}{2U}C_{xq}\dot{\theta} - C_w(cos\Theta)\theta\right] = C_{Fxa} \quad (25)$$

$$-(C_{zu}{}'_{u}) + \left[\left(\frac{mU}{Sq} - \frac{c}{2U}C_{z\dot{\alpha}}\right)'_{\dot{\alpha}} - C_{z\alpha}{}'_{\alpha}\right] + \left[\left(-\frac{mU}{Sq} - \frac{c}{2U}C_{zq}\right)\dot{\theta} - C_w(sin\Theta)\theta\right] = C_{Fza} \quad (26)$$

$$(-C_{mu}{}'_{u}) + \left(-\frac{c}{2U}C_{m\dot{\alpha}}{}'_{\dot{\alpha}} - C_{m\alpha}{}'_{\alpha}\right) + \left(\frac{I_y}{Sqc}\dot{\theta} - \frac{c}{2U}C_{mq}\dot{\theta}\right) = C_{ma} \quad (27)$$

Estas ecuaciones asumen que:

Los ejes X y Z recaen en el plano de simetría y el origen del sistema de ejes está en el centro de gravedad del cohete.

La masa del cohete es constante.

El cohete se asume como cuerpo rígido.

La tierra es la referencia inercial.

Las perturbaciones en el equilibrio son pequeñas.

Para la solución de las ecuaciones de movimiento es necesario obtener inicialmente la solución de la ecuación homogénea, para tal caso $C_{ma} = C_{Fza} = C_{Fxa} = 0$. Tomando la transformada de Laplace de las ecuaciones (26, 25 y 27) y despreciando $C_{x\dot{\alpha}}$, C_{xq} y C_{mu}, se obtiene:

$$\left(\frac{mU}{Sq}s - C_{xu}\right)'_{u(s)} - C_{x\alpha}{}'_{\alpha(s)} - C_w(cos\Theta)\theta(s) = 0 \quad (28)$$

$$-C_{zu}{}'_{u(s)} + \left[\left(\frac{mU}{Sq} - \frac{c}{2U}C_{z\dot{\alpha}}\right)s - C_{z\alpha}\right]'_{\alpha(s)} + \left[\left(-\frac{mU}{Sq} - \frac{c}{2U}C_{zq}\right)s - C_w(sin\Theta)\right]\theta(s) = 0 \quad (29)$$

$$\left(-\frac{c}{2U}C_{m\dot{\alpha}}s - C_{m\alpha}\right)'_{\alpha(s)} + \left(\frac{I_y}{Sqc}s^2 - \frac{c}{2U}C_{mq}s\right)\theta(s) = 0 \quad (30)$$

4.4.Aproximación de periodo corto

La aproximación de periodo corto se toma cuando se asume una velocidad constante hacia adelante, es decir, ‘u=0; por lo tanto, no existen cambios en las fuerzas que se generan en esta dirección ya que no hay cambio de velocidad. Cabe aclarar que el sistema de control va a actuar en toda la etapa de ascenso, aun cuando existan cambios grandes de velocidad en la etapa de propulsión, pero en este caso este comportamiento se toma como una limitación.

Al eliminar la ecuación de movimiento en el eje X de la ecuación (30) queda de la siguiente forma:

$$\left(\frac{mU}{Sq}s - C_{z\alpha}\right)\alpha(s) + \left(-\frac{mU}{Sq}s - C_w sin\Theta\right)\theta(s) = C_{z\delta e}\delta_e \quad (31)$$

$$\left(-\frac{d}{2U}C_{m\dot{\alpha}}s - C_{m\alpha}\right)\alpha(s) + \left(\frac{I_y}{Sqd}s^2 - \frac{d}{2U}C_{mq}s\right)\theta(s) = C_{m\delta e}\delta_e \quad (32)$$

4.5. Caracterización del cohete

Como se mencionó anteriormente, se trabajó con el cohete Ainkaa V construido por la Universidad de San Buenaventura, sede Bogotá [13]. Para este análisis se tomó en cuenta la tercera etapa, la cual se muestra en la figura 9.

Figura 9. Modelo físico del Cohete etapa Ainkka V, tercera etapa

A continuación se muestran los parámetros que conforman el diseño del cohete:

Tabla 1. Dimensiones generales del cohete

Dimensiones generales		
Peso total cohete	5,56	Kg
Largo del fuselaje	0,87	m
Diámetro del fuselaje	0,082	m
Material del fuselaje	PVC	-
Largo de ojiva	0,22	m
Material de la ovija	Madera	-
Largo total	1,09	m

Envergadura	0,2	m

Tabla 2. Dimensiones de la aleta

Dimensiones de la aleta		
Cuerda de raíz	0,165	m
Cuerda de punta	0,09	m
Envergadura de aleta	0,06	m
Área	0,0255	m^2
Espesor	0,003	m
Material	Aluminio	AI I050

De igual forma, se tomaron en cuenta las condiciones del ambiente en donde se va a realizar el lanzamiento, ya que se pretende realizar en la base de Marandua, Vichada Colombia, donde la temperatura promedio es de 30°c. Resulta ser un espacio semidesértico y despoblado, perfecto para este tipo de actividades.

Para poder simular el vuelo del cohete fue necesario utilizar Rocksim V9.1.3®, el cual es un programa que permite diseñar cualquier tipo de cohete experimental, para luego simular su lanzamiento. Además de esto permite cumplir con los criterios de peso, velocidad y altura que se desee [14].

Para la primera fase en Rocksim V9.1.3® se realiza el diseño de cada una de las secciones que componen el cohete, se empieza por la nariz cónica ingresando valores como la longitud, el diámetro, el material con la que fue construida. Para la sección del cuerpo del

cohete se debe tener en cuenta la longitud y el diámetro del tubo, la longitud del bloque del motor donde se ubica el combustible y finalmente las dimensiones de las aletas y su localización en el cuerpo del cohete.

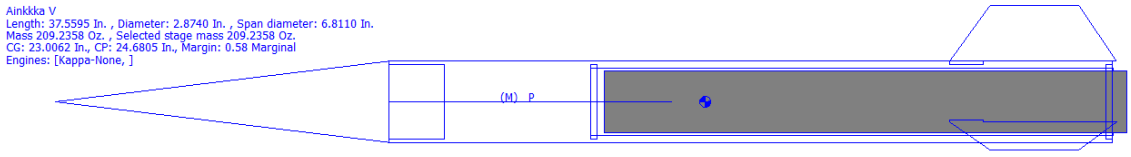


Figura 10. Diseño del cohete en Rocksim V9.1.3®

Para cargar el motor en la simulación es necesario crearlo en la biblioteca de motores, por medio de la interfaz de *EngEdit* [35], el cual a partir de la longitud, el diámetro, la masa inicial y la masa del propelente, genera el empuje promedio, el empuje máximo generado, el tiempo de quemado y el impulso específico.

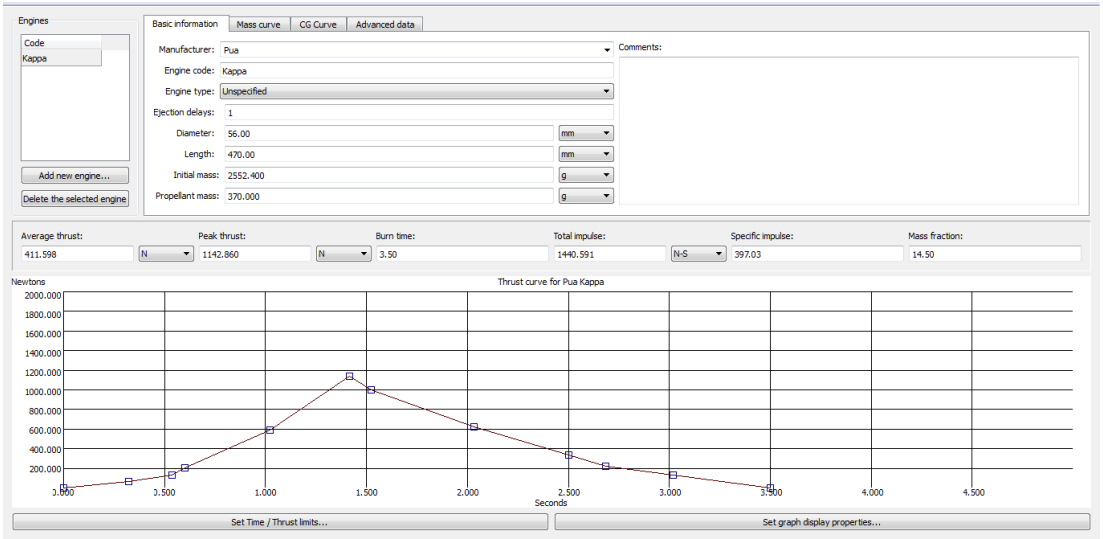


Figura 11. Carga del motor en EngEdit

Después de haber diseñado el cohete y cargado el motor, se declaran las condiciones en las cuales se va a realizar la simulación. Dentro de estas tenemos la temperatura, la densidad y la presión del ambiente, condiciones esenciales para determinar el punto geográfico de lanzamiento del cohete.

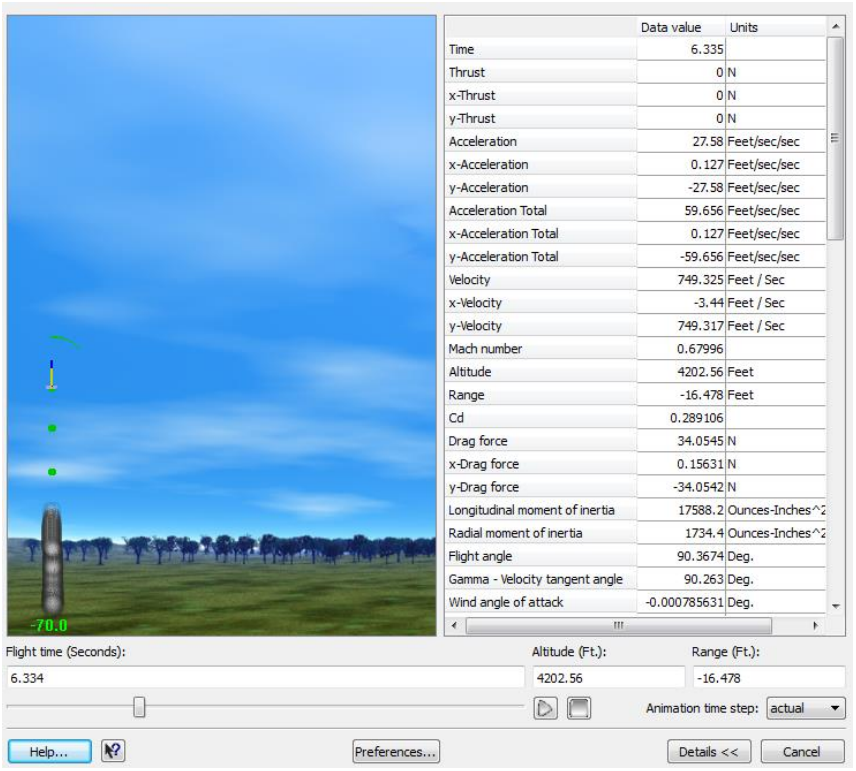
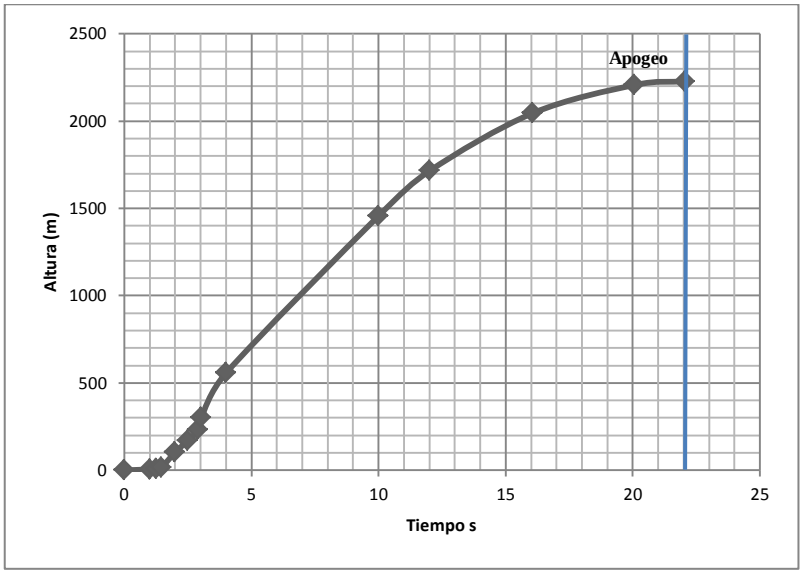


Figura 12. Simulación del vuelo del cohete en Rocksim V9.1.3®

Gracias a Rocksim V9.1.3® se encontraron datos como la velocidad, el empuje generado, la altura alcanzada, el tiempo de vuelo, el momento de inercia y los centros de gravedad, entre otros datos [21] (estos datos son la base para el análisis dinámico y el sistema de control). Además se obtienen gráficas donde se muestra el comportamiento del cohete desde el punto cero hasta el punto del apogeo.

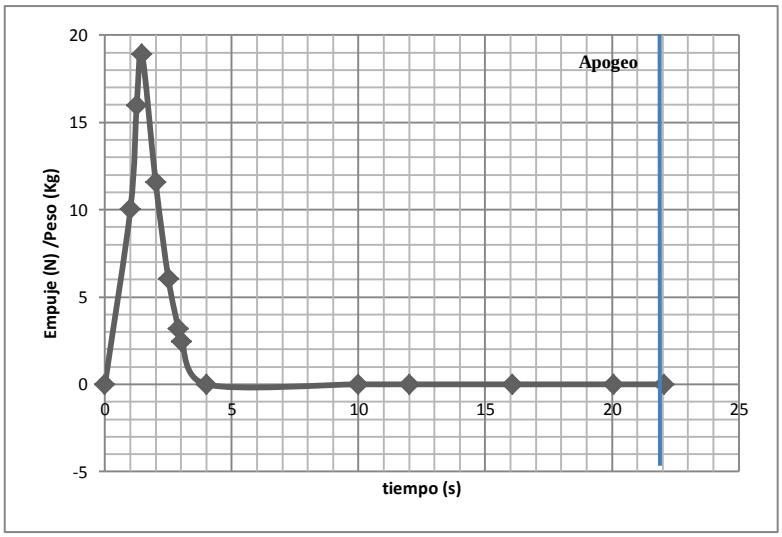
En la gráfica 1 se observa la altura que alcanza el cohete respecto al tiempo y se establece el punto de apogeo a una altura de 2227,92 m para un tiempo de 22,05 segundos.



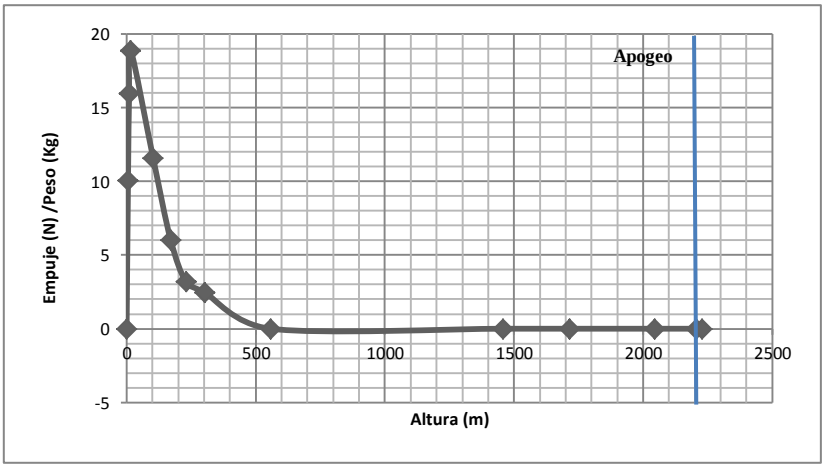
Gráfica 1. Altura VS Tiempo del cohete Ainkka V

El empuje es parte importante en la caracterización de la capacidad de elevación de un motor cohete. Para esto se utiliza el impulso total del motor cohete que reúne el tiempo o duración del empuje y se define como la integral del empuje sobre la duración operativa del motor [14].

En Rocksim V9.1.3®, también se puede realizar la estimación del empuje y el peso, teniendo en cuenta el tiempo y la altura, así como se observa en las gráficas 2 y 3.

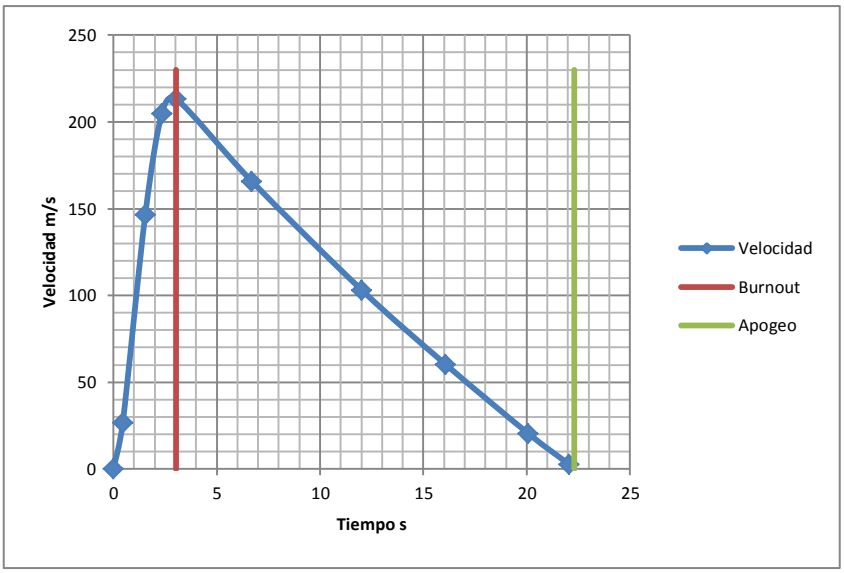


Gráfica 2. Empuje/peso VS tiempo



Gráfica 3. Empuje/peso VS altura

El empuje máximo que presenta el cohete Ainkaa V es en el tiempo de 1,46 s con un empuje de 1080,64 N; sin embargo el combustible tiene un tiempo de quemado rápido haciendo que en el segundo 3,03 se genere el empuje total y seguidamente el cohete actúe bajo la inercia que lleva consigo, la cual resulta ser elevada y al mismo tiempo alcanza así una mayor altura.



Gráfica 4. Velocidad VS Tiempo

Por medio de Rocksim V9.1.3® fue posible obtener la gráfica de velocidad vs tiempo, en la cual se puede apreciar el momento en el que se consume todo el combustible (Burnout) en un tiempo aproximado de 3,03 segundos. En este punto se observa la velocidad máxima alcanzada por el cohete: 213 m/s. Desde este punto el cohete empieza a desacelerar a una razón de aproximadamente -11,052m/s^2, ver ecuación 33.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-213}{22-3} = -11{,}052\ m/s^2 \qquad (33)$$

Así mismo en la gráfica 4 se observa una línea de apogeo la cual indica que a los 22 segundos llega a su altura máxima evidenciando una pérdida en la energía cinética, que se obtuvo por el empuje inicial del motor cohete.

A partir de los datos arrojados por Rocksim V9.1.3® se estiman algunas condiciones aerodinámicas, teniendo en cuenta los centros de gravedad y centros de presiones de cada elemento que conforma el cohete [18].

Primero se analizó la parte superior del cohete. Generalmente la fuerza normal en la nariz es similar para todas las formas, en este caso es una nariz cónica y el valor del coeficiente de la fuerza normal es de 2 [15].

$$(C_{N\alpha})_n = 2 \qquad (34)$$

Mientras que la ubicación del centro de presión tiende a variar de acuerdo a la geometría que tenga la nariz, para el caso de la nariz cónica se tiene la siguiente ecuación:

$$\overline{X_n} = \frac{2}{3}L \qquad (35)$$

Luego se analizaron las aletas, donde según las dimensiones, se obtiene el coeficiente de la fuerza que ejercen estas en el cohete, con la siguiente ecuación:

$$(\mathrm{C_{N\alpha}})\mathrm{f} = \frac{4\mathrm{n}\left(\frac{\mathrm{al}}{\mathrm{d}}\right)\text{^}2}{1+\sqrt{1+\left(\frac{2\mathrm{l}}{C_r+C_t}\right)\text{^}2}} \qquad (36)$$

$$(\mathrm{C_{N\alpha}})\mathrm{f} = \frac{4 * 4\left(\frac{6\mathrm{cm}}{8{,}2\mathrm{cm}}\right)\text{^}2}{1+\sqrt{1+\left(\frac{2 * 6\mathrm{cm}}{16{,}5\mathrm{cm}+9\mathrm{cm}}\right)\text{^}2}} = 4{,}0691$$

Donde n es el número de aletas, que en este caso son 4.

Además de esto, se debe tener en cuenta que el flujo de aire sobre las aletas está influenciado por el flujo de aire sobre la sección del cuerpo a la que las aletas están unidas. Es por esto que el coeficiente de la fuerza para cada aleta es multiplicada por un factor de interferencia.

$$\mathrm{K_{fb}} = 1 + \frac{\mathrm{r}}{\mathrm{al}+\mathrm{r}} \qquad (37)$$

$$\mathrm{K_{fb}} = 1 + \frac{4{,}1\mathrm{cm}}{6\mathrm{cm}+4{,}1\mathrm{cm}} = 1{,}4059$$

Teniendo en cuenta el factor de interferencia el coeficiente de la fuerza que generan las aletas en presencia del cuerpo del cohete está dado por la siguiente ecuación:

$$(C_{N\alpha})fb = K_{fb}(C_{N\alpha})f \qquad (38)$$

$$(C_{N\alpha})fb = 1{,}4059 * 4{,}0691 = 5{,}7207$$

La ubicación del centro de presión para las aletas se toma en cuenta desde la raíz de la nariz hasta el valor $\bar{X}_f$.

$$\bar{X}_f = X_f + \Delta X_f \qquad (38)$$

$$\bar{X}_f = 92{,}5\text{cm} + 5{,}3083 = 97{,}8083\text{cm}$$

Este punto se puede ver claramente en la figura 13, la cual muestra el modelo de la aleta, el cual permite obtener las características de cada elemento del cohete y especifica el centro de gravedad, que es donde se intersecan la línea roja y verde.

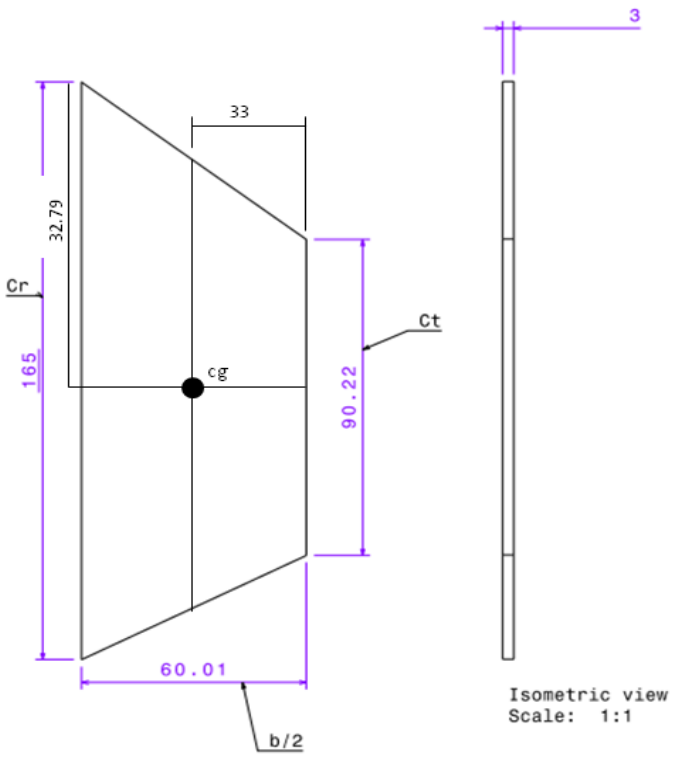


Figura 13. Centro de gravedad y ubicación del mecanismo de control

Para perfiles simétricos como en este caso en vuelo subsónico, el centro aerodinámico se localiza aproximadamente al 25% de la cuerda desde el borde de ataque del perfil, así mismo el momento generado en este punto es cero [36], por esta razón se debe implementar el mecanismo de control en esta zona, el cual va a ser capaz de mover la aleta, en la dirección pertinente para que se pueda mantener en una trayectoria recta.

Después de hallar el coeficiente de la fuerza normal que actúa en la nariz cónica y en las aletas se halla el coeficiente de la fuerza normal total.

$$C_{N\alpha} = (C_{N\alpha})n + (C_{N\alpha})fb \qquad (40)$$

$$C_{N\alpha} = 2 + 5{,}7207 = 7{,}7207$$

Y el centro de presión del cohete medido desde la punta de la ovija.

$$\overline{X} = \frac{((C_{N\alpha})n * \overline{X_n} + ((C_{N\alpha})fb * \overline{X_f}}{C_{N\alpha}} \quad (41)$$

$$\overline{X} = \frac{2(14{,}666) + 5{,}7207(97{,}8083)}{7{,}7207} = 76{,}2707\text{cm}$$

Donde se relaciona la fuerza normal total $C_{N\alpha}$, con el centro de presión de la nariz $\overline{X_n}$, la fuerza total de las aletas $(C_{N\alpha})fb$ y el centro de presión de las mismas $\overline{X_f}$.

Ahora se analizó la posición del centro de gravedad del cohete medido desde la punta de la ojiva, ya que con este valor se determina si el vuelo es seguro y estable.

$$X_{CG} = \overline{X} - d \quad (42)$$

$$X_{CG} = 76{,}2707\text{cm} - 8{,}2\text{cm} = 68{,}0707\text{cm}$$

Según los resultados obtenidos la distancia entre el centro de gravedad al centro de presión es de 8,2 cm. Según los datos encontrados, se determina que el cohete es estáticamente estable, ya que el centro de gravedad se encuentra delante del punto del centro de presión en un rango mayor o igual al diámetro del fuselaje [37].

4.6. Estimaciones para la simulación

De acuerdo a los resultados arrojados por Rocksim V9.1.3® en la simulación, se toman en cuenta los valores que se relacionan como datos de entrada que debe tener el sistema, que en este caso son:

$$q = 29356{,}013 Kg/m\text{^}2$$

$$I_y = 0{,}287 \text{Kg m}^2$$

$$U = 213\frac{m}{s}$$

$$T = 1180\text{N}$$

Donde q es la presión dinámica, I_y es el momento de inercia, U es la velocidad máxima alcanzada y T es el empuje del combustible sólido del cohete.

Dentro de las estimaciones para encontrar la función de transferencia; se encuentra las derivadas del cabeceo, la amortiguación, el valor de la pendiente de la fuerza normal, el coeficiente de momento de cabeceo y el coeficiente de deflexión del elevador o aleta.

Derivadas de guiñada para el ala C_{Lq}

$$C_{Lq} = \left(\frac{1}{2} + 2\frac{\bar{x}}{\bar{c}}\right) C_{L\alpha} \qquad (43)$$

Donde $\frac{\bar{x}}{\bar{c}}$ es la distancia entre el centro de gravedad y el centro aerodinámico y se expresa de la siguiente manera:

$$\frac{\bar{x}}{\bar{c}} = \frac{x_{a.c.}}{\bar{c}} - \frac{x_{c.g}}{\bar{c}} \qquad (44)$$

$$\frac{\bar{x}}{\bar{c}} = 0{,}324 - 0{,}045 = 0{,}278$$

$$C_{L\alpha} = 8\tan^{-1}\frac{\pi A}{16+\frac{\pi A}{1+2\lambda\tan\Lambda_{LE}}} \qquad (45)$$

$$C_{L\alpha} = 8 * \tan^{-1}\left(\frac{\pi * 1{,}568}{16 + \left(\frac{\pi * 1{,}568}{1 + 2 * 0{,}545 * \tan 0{,}749}\right)}\right) = 2{,}060$$

Para hallar la pendiente de la curva de sustentación de la aleta se aplica la siguiente ecuación:

$$\frac{X_{a.c}}{c_r} = \frac{\frac{2}{3}(1-\lambda)+\frac{1}{2}\left(1-\frac{\lambda^2}{1+\lambda}\right)\pi\log\left(1+\frac{A}{5}\right)}{1+\pi\log\left(1+\frac{A}{5}\right)} \qquad (46)$$

$$\frac{X_{a.c}}{c_r} = \frac{\frac{2}{3}*(1-0{,}545)+\frac{1}{2}*\left(1-\frac{(0{,}545)^2}{1+0{,}545}\right)*\pi\log\left(1+\frac{1{,}568}{5}\right)}{1+\pi\log\left(1+\frac{1{,}568}{5}\right)} = 0{,}324$$

$$C_{m\alpha} = \left(n*\frac{X_{a.c}}{Cr}\right)*\left(\frac{Cr}{\bar{c}}\right)*(C_{L\alpha}) \qquad (47)$$

$$C_{m\alpha} = (0{,}5-0{,}324)*(1{,}257)*(2{,}060) = 0{,}456$$

Derivadas de guiñada para el ala C_{mq}

$$(C_{mq})_{M\approx 0.2} = -0.7C_{l\alpha}\cos\Lambda_{c/4}\left\{\frac{A\left[\frac{1}{2}\frac{\bar{x}}{\bar{c}}+2\left(\frac{\bar{x}}{\bar{c}}\right)^2\right]}{A+2\cos\Lambda_{c/4}}+\frac{1}{24}\left(\frac{A^3\tan^2\Lambda_{\frac{c}{4}}}{A+6\cos\Lambda_{\frac{c}{4}}}\right)+\frac{1}{8}\right\} \qquad (48)$$

$$(C_{mq})_{M\approx 0.2} = -0.7*2{,}060$$

$$*0{,}999\left(\frac{1{,}568\left(\frac{1}{2}*0{,}278+2*(0{,}278^2)\right)}{1{,}568+2*0{,}999}+\frac{1}{24\left(\frac{1{,}568^3*0{,}001}{1{,}568+6*0{,}999}\right)}+\frac{1}{8}\right) = -0{,}061$$

Cuando el número de Mach resulta ser mayor a 0.2, se debe aplicar la ecuación 49, teniendo en cuenta que para este caso el Mach resulta ser de 0.6.

$$\left(C_{mq}\right)_{M>0.2} = \left[\frac{\frac{A^3 \tan^2 \Lambda_{\frac{c}{4}}}{AB+6\cos\Lambda_{\frac{c}{4}}}+\frac{3}{B}}{\frac{A^3 \tan^2 \Lambda_{\frac{c}{4}}}{A+6\cos\Lambda_{\frac{c}{4}}}+3}\right]\left(C_{mq}\right)_{M\approx 0.2} \quad (49)$$

$$B = \sqrt{1 - M^2 \cos^2 \Lambda_{c/4}} \quad (50)$$

$$B = \sqrt{1-(0{,}6)^2 * 0{,}998} = 0{,}800$$

$$\left(C_{mq}\right)_{M>0.2} = \left(\frac{\frac{1{,}568^3 * 0{,}001}{(1{,}568 * 0{,}800)+6*0{,}999}+\frac{3}{0{,}800}}{\frac{1{,}568^3 * 0{,}001}{1{,}568+6*0{,}999}+3}\right) = -0{,}077$$

$$C_{m\delta} = -\frac{Tl}{Sqd} \quad (51)$$

$$C_{m\delta} = -\frac{1180 * 0{,}349}{0{,}025 * 29356{,}013 * 0{,}082} = -6{,}72$$

$$C_w = \frac{mg}{Sq} \quad (52)$$

$$C_w = \frac{5{,}569\ \text{Kg} * 9{,}8\ \text{m/s}^2}{0{,}025\ \text{m} * 29356{,}013\text{Kg/m\^2}} = 0{,}076$$

$$C_{z\delta e} = \left(\frac{\bar{c}}{l_t}\right) C_{m\delta e} \quad (53)$$

$$C_{z\delta e} = \left(\frac{0{,}131}{0{,}34}\right)(-0.71) = -0{,}266$$

$$C_{z\delta} = -\frac{T}{Sq} \quad (54)$$

$$C_{z\delta} = -\frac{1180}{0{,}025 * 29356{,}013} = -1{,}36e9$$

Teniendo en cuenta los coeficientes hallados anteriormente, se procede a sustituir los valores en la ecuación 30, obteniendo lo siguiente:

$$\left(1{,}586279\ s - (-4)\right)\alpha(s) + (-1{,}58628\ s - 0{,}0766628)\theta(s) = -0{,}26625\ \delta e(s)$$

$$(-0{,}45602)\alpha(s) + (0{,}004677\ s^2 - (-1{,}4893x10^{-5}\ s)\theta(s) = -0{,}71\ \delta e(s)$$

Para dar solución a estas ecuaciones, se formula la función de transferencia como una división, de modo que el numerador es la determinante de la ecuación no homogénea, ubicando los coeficientes en la columna de la variable a controlar (cabeceo en este caso);

$$\begin{vmatrix} -1{,}58627941\ s & 4 & -0{,}26625298 \\ -0{,}45602432 & 0 & -0{,}71 \end{vmatrix}$$

Solucionando la determinante se obtuvieron los siguientes resultados: 1,126 s y -2,961.

El denominador es el determinante de la ecuación homogénea:

$$\begin{vmatrix} 1{,}58627941\ s & 4 & -1{,}58627941\ s & -0{,}076662821 \\ -0{,}45602432 & 0 & 0{,}00467716\ s^2 & 1{,}48932E\text{-}05 \end{vmatrix}$$

Obteniendo los siguientes resultados: $0{,}007\ s^3 - 0{,}018\ s^2 - 0{,}723$ s y -0,034.

Quedando la función de transferencia de la siguiente manera:

$$\frac{\theta(s)}{\delta(s)} = \frac{1{,}1262\,s - 2{,}7185}{0{,}0074\,s^3 + 0{,}0186\,s^2 + 0{,}7233\,s - 0{,}0349}$$

Donde la salida del sistema $\theta(s)$ es el ángulo de salida y la entrada $\delta(s)$ se refiere a la deflexión de las aletas.

Para el desarrollo del sistema de control es necesario saber inicialmente si la función de transferencia de la planta es inherentemente estable o no. Esto se hace mediante el diagrama de polos y ceros de "Root locus", en donde el plano izquierdo y derecho indican si es o no estable. En el plano izquierdo se deben ubicar los puntos que generan una señal estable, con el fin de que la respuesta al estar más lejos del cero sea más rápida.

Figura 14. Diagrama de bloques en lazo abierto en Simulink

Al realizar el análisis de la función de transferencia del modelo sin el controlador, se obtiene una respuesta a partir de una señal, la cual se muestra a continuación:

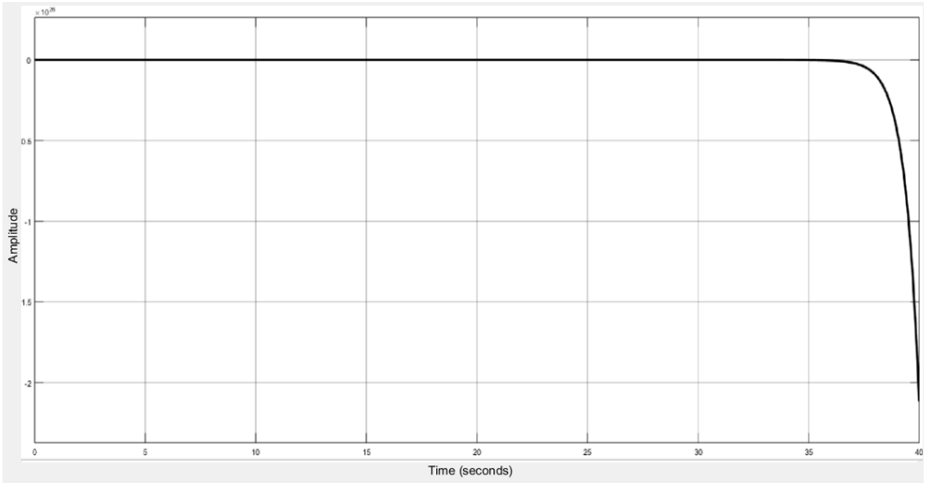


Figura 15. Respuesta al sistema en lazo abierto

En esta señal se observa un sistema que no es capaz de estabilizarse, ya que tiende a una caída y por lo tanto nunca se llega al valor de entrada (*step)* ingresado.

A continuación se encuentra el diagrama de Root locus del sistema en lazo abierto de la función de transferencia que fue encontrada anteriormente, donde se relaciona la entrada del sistema junto con la planta y finalmente la respuesta de la señal.

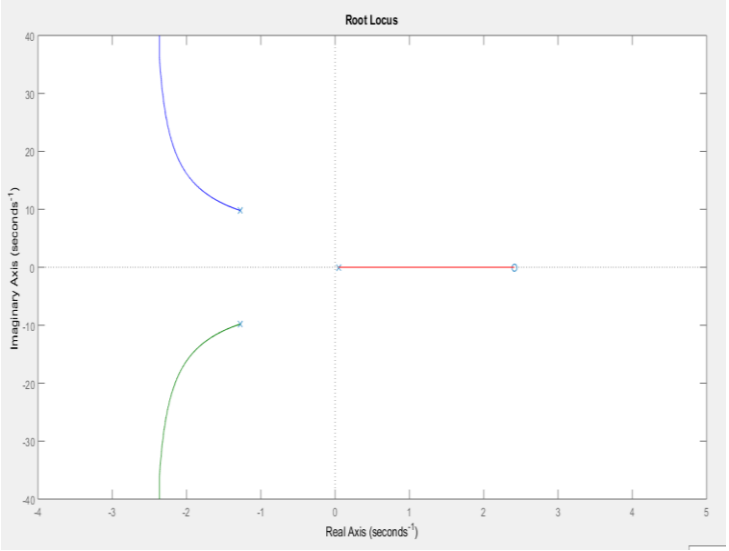


Figura 16. Diagrama de Root Locus, del sistema en lazo abierto

En el diagrama Root locus (polos y ceros) de la figura 16 se observa que el sistema (sin controlador) resulta ser inestable, con valores en los polos de $S^3 = -1.264$, $S^2 = -1.264$, $S = 0.083$ y cero de $S = 2.415$, indicando que deben hacerse modificaciones en el diagrama de bloques, esto de tal modo que los polos y ceros se encuentren en el plano izquierdo, y así el sistema sea estable.

4.7. Sistema de control

En la adaptación de un controlador que sea capaz de estabilizar el sistema con la planta, se realiza el ajuste de un controlador PID, el principio básico del esquema del control PID es que actúa sobre la variable a ser manipulada por medio de la combinación de tres acciones de control, la primera es la acción de control proporcional donde la acción de control es proporcional a la señal de error actuante, la cual es la diferencia entre la entrada y la señal de realimentación; la segunda es la acción de control integral donde la acción de control es proporcional a la integral de la señal de error actuante y la tercera es la acción de control derivativa donde la acción de control es proporcional a la derivada de la señal del error actuante.

La acción del control PID en controladores analógicos está dado por:

$$\mathrm{m(t)} = \mathrm{K}\left[\mathrm{e(t)} + \frac{1}{\mathrm{T_i}}\int_0^{\mathrm{t}} \mathrm{e(t)dt} + \mathrm{T_d}\frac{\mathrm{de(t)}}{\mathrm{dt}}\right] \quad (54)$$

Donde $\mathrm{e(t)}$ es la entrada al controlador, $\mathrm{m(t)}$ es la salida del controlador, K es la ganancia proporcional, $\mathrm{T_i}$ es el tiempo integral y $\mathrm{T_d}$ es el tiempo derivativo [15].

De acuerdo a lo anterior se genera un lazo de retroalimentación, en donde gracias a la herramienta Tune de *Simulink*®, se obtienen las constantes Kp, Kd y Ki automáticamente, las

cuales conforman la función de transferencia del PID [21]. La herramienta Tune implementa la siguiente ecuación:

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Por medio del método *Robust response time* el cual varia los valores de las constantes a partir del tiempo de respuesta y comportamiento de la señal que el usuario desee [42].

Además del método realizado en el software de Matlab, se pueden generar lso valores de las constantes del PID por medio del método de Ziegler – Nichols, en el cual se determina la ganancia última K_{cu} y el periodo ultimo P_u. Los parámetros del PID asociados a esta ganacia y periodo ultimo se muestra en la siguiente tabla:

CONTROLADOR	**Kc**	$\boldsymbol{Ti}$	$\boldsymbol{Td}$
P	0.5Kcu	-	-
PI	0.45Kcu	Pu/1.2	-
PID	0.6Kcu	Pu/2	Pu/8

Tabla 3. Método de Ziguer Nichols

Con las constantes del PID, la función de transferencia del actuador y el sensor ideal [39] se obtiene el diagrama de bloque en lazo cerrado con la función de transferencia de la planta, así como se muestra a continuación:

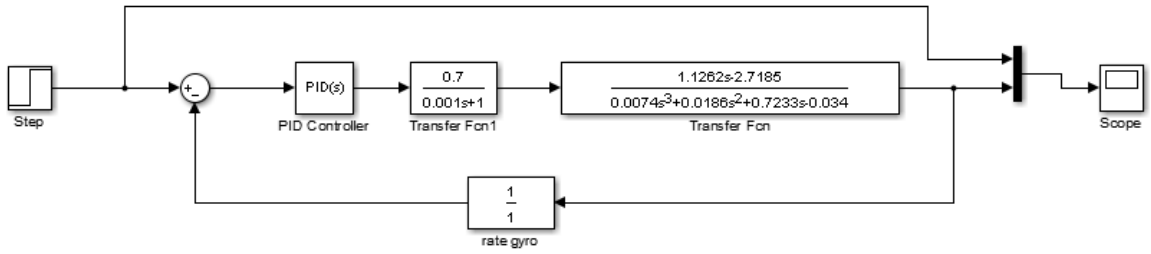


Figura 17. Diagrama de bloque en lazo cerrado con el PID en Simulink

La respuesta al sistema en lazo cerrado se obtiene mediante el *scope*, generando la siguiente señal estabilizada.

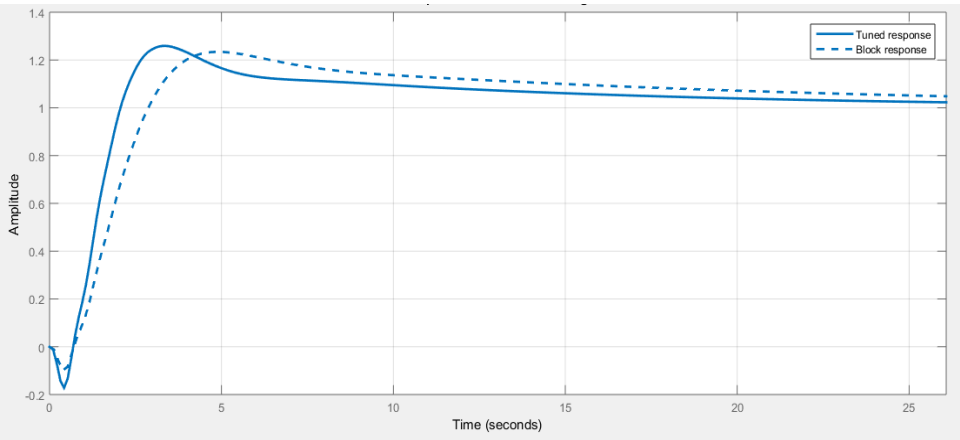


Figura 18. Respuesta del sistema de control

En la figura 18 se observa que el tiempo de respuesta de la señal es de 12s, el tiempo de subida o también llamado *Rise time* es de 1,6s con un sobreimpulso de 18%, un pico de amplitud de 1.27, un margen de ganancia de 8.36dB@2.36rad/s y un margen de fase de 46.3deg@0.803rad/s.

Donde las ganancias para el controlador se presentan como: proporcional $K_P = -0.2032$, integral $K_i = -0.0159$ y derivativa $K_d = 0.0693$.

Con los datos anteriores se puede encontrar un factor de amortiguamiento:

$$\zeta = \frac{-\ln\left(\frac{os}{100}\right)}{\sqrt{\pi^2+\ln\left(\frac{os}{100}\right)}} \quad (55)$$

$$\zeta = \frac{-\ln\left(\frac{18}{100}\right)}{\sqrt{\pi^2+\ln\left(\frac{18}{100}\right)}} = 0.4791$$

Y la frecuencia natural [40]:

$$w_n = \frac{4}{\zeta * ts} \quad (56)$$

$$w_n = \frac{4}{0.4791 * 12} = 0.6957\, rad/s$$

4.8.Discretización del sistema

Actualmente los sistemas de control se manejan en su gran mayoría bajo control digital, ya que estos resultan ser adaptables a microcontroladores y tarjetas de datos, entre otros, siendo más exactos en la respuesta del sistema, teniendo en cuenta que manejan algoritmos complejos, que se ajustan y modifican según las especificaciones de entrada. Es por eso que al ver la necesidad de implementar un sistema digital, para luego ser unificado con un microcontrolador, se debe pasar el sistema que se tiene inicialmente continuo a un sistema discreto, esto por medio de una discretización.

El proceso de discretización es convertir un modelo continuo a un modelo discreto por medio de Matlab®. Esta operación se realiza mediante el comando de [Nz,Dz] =c2dm

(N,D,Ts,'metodo'), donde se toma en cuenta la función de transferencia del modelo continuo, la cual viene dada por los polinomios numerador y denominador (N y D). Además de esto existe el tiempo de muestreo (Ts) y por último se especifica el carácter con el cual se ejecuta la discretización (método). Dentro de estos métodos se encuentra la discretización utilizando un retenedor de orden cero (ZOH) ya que por su simplicidad es el mas utilizado en sistemas de control [19]. Con el comando dstep(Nz,Dz), se calcula la respuesta temporal de un sistema discreto a una secuencia de escalón, en donde se muestran múltiplos del periodo de muestreo.

Para el proceso de discretización se genera un código en *Matlab*®, el cual se muestra a continuación:

```
num=[0.078 -0.417 0.534 0.043]
den=[0.007 0.096 0.306 0.499 0.043]
planta=tf(num,den)
plantad=c2d(planta,1,'zoh')
Go=feedback(pidcontinuo*planta,1)
God=feedback(piddiscreto*plantad,1)
step(Go)
axis([0 70 0 1])
hold on
figure
step(God)
```

Este código arroja una función de transferencia en tiempo discreto de la planta y se vuelve a generar un diagrama de bloque con su respectivo PID.

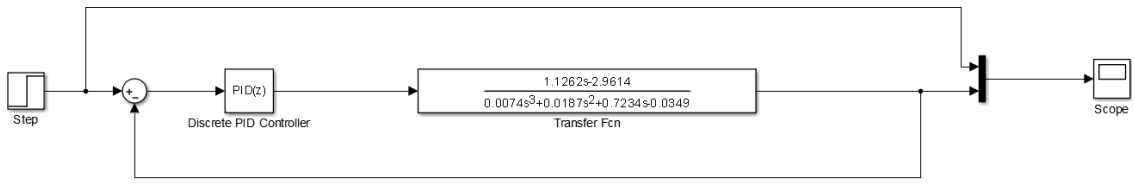


Figura 19. Diagrama de bloque del sistema en tiempo discreto

Según la respuesta obtenida en el *scope* del lazo cerrado, se genera la gráfica de la figura 19, la cual muestra la señal de respuesta por medio de pulsos (figura 20).

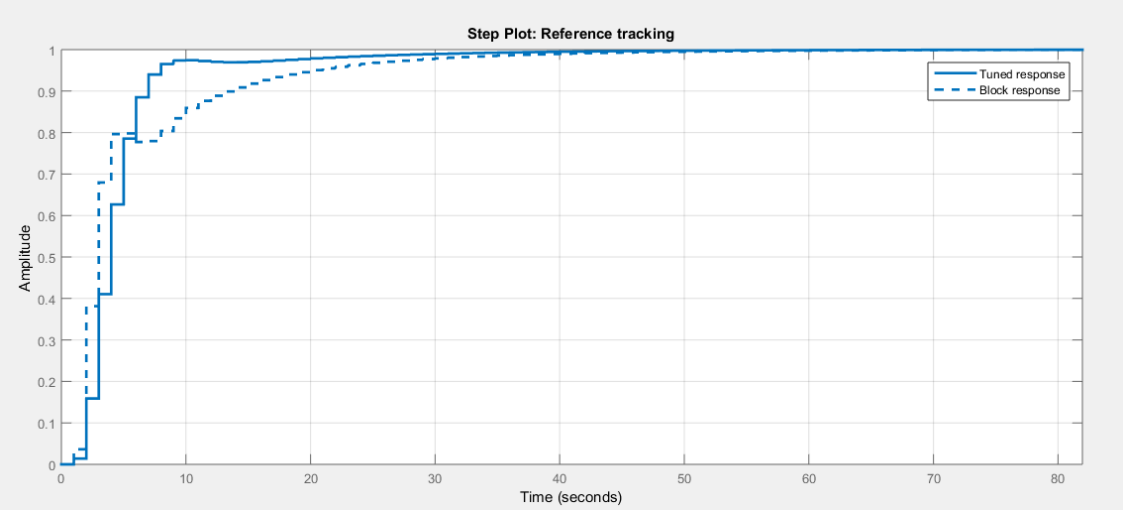


Figura 20. Respuesta del sistema en lenguaje digital

En la figura 20 se encuentra representada la señal de respuesta del sistema en un lenguaje digital, lo que quiere decir que los valores son discretos, este utiliza numeración binaria donde las cantidades se representan utilizando los números 0 o 1, a lo que se llama bit. Al mismo tiempo se obtienen las ganancias para el controlador PID [13] que son:

Donde las ganancias para el controlador se presentan como: proporcional $K_P = -0.1329$, integral $K_i = -0.00618$ y derivativa $K_d = 0.0571$.

Obteniendo un valor pico de 1.18, un margen de ganancia de 7.23dB@1.95rsd/s y margen de fase de 64.5deg@0.5rad/s.

Según los datos anteriores se encontró una frecuencia de muestreo de 0.11Hz, cumpliendo asi que este valor sea 10 veces menos al tiempo que demora la señal en estabilizarce, dato

escencial para que el usuario seleccione el convertidor adecuado que será parte del microcontrolador.

Se recomienda seleccionar que tenga la resolución suficiente para evitar el "Aliasing" el cual impide recuperar correctamente la señal cuando las muestras de esta se obtienen a intervalos de tiempo demasiado grandes.

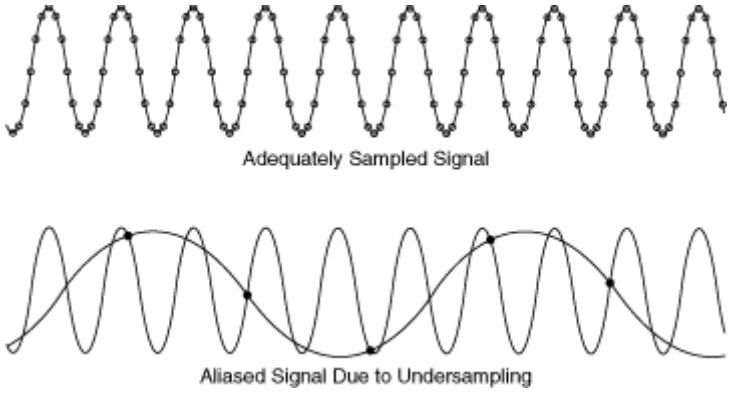


Figura 21. Efecto de aliasing [43]

4.9.Desacople del sistema

Por desacople del sistema se entiende que el sistema de control deja de operar y las aletas quedan fijas y paralelas al eje longitudinal del cohete, ahora, como se habló desde el inicio, el control por medio de las aletas del cohete es necesario para mantener una trayectoria rectilínea en su recorrido de ascenso hasta que llegue al apogeo, de tal manera que al momento de descender este control ya no es necesario, ya que si el sistema de control sigue funcionando podría interferir en la trayectoria de descenso a causa de la posición invertida del cohete.

El sistema de control se debe inhabilitar en las siguientes condiciones:

1. Cuando la velocidad sea de 2.70 m/s.

2. Cuando el cohete alcance una altura de 1715 metros sobre el punto de lanzamiento.

Las condiciones anteriormente descritas, se escogieron de acuerdo a: se pretende desacoplar el sistema cuando el cohete empieze a descender, en este caso el punto de apogeo se toma como referencia para tal fin, punto en el cual la velocidad total del cohete llega a ser aproximadamente cero, por eso en la primera condición se escojió aleatoriamente una velocidad de 2.70 m/s, pero hay que recordar que al momento del despegue existen velocidades cercanas a cero, como segunda condición se escojió una altura de 1715 metros sobre el punto de lanzamiento, este valor se elijio de manera aleatoria para evitar que el sistema se desacople en el despegue.

5. ANÁLISIS DE RESULTADOS

Para el análisis de resultados se realizó una simulación del sistema por medio de una interfaz gráfica, en donde se programaron todas las ecuaciones de la dinámica del cohete y las necesarias para la obtención de la función de transferencia.

5.1.Simulación del sistema

Para el proceso de simulación se realiza una interfaz gráfica de usuario en Matlab®, esto con el ánimo de hacerlo interactivo y que sea aplicable a otros modelos de cohetes, para esto se utilizó la interfaz gráfica GUIDE, la cual es un entorno de programación visual que realiza y ejecuta programas que necesiten ingreso de datos [27].

Inicialmente se obtiene la opción de interfaz gráfica de usuario en blanco, en donde se empieza a diseñar el programa.

Se requiere que el usuario introduzca los siguientes valores de entrada:

1. Masa.
2. Empuje.
3. Tiempo de quemado del motor.
4. Velocidad del paracaídas.

Con estos datos se realiza la programación de las respectivas ecuaciones para generar las siguientes respuestas, (la programación se encuentra en el anexo B):

1. Velocidad máxima.
2. Altura máxima.
3. Tiempo de apogeo.
4. Aceleración.

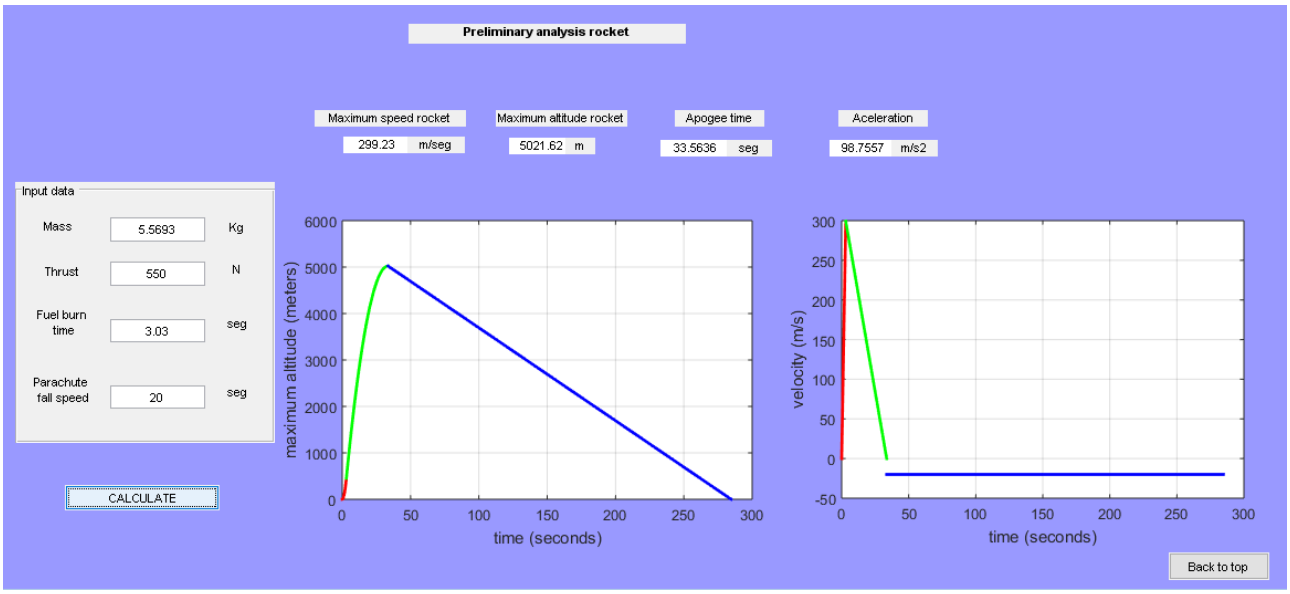


Figura 22. Interfaz gráfica del análisis preliminar del cohete

En este caso se puede apreciar la gráfica de altura versus tiempo, en donde se analiza que en la primera fase existe un desplazamiento corto en el cual el combustible estará quemándose, para luego alcanzar una segunda fase en donde el cohete tiene acumulada energía cinética la cual

se consume en el resto del recorrido hasta llegar a su altura máxima o también llamado apogeo, donde empezará a descender con una velocidad constante gracias al paracaídas, hasta terminar su descenso.

Para la gráfica de velocidad versus tiempo se observa que el cohete alcanza una velocidad máxima cuando el combustible se quema por completo, es decir al termino de la fase uno.

Para la segunda opción de interfaz se establece un análisis dinámico y la función de transferencia, para esto el usuario debe introducir los siguientes valores:

1. Masa.
2. Velocidad máxima.
3. Área de la aleta.
4. Densidad.
5. Cuerda de raíz.
6. Cuerda de punta.
7. Envergadura.
8. Diámetro del cohete.
9. Distancia desde el eje del ala al centro de la cuerda aerodinámica.
10. Mach.
11. Ángulo de cabeceo en grados.
12. Momento de inercia.
13. Coeficiente de momento de deflexión.
14. Distancia de centros de gravedad.
15. Ángulo de ataque.

Con estos valores se genera la función de transferencia (programación se encuentra en el anexo C), y así mismo la gráfica del comportamiento de la amplitud versus tiempo.

Además de esto se pretende hacer que la simulación genere una señal estable que muestre el tiempo de respuesta del sistema, para eso se debe tener en cuenta un controlador que en este caso es un controlador PID, el cual se compone de una ganancia proporcional, integral y derivativa.

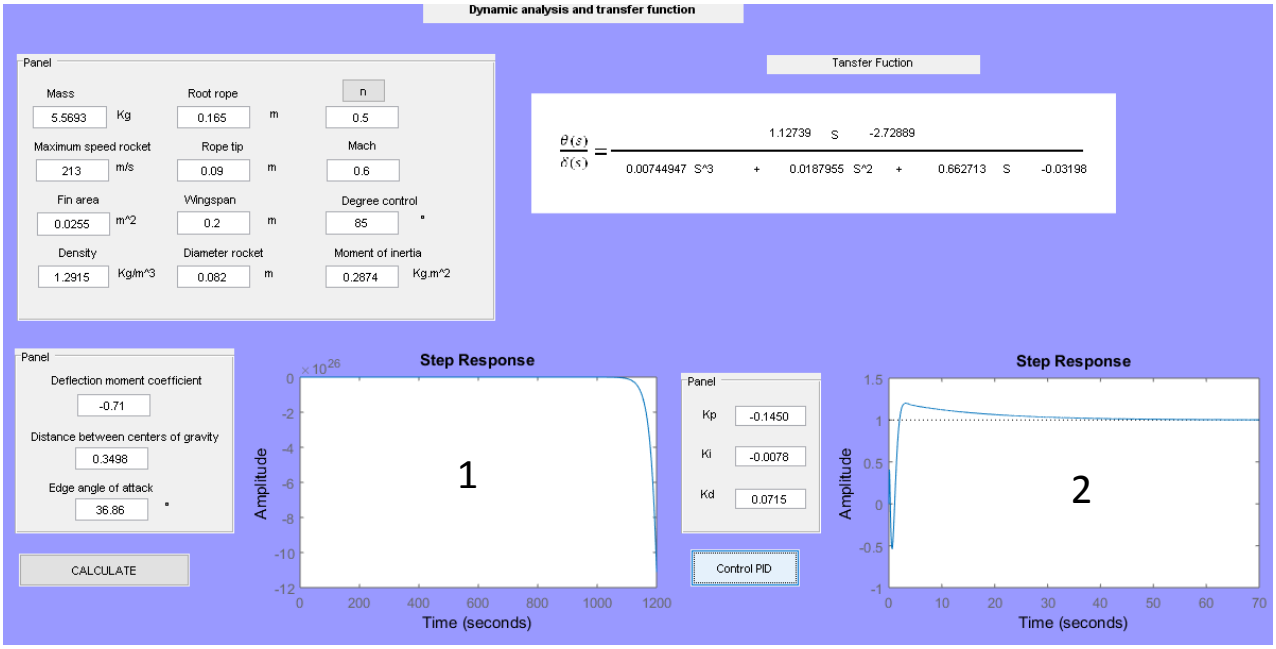


Figura 23. Interfaz gráfica del análisis dinamico y función de transferencia

De acuerdo al recuadro 1 de la figura 22 se puede apreciar los resultados de la simulación, donde se tiene la gráfica de amplitud versus tiempo que inicialmente no se encuentra controlada ya que presenta un sistema abierto sin retroalimentación y asi mismo carece de un controlador es por esto la amplitud no se llega a estabilizar, caso contrario sucede en el recuadro 2 de la figura 22 en donde se detalla una señal estable gracias a la implementación de un controlador PID.

CONCLUSIONES

Por medio de Rocksim V9.1.3® se simularon los parámetros del vuelo del cohete Ainkka V, los cuales se utilizaron para el desarrollo del sistema de control.

Por medio de análisis dinámicos se pudo generar un modelo capaz de estabilizar el cohete en su ángulo de cabeceo, el cual implementa un control PID, que llega a estabilizar el sistema conforme a un valor de entrada deseado (ángulo de cabeceo).

Este sistema de control pretende ser implementado físicamente en futuros proyectos. Es por eso que se buscó determinar qué lenguaje resultaba ser más efectivo para su implementación, implicando así el uso de lenguaje discreto para el control PID. Sin embargo se puede evidenciar que el tiempo de respuesta para los dos sistemas; resulta ser un poco prolongado, teniendo en cuenta el tiempo que el cohete tendrá desde su punto cero hasta la altura del apogeo, por esta razón es necesario adicionar otro controlador que pueda modificar o mejorar ese tiempo de respuesta. No obstante el mecanismo del sistema de control se debe ubicar en las aletas, en la cuerda de raíz a la altura del centro de presión, según los análisis expuestos anteriormente; este sistema deberá estar conformado por cuatro servos cada uno para una aleta respectivamente, una plataforma que tenga un entorno que implemente el lenguaje de programación en este caso un arduino, también es necesario una unidad de medición inercial (IMU) la cual detecta la tasa de aceleración usando acelerómetros, además de cambios en el cabeceo, guiñada y alabeo usando uno o más giróscopos y una batería que se encarga de alimentar todo el sistema.

De acuerdo al análisis dinámico, se deben tener en cuenta los datos de entrada y salida, así como las especificaciones de vuelo del cohete, ya que a partir de estos se genera las ecuaciones de movimiento, para luego determinar la función de transferencia, la cual será la planta del

sistema que tendrá lugar en Simulink®, para desarrollar el lazo cerrado junto con el controlador y así poder generar la señal de respuesta.

Para la primera opcion de la interfaz gráfica la cual resulta ser un análisis preliminar del cohete, se sugiere que el usuario use plataformas de simulación como Rocksim V9.1.3®, en donde se obtienen resultados mas exactos los cuales son fundamentales para el análisis dinámico y la generación de la función de transferencia que resulta ser la planta para diseñar el sistema de control que se requiere.

Se publico el documento científico, en el libro DESARROLLO E INNOVACIÓN EN INGENIERÍA primera edición, acta de Ingenieira Vol. 2, pp.135-142, 2016 del Instituo Antioqueño de investigación, gracias a la participación en la CONFERENCIA INTERNACIONAL DE INGENIERIA, llevada a cabo del 23 al 25 de Agosto en la ciudad de Medellin.

ANEXOS

Anexo 1. Cálculos

Parametros del cohete para el analisis dinamico							
t (S)	h(m)	U(m/s)	m(Kg)	I_y (Kg m^2)	cg (m)	CD	Mach
1,005	5,3909	26,544	5,8783	0,2962	0,5826	0,3274	0,0778
3,03	302	213	5,5693	0,2874	0,5747	0,275	0,6276
12,015	1715,8	103,1354	5,5693	0,2871	0,5747	0,2947	0,3086
16,065	2045,68	60,4061	5,5693	0,2871	0,5747	0,3088	0,1814
20,07	2206,812	20,529	5,5693	0,2871	0,5747	0,3467	0,061
22,05	2227,92	2,7075	5,5693	0,2871	0,5747	1,96	0,0081

Simbolo	Valor	Unidad
C_r	0,165	m
C_t	0,09	m
b	0,2	
λ	0,545454545	m
A	1,568627451	
S	0,0255	m^2
$\bar{c}$	0,131176471	
y_{mac} $\frac{x_{c.g}}{c_r}$	0,045098039	m
$\frac{c_r}{\bar{c}}$	1,257847534	
Λ_{LE}	0,643328362	grados
$C_{L\alpha}$	2,060223155	
$tan\Lambda_{LE}$	0,749730119	
$Atan\Lambda_{LE}$	1,5727	
β	0,800241854	
$\beta/tan\Lambda_{LE}$	1,067373224	
$\frac{x_{a.c}}{c_r}$	0,32402712	m
$\frac{\bar{x}}{\bar{c}}$	0,27892908	m
$\Lambda_{c/4}$	0,032794118	
$\cos(\Lambda_{c/4})$	0,999462321	
$tan^2\Lambda_{c/4}$	0,001076226	
M	0,6	
d	0,082	m
l	1,09	m

Simbolo	Valor	Unidad
l	1,09	m
q	29356,01309	kg/m^2
$\frac{d}{2U}C_{mq}$	-1,48932E-05	s
$\frac{mU}{Sq}$	1,59E+00	s
$\frac{mg}{Sq}$	0,07695566	
$\frac{I_y}{Sqd}$	4,68E-03	s^2
n	0,5	
ρ	1,2915	kg/m^3
Θ	85	grados
l_t	0,3498	m
$C_w sen\Theta$	0,07666282	
C_{xu}	0,5894	
T	1180	N

Aerodinamica del cohete		
Descripcion	**Simbolo**	**Valor**
Area transversal		**0,005281017**
Calculo coeficiente de Drag	C_d	0,352056226
DRAG	D	263,5416633
Calculo coeficiente de Lift	C_l	1
Velocidad de perdida	V_{stall}	57,57196418

ECUACIONES DE MOVIMIENTO																
$\left(\frac{mU}{Sq}s - C_{z\alpha}\right)\alpha(s) + \left(-\frac{mU}{Sq}s - C_w sin\Theta\right)\theta(s) = C_{z\delta_e}\delta_e$																
1	1,586279415	s	-	-4	α(S)	+	-1,58627941	s	-	0,076662821	θ(S)	=	-0,26625298	δ_e (S)		
$\left(-\frac{d}{2U}C_{m\dot{\alpha}}s - C_{m\alpha}\right)\alpha(s) + \left(\frac{I_y}{Sqd}s^2 - \frac{d}{2U}C_{mq}s\right)\theta(s) = C_{m\delta_e}\delta_e$																
2			-0,45602432	α(S)	+	0,00467716	S^2	-	-1,48932E-05	s	θ(S)	=	-0,71	δ_e(S)		

NUM (θ (s))

		-1,58627941	s	4	-0,26625298	
		-0,45602432			-0,71	

s	1,12625838
N	-2,96141784

DEN (δ (s))

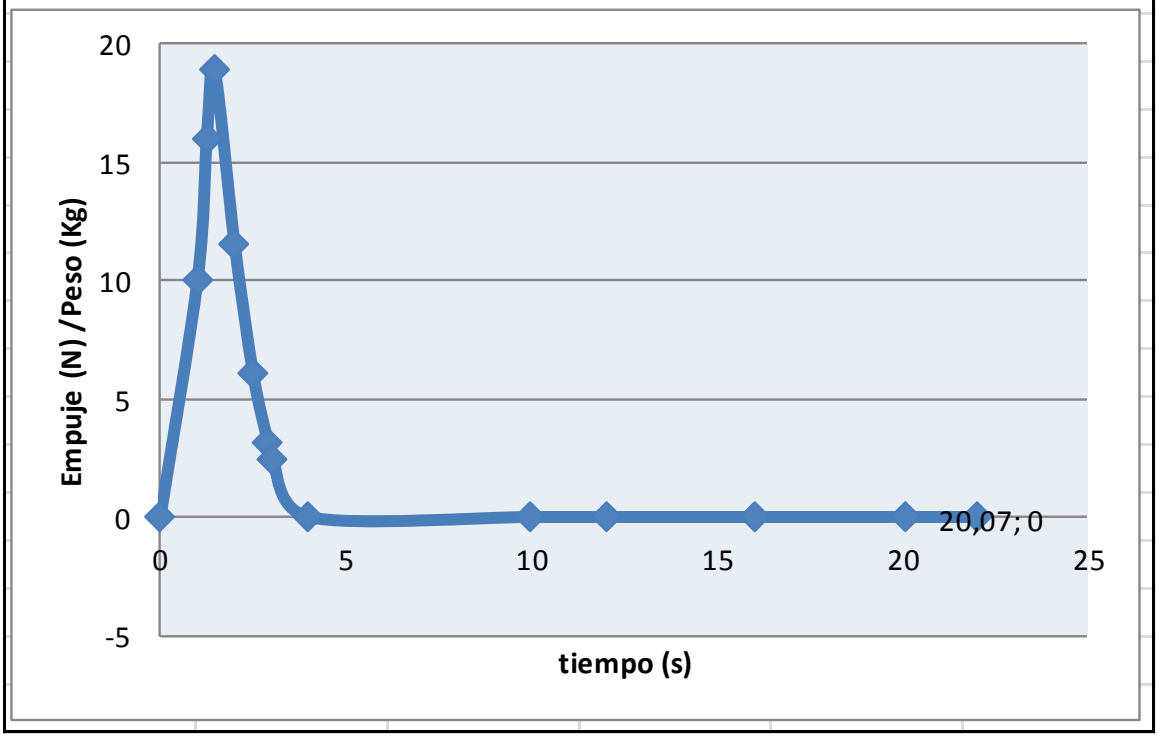
	1,58627941	s	4	-1,58627941	s	-0,076662821
	-0,45602432			0,00467716	S^2	1,48932E-05

S^3	0,00741928
S^2	0,01873226
s	0,72344157
N	-0,03496011

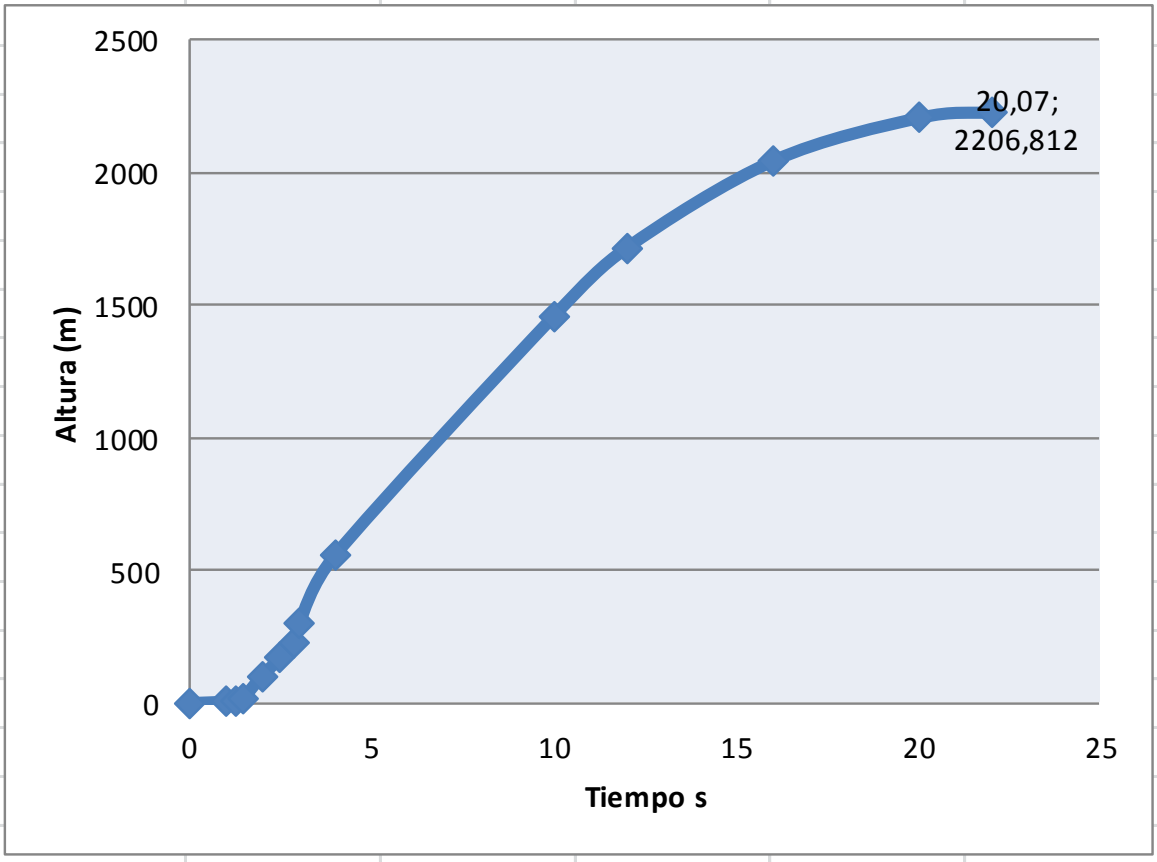
FUNCION DE TRANSFERENCIA

$$\frac{\theta(s)}{\delta(s)} = \frac{1{,}12625838\,S + (-2{,}96141784)}{0{,}00741928\,S^3 + 0{,}01873226\,S^2 + 0{,}72344157\,s + (-0{,}03496011)}$$

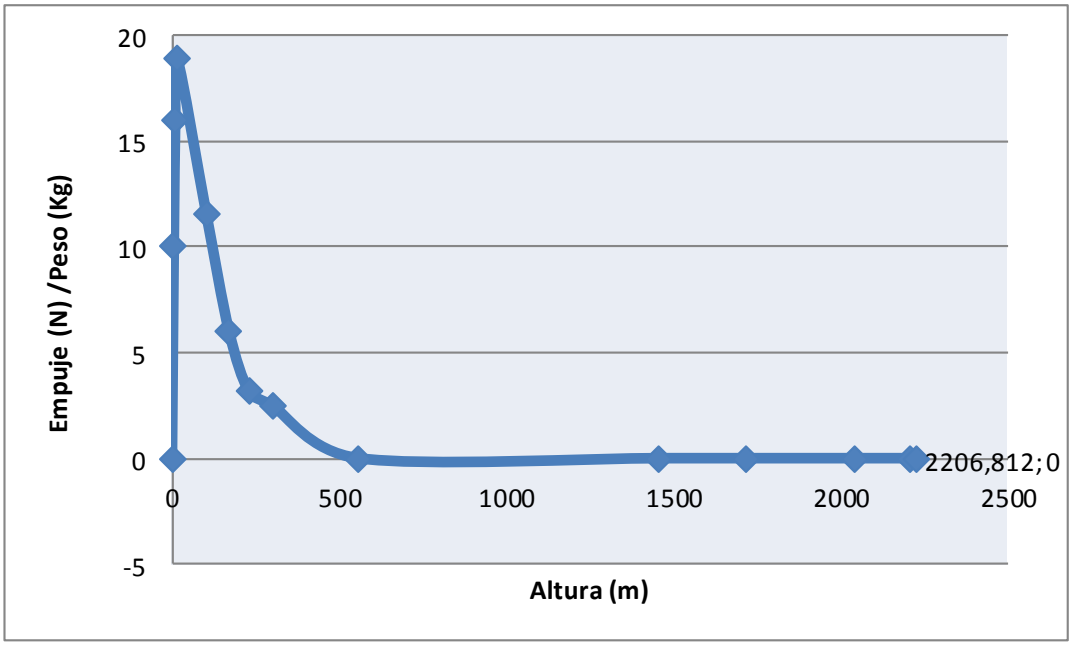
T (N)	t (S)	w(N)	T/w
0	0	0	0
579,152	1,005	57,60734	10,0534411
912,794	1,2525	57,13106	15,9771935
1080,74	1,46	57,1781	18,9012926
642,58	2	55,52582	11,5726341
331,348	2,505	54,90744	6,03466488
173,522	2,9025	54,6791	3,17346116
134,81	3,03	54,62324	2,46799714
0	4	54,53602	0
0	10	54,53602	0
0	12,015	54,53602	0
0	16,065	54,53602	0
0	20,07	54,53602	0
0	22,05	54,53602	0



h(m)	t (S)
0	0
5,3909	1,005
9,654	1,2525
15,98	1,46
102,76	2
170,457	2,505
230,65	2,9025
302	3,03
556,85	4
1456,89	10
1715,8	12,015
2045,68	16,065
2206,812	20,07
2227,92	22,05



T/W	h(m)
0	0
10,0534411	5,3909
15,9771935	9,654
18,9012926	15,98
11,5726341	102,76
6,03466488	170,457
3,17346116	230,65
2,46799714	302
0	556,85
0	1456,89
0	1715,8
0	2045,68
0	2206,812
0	2227,92



Anexo 2. Artículo publicado

Actas de Ingeniería
Vol. 2, pp. 135-142, 2016

http://fundacioniai.org/actas

Simulated control system for keeping the trajectory of an atmospheric ballistic rocket, not remote-controlled

Sistema de control simulado para mantener la trayectoria de un cohete balístico atmosférico no-teledirigido

Angélica Rincón C.[1], Yesid Parra R.[2], José Urrego P.[3]
[1]*amrincon(AT)academia.usbbog.edu.co,* [2]*ycparra(AT)academia.usbbog.edu.co,* [3]*jurrego(AT)usbbog.edu.co*
Universidad de San Buenaventura. Bogotá – Colombia

Artículo de Investigación

Abstract
This project presents the design of a control system capable of maintaining a straight path modifying the pitch attitude in an atmospheric rocket, this will be done under the design specifications of Ainkka V rocket built previously at the University of San Buenaventura, Bogotá D.C. The system is based on the calculation of the transfer function relating the output angle (which in this case is the pitch) with fins deflection, then a controlled is implement, which by means of a feedback signal is capable of stabilize the system, to obtain the desired response.

Keywords: Atmospheric rocket, transfer function, controller, dynamic analysis.

Resumen
Este proyecto presenta el diseño de un sistema de control capaz de mantener una trayectoria recta modificando la posición de cabeceo en un cohete atmosférico. Esto se hizo bajo las especificaciones de diseño del cohete Ainkka V construido en la Universidad de San Buenaventura, sede Bogotá. El sistema se basó en el cálculo de la función de transferencia que relaciona el ángulo de salida (que en este caso es el de cabeceo) con la deflexión de las aletas. Posteriormente se implementó un controlador que por medio de una señal retroalimentada es capaz de estabilizar el sistema, para obtener el resultado deseado.

Palabras clave: Cohete atmosférico, función de transferencia, controlador, análisis dinámico.

Citación
Rincón, A., Parra, Y. and Urrego J. 2016. Sistema de control simulado para mantener la trayectoria de un cohete balístico atmosférico no-teledirigido. Actas de Ingeniería 2, 135-142.

1. Introducción

Tras la fase de impulso, un cohete sigue una trayectoria determinada de acuerdo a la misión con la cual este fue diseñado. Ahora, en el espacio aéreo en el cual un cohete está volando, no siempre se encontrarán las condiciones esperadas para un vuelo normal y sin complicaciones, es decir, que se produzcan condiciones del medio ambiente que entorpezcan la misión. Estas condiciones están ligadas con fenómenos tales como ráfagas o vientos cortantes que pueden causar el desvió de la trayectoria o rumbo del cohete. Esta desviación se puede evidenciar en grados, por ejemplo, un cohete con aletas estáticas puede desviarse hasta cierto grado en donde las aletas lo logran estabilizar. Pero pueden existir situaciones en las que estas desviaciones sean mayores, como cuando la desviaciones del cohete superan aproximadamente los 15 grados con respecto al eje de la trayectoria, si es así, el cohete perderá el control, entrará en perdida y por consiguiente se precipitará, terminando en un total fracaso la misión, generando muchos atrasos en la investigación u objetivo del lanzamiento, así como la perdida de altas sumas de dinero.

Es por lo anterior que el proyecto hace referencia a la simulación de un sistema de control activo para un cohete atmosférico, no teledirigido y así evitar los problemas ya mencionados, que puede llegar a tener una misión con un cohete.

De acuerdo a las características de los cohetes, estos poseen 3 ejes de movimiento: guiñada, alabeo y cabeceo. En este caso se escogió el eje de cabeceo, encargado del movimiento de cabeceo del cohete. Como resultado de la simetría de Y y Z tal como se aprecia en la figura 1, viendo el cohete desde el eje longitudinal, la función de transferencia que se va a realizar para el control del eje de cabeceo también puede ser usada para el control del eje de guiñada.

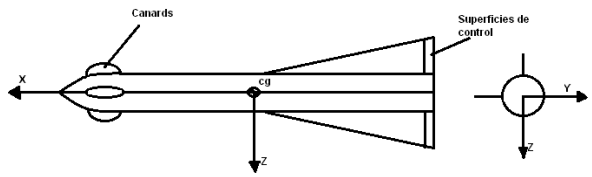


Figura 24 Esquema de un misil aerodinámico y sus ejes.

Se modeló el sistema de control activo a partir de ecuaciones diferenciales lineales e invariantes en el tiempo. Con estas se generó una función de transferencia la cual por medio de Simulink®, el cual es una plataforma de Matlab®, se puede modelar y simular sistemas dinámicos.

El proceso de simulación en Simulink® posee dos fases, la primera se refiere al diseño del modelo de acuerdo a las especificaciones de la planta y la segunda fase es el análisis del modelo junto con el controlador que estabilizará el sistema en su eje de cabeceo.

Conforme a los resultados de la simulación se obtuvieron diagramas en donde se puede observar la respuesta del sistema respecto al tiempo.

2. Análisis dinámico del sistema

Las fuerzas que actúan en un cohete son la sustentación, el arrastre y el empuje. La primera es la resultante de la fuerza aerodinámica y debe ser perpendicular al viento relativo; el arrastre también es el resultante de la fuerza aerodinámica, es paralelo al viento relativo, y es producido por la fuerza de presión y la fuerza de fricción que actúan en la superficie; y por último el empuje que es la fuerza generada por la expulsión de gases de la cámara de combustión [1].

La siguiente figura hace referencia a los ejes X, Y, Z y a los movimientos del cohete P, Q y R, que son el movimiento de alabeo, cabeceo y de guiñada respectivamente [2].

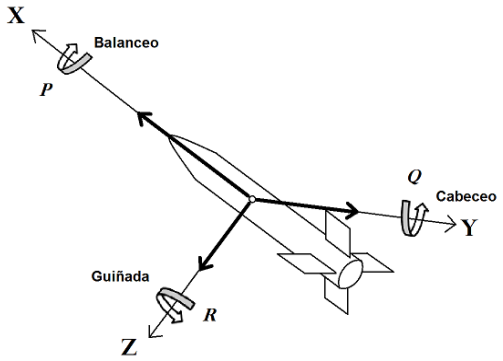


Figura 25 Grados de libertad del cohete [2]

Para el periodo de análisis se asume que el cohete balístico tiene una masa constante y es considerado como un cuerpo rígido.

2.1 Desarrollo de las ecuaciones de movimiento
El movimiento del cohete se rige bajo la segunda ley de Newton, la cual declara que la suma de todas las fuerzas externas que actúan en un cuerpo deben ser igual a la tasa de cambio de su momentum, ecuación (1) y la suma de los momentos externos que actúan en un cuerpo deben ser igual a la tasa de cambio de su momentum angular, ecuación (2). Las tasas de cambio son tomadas respecto a un espacio inercial [1].

$$\sum \mathrm{F} = \frac{\mathrm{d}}{\mathrm{dt}}(\mathrm{mV_T})\Big]_{\mathrm{I}} \quad (1)$$

$$\sum M = \frac{dH}{dt}\Big]_I \quad (2)$$

Para el análisis se debe asumir lo siguiente:
La masa del cohete permanece constante durante el vuelo; este es un cuerpo rígido, y la tierra se toma como referencia inercial.

De acuerdo a la ecuación (1), se derivan las siguientes tres ecuaciones que corresponden a las ecuaciones del movimiento lineal, en los ejes X, Y y Z:

$$\sum \Delta F_x = m(\dot{U} + WQ - VR) \quad (3)$$
$$\sum \Delta F_Y = m(\dot{V} + UR - WP) \quad (4)$$
$$\sum \Delta F_z = m(\dot{W} + VP - UQ) \quad (5)$$

Donde V y W son las componentes de la velocidad lineal y P, Q y R son los componentes de la velocidad angular. Estos son tomados respecto al eje de referencia que es la tierra.

Las ecuaciones de movimiento angular se derivan de la ecuación (2) y son presentadas a continuación:

$$\sum \Delta \ell = \dot{P} I_x - \dot{R} J_{xz} + QR(I_z - I_y) - PQ J_{xz} \quad (6)$$
$$\sum \Delta \mathcal{M} = \dot{Q} I_y + PR(I_x - I_z) - (P^2 - R^2) J_{xz} \quad (7)$$
$$\sum \Delta \aleph = \dot{P} I_z - \dot{P} J_{xz} + PQ(I_y - I_x) + QR J_{xz} \quad (8)$$

De acuerdo a lo anterior, es necesario determinar seis ecuaciones de movimiento simultáneas para describir el comportamiento del cohete completamente. En algunos casos se pueden hacer ciertas asunciones para dividir estas en dos grupos de tres ecuaciones simultáneas.

Para estos análisis se debe considerar el cohete en un vuelo recto y sin aceleración, para luego ser distorsionado por la deflexión del elevador.

De acuerdo a la figura 3, la deflexión del elevador en el eje OY genera un momento de cabeceo en este mismo eje y a causa de este se generan cambios en las fuerzas F_x y F_z. Esto no genera momento de rotación ni de guiñada, por lo tanto no hay cambios en F_y, por ende P=R=V=0 y $\sum \Delta F_Y$, $\sum \Delta \ell$, y $\sum \Delta \aleph$ son descartados.

$$\sum \Delta F_x = m(\dot{U} + WQ) \quad (9)$$
$$\sum \Delta F_z = m(\dot{W} - UQ) \quad (10)$$
$$\sum \Delta \mathcal{M} = m(\dot{W} - UQ) \quad (11)$$

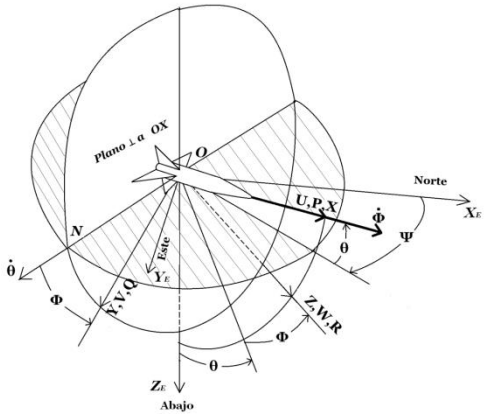


Figura 26 Diagrama de ejes para un cohete.

2.2 Ecuaciones de movimiento longitudinal
Es necesario expandir las ecuaciones de fuerzas y momentos aplicados, y expresarlos en los cambios de estas mismas a causa de perturbaciones.

$$\left(\frac{mU}{Sq}\,'\dot{u} - C_{xu}\,'u\right) + \left(-\frac{c}{2U}\,C_{x\dot{\alpha}}\,'\dot{\alpha} - C_{x\alpha}\,'\alpha\right) + \left[-\frac{c}{2U}C_{xq}\dot{\theta} - C_w(cos\Theta)\theta\right] = C_{Fxa} \quad (12)$$

$$-(C_{zu}\,'u) + \left[\left(\frac{mU}{Sq} - \frac{c}{2U}C_{z\dot{\alpha}}\right)'\dot{\alpha} - C_{z\alpha}\,'\alpha\right] + \left[\left(-\frac{mU}{Sq} - \frac{c}{2U}C_{zq}\right)\dot{\theta} - C_w(sin\Theta)\theta\right] = C_{Fza} \quad (13)$$

$$(-C_{mu}\ 'u) + \left(-\frac{c}{2U}C_{m\dot{\alpha}}\ '\dot{\alpha} - C_{m\alpha}\ '\alpha\right) + \left(\frac{I_y}{Sqc}\ddot{\theta} - \frac{c}{2U}C_{mq}\dot{\theta}\right) = C_{ma} \quad (14)$$

Estas ecuaciones asumen que:

- Los ejes X y Z recaen en el plano de simetría y el origen del sistema de ejes está en el centro de gravedad del cohete.
- La masa del cohete es constante.
- El cohete se asume como cuerpo rígido.
- La tierra es la referencia inercial.
- Las perturbaciones en el equilibrio son pequeñas.

Para la solución de las ecuaciones de movimiento es necesario obtener inicialmente la solución de la ecuación homogénea, para tal caso $C_{ma} = C_{Fza} = C_{Fx\alpha} = 0$. Tomando la transformada de Laplace de la ecuación (6) y despreciando $C_{x\dot{\alpha}}$, C_{xq} y C_{mu}, se obtiene:

$$\left(\frac{mU}{Sq}s - C_{xu}\right)\ 'u(s) - C_{x\alpha}\ '\alpha(s) - C_w(cos\Theta)\theta(s) = 0 \quad (15)$$

$$-C_{zu}\ 'u(s) + \left[\left(\frac{mU}{Sq} - \frac{c}{2U}C_{z\dot{\alpha}}\right)s - C_{z\alpha}\right]\ '\alpha(s) + \left[\left(-\frac{mU}{Sq} - \frac{c}{2U}C_{zq}\right)s - C_w(sin\Theta)\right]\theta(s) = 0 \quad (16)$$

$$\left(-\frac{c}{2U}C_{m\dot{\alpha}}s - C_{m\alpha}\right)\ '\alpha(s) + \left(\frac{I_y}{Sqc}s^2 - \frac{c}{2U}C_{mq}s\right)\theta(s) = \quad (17)$$

2.3 Aproximación de periodo corto

La aproximación de periodo corto se toma cuando se asume una velocidad constante hacia adelante, es decir, 'u=0; por lo tanto, no existen cambios en las fuerzas que se generan en esta dirección ya que no hay cambio de velocidad. Al eliminar la ecuación de movimiento en el eje X de la ecuación (7) queda de la siguiente forma:

$$\left(\frac{mU}{Sq}s - C_{z\alpha}\right)\alpha(s) + \left(-\frac{mU}{Sq}s - C_w sin\Theta\right)\theta(s) = C_{z\delta e}\delta_e \quad (18)$$

$$\left(-\frac{d}{2U}C_{m\dot{\alpha}}s - C_{m\alpha}\right)\alpha(s) + \left(\frac{I_y}{Sqd}s^2 - \frac{d}{2U}C_{mq}s\right)\theta(s) = C_{m\delta e}\delta_e \quad (19)$$

2.4 Caracterización del cohete

Como se mencionó anteriormente, se trabajó con el cohete Ainkka V construido por la Universidad de San Buenaventura, sede Bogotá. Para este análisis se tomó en cuenta la tercera etapa, la cual se muestra en la figura 3 y 4.

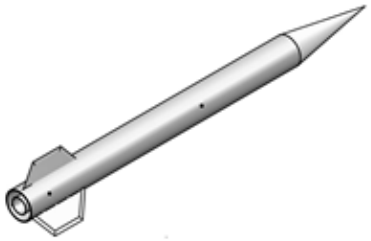


Figura 27 Modelo en Catia V5 del Cohete etapa Ainkka V, tercera etapa.

Figura 28 Modelo físico del Cohete etapa Ainkka V, tercera etapa.

A continuación se muestran los parámetros que conforman el diseño del cohete:

Tabla 4 Dimensiones generales del cohete

Dimensiones generales		
Peso total cohete	5,56	Kg
Largo de tubo	0,87	m
Diámetro de tubo	0,082	m
Material del tubo	PVC	-
Largo de ojiva	0,22	m
Material de la ojiva	Madera	-
Largo total	1,09	m
Envergadura	0,2	m

Tabla 5 Dimensiones de la aleta

Dimensiones de la aleta		
Cuerda de raíz	0,165	m
Cuerda de punta	0,09	m
Envergadura de aleta	0,06	m
Área	0,0255	m^2
Espesor	0,003	m
Material	Aluminio	Al 1050

De igual forma, se tomaron en cuenta las condiciones del ambiente en donde se va a realizar el lanzamiento, ya que se pretende realizar en la base de Marandua, Vichada Colombia, donde la temperatura promedio es de 25°c. Resulta ser un espacio semidesértico y despoblado, perfecto para este tipo de actividades.

Para poder simular el vuelo del cohete fue necesario utilizar *Rocksim V9.1.3*®, el cual es un programa informático que permite diseñar cualquier tipo de cohete, luego simular su vuelo para ver qué tan alto y rápido puede volar, esto con el fin de darse cuenta si el modelo va a ser estable y seguro para poner en marcha. Además de esto permite cumplir con los criterios de peso, velocidad y altura que se desee [3].

Para la primera fase en *Rocksim V9.1.3*® se realiza el diseño de cada una de las secciones que componen el cohete, se empieza por la nariz cónica ingresando valores como la longitud, el diámetro, el material con la que fue construida. Para la sección del cuerpo del cohete se debe tener en cuenta la longitud y el diámetro del tubo, la longitud del bloque del motor donde se ubica el combustible y finalmente las dimensiones de las aletas y su localización en el cuerpo del cohete.

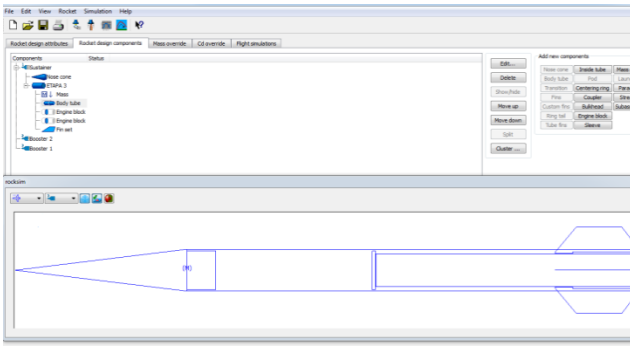


Figura 29 Diseño del cohete en Rocksim.

Para cargar el motor en la simulación es necesario crearlo en la biblioteca de motores, por medio de la interfaz de *EngEdit*, el cual a partir de la longitud, el diámetro, la masa inicial y la masa del propelente, genera el empuje promedio, el empuje máximo generado, el tiempo de quemado y el impulso específico.

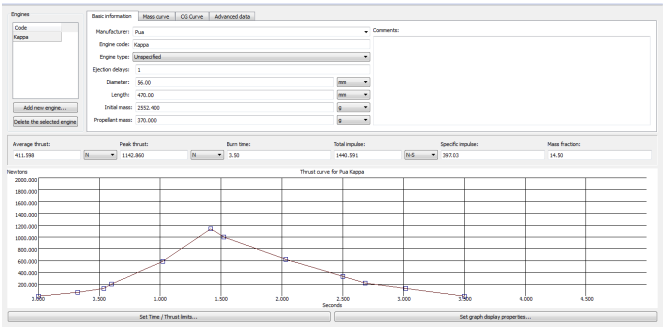


Figura 30 Carga del motor en EngEdit.

Después de haber diseñado el cohete y cargado el motor, se declaran las condiciones en las cuales se va hacer la simulación. Dentro de estas tenemos la temperatura, la densidad y la presión del ambiente, condiciones esenciales para determinar el punto geográfico de lanzamiento del cohete.

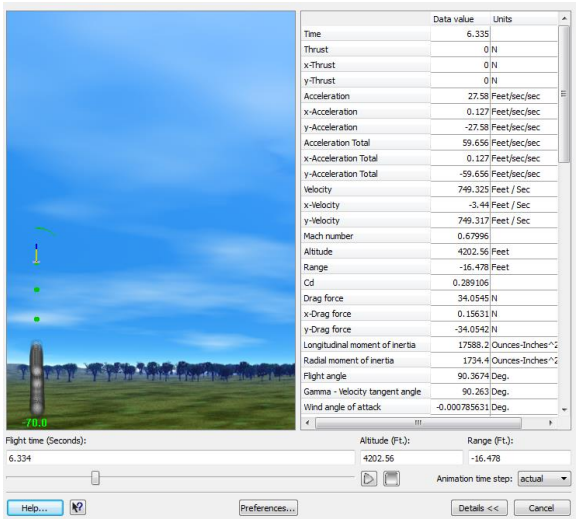


Figura 31 Simulación del vuelo del cohete en Rocksim.

Gracias a *Rocksim V9.1.3*® se encontraron datos como la velocidad, el empuje generado, la altura alcanzada, el tiempo de vuelo, el momento de inercia y los centros de gravedad, entre otros datos

[11] (estos datos son la base para el análisis dinámico y el sistema de control). Además se obtienen gráficas donde se muestra el comportamiento del cohete desde el punto cero hasta el punto del apogeo.

En la figura 9 se observa que el comportamiento de la altura es directamente proporcional al tiempo y se establece el punto de apogeo a la altura de 2227,92 m para un tiempo de 22,05 segundos.

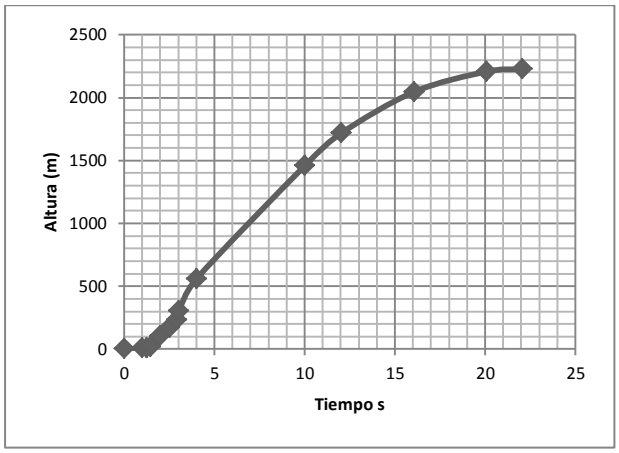


Figura 32 Altura vs Tiempo del cohete Ainkka V.

El empuje es parte importante en la caracterización de la capacidad de elevación de un motor cohete, pero para saber qué tan alto será propulsado es necesario medir la salida total. Para esto se utiliza el impulso total del motor cohete que reúne el tiempo o duración del empuje y se define como la integral del empuje sobre la duración operativa del motor [4].

En *Rocksim V9.1.3*®, también se puede realizar la estimación del empuje y el peso, teniendo en cuenta el tiempo y la altura, así como se observa en las figuras 10 y 11.

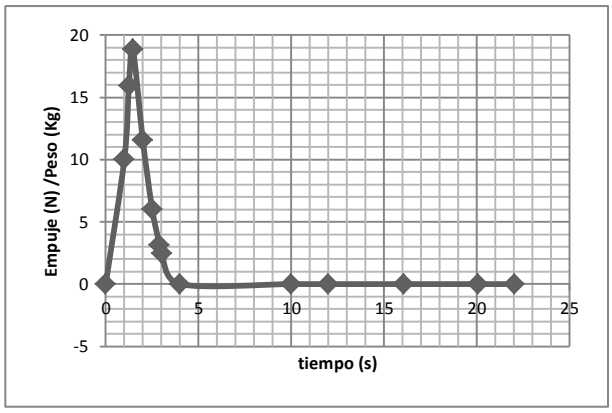


Figura 33 Empuje/peso vs tiempo.

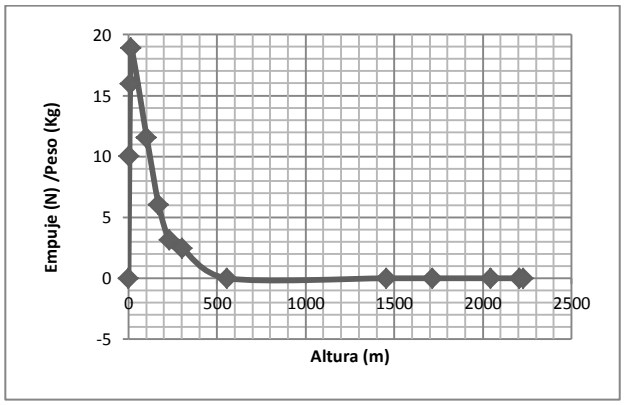


Figura 34 Empuje/peso vs altura.

El empuje máximo que presenta el cohete *Ainkka V* es en el tiempo de 1,46 s con un empuje de 1080,64 N; sin embargo el combustible tiene un tiempo de quemado rápido haciendo que en el segundo 3,03 se genere el empuje total y seguidamente el cohete actúe bajo la inercia que lleva consigo, la cual resulta ser elevada y al mismo tiempo alcanza así una mayor altura.

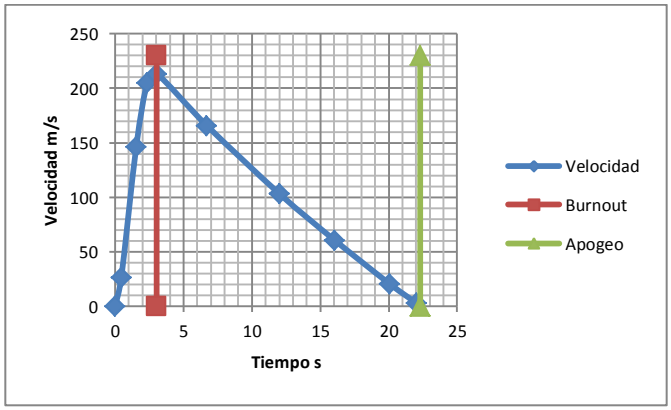


Figura 35 Velocidad vs Tiempo.

Por medio de *Rocksim V9.1.3*® fue posible obtener la gráfica de velocidad vs tiempo, en la cual se puede apreciar el momento en el que se consume todo el combustible (*Burnout*) en un tiempo aproximado de 3,03 segundos. En este punto se observa la velocidad máxima alcanzada por el cohete: 213 m/s. Desde este punto el cohete empieza a desacelerar a una razón de aproximadamente -11,038 m/s^2, esto gracias a la pendiente de una línea recta aproximada desde el punto de velocidad máxima con las coordenadas (3, 213) hasta el punto de velocidad cero con coordenadas (22, 3).

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-213}{22-3} = -11{,}05\ m/s^2 \qquad (20)$$

Así mismo en la figura 12 se observa una línea de apogeo la cual indica que a los 22 segundos llega a su altura máxima evidenciando una pérdida en la energía cinética, que se obtuvo por el empuje inicial del motor cohete.

A partir de los datos arrojados por *Rocksim V9.1.3*® se estiman algunas condiciones aerodinámicas, teniendo en cuenta los centros de gravedad y centros de presiones de cada elemento que conforma el cohete [8].

Primero se analizó la parte superior del cohete. Generalmente la fuerza normal en la nariz es similar para todas las formas, en este caso es una nariz cónica y el valor para la fuerza normal es de 2 [5].

$$(C_{N\alpha})_n = 2 \qquad (21)$$

Mientras que la ubicación del centro de presión tiende a variar de acuerdo a la geometría que tenga la nariz, para el caso de la nariz cónica se tiene la siguiente ecuación:

$$\overline{X_n} = \frac{2}{3}L \qquad (22)$$

Luego se analizaron las aletas, donde según las dimensiones, se obtiene la fuerza que ejercen estas en el cohete, con la siguiente ecuación:

$$(\mathrm{C_{N\alpha}})\mathrm{f} = \frac{4\mathrm{n}\left(\frac{\mathrm{s}}{\mathrm{d}}\right)\text{^}2}{1+\sqrt{1+\left(\frac{2\mathrm{l}}{\mathrm{a+b}}\right)\text{^}2}} \qquad (23)$$

Donde n es el número de aletas, que en este caso son 4.

Además de esto, se debe tener en cuenta que el flujo de aire sobre las aletas está influenciado por el flujo de aire sobre la sección del cuerpo a la que las aletas están unidas. Es por esto que la fuerza para cada aleta es multiplicada por un factor de interferencia [5].

$$\mathrm{K_{fb}} = 1 + \frac{\mathrm{R}}{\mathrm{S+R}} \qquad (24)$$

Teniendo en cuenta esa fuerza de interferencia la fuerza que generan las aletas en presencia del cuerpo del cohete está dada por la siguiente ecuación:

$$(\mathrm{C_{N\alpha}})\mathrm{fb} = \mathrm{K_{fb}}(\mathrm{C_{N\alpha}})\mathrm{f} \qquad (25)$$

La ubicación del centro de presión para las aletas se toma en cuenta desde la raíz de la nariz hasta el valor $\overline{\mathrm{X}}_\mathrm{f}$.

$$\bar{X}_f = X_f + \Delta X_f \qquad (26)$$

$$\overline{\mathrm{X}}_\mathrm{f} = 92{,}5\mathrm{cm} + 5{,}3083 = 97{,}8083\mathrm{cm}$$

Este punto se puede ver claramente en la figura 13, la cual muestra el modelo de la aleta por medio del programa de diseño *Catia V5*, el cual permite obtener las características de cada elemento del cohete y especifica el centro de gravedad, que es donde se intersectan la línea roja y verde.

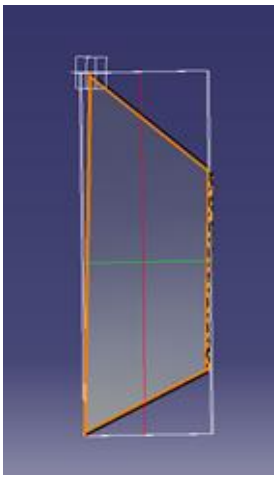


Figura 36 Centro de gravedad por medio de Catia V5.

Después de hallar la fuerza normal que actúa en la nariz cónica y en las aletas se halla la fuerza normal total.

$$C_{N\alpha} = (C_{N\alpha})n + (C_{N\alpha})fb \quad (27)$$

Y el centro de presión del cohete.

$$\bar{X} = \frac{((C_{N\alpha})n * \overline{X_n} + ((C_{N\alpha})fb * \bar{X}_f}{C_{N\alpha}} \quad (28)$$

$$\bar{X} = \frac{2(14{,}666) + 5{,}7207(97{,}8083)}{7{,}7207} = 76{,}2707cm$$

Donde se relaciona la fuerza normal total $C_{N\alpha}$, con el centro de presión de la nariz $\overline{X_n}$, la fuerza total de las aletas ($C_{N\alpha}$)fb y el centro de presión de las mismas
$\bar{X}_f$.

Se analizó la posición del centro de gravedad del cohete, ya que con este valor se determina si el vuelo es seguro y estable.

$$X_{CG} = \bar{X} - d \quad (29)$$

$$X_{CG} = 76{,}2707\text{cm} - 8{,}2\text{cm} = 68{,}0707\text{cm}$$

Según los datos encontrados se determina que el cohete es estáticamente estable, ya que el centro de gravedad se encuentra delante del punto del centro de presión.

3. Estimaciones para la simulación

De acuerdo a los resultados arrojados por *Rocksim V9.1.3®,* en la simulación se toman en cuenta los valores que se relacionan como la presión dinámica,

$$q = \frac{1}{2}\rho U^2 \quad (30)$$

$$q = 29356{,}01309Kg/m\text{^}2$$

el momento de inercia $I_y = 0{,}2874\, Kgm^2$, la velocidad máxima alcanzada $U = 213m/s$ y el empuje del motor sólido del cohete $T = 1180N$.

Dentro de las estimaciones para encontrar la función de transferencia se encuentra las derivadas del cabeceo, la amortiguación, el valor de la pendiente de la fuerza normal, el coeficiente de momento de cabeceo y el coeficiente de deflexión del elevador o aleta [7].

De acuerdo a las especificaciones y según las ecuaciones de movimiento (8) se tiene:

$$(1{,}586279\ \text{s} - (-4))\alpha(s) +$$
$$(-1{,}58628\ \text{s} - 0{,}0766628)\theta(s) = -0{,}26625\ \delta e(s) \quad (31)$$

$$(-0{,}45602)\alpha(s) +$$
$$(0{,}004677\ \text{s}^2 - (-1{,}4893\text{x}10^{-5}\ \text{s})\theta(s) =$$
$$-0{,}71\ \delta e(s) \quad (32)$$

Para dar solución a estas ecuaciones se formula la función de transferencia como una división, de modo que el numerador es la determinante de la ecuación no homogénea, ubicando los coeficientes en la columna de la variable a controlar (cabeceo en este caso).

$$\begin{vmatrix} \text{-1,58627941 s} & 4 & \text{-0,26625298} \\ \text{-0,45602432} & & \text{-0,71} \end{vmatrix} \quad (33)$$

s	1,12625838
N	-2,96141784

Tabla 6 Constantes del numerador de la función de tranferencia

Y el denominador es la determinante de la ecuación homogénea.

$$\begin{vmatrix} 1{,}58627941\ \text{s} & 4 & \text{-1,58627941 s} & \text{-0,076662821} \\ \text{-0,45602432} & & 0{,}00467716\ s^2 & 1{,}48932\text{E-}05 \end{vmatrix} \quad (34)$$

s^3	0,00741928
s^2	0,01873226
s	0,72344157
N	-0,03496011

Tabla 7 Constantes del denominador de la función de tranferencia

Quedando la función de transferencia de la siguiente manera:

$$\frac{\theta(s)}{\delta(s)} = \frac{1{,}1262\ s - 2{,}7185}{0{,}0074\ s^3 + 0{,}0186\ s^2 + 0{,}7233\ s - 0{,}0349} \quad (35)$$

Donde la salida del sistema θ(s) es el ángulo de salida y la entrada δ(s) se refiere a la deflexión de las aletas.

4. Sistema de control

Para el desarrollo del sistema de control es necesario saber inicialmente si la función de transferencia de la planta es inherentemente estable o no. Esto se hace mediante el diagrama de polos y ceros de "Root locus", en donde el plano izquierdo y derecho indican si es o no estable. En el plano izquierdo se deben ubicar los puntos que generan una señal estable, con el fin de que la respuesta al estar más lejos del cero sea más rápida.

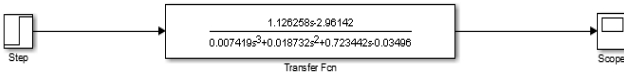


Figura 37 Diagrama de bloques en lazo abierto.

Al realizar una retroalimentación del modelo sin el controlador se obtiene una respuesta a partir de una señal, la cual se muestra a continuación:

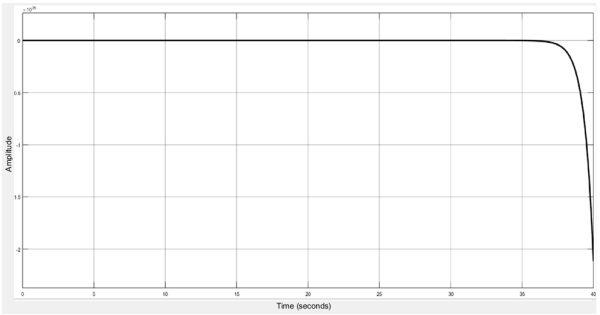


Figura 38 Respuesta al sistema en lazo abierto.

En esta señal se observa un sistema que no es capaz de estabilizarse, ya que tiende a una caída y por lo tanto nunca se llega al valor de entrada (*step*) ingresado.

A continuación se encuentra el diagrama de Root locus del sistema en lazo abierto de la función de transferencia, que fue encontrada anteriormente, donde se relaciona la entrada del sistema junto con la planta y finalmente la respuesta de la señal.

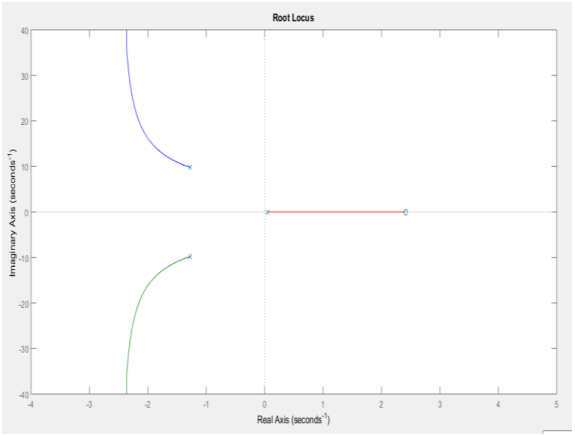


Figura 39 Diagrama Root Locus del sistema en lazo abierto.

En el diagrama Root locus (polos y ceros) de la figura 16 se observa que el sistema (sin controlador) resulta ser inestable, ya que existen ceros en el plano derecho indicando que deben hacerse modificaciones en el diagrama de bloques, esto de tal modo que los polos y ceros se encuentren en el plano izquierdo, y así el sistema sea estable [10].

4.1 Controlador

En la adaptación de un controlador que sea capaz de estabilizar el sistema con la planta se realiza el ajuste de un controlador PID. El principio básico del esquema del control PID es que actúa sobre la variable a ser manipulada por medio de la combinación de tres acciones de control, la primera es donde la acción de control es proporcional a la señal de error actuante, que es la diferencia entre la entrada y la señal de realimentación; la segunda es donde la acción de control es proporcional a la integral de la señal de error actuante, y la tercera es donde la acción de control es proporcional a la derivada de la señal del error actuante [9].

La acción del control PID en controladores analógicos está dado por:

$$m(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right] \qquad (36)$$

Donde $e(t)$ es la entrada al controlador, $m(t)$ es la salida del controlador, K es la ganancia proporcional, T_i es el tiempo integral y T_d es el tiempo derivativo [6].

De acuerdo a lo anterior se genera un lazo de retroalimentación, en donde gracias a la herramienta Tune de *Simulink*®, se obtienen las constantes Kp, Kd y Ki automáticamente, las cuales conforman la función de transferencia del PID [12].

Con las constantes del PID se obtiene el diagrama de bloque en lazo cerrado con la función de transferencia de la planta, así como se muestra a continuación:

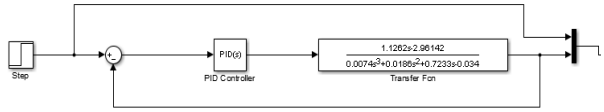


Figura 40 Diagrama de bloque en lazo cerrado con el PID.

La respuesta al sistema en lazo cerrado se obtiene mediante el *scope*, generando la siguiente señal estabilizada.

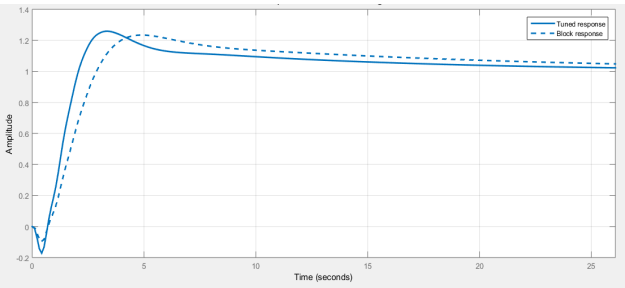


Figura 41 Respuesta del sistema de control.

Donde las ganancias para el controlador se presentan como: proporcional $K_P = -0.2032$, integral $K_i = -0.0159$ y derivativa $K_d = 0.0693$.

Obteniendo un pico de amplitud de 1.27, un margen de ganancia de 8.36dB@2.36rad/s, un margen de fase de 46.3deg@0.803rad/s y un tiempo de estabilización de 15s.

4.2 Discretización del sistema

Actualmente los sistemas de control se manejan en su gran mayoría bajo control digital, ya que estos resultan ser adaptables a microcontroladores y tarjetas de datos, entre otros, siendo más exactos en la respuesta del sistema, teniendo en cuenta que manejan algoritmos complejos, que se ajustan y modifican según las especificaciones de entrada. Es por eso que al ver la necesidad de implementar un sistema digital, para luego ser unificado con un microcontrolador, se debe pasar el sistema que se tiene inicialmente continuo a un sistema discreto, esto por medio de una discretización.

El proceso de discretización es convertir un modelo continuo a un modelo discreto por medio de *Matlab*®. Esta operación se realiza mediante el comando de [Nz,Dz] =c2dm (N,D,Ts,'metodo'), donde se toma en cuenta la función de transferencia del modelo continuo, la cual viene dada por los polinomios numerador y denominador (N y D). Además de esto existe el tiempo de muestreo (Ts) y por último se especifica el carácter con el cual se ejecuta la discretización (método). Dentro de estos métodos se encuentra el método ZOH, el cual realiza la discretización utilizando mantenedor de orden cero que es uno de los más usados. Con el comando dstep(Nz,Dz), se calcula la respuesta temporal de un sistema discreto a una secuencia de escalón, en donde se muestran múltiplos del periodo de muestreo.

Para el proceso de discretización se genera un código en *Matlab*®, el cual se muestra a continuación:

```
num=[1.1262 -2.7185]
den=[0.0074 0.0186 0.7233 -0.0349]
planta=tf(num,den)
plantad=c2d(planta,1,'zoh')
Go=feedback(pidcontinuo*planta,1)
God=feedback(piddiscreto*plantad,1)
step(Go)
axis([0 70 0 1])
hold on
figure
step(God)
```

Este código arroja una función de transferencia en tiempo discreto de la planta y se vuelve a generar un diagrama de bloque con su respectivo PID.

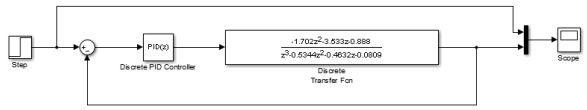


Figura 42 Diagrama de bloque del sistema en tiempo discreto.

Según la respuesta obtenida en el *scope* del lazo cerrado, se genera la gráfica de la figura 19, la cual muestra la señal de respuesta por medio de pulsos.

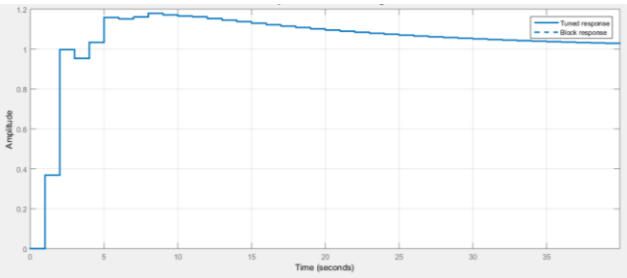


Figura 43 Respuesta del sistema en lenguaje digital.

La figura 20 se encuentra representada en forma digital, lo que quiere decir que los valores son discretos. Este utiliza numeración binaria donde las cantidades se representan utilizando los números 0 o 1, a lo que se llama *bit*. Al mismo tiempo se obtienen las ganancias para el controlador PID que son [13]:

Donde las ganancias para el controlador se presentan como: proporcional $K_P = -0.1329$, integral $K_i = -0.00618$ y derivativa $K_d = 0.0571$.

Obteniendo un valor pico de 1.18, un margen de ganancia de 7.23dB@1.95rsd/s y margen de fase de 64.5deg@0.5rad/s.

5. Conclusiones

Por medio de análisis dinámicos se pudo generar un modelo capaz de estabilizar el cohete en su ángulo de cabeceo, lo cual se puede evidenciar en la simulación del diagrama de bloques de la figura 19, que implementa un controlador PID y en la figura 20 donde se observa como el sistema llega a estabilizarse conforme a un valor de entrada deseado (ángulo cabeceo).

Este sistema de control pretende ser implementado físicamente en futuros proyectos. Es por eso que se buscó determinar qué lenguaje resultaba ser más efectivo para su implementación, implicando así el uso de lenguaje discreto para el control PID, ya que como se mencionó anteriormente, estos sistemas son más utilizados por la industria y generan más exactitud en las variables que manipula. Sin embargo se puede evidenciar que el tiempo de respuesta para los dos sistemas resulta ser un poco prolongado, teniendo en cuenta el tiempo que el cohete tendrá desde su punto cero hasta la altura del apogeo, por esta razón es necesario adicionar otro controlador que pueda modificar o mejorar ese tiempo de respuesta.

No obstante el mecanismo del sistema de control se debe ubicar en las aletas, en la cuerda de raíz donde se ubica el centro de presión. Según los análisis expuestos anteriormente, este sistema deberá estar conformado por cuatro servos cada uno para una aleta respectivamente, una plataforma que tenga un entorno que implemente el lenguaje de programación en este caso un arduino. También es necesaria una unidad de medición inercial (*IMU*) la cual detecta la tasa de aceleración usando acelerómetros, además de cambios en el cabeceo, guiñada y alabeo usando uno o más giróscopos y una batería que se encarga de alimentar todo el sistema. Para que exista una transferencia de información y energía entre el arduino, los servos, el IMU y la batería se usan cables.

De acuerdo al análisis dinámico se deben tener en cuenta los datos de entrada y salida, así como las especificaciones de vuelo del cohete, ya que a partir de estos se genera las ecuaciones de movimiento, para luego determinar la función de transferencia, la cual será la planta del sistema que tendrá lugar en *Simulink*®, para desarrollar el lazo cerrado junto con el controlador y así poder generar la señal de respuesta.

Referencias

[1] Blakelock, J. (1991). Automatic control of Aircraft and Missiles. A Wiley-Interscience publication, pp. 60-272.

[2] George, M. (2004). Missile Guidance and Control Systems. Springer-Verlag New York, pp. 24.

[3] Apogee Components (2016). Rocksim Information. https://www.apogeerockets.com/Rocksim/Rocksim_information

[4] Nakka R. (2001) Teoría Sobre Motores Cohete De propelente Sólido. http://www.nakka-

rocketry.net/articles/teoria_de_los_motores_cohete.pdf
[5] Barrowman, J. (2010). Calculating the Center of Pressure of a Model Rocket. Estes-Cox.
[6] Ogata, K. (1996). Sistemas de control en Tiempo Discreto. Prentice Hall Hispanoamericana s.a.
[7] Fink, R. Macdonnell Douglas Corporation. (1978). USAF Stability and control datcom. Long Beach, California.
[8] Bandu, N. (1998). Performance, stability, dynamics and control of airplanes. American Institute of Aeronautics and Astronautics.
[9] Ogata, K. (2010). Ingenieria de control Moderna. Pearson educación, s.a., madrid.
[10] Vargas, M. (2005). Tutorial de análisis y control de sistemas usando matlab. http://www.esi2.us.es/~vargas/docencia/cpc/guiasMatlab/tutorialControlToolbox.pdf
[11] Millign, T. (2012). Determining the Drag Coefficient Of A Model Rocket Using Acclerometer Based Payloads. https://www.apogeerockets.com/downloads/Cd_determination_using_Coast_Data.pdf
[12] Norman, S. (2011). Control System Engineering. USA. John Wiley.
[13] Sistema digital. (2009). http://www.profesormolina.com.ar/electronica/componentes/int/sist_digi.htm

Anexo 3. Programación para la primera opción de la interfaz análisis de los parámetros del cohete.

```
function varargout = guide3(varargin)
% GUIDE3 MATLAB code for guide3.fig
%      GUIDE3, by itself, creates a new GUIDE3 or raises the existing
%      singleton*.
%
%      H = GUIDE3 returns the handle to a new GUIDE3 or the handle to
%      the existing singleton*.
%
%      GUIDE3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIDE3.M with the given input arguments.
%
%      GUIDE3('Property','Value',...) creates a new GUIDE3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guide3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guide3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guide3

% Last Modified by GUIDE v2.5 29-Jul-2016 23:05:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guide3_OpeningFcn, ...
                   'gui_OutputFcn',  @guide3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before guide3 is made visible.
function guide3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to guide3 (see VARARGIN)

% Choose default command line output for guide3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guide3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = guide3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
m=handles.masa;
e=handles.empuje;
tb=handles.edit3;
vparac=handles.edit4;
a=e/m;
vf=a*tb;
h1=0.5*a*(tb)^2;
t2=vf/9.8;
h2=vf*t2-0.5*9.8*(t2)^2;
tapogeo=tb+t2;
t3=((h1+h2)/vparac);
H=h1+h2;
tt=tb+t2+t3;
set(handles.text22,'String',a);
set(handles.text12,'String',vf);
set(handles.text14,'String',H);
set(handles.text16,'String',tapogeo);
axes(handles.axes1);
%grafica altitud vs tiempo
T1=linspace(0,tb,100);
d1=(0.5*a).*(T1.^2);
plot(T1,d1,'ro');
hold on
axes(handles.axes1);
T2=linspace(tb,t2+tb,100);
T22=linspace(0,t2,100);
d2=h1+(vf.*T22)-(0.5*9.8).*(T22.^2);
plot(T2,d2,'gx');
hold on
axes(handles.axes1);
```

```
T3=linspace(t2+tb,tt,100);
T33=linspace(0,t3,100);
d3=(h2+h1)-vparac.*T33;
plot(T3,d3,'b*');
grid on
%grafica de y versus tiempo
xlabel('tiempo');
ylabel('altura maxima');
title('Altitud');
axes(handles.axes3);
%velocidad tiempo
V1=a.*T1;
plot(T1,V1,'ro');
hold on
V2=vf-9.8.*T22;
plot(T2,V2,'gx');
hold on
V3=ones(1,100)*(-vparac);
plot(T3,V3,'b*');
grid on
 %grafica de velocidad versus tiempo
xlabel('tiempo');
ylabel('velocidad');
title('Velocidad');
function masa_Callback(hObject, eventdata, handles)
% hObject    handle to masa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.masa=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of masa as text
%        str2double(get(hObject,'String')) returns contents of masa as a
double
% --- Executes during object creation, after setting all properties.
function masa_CreateFcn(hObject, eventdata, handles)
% hObject    handle to masa (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function empuje_Callback(hObject, eventdata, handles)
% hObject    handle to empuje (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.empuje=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of empuje as text
%        str2double(get(hObject,'String')) returns contents of empuje as a
double
```

```
% --- Executes during object creation, after setting all properties.
function empuje_CreateFcn(hObject, eventdata, handles)
% hObject    handle to empuje (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit3=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit4=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
botox=uicontrol('Style','pushbutton','Units','normalized','String','EMPEZAR','
Callback','clear all;close all;clc;guide2;');
```

Anexo 4. Programación para la segunda opción de la interfaz de análisis dinámico y la función de transferencia.

```
function varargout = guidefinal(varargin)
% GUIDEFINAL MATLAB code for guidefinal.fig
%      GUIDEFINAL, by itself, creates a new GUIDEFINAL or raises the existing
%      singleton*.
%
%      H = GUIDEFINAL returns the handle to a new GUIDEFINAL or the handle to
%      the existing singleton*.
%
%      GUIDEFINAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIDEFINAL.M with the given input arguments.
%
%      GUIDEFINAL('Property','Value',...) creates a new GUIDEFINAL or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guidefinal_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guidefinal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guidefinal

% Last Modified by GUIDE v2.5 30-Jul-2016 15:55:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guidefinal_OpeningFcn, ...
                   'gui_OutputFcn',  @guidefinal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT
% --- Executes just before guidefinal is made visible.
function guidefinal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guidefinal (see VARARGIN)
% Choose default command line output for guidefinal
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guidefinal wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = guidefinal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit1=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit2=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
% --- Executes during object creation, after setting all properties.
```

```
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit3=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit4=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit5=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit6=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit7=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit8=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit9=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit10=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit11=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double



% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit12=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a
double
% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit13=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a
double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edti14=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a
double
% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%variables de entrada
m=handles.edit15;
vm=handles.edit16;
area=handles.edit17;
densidad=handles.edit18;
angulo=handles.edit22;
lt=handles.edit28;
cmde=handles.edit27;
n=handles.edit23;
cr=handles.edit19;
ct=handles.edit20;
b=handles.edit21;
angulole=handles.edit24;
iy=handles.edit26;
dcohete=handles.edit25;
Mach=handles.edit29;

%presion dinamica
q=((1/2)*(densidad*vm^2))
%E42
musq=((m*vm)/(area*q))
%E60
czalpha=-4
%E49
cw=0.07695
radianes=(angulo*pi())/180
cwsen=cw*sin(radianes)
%taper ratio
taperr=ct/cr
%cuerda media
c=(2/3)*(cr)*((1+taperr+(taperr^2))/(1+taperr))
%E64
czde=((c/lt)*(cmde))
```

```
%aspect ratio
aspectr=((2*b)/(cr*(1+taperr)))
%distancia borde de ataque a centro aerodinamico
xacr=((0.666*(1-taperr))+(0.5*(1-
(taperr^2/1+taperr))*(pi()*log(1+aspectr/5)))/(1+(pi()*log(1+aspectr/5))))
%crc
crc=cr/c
radangulole=(angulole*pi())/180
%tanle
tanle=tan(radangulole)
%pendiente curva de sustentacion
clalpha=8*atan(pi()*aspectr/(16+(pi()*aspectr)/(1+2*taperr*tanle)))
%coeficiente momento de cabeceo
cmalpha=(n-xacr)*(crc)*(clalpha)
%iy/sqd
iysqd=((iy)/(area*q*dcohete))
%25de la cuerda
porc=(c*25)/100
%ymac
ymac=(b/6)*(1+2*taperr)/(1+taperr)
%E33
xc=xacr-ymac
%tan al cuadro
tanporc=tan(porc)^2
%amortiguacion cabeceo
cmq1=-
0.7*clalpha*cos(porc)*((aspectr*(0.5*xc+(2*(xc)^2))/(aspectr+2*cos(porc))))+(0
.04166*(((aspectr)^3*tanporc)/(aspectr+6*cos(porc)))+(0.125))
%coseno del porcentaje
cosporc=cos(porc)
%beta
B=sqrt(1-(Mach^2)*((cosporc)^2))
%E57
cmq2=((aspectr^3*tanporc/(aspectr*B+(6*cosporc))+3/B)/((aspectr^3*tanporc)/(as
pectr+(6*cosporc))+3))*cmq1
%E41
ducmq=(dcohete/(2*vm))*(cmq2)
%s^3
stres=musq*iysqd
%s^2
sdos=(musq*-ducmq)+(-czalpha*iysqd)
%s denominador
sden=(-czalpha*-ducmq)+(cmalpha*musq)
%N denominador
nden=-(-cmalpha*-cwsen)
%s numerador
snum=(-musq*cmde)
%N numerador
nnum=(-czalpha*cmde)-(cmalpha*czde)

set(handles.text58,'String',snum);
set(handles.text59,'String',nnum);
set(handles.text60,'String',stres);
set(handles.text61,'String',sdos);
set(handles.text62,'String',sden);
set(handles.text63,'String',nden);
axes(handles.axes4);
```

```
num=[snum nnum]
den=[stres sdos sden nden]
S=tf(num,den)
step(S)
handles.snum=snum;
guidata(hObject,handles);
handles.nnum=nnum;
guidata(hObject,handles);
handles.stres=stres;
guidata(hObject,handles);
handles.sdos=sdos;
guidata(hObject,handles);
handles.sden=sden;
guidata(hObject,handles);
handles.nden=nden;
guidata(hObject,handles);

function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit27=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27 as a
double

% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit28_Callback(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit28=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit28 as text
%        str2double(get(hObject,'String')) returns contents of edit28 as a
double

% --- Executes during object creation, after setting all properties.
function edit28_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit15=NewVal;
guidata(hObject,handles);

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit16=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a
double

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit17=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a
double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit18=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a
double

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit19=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a
double

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit20=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a
double

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit21_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit21=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a
double

% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit22=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a
double

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
```

```
handles.edit23=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a
double

% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit24=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a
double

% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit25=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit26=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26 as a
double

% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit29=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit29 as text
%        str2double(get(hObject,'String')) returns contents of edit29 as a
double

% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function text58_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text58 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes4
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes7);
kp=handles.edit31;
ki=handles.edit32;
kd=handles.edit33;
snum=handles.snum;
nnum=handles.nnum;
stres=handles.stres;
sdos=handles.sdos;
sden=handles.sden;
nden=handles.nden;
axes(handles.axes7);
num=[snum nnum]
den=[stres sdos sden nden]
S=tf(num,den)
control=pid(kp,ki,kd)
Go=feedback(control*S,1)
step(Go)

function edit31_Callback(hObject, eventdata, handles)
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit31=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit31 as text
%        str2double(get(hObject,'String')) returns contents of edit31 as a
double

% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit32=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32 as a
double


% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit33_Callback(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Val=get(hObject,'string');
NewVal=str2double(Val);
handles.edit33=NewVal;
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of edit33 as text
%        str2double(get(hObject,'String')) returns contents of edit33 as a
double


% --- Executes during object creation, after setting all properties.
function edit33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function axes7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes7

% --- Executes during object creation, after setting all properties.
function axes12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
axes(hObject)
imshow('linea.png')
% Hint: place code in OpeningFcn to populate axes12
```

Anexo 5. Guia en Matlab para simular el sistema de control

El diseño de la interfaz gráfica de usuario de Matlab para la simulación de un sistema de control para un cohete balístico, no teledirigido, se realizó utilizando GUIDE, el cual es un entorno de programación que realiza y ejecuta programas que requieran el ingreso de datos.

En esta guía el usuario podrá ejecutar dos simulaciones, la primera se basa en la obtención de los parámetros de un análisis preliminar del cohete que se desea diseñar y la segunda opción se basa en un análisis dinámico donde por medio del ingreso de datos acerca del cohete se obtiene una función de transferencia.

Para dar inicio al GUIDE se debe ingresar al programa Matlab, después:

Ejecutar en la ventana de comandos la palabra “guide”.

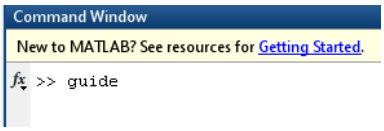


Figura 44. Ventana de comando Matlab

Al dar ENTER al comando “guide” se presenta el siguiente cuadro de diálogo:

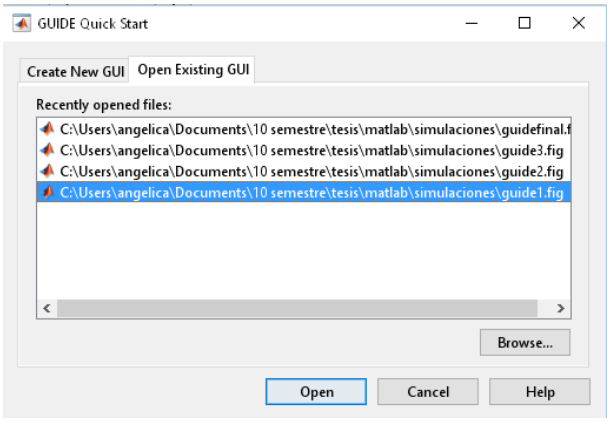


Figura 45. Cuadro de inicio del GUIDE

En donde se debe seleccionar el “guide1” y luego oprimir el botón “Open”.

Al ejecutar Open se abre la siguiente ventana:

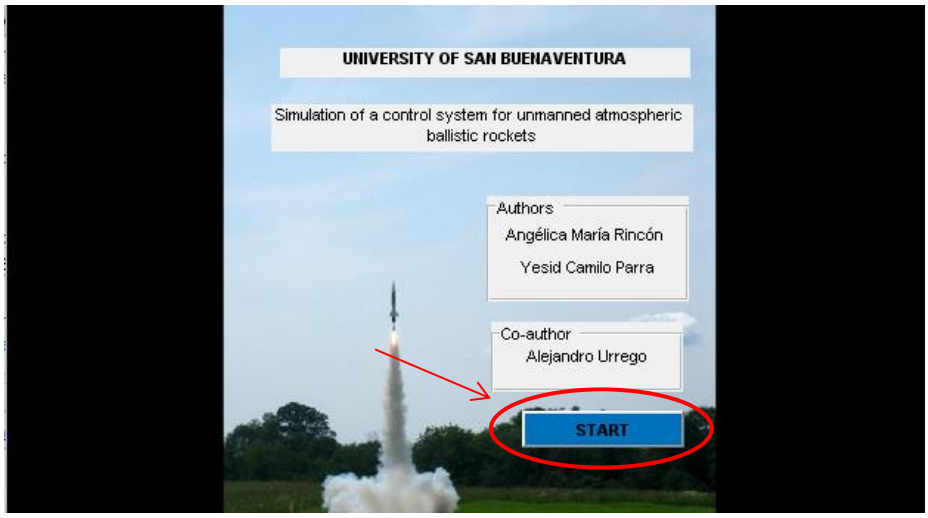


Figura 46. Ventana de inicio de la simulación

En esta se muestra información inicial sobre simulación, como lo es el lugar donde fue desarrollado y sus respectivos autores.

Para empezar el usuario debe oprimir el botón “START”, en donde se abrirá la siguiente ventana:

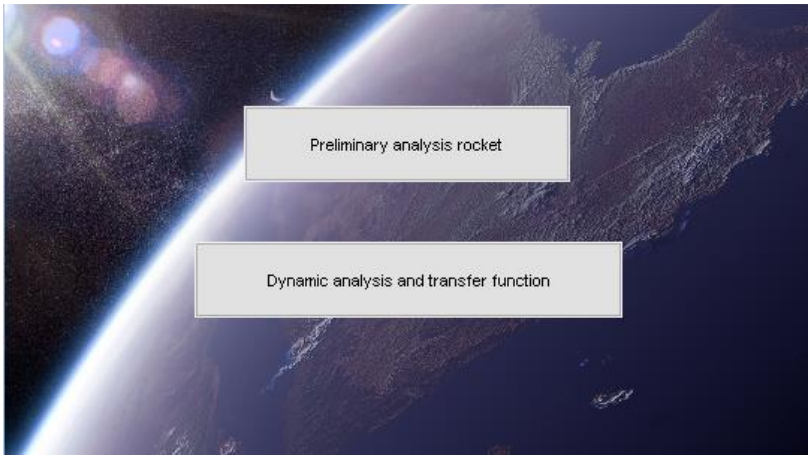


Figura 47. Ventana de las dos opciones para el usuario

En esta ventana el usuario podrá seleccionar 2 opciones, para quienes deseen determinar los parámetros principales de su cohete deben oprimir el primer recuadro “Preliminary analysis

rocket" que genera un icono llamado "OPTION 1", y para los usuarios que desean un análisis para desarrollar su sistema de control, debe oprimir el recuadro "Dynamic analysis and transfer function" que genera un icono llamado "OPTION 2".

En la siguiente imagen se muestra el icono generado cuando se selecciona la primera opción (primer recuadro).

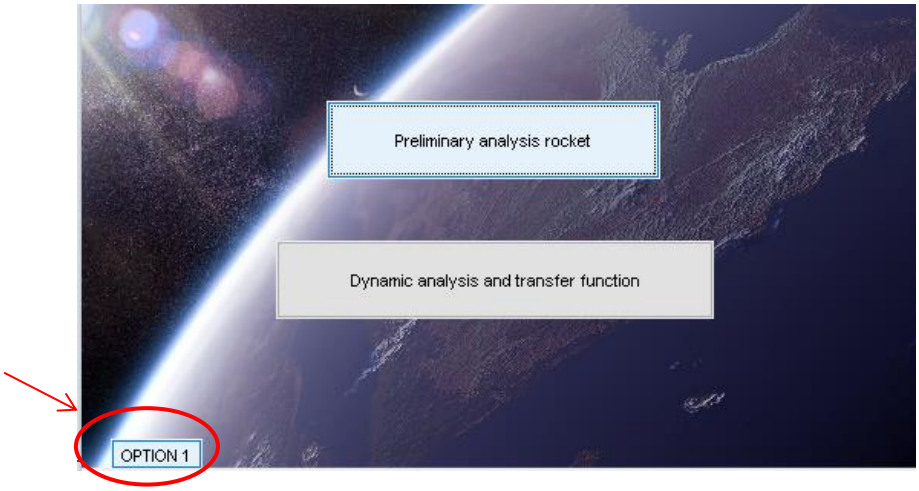


Figura 48. Ventana señalando la primera opción

En el caso de haber seleccionado la primera opción (primer recuadro) aparecerá la siguiente ventana:

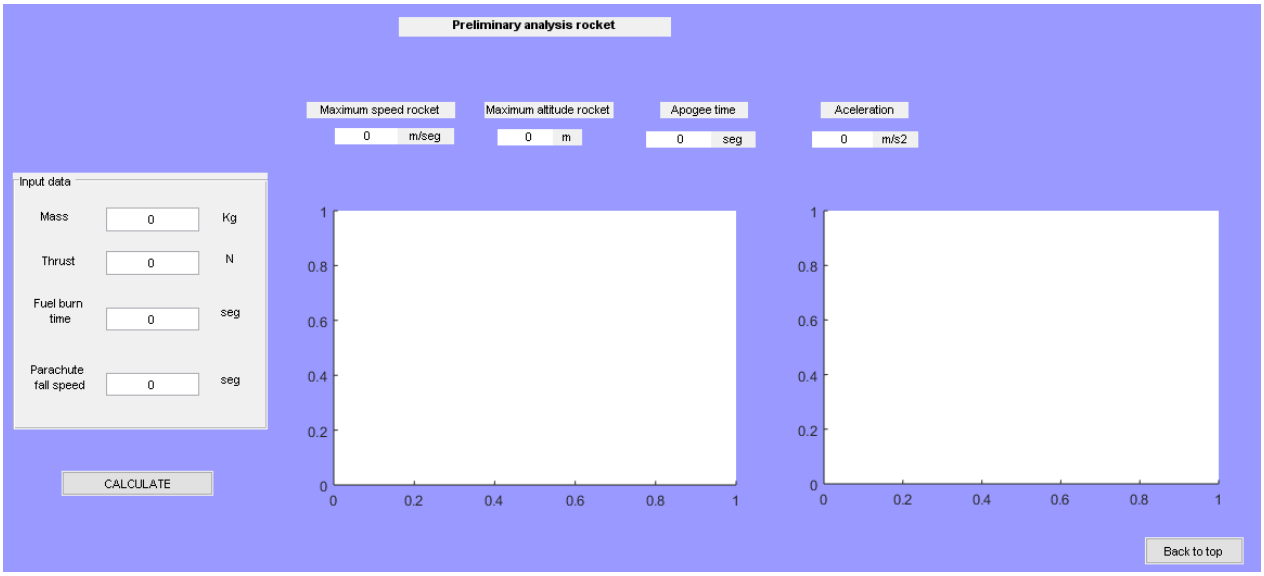


Figura 49. Ventana de análisis preliminar del cohete

En esta pantalla el usuario debe ingresar los datos de entrada que se piden respecto al cohete que este quiere desarrollar, estos son la masa del cohete, el empuje promedio que genera el combustible, el tiempo de quemado del combustible y por último la velocidad de caída del paracaídas, se debe recalcar que es necesario que se ingresen todos los datos para que se pueda calcular.

Después de ingresar todos los datos de entrada se oprime el botón "CALCULATE" y seguidamente se genera la respuesta de los siguientes parámetros: la velocidad máxima del cohete, la altura máxima alcanzada del cohete, el tiempo de apogeo y la aceleración, así mismo se generan automáticamente dos gráficas, la gráfica de la izquierda se encuentra la relación entre altura versus tiempo y la gráfica de la derecha está la relación velocidad versus tiempo.

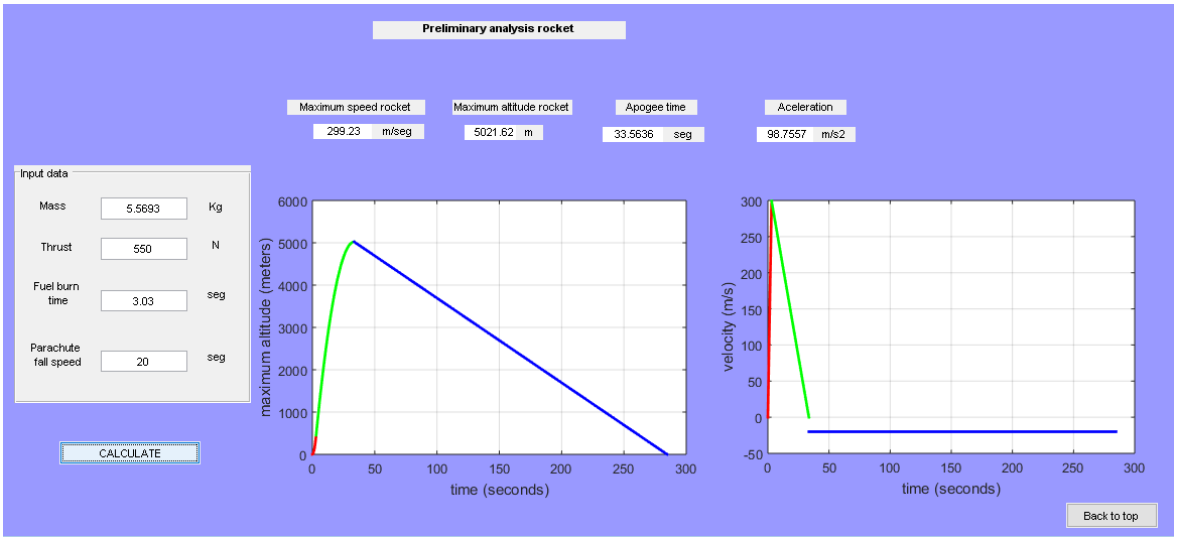


Figura 50. Ventana con datos ingresados y muestra de resultados para la primera opción

Para los usuarios que deseen regresar a la pantalla de opciones y elegir la segunda opción, deben seleccionar "Back" y oprimir el segundo recuadro "Análisis dinámico y función de transferencia" para luego oprimir el botón generado, como se observa en la siguiente figura:

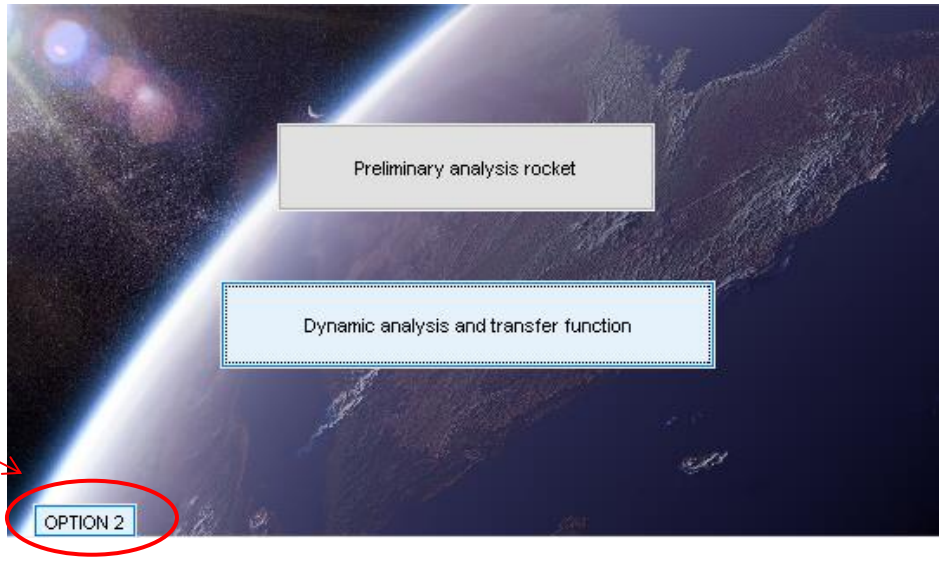


Figura 51. Ventana señalando la segunda opción

Al oprimir el botón “OPTION 2” aparece la siguiente pantalla:

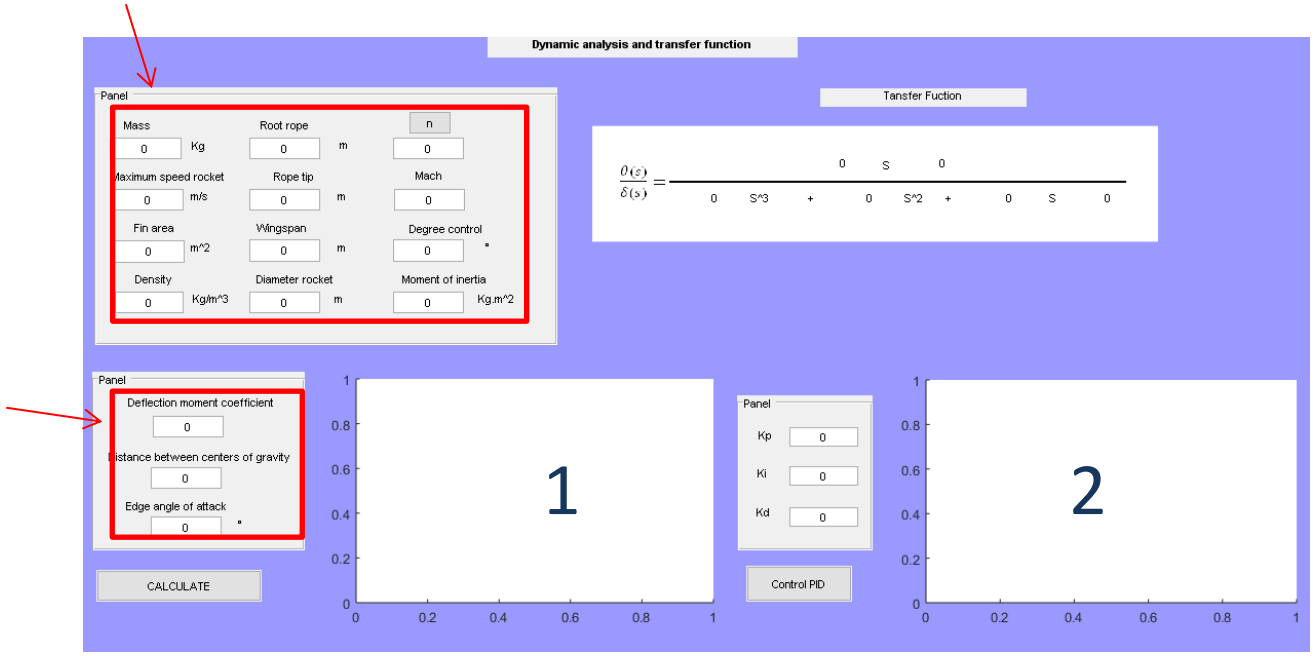


Figura 52. Ventana dela análisis dinámico y función de transferencia

En esta se encuentran resaltados en el recuadro rojo los datos necesarios para poder hallar la función de transferencia, dentro del valor que está representado como n se encuentra un mensaje que indica su significado, como se aprecia en la siguiente figura.

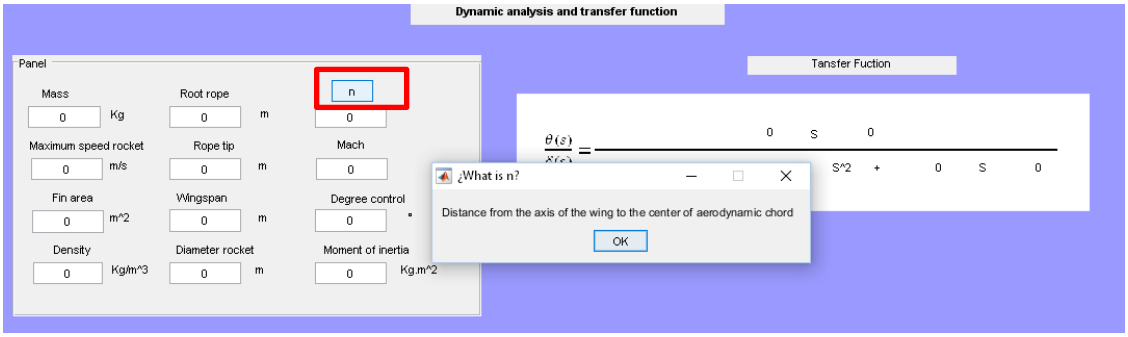


Figura 53. Aviso del significado de n

Después de que el usuario haya insertado todos los valores indicados, se debe oprimir el botón "CALCULATE", el cual a partir de la programación interna de ecuaciones generará los valores de la función de transferencia y en el cuadro 1 la señal de respuesta de esta, que se encuentra dada por la relación entre amplitud y el tiempo, tal como se observa en la siguiente figura:

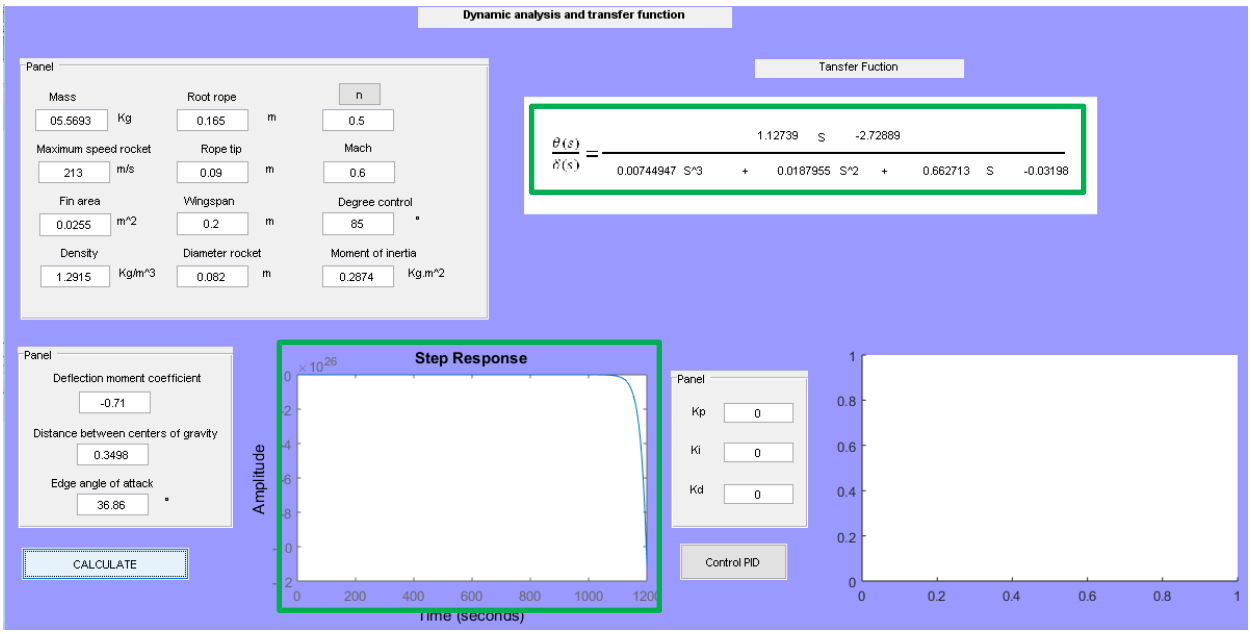


Figura 54. Ventana con datos ingresados y muestra de resultados para la segunda opción

Luego de obtener la primera gráfica, se requiere que el usuario introduzca los valores de las ganancias proporcional, integral y derivativo de su controlador PID, después, al oprimir el

botón “Control PID” se generará la gráfica de la señal ya estabilizada donde se podrá visualizar el tiempo de respuesta de ese sistema de control, de allí se podrá saber si ese sistema es adecuado para la misión que se quiere realizar o si por el contrario es necesario cambiar alguno de los datos introducidos relacionados con el diseño.

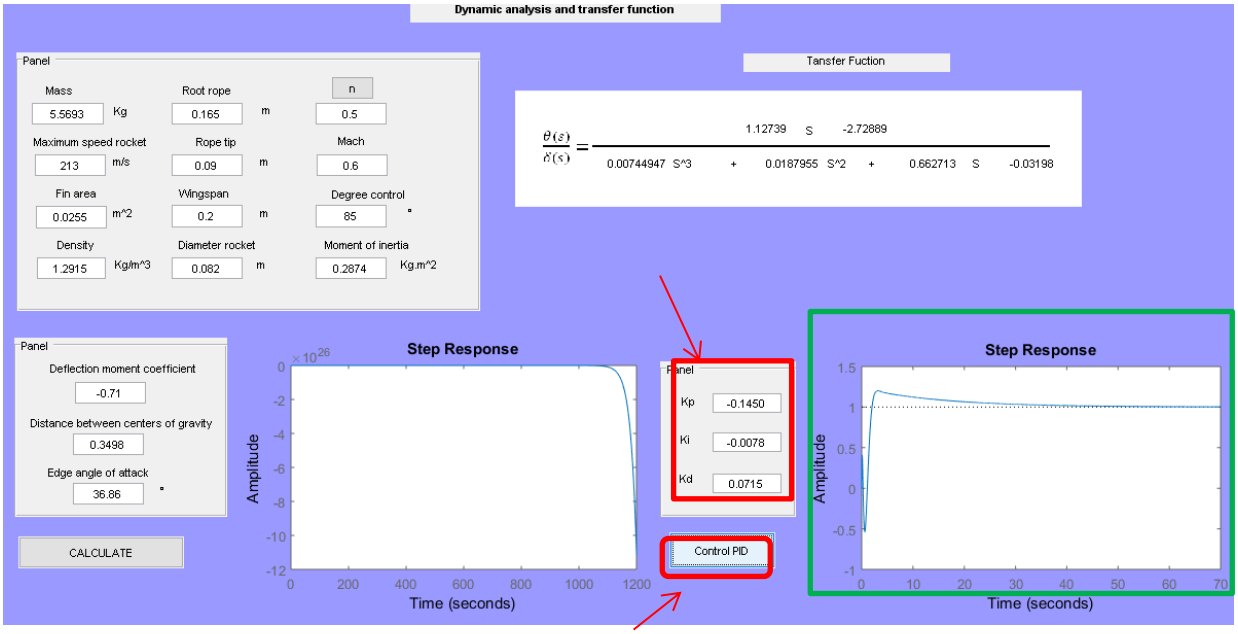


Figura 55. Ingreso de los datos del controlador y respuesta al sistema

BIBLIOGRAFÍA

[1] John D. Anderson Jr. Fundamentals of Aerodynamics McGraw Hill Science Engineering Math 2010.pdf, página 391.

[2] Siefert, R. Dedign of an autopilot for small unmanned aerial vehicles. 2004. [En línea]. Disponible en: http://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=1042&context=etd

[3] Honeywell. Vehicle health management.2002. [En línea]. Disponible en: http://www51.honeywell.com/aero/technology/common/documents/defense_space_management_brochure.pdf

[4] Mutter, F. Gareis, S. Model driven in the loop validation: simulation based testing of UAV software using virtual envirotments. 2011. [En línea]. Disponible en: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5934784

[5] Buendia, L. Lezama, R. Delgado, D. F-16 Autopilot design. 2011. [En línea]. Disponible en: http://users.encs.concordia.ca/~ymzhang/courses/MECH480-6091/CourseProjects&Labs/MECH6091/F2011/MECH6091_Presentation_Group1.pdf

[6] Johansen, I. Autopilot design for unmanned aerial vehicles. 2012. [En línea]. Disponible en: https://brage.bibsys.no/xmlui/handle/11250/260645

[7] Goyal, S. Study of a Longitudinal Autopilot For Different Aircrafts. [En línea]. Disponible en:. http://plaza.ufl.edu/siddgoya/Homepage/Publications_files/college_Project.pdf

[8] George, M. "Missile Guidance and Control Systems," Springer-Verlag New York, Inc 2004, pp. 24.

[9] Apogee Components (2016). Rocksim Information. [En linea]. Disponible en: https://www.apogeerockets.com/Rocksim/Rocksim_information

[10] Omar, D. Balística externa. [En línea]. Disponible en: http://clasev.net/v2/pluginfile.php/75852/mod_resource/content/1/balistica%20externa.pdf

[11] Escuela de Ingeniería Mecánica – ULA. Teoria de control . [En línea]. Disponible en: http://webdelprofesor.ula.ve/ingenieria/djean/index_archivos/Documentos/TC12_Intro_Sistemas_Digitales.pdf

[12] Blakelock, J. "Automatic control of Aircraft and Missiles," A Wiley-Interscience publication, (1991). pp. 60-272.

[13] Ibáñez, F. Marlon R. Daniel, R. Diseño y construcción de una misión de cohetería experimental con propelente tipo sólido para alcances estratosféricos. Universidad de San Buenaventura. 2016.

[14] Barrowman, J. "Calculating the Center of Pressure of a Model Rocket," Estes-Cox Corp 2012.

[15] Ogata, K "Sistemas de control en Tiempo Discreto," Prentice Hall Hispanoamericana s.a. (1996).

[16] Fink, R. "USAF Stability and control datcom," Macdonnell Douglas Corporation. 1978 Long Beach, California.

[17] Bandu, N. "Performance, stability, dynamics and control of airplanes," American Institute of Aeronautics ans Astronautics, Inc (1998).

[18] Ogata, K. "Ingenieria de control Moderna," Pearson educación, s.a., madrid, 2010.

[19] Vargas, M. "Tutorial de análisis y control de sistemas usando matlab," Este tutorial está basado en un trabajo original de: Manuel Berenguel Soria y Teodoro Álamo Cantarero (2005).

[20] Norman, S. "Control System Engieenering". John Wiley & Sons, Inc (2011, 2006, 2003, 1996). Sistema digital. [En linea]. Disponible en: http://www.profesormolina.com.ar/electronica/componentes/int/sist_digi.htm

[21] Incaes Aerospace, Empresa del conocimiento en ciencia, ingeniería, tecnología e industrial del campo aeroespacial, Antioquia-Colombia, [En linea]. Disponible en: http://ingesaerospace.blogspot.com/.

[22] Pirateque Bolívar, Marco & Esteban Savogal Aldo. Historia preliminar de la cohetería en Colombia. 26 de diciembre de 2011. Pág. 10 -18.

[23] MathWorks, 1994. Aerospace Blockset. [En linea]. Disponible en: http://www.mathworks.com/products/aeroblks/?refresh=true.

[24] Sistemas de control, 2012. [En linea]. Disponible en: http://www.isa.cie.uva.es/~felipe/docencia/ra12itielec/tema1_trasp.pdf

[25] Control-PID. Control PID, metodología y aplicaciones, 2016. [En linea]. Disponible en:de http://control-pid.wikispaces.com/

[26] Apogee components. Rocksim information, 2014. [En linea]. Disponible en:https://www.apogeerockets.com/Rocksim/Rocksim_information

[27] Virginia Mazzone. Controladores PID, 2002. [En linea]. Disponible en: http://www.eng.newcastle.edu.au/~jhb519/teaching/caut1/Apuntes/PID.pdf

[28] Artículo, "Misiones de cohetería experimental con propelente solido: misión Seneca, cohete AINKAA" José Alejandro Urrego, Fabio Arturo Rojas, 2010. [En linea]. Disponible en: http://www.scielo.org.co/scielo.php?pid=S0124-81702010000200004&script=sci_arttext.

[29] Misiones de cohetería experimental con propelente solido: misión Seneca VII, cohete AINKAA V" José Alejandro Urrego, 2015. [En linea]. Disponible en: https://pua.uniandes.edu.co/doku.php?id=misiones:mision14.

[30] "Rocket Roll Dynamics and Disturbance – Minimal Modeling and System Identification", Publicacion, Christopher E Hann, Alex Keall, XiaoQi Chen y otros. Universidad de Canterbury, New Zealand, 2009.

[31] Trabajo de grado de Maestría en Ingeniería Aeroespacial, "Control activo del movimiento roll para un cohete", 2012, Ing. Adriano Arcadipane, Universidad Degli Di Palermo de Italia.

[32] Barragán, O. 2010, "Manual de interfaz grafica de usuario en matlab, parte I", [En linea]. Disponible en: https://www.dspace.espol.edu.ec/bitstream/123456789/10740/19/%255Bmatlab%255D_MATLAB_GUIDE.pdf

[33] Barco, R. Discretización de señales. Agosto 2013. [En linea]. Disponible en: http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-RodrigoBarco.pdf

[34] Nakka, R. Teoría Sobre Motores Cohete De propelente Sólido, 1997. [En linea]. Disponible en: www.nakka-rocketry.net

[35] Victoria, Poveda, S, O (2012) Misión séneca III: lanzamiento de un cohete balístico multietapa con combustible tipo candy. Bogotá 2012. Universidad de los Andes. Facultad de Ingeniería. Departamento de Ingeniería Mecánica.

[36] Nasa Aeronautics and Space Administration. Aerodynamic center 2015. [En linea]. Disponible en: https://www.grc.nasa.gov/www/k-12/airplane/ac.html

[37] Rocktreviews.com. Center of pressure CP 2014. [En linea]. Disponible en: http://www.rocketreviews.com/what-is-cp.html

[38] Discretización de funciones de transferencias, capitulo 2. Universidad Popular del Cesar. [En linea]. Disponible en: http://es.slideshare.net/davinso1/unidad-3-c2control2-1

[39] Aponte, J. Amaya, D. Rubiano, A. Prada, V. 2010. Modelado, diseño y construcción de un sistema activo de control de estabilidad de bajo costo para cohetes experimentales tipo aficionado. Universidad Militar Nueva Granada.

[40] Arias, L. Gonzalez, J. 2015. Implementacion de un simulador de vuelo para un vehiculo aereo no tripulado para el monitoreo de cultivos. Universidad de San Buenaventura.

[41] Conversor Analógico – Digital (A/D), ARDUINO. 2016. [En linea]. Disponible en: http://playground.arduino.cc/ArduinoNotebookTraduccion/Appendix6.

[42] MathWorks, 1994. Design Compensator using automated PUD tuning and graphical bode design. [En linea]. Disponible en: http://www.mathworks.com/help/slcontrol/ug/design-compensator-in-simulink-using-automated-pid-tuning.html.

[43] Aliasing. 2009. [En linea]. Disponible en: http://zone.ni.com/reference/en-XX/help/370051M-01/cvi/libref/analysisconcepts/aliasing/