Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2613 - Control Automático

Proyecto: Control de cohete

Catalina Sarmiento

Introducción

Para el presente informe, se busca encontrar un controlador en MATLAB adecuado para un cohete basado en el cohete Falcon 9 de la empresa SpaceX, utilizando las ecuaciones entregadas por los ayudantes. Como estas ecuaciones son no lineales, y cada entrada afecta a cada salida del sistema, se procedió a sintonizar el controlador por lazos separados. Debido a la alta sensibilidad a los pequeños cambios de fuerza entregada por el thruster del cohete, se decidió hacer varias pruebas del controlador acoplado, en vez de analizar la respuesta del cohete por cada coordenada esférica. De esa forma, no es necesario buscar los parámetros utilizados finalmente en un proceso de optimización separado.

Teoría del controlador base

El controlador debe ser del tipo discreto, ya que MATLAB trabaja con vectores, se utilizaron como base las ecuaciones de controlador PD y PID discretos vistas en clases.
Controlador PD:

$$u_k = (K_p + K'_d)e_k - K'_d e_{k-1} \tag{1}$$

$$K'_d = \frac{K_d}{\triangle T} \tag{2}$$

Para obtener una mayor responsividad del controlador, necesaria para controlar un cohete que va a alta velocidad, se utiliza $\triangle T = 0.1$:

$$K'_d = 10K_d \tag{3}$$

$$u_k = (K_p + 10K_d)e_k - 10K_d e_{k-1} \tag{4}$$

Con un desarrollo análogo, y considerando el tiempo de muestreo distinto de 1, se obtiene la ecuación para un controlador PID discreto:

$$u_k = u_{k-1} + (K_p + 10K_d + K_i)e_k - (K_p + 20K_d)e_{k-1} + 10K_d e_{k-2} \tag{5}$$

Como la simulación parte desde el tiempo 0, se deben considerar condiciones de borde:

- En el tiempo 0, el controlador no actúa. El módulo del modelo del cohete calcula el primer valor de la fuerza en cada coordenada esférica.
- Cuando el tiempo es mayor a 0: Se comienza a calcular los errores y se almacenan en variables auxiliares que cambian en cada iteración de la simulación
- Cuando el tiempo es 1 tiempo de muestreo: Si el controlador es un PD sólo se considera:

$$u_k = (K_p + 10K_d)e_k \tag{6}$$

  Y el controlador es PID tiene la siguiente forma:

$$u_k = (K_p + 10K_d + K_i)e_k \tag{7}$$

- Cuando el tiempo es 2 tiempos de muestreo: El controlador PD actúa con todos sus términos, debido a que solo depende del valor inmediatamente anterior del error. En cambio, el controlador PID aún carece de un término, puesto que ese depende de los errores de dos iteraciones anteriores:

$$u_k = u_{k-1} + (K_p + 10K_d + K_i)e_k - (K_p + 20K_d)e_{k-1} \tag{8}$$

- En el caso en que el tiempo es mayor a 2 tiempos de muestreo, el controlador PID actúa con todos sus términos.

Para encontrar las constantes de cada controlador, por cada lazo, se comienza utilizando un controlador PD, ya que las constantes Kd y Kp son las encargadas de llevar la salida lo más cercana a la referencia en el menor tiempo posible. Luego, se cambia el controlador PD por un controlador PID para reducir errores de baja frecuencia, pero solo en el lazo del control del thruster. Los lazos de elevación y azimut no requieren el término integral, pues no enfrentan tanta inercia como sí lo hace el lazo del thruster.
Es por esto último que también se agrega prealimentación al lazo del thruster y al de elevación, equivalente a la energía potencial que posea en aquel momento.

I. Parte: Llegada a punto de lanzamiento de la carga

1. Setpoints en coordenadas esféricas

Se calculan los setpoints transformando las coordenadas cartesianas del punto donde se debe soltar la carga a sus equivalentes esféricas. Debido a que el cohete está en movimiento constante, y se ha formulado en coordenadas relativas, se considera el cálculo de estos setpoints de forma incremental. Esto quiere decir que se calcula la coordenada esférica necesaria para llegar al punto meta respecto a la posición inmediatamente anterior del cohete.
La única que no se vuelve coordenada esférica es la altura del cohete:

$$Altura_SP = z_k - z_{k-1} \tag{9}$$

Las coordenadas esféricas angulares tienen asociados los siguientes setpoints:

$$\phi_{SP} = atan(\frac{\sqrt{((x_{SP} - x_{k-1})^2 + (y_{SP} - y_{k-1})^2)}}{(z_{SP} - z_{k-1})}) \tag{10}$$

$$\theta_{SP} = atan(\frac{(y_{SP} - y_{k-1})}{(x_{SP} - x_{k-1})}) \tag{11}$$

Para el setpoint de la coordenada Phi, este se ajusta para quedar en el rango de 0 a 2π, sea negativo, se le suma $\pi/2$. En caso contrario, se lo resta a $\pi/2$. Esto es por la definición de coordenadas de elevación dada en el enunciado.

2. Condiciones iniciales utilizadas en los experimentos

Corresponden a un cohete partiendo del reposo en 90 grados respecto a la superficie terrestre. Este cohete tiene una masa de 2000 kg, dentro de la que va considerada su carga de 300 kg. El vector de condiciones iniciales es el siguiente:

$$\begin{bmatrix} Altura_z & Velocidad_z & Masa & Ang_\phi & Ang_\theta & VelAng_\phi & VelAng_\theta & Delay_\phi & Delay_\theta \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} 0 & 0 & 2000 & 1.571 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

Ang es abreviación de la palabra "Ángulo", mientras que Vel es abreviación de la palabra "Velocidad". Las condiciones iniciales son las mismas dadas en el enunciado del proyecto y en la ayudantía asociada.

3. Controlador azimut: Coordenada θ

En este lazo se utilizó un controlador PD, pues no requiere ajustarse por error permanente. Llega a su setpoint sin problemas, como se podrá apreciar. Las ganancias utilizadas son:

- $K_p = 0.5005$

- $K_d = 6.5065$

Estas se obtuvieron por el método de prueba y error, el que es detallado en la sección VI del presente informe.

4. Controlador elevación: Coordenada ϕ

En este lazo se utilizó un controlador PD, pues no requiere ajustarse por error permanente. Llega a su setpoint sin problemas, como se podrá apreciar. Las ganancias utilizadas son:

- $K_p = 0.5$
- $K_d = 1.5$

Estas se obtuvieron por el método de prueba y error, el que es detallado en la sección VI del presente informe. Se agrega además un término de prealimentación, definido de la siguiente forma:

$$prealimentacion_{\phi} = 0.000022 * masa_{k-1} * g0 * sin(\phi_{k-1}) \tag{14}$$

Donde g0 es la aceleración de gravedad variable, definida como:

$$g0 = 9.8065 \tag{15}$$

5. Controlador elevación: Coordenada z

En este lazo se utilizó un controlador PID, pues requiere ajustarse por error permanente. Llega a su setpoint sin problemas, como se podrá apreciar. Las ganancias utilizadas son:

- $K_p = 0.12 * abs((e_k - e_{k-1})/10000)$
- $K_d = 1.25 * abs((e_k - e_{k-1})/10000)$
- $K_i = 500 * 10^(- 10)$

Estas se obtuvieron por el método de prueba y error, el que es detallado en la sección VI del presente informe. Las ganancias hacen variables para que las constantes posean una mayor magnitud cuando el cohete se encuentre más lejos del setpoint. Por otro lado, cuando el cohete esté en la vecindad del setpoint, estas constantes forzarán a que el cohete vaya más lento, y así pueda controlar con menor esfuerzo los pequeños cambios angulares que deba realizar. Este método ha sido empleado ampliamente en sistemas de vuelo (referencia [1]).
Se divide el factor multiplicador que da la variabilidad a las ganancias por 10000 debido a que es la mejor forma de normalizar el error, y dejar la salida del controlador desagregada por cada término dentro del rango permitido para la magnitud de la señal de control del thruster.
Empíricamente se comprobó que se pueden disminuír las oscilaciones del cohete en elevación

gracias a un decaimiento exponencial del término derivativo. Por otro lado, se penaliza al controlador por utilizar movimientos demasiado bruscos, agregando un término proporcional que opera sobre la variación instantánea del valor de la señal de control del thruster. Finalmente, también se agrega prealimentación, en este caso posee una magnitud mayor que aquella que se le da en el lazo de elevación.
La ecuación del controlador PID modificado es la siguiente:

$$\begin{aligned} U_{tk} = K_p * (e_k - e_{k-1}) + K_d * exp(-k * 10^(- 7)) * ((e_k - e_{k-1}) - (e_{k-1} - e_{k-2})) \\ + K_i * \sum_0^k e_k + 1.5 * K_d * (U_{tk} - U_{tk-1}) + 0.00011 * masa_{k-1} * g0 * sin(\phi_{k-1}) \end{aligned} \tag{16}$$

Se puede operar sin prealimentación y llegar al setpoint, pero el proceso resultaría ser demasiado lento. Por otro lado, el decaimiento exponencial del término derivativo es una forma de eliminar oscilaciones en régimen permanente, pues el término derivativo debería ser prácticamente cero alrededor del setpoint, pero el radio de maniobras del cohete no lo permite.
Si en algún momento el término integral:

$$termino_{integral} = K_i * \sum_0^k e_k \tag{17}$$

Es mayor a 0.5, se lo fuerza a saturar en 0.5, para que no pueda aumentar su magnitud. El aumento excesivo del término integral en casos extremos podría saturar la salida de este controlador, por ello es necesaria esta medida de limitación. No se fuerza que sea de magnitud negativa mayor o igual a -0.5 debido a que la naturaleza del problema no permite que este término se haga negativo en ascenso.

6. Resultados de experimentos en caída libre

Antes de ver las simulaciones, téngase presente que el controlador solo comienza a actuar después de que el cohete ha completado una ascensión vertical hasta la mitad de la altura final a la que debe llegar.

6.1. Prueba de maniobras aéreas

Se utilizó el siguiente setpoint $[x, y, z]$ en metros, junto con las condiciones iniciales ya expuestas:

$$setpoint_{cartesianas} = \begin{bmatrix} 5000 & 1000 & 10000 \end{bmatrix} \tag{18}$$

Se eligieron estas condiciones iniciales pues la altura no causa un uso excesivo de combustible, y la relación entre x e y debiera forzar al cohete a doblar al menos dos veces en el aire para adquirir una actitud que le permita llegar a un punto como aquel en el plano $[x, y]$. El resultado es el siguiente:

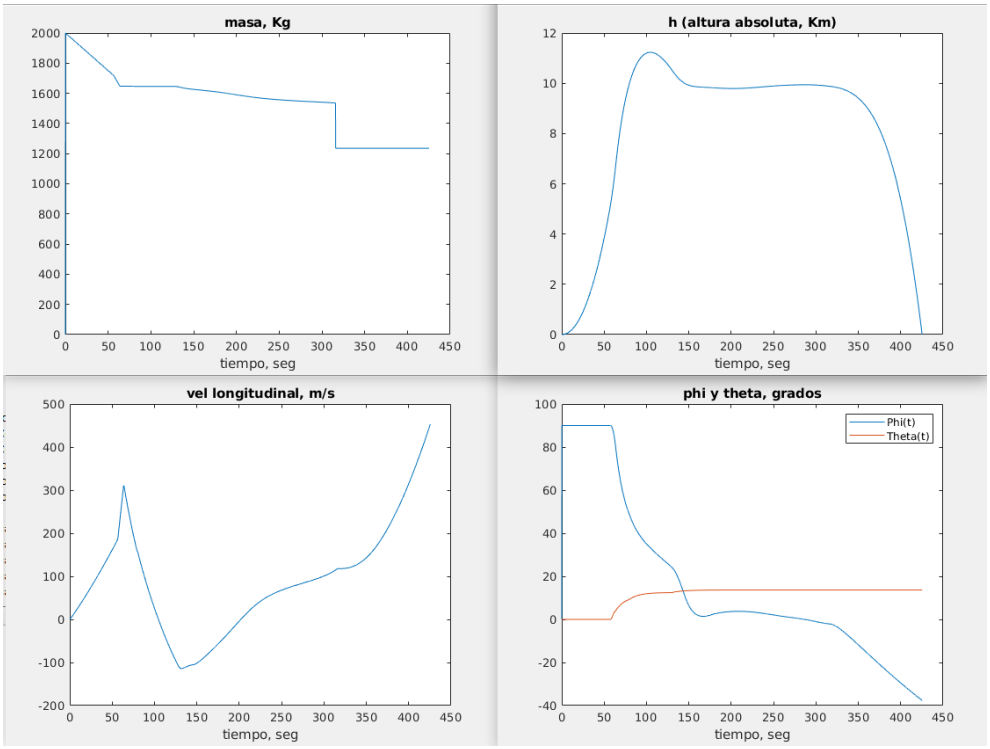

Figura 1: Llegada al setpoint y caída libre

El cohete llega al setpoint en $t = 316 \triangle T$, y toca tierra en $t = 425 \triangle T$. Esto último quiere decir que se demora $t = 109 \triangle T$ en caer de forma libre de vuelta a la superficie terrestre, después de soltar la carga.
Settling time es aproximadamente de $t = 150 \triangle T$ para el lazo de control de altura del cohete. La magnitud del overshoot máximo es de aproximadamente 1 km, pero esto es engañoso en 2D. No está ocurriendo una situación en la que el controlador de altura no esté respondiendo adecuadamente, lo que pasa es que el cohete en ese momento está girando en el aire para acomodar su actitud y apuntar lo más rápidamente posible hacia el setpoint.
Cabe destacar la variación de velocidad que experimenta el cohete: Al dar las vueltas en el aire, su velocidad cambia de signo debido a la reorientación, siendo primero 300 m/s, y finalmente -100 m/s. Al comenzar la caída libre, esta cruza nuevamente por cero y se hace positiva, subiendo debido a que el cohete no se reorienta y solo cae sin más.
El punto de partida es el orígen del plano cartesiano. El punto en metros donde se suelta la carga es:

$$\begin{bmatrix} 4703 & 928 & 9879 \end{bmatrix} \tag{19}$$

El error de control final se encuentra representado en un vector cuyos componentes se interpretan como diferencias en coordenadas cartesianas: $[ErrorenX \; ErrorenY \; ErrorenZ]$:

$$\begin{bmatrix} 297(0,059\,\%) & 72(0,072\,\%) & 121(0,012\,\%) \end{bmatrix} \tag{20}$$

En vista de lo anterior, el error porcentual promedio en las tres coordenadas cartesianas es

de 0,047 %. El punto en metros donde cae el cohete es:

$$\begin{bmatrix} 26294 & 6179 & 0 \end{bmatrix} \tag{21}$$

Como ver las variaciones de ángulos en 2D no permite visualizar correctamente el movimiento del cohete en 3D, también se ha generado el gráfico de su trayectoria en el espacio 3D cartesiano:

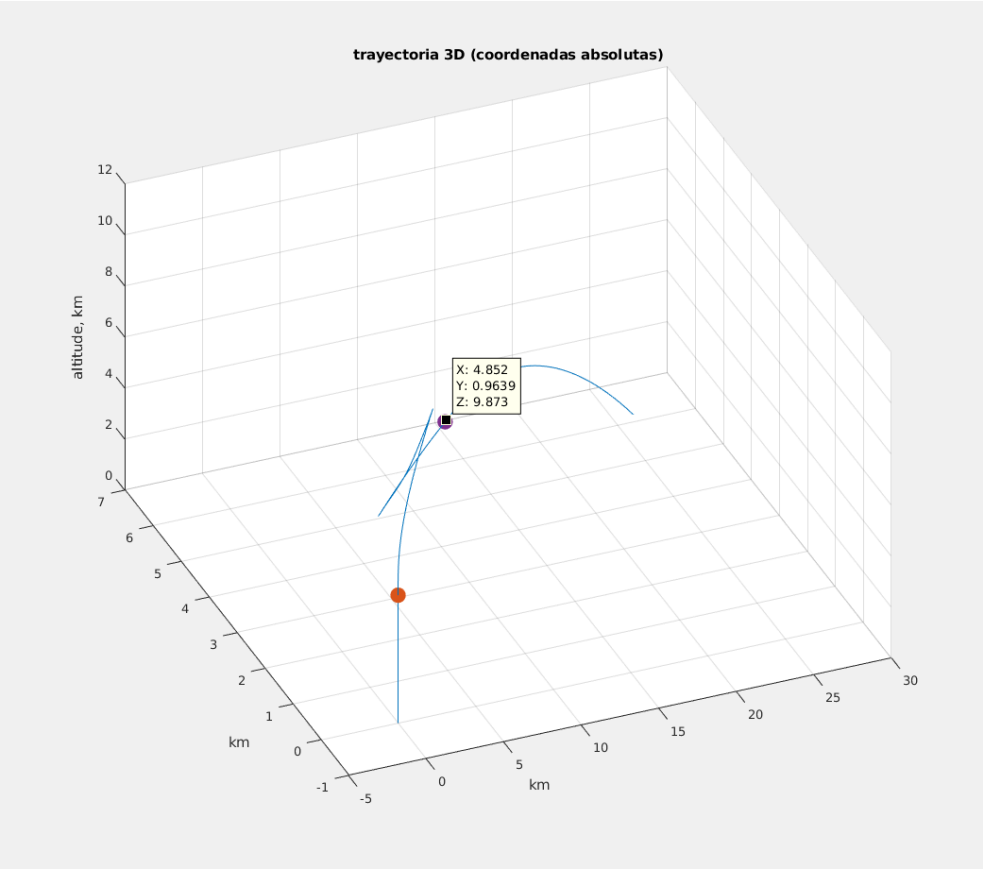

Figura 2: Movimiento en 3D del cohete

El punto naranjo marcado es aquel en que el controlador comienza a actuar, después de la fase de ascensión. El punto morado corresponde al punto en el que se llega al setpoint y la carga del cohete es soltada.

6.2. Alto consumo de combustible, maniobras de precisión tempranas

Se utilizó el siguiente setpoint $[x, y, z]$ en metros, junto con las condiciones iniciales ya expuestas:

$$setpoint_{cartesianas} = \begin{bmatrix} 5000 & 1000 & 100000 \end{bmatrix} \tag{22}$$

Se eligieron estas condiciones iniciales pues la altura causa un uso excesivo de combustible, y la relación entre x e y debiera forzar al cohete a doblar al menos dos veces en el aire para adquirir una actitud que le permita llegar a un punto como aquel en el plano $[x, y]$. Adicionalmente, x e y son mucho menores que la altura a alcanzar, por lo que el cohete necesariamente debe alcanzar este punto sobre el plano XY en la menor cantidad de ajustes aéreos posibles. El resultado es el siguiente:

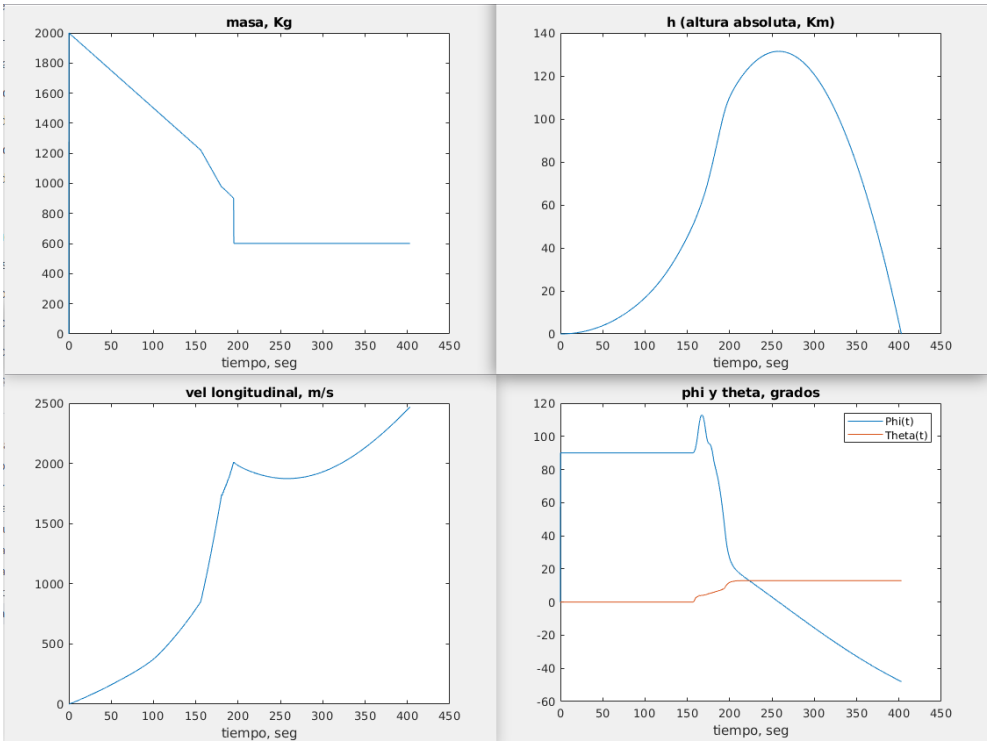

Figura 3: Llegada al setpoint y caída libre

El cohete llega al setpoint en $t = 195 \triangle T$, y toca tierra en $t = 403 \triangle T$. Esto último quiere decir que se demora $t = 208 \triangle T$ en caer de forma libre de vuelta a la superficie terrestre, después de soltar la carga.
Settling time es aproximadamente de $t = 200 \triangle T$ para el lazo de control de altura del cohete. Esto aunque el controlador no alcanza a estabilizarse en una altura antes de soltar la carga y dejar caer libremente al cohete.
Cabe destacar la variación de velocidad que experimenta el cohete: En este caso el cohete no necesita dar vueltas en el aire. Solo aumenta su velocidad hasta el punto en que se suelta la carga. Luego de ese punto, la velocidad disminuye mientras el cohete alcanza su máxima altura en caída libre, y luego comienza a aumentar debido a la caída.
El punto de partida es el orígen del plano cartesiano. El punto en metros donde se suelta la carga es:

$$\begin{bmatrix} 4760 & 950 & 104760 \end{bmatrix} \tag{23}$$

El error de control final se encuentra representado en un vector cuyos componentes se interpretan como diferencias en coordenadas cartesianas: $[ErrorenX\, ErrorenY\, ErrorenZ]$:

$$\begin{bmatrix} 240(0,048\,\%) & 50(0,05\,\%) & 4760(0,047\,\%) \end{bmatrix} \tag{24}$$

En vista de lo anterior, el error porcentual promedio en las tres coordenadas cartesianas es de 0,048 %. El punto en metros donde cae el cohete es:

$$\begin{bmatrix} 372960 & 85210 & 0 \end{bmatrix} \tag{25}$$

Como ver las variaciones de ángulos en 2D no permite visualizar correctamente el movimiento del cohete en 3D, también se ha generado el gráfico de su trayectoria en el espacio 3D cartesiano:

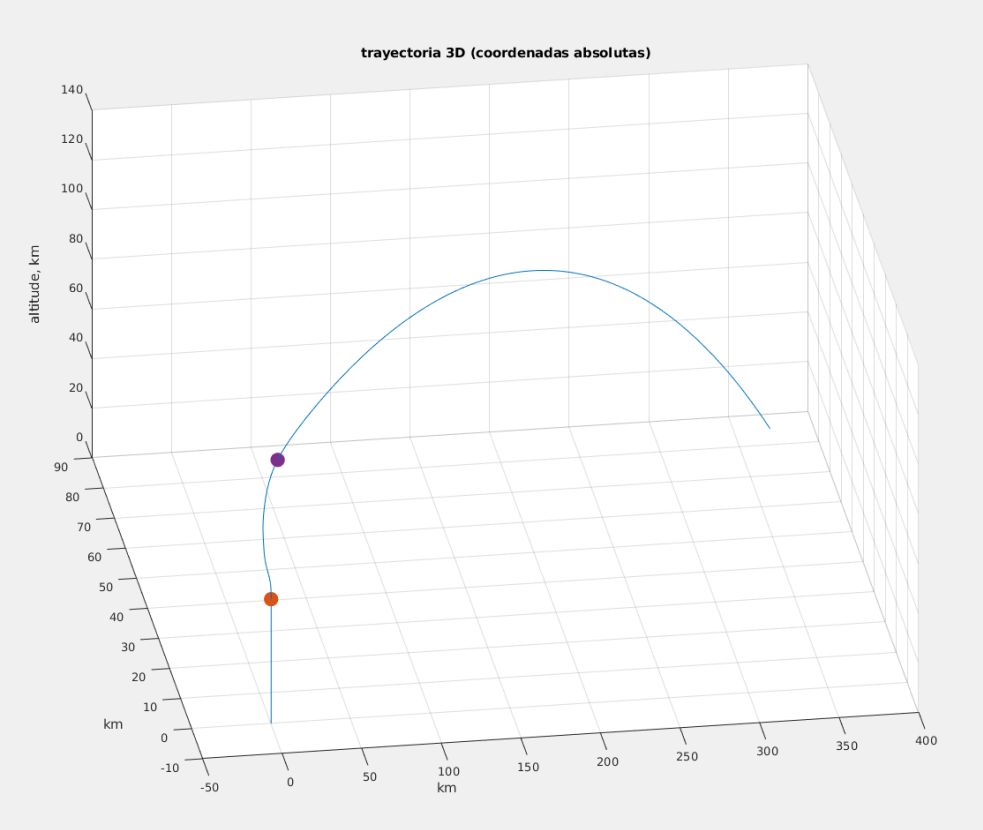

Figura 4: Movimiento en 3D del cohete

El punto naranjo marcado es aquel en que el controlador comienza a actuar, después de la fase de ascensión. El punto morado corresponde al punto en el que se llega al setpoint y la carga del cohete es soltada.

II. Parte: Llegada a punto de lanzamiento de la carga y aterrizar suavemente

El aterrizaje se realiza en dos fases: Primero se intenta disminuir la energía cinética del cohete, para estabilizarlo en un ángulo alrededor de $\pi/2$ mientras aterriza. En principio, esto es necesario para que la velocidad no aumente rápidamente al hacer que la tobera apunte hacia el suelo mientras el cohete baja.
El error de control se plantea respecto a la energía cinética, calculada como sigue:

$$K_k = 0.5 * masa_k * (\sqrt{(h'_k)^2 + (\phi'_k)^2 + (\theta_k)^2})^2 \tag{26}$$

El error de control es:

$$E_k = -K_k \tag{27}$$

Debido a que se requiere alcanzar una energía cinética igual a cero al aterrizar. El controlador varía en su expresión para acomodarse a esta nueva necesitad. Con el objetivo de no aumentar la energía cinética, $U_y = 0$, así el cohete no estará aumentando su velocidad y gastando combustible en girar gracias a la coordenada θ. Luego de llegar al setpoint, se utiliza un controlador PD con la siguiente fórmula (mismas ganancias variables de antes):

$$\begin{aligned} U_{tk} = (K_p * (e_k - e_{k-1}) + K_d * exp(-k * 10^(- 7)) * ((e_k - e_{k-1}) - (e_{k-1} - e_{k-2})) \\ +0.000003 * (-K_k)) * 0.1 \end{aligned} \tag{28}$$

Durante el uso de este controlador en altura, se utiliza el mismo controlador anterior de elevación, con la misma fórmula presentada en la sección 1. El setpoint se fija en en z variable para ambos. Así se recalculaba también el setpoint en ϕ. ref_z es la altura a la que el cohete soltó la carga. La fórmula de variación de setpoint es:

$$z_{SP} = z_{SP} * (1 + 0.000000015 * (-K_k) * \frac{z_{k-1} + 1}{ref_z * 1000}) \tag{29}$$

Se partía desde el setpoint de altura inicial.
El controlador de altura es reemplazado por uno con constantes distintas cuando la altura alcanza un 88 por ciento de su valor en la referencia donde se soltó la carga. La fórmula del controlador que finalmente hace aterrizar al cohete es:

$$\begin{aligned} U_{tk} = K_p * (e_k - e_{k-1}) + K_d * exp(-k * 10^(- 7)) * ((e_k - e_{k-1}) - (e_{k-1} - e_{k-2})) \\ +0.0000000001 * (-h'_k) \end{aligned} \tag{30}$$

Este controlador opera hasta que el cohete llega a estar 16 metros sobre la superficie terrestre. Luego, U_t baja según la relación:

$$U_{tk} = U_{tk-1} * 0.99 \tag{31}$$

Hasta que el cohete toca tierra.
En este caso, como la energía cinética ya ha disminuido mientras el cohete baja con la tobera hacia el suelo, se controla con respecto a la velocidad la coordenada cartesiana z. Cuando la velocidad es menor que 50 m/s, lo que indica que el cohete está cerca de aterrizar suavemente, el setpoint de altura varía según la siguiente fórmula:

$$z_{SP} = z_{SP} * 0.997 * (1 + 0.00011 * (-h'_{k-1}) + 0.005 * (\frac{ref_z}{(z_{k-1} + 1) * 1000})) \tag{32}$$

El setpoint de elevación se recalcula en cada tiempo de muestreo como si el cohete tuviera un setpoint de altura de gran magnitud, de forma que este setpoint haga que el cohete tienda a inclinarse en 90 grados:

$$\phi_{SP} = atan(\frac{\sqrt{((x_{SP} - x_{k-1})^2 + (y_{SP} - y_{k-1})^2)}}{(1e10 - z_{k-1})}) \tag{33}$$

1. Resultados de pruebas con aterrizaje suave

1.1. Consumo moderado de combustible, maniobras de precisión tempranas

Se utilizó el siguiente setpoint $[x, y, z]$ en metros, junto con las condiciones iniciales ya expuestas:

$$setpoint_{cartesianas} = \begin{bmatrix} 5000 & 1000 & 10000 \end{bmatrix} \tag{34}$$

Se eligieron estas condiciones iniciales pues la altura causa un uso excesivo de combustible, y la relación entre x e y debiera forzar al cohete a doblar al menos dos veces en el aire para adquirir una actitud que le permita llegar a un punto como aquel en el plano $[x, y]$. El resultado es el siguiente:

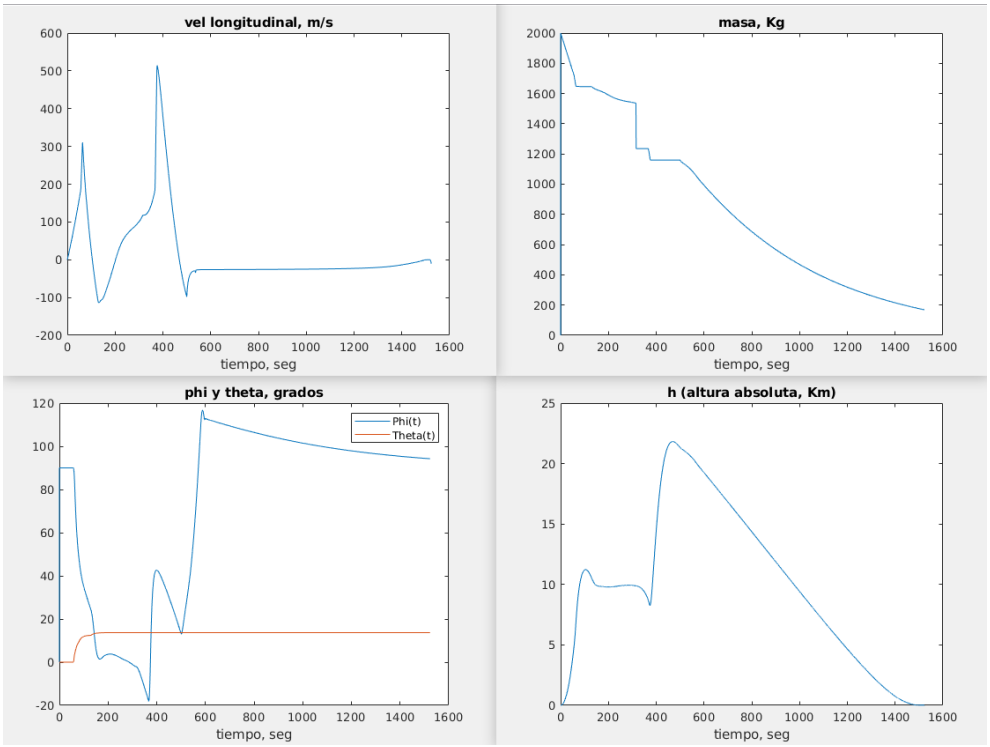

Figura 5: Llegada al setpoint y aterrizaje suave

El cohete llega al setpoint en $t = 316 \triangle T$, y toca tierra en $t = 1522 \triangle T$. Esto último quiere decir que se demora $t = 1206 \triangle T$ en caer de forma controlada de vuelta a la superficie terrestre, después de soltar la carga.
Settling time es aproximadamente de $t = 200 \triangle T$ para el lazo de control de altura del cohete. El controlador se estabiliza alrededor de su setpoint en altura por $t = 200 \triangle T$ adicionales, con el objetivo de cambiar el signo de su velocidad para poder adoptar una elevación cercana a los 120 grados sexagesimales.
En este caso el cohete no necesita dar vueltas en el aire. Solo aumenta su velocidad hasta el punto en que se suelta la carga. Luego de ese punto, la velocidad cambia de signo mientras disminuye la energía cinética. Finalmente, el controlador de aterrizaje lleva el cohete a una posición en 90 grados de elevación, y asegura un aterrizaje suave con velocidad final de -9.75 m/s y ángulo de elevación de 94 grados sexagesimales.
El combustible se acaba hacia el final del trayecto de aterrizaje, debido a la necesidad de mantener propulsión para que la velocidad de aterrizaje no suba en magnitud. Si el cohete se llevara a un setpoint más alto, y se le hiciera soltar su carga, no sería capaz de mantener la propulsión necesaria para aterrizar suavemente.
El punto de partida es el origen del plano cartesiano. El punto en metros donde se suelta la carga es:

$$\begin{bmatrix} 4702 & 927 & 9879 \end{bmatrix} \tag{35}$$

El error de control final se encuentra representado en un vector cuyos componentes se inter-

pretan como diferencias en coordenadas cartesianas: $[ErrorenX\, ErrorenY\, ErrorenZ]$:

$$\begin{bmatrix} 298(0,059\,\%) & 73(0,073\,\%) & 121(0,012\,\%) \end{bmatrix} \tag{36}$$

En vista de lo anterior, el error porcentual promedio en las tres coordenadas cartesianas es de 0,048 %. El punto en metros donde cae el cohete es:

$$\begin{bmatrix} 33393 & 7906 & 0 \end{bmatrix} \tag{37}$$

Como ver las variaciones de ángulos en 2D no permite visualizar correctamente el movimiento del cohete en 3D, también se ha generado el gráfico de su trayectoria en el espacio 3D cartesiano:

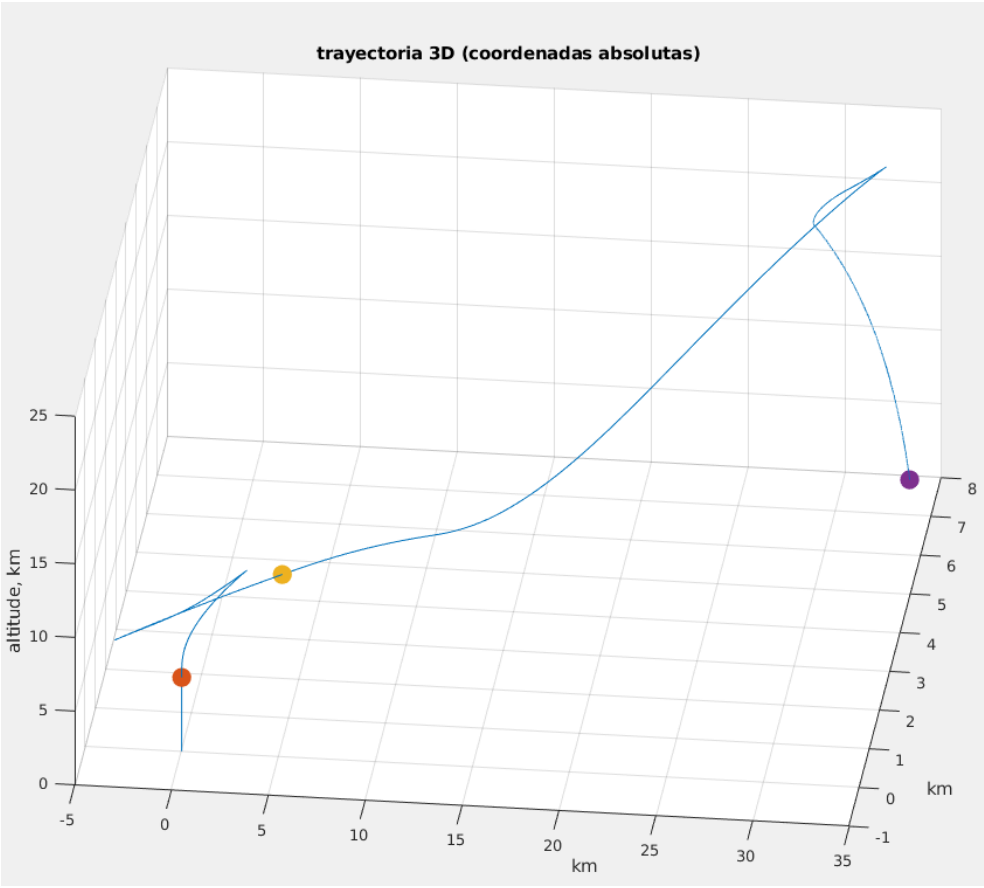

Figura 6: Movimiento en 3D del cohete

El punto naranjo marcado es aquel en que el controlador comienza a actuar, después de la fase de ascensión. El punto amarillo corresponde al punto en el que se llega al setpoint y la carga del cohete es soltada. Finalmente, el punto morado es aquel en que la magnitud de la velocidad del cohete se vuelve menor a 10 m/s. Se estima óptimo que la velocidad no supere aquel límite, debido a que en general el choque de un objeto a esa velocidad ni

siquiera es capaz de herir a un ser humano. Teóricamente es imposible llegar a velocidad que sea exactamente cero sin tocar tierra, sin que se tenga que controlar el cohete hasta un tiempo que tiende a infinito.

III. Explicación de código MATLAB

Inicialmente se definen todas las variables que llevan los valores de errores de control, ganancias, y valores binarios para controlar la serie de rutinas de controladores. También se definen los vectores donde se guardan las posiciones del cohete en cada instante de simulación, de tiempo transcurrido, y de condiciones iniciales.
Posteriormente, se comienza la rutina de ascenso vertical del cohete, dependiente de que la variable llamada "fase2 "sea igual a cero, y de que no se ha llegado a la mitad de la altura dada por el setpoint. Al finalizar el ascenso, se guarda el punto donde se pasan las variables controladas al primer controlador:

```
if ((set_z - pz(iter-1)) > set_z*0.5)&&(fase2==0)
    ut=5; % se asciende a media velocidad para entregar al controlador un cohete menos descontrolado
    ux=0;
    uy=0;
    disp("ASCENDIENDO...")
else
    if fase2 == 0
        fase2=1;
        p1=[px(iter-1) py(iter-1) pz(iter-1)];
        disp(" === TÉRMINO DE ASCENSO ===")
    end
end
```

Figura 7: Fase de ascenso

El controlador opera con las operaciones descritas anteriormente, hasta que detecta que ninguna de las coordenadas cartesianas de su posición actual está más lejos que una tolerancia de 300 metros del setpoint. Cuando esto se cumple, comienza la caída libre o el descenso controlado. La caída libre se da cuando la variable çaidalibre .es 1, en caso contrario se realiza el descenso controlado:

```
% variables manipuladas para angulo de toberas
ux=Kp_phi*errPhi+Kd_phi*evPhi  + 0.2*preK*x0(end,3)*g0*sin(x0(end,4)) ; %rad
uy=Kp*errTheta+Kd*evTheta; %rad
integral = Ki*acum_z;
if integral > 0.5
    integral = 0.5;
end

if (((abs(set_z - pz(iter-1)) > tolerance)&&((set_z - pz(iter-1)) > 0))||((abs(set_y - py(iter-1)) > tolerance))||((abs(set_x - px(iter-1)) > tolerance)))&&(flagg==0)
    ut=Kp_impulso*errR+Kd_impulso*exp(-iter*nr)*evR + integral + Kd_ur*d_r  + preK*x0(end,3)*g0*sin(x0(end,4));
```

Figura 8: Controlador hacia el setpoint donde se suelta la carga

La variable "flagg "denota globalmente el momento en que termina el control hacia el setpoint. Cuando se llega al setpoint, entonces se suelta la carga y se cambia inexorablemente "flagg "hacia el valor 1, de forma de que comience el descenso. En el caso de la caída libre, el código aparte de guardar el punto donde comienza la caída no hace más que mantener todas las variables de control en cero. Cuando el cohete posee una altura igual o menor que cero,

entonces se considera que ha chocado contra la tierra, y se muestra el punto de contacto para graficarlo:

```
if flagg == 0
    flagg = 1;
    ref = pz(iter-1);
    p2=[px(iter-1) py(iter-1) pz(iter-1)];
    p3=[px(iter-1) py(iter-1) pz(iter-1)];
    disp("------ LLEGADA A SETPOINT ------")
    x0(end,3) = x0(end,3) - Mcarga;
    disp(last_t)
    disp([px(iter-1) py(iter-1) pz(iter-1)])
end
ut = 0;
ux = 0;
uy = 0;
if (pz(iter-1) <= 0)
    goal=1;
    disp(" -------- CRASHED SUCCESFULLY -------- ")
    disp(last_t)
    disp([px(iter-1) py(iter-1) pz(iter-1)])
end
```

Figura 9: Código de caída libre

En el caso de que la variable çaidalibre "sea cero, entonces se da paso al control de velocidad de aterrizaje. Este tiene dos fases que se separan al momento de definir los setpoints con los que opera cada controlador. La clave de cambio de controlador y de setpoint es la variable "landing ": Si esta es igual a cero, entonces se emplea control por energía cinética:

```
if caidalibre==0
    if landing == 0 % control por energía cinética
        k = 0.5*x(end,3)*(sqrt((x(end,2))^2 + (x(end,6))^2 + (x(end, 7))^2  ))^2;
        errvel = -k;
        d_errvel = errvel - vel_ant;

        uy = 0; % el cohete no se mueve en la coordenada esférica theta
        ut=(Kp_impulso*errR+Kd_impulso*exp(-iter*nr)*evR + 0.000003*errvel)*0.1; % controlador de energía cinética

        if flagg == 0
            flagg = 1;
            ref = pz(iter-1);
            p2=[px(iter-1) py(iter-1) pz(iter-1)]; % guardar punto donde se llega a setpoint
            disp("------ LLEGADA A SETPOINT ------")
            x0(end,3) = x0(end,3) - Mcarga; % soltar la carga
            disp(last_t)
            disp([px(iter-1) py(iter-1) pz(iter-1)])
        end

    else % control por velocidad en coordenada z, justo antes de efectivamente aterrizar sobre la superficie terrestre
        disp("Landing...")
```

Figura 10: Código de energía cinética

Cuando esta es igual a uno, entonces se pasa a control por velocidad en la coordenada cartesiana z. A continuación, se podrá apreciar cómo el código distingue explícitamente entre ambos controladores:

```
else % control por velocidad en coordenada z, justo antes de efectivamente aterrizar sobre la superficie terrestre
    disp("Landing...")
    disp([x(end,2) x(end,4)*180/pi])
    disp(errR)

    uy = 0; % 0.3*pi/180;
    if pz(iter-1) > 16 % control de velocidad suave
        ut= Kp_impulso*errR+Kd_impulso*exp(-iter*nr)*evR + 0.0000000001*(-x(end,2)) ;
    else % fase final de contacto con la tierra
        ut=ut*0.99;
    end
    disp("----------------------------")
    disp([PhiSP set_z pz(iter-1)])
    disp([ux uy ut])

    if x(end,2) <= 10
        p3=[px(iter-1) py(iter-1) pz(iter-1)];
    end

    if (pz(iter-1) <= 0)
        goal=1;
        disp(" -------- CRASHED SUCCESFULLY -------- ")
        disp(last_t)
        disp([px(iter-1) py(iter-1) pz(iter-1)])
    end
end
```

Figura 11: Código de control de descenso suave

Con el objetivo de demostrar de manera transparente la forma en la que varían distintas estadísticas del cohete mientras aterriza, también se imprimen en consola los valores de: la velocidad en la coordenada z, la posición angular en elevación, el error de control de altura (errR), el setpoint de elevación, el setpoint de altura, la altura en la iteración anterior, y finalmente todas las variables de control.
En cada iteración, luego de calcular las variables de control de la forma que se ha expuesto, se corre el modelo con las condiciones iniciales dadas por su salida en la iteración anterior. Luego de ello, se actualizan las nuevas condiciones iniciales con la salida actual, se guardan posiciones y tiempos de simulación, y se limita la altura a que sea mayor o igual que cero. Cuando en caída libre o en control de descenso suave se detecta que se ha tocado tierra, la variable goal se hae 1 para parar la simulación en ese instante de tiempo:

```
% ************************************************
% ********   integracion numerica variables cohete   ***********
options=odeset('RelTol',1e-4,'AbsTol',1e-6);
[t,x]=ode23(@(t,x) cohete_modelov2(t,x,ut,ux,uy),[0 dT],x0,options);

% paso a coordenadas absolutas px, py, pz
px(iter)=px(iter-1)+x(end,2)*cos(x(end,4))*cos(x(end,5))*dT;
py(iter)=py(iter-1)+x(end,2)*cos(x(end,4))*sin(x(end,5))*dT;
          disp(x)

aux=pz(iter-1)+x(end,2)*sin(x(end,4))*dT;
if aux < 0
    aux=0;
end
pz(iter)=aux;

T=[T;t+dT*(iter-1)];
setphi=[setphi;PhiSP];
X=[X;x];
x0=x(end,:);
u_ant = ut;
vel_ant= errvel;

if goal==1
    break
end
last_t = t+dT*(iter-1);
```

Figura 12: Código de simulación en general

Respecto a la determinación de setpoints, se procede con el cálculo de setpoints incrementales descrito en secciones anteriores del presente informe. En código, primero se calculan los setpoints en el caso del controlador para llegar a un punto específico en el espacio:

```
for iter=2:fix(Tf/dT)
    % ***************CALCULO SETPOINTS INCREMENTALES**********************
    if flagg==0 % ascenso hacia setpoint
        % actualizar el ángulo de elevación objetivo
        directive = wrapTo2Pi(atan(sqrt((set_x-px(iter-1))^2 + (set_y-py(iter-1))^2)/(set_z-pz(iter-1)) ));

        % transformar a coordenadas de elevación dadas en el enunciado del
        % proyecto
        if directive >= 0
            directive = pi/2-directive;
        end
        if directive < 0
            directive = directive+pi/2;
        end
        PhiSP = directive;

        % actualizar setpoints en z y en theta (el de z es solo
        % referencial, no se usa en el cálculo del error)
        ThetaSP = wrapTo2Pi(atan((set_y-py(iter-1))/(set_x-px(iter-1))));
        AlturaSP = sqrt((set_y-py(iter-1))^2 + (set_y-py(iter-1))^2 + (set_z-pz(iter-1))^2);
```

Figura 13: Cálculo de setpoints en coordenadas esféricas para llegar a setpoint en coordenadas cartesianas

Cuando se elige el descenso controlado para aterrizaje suave, se sigue el cálculo de los dos setpoints, correspondientes a cada controlador expuesto anteriormente:

```
else % etapas de descenso controlado
    if landing == 0 % controlador en base a energía cinética
        set_z = set_z*(1 + 0.000000015*errvel*( ((pz(iter-1)+1)*1)/(ref*1000) ) );
        if (pz(iter-1) > ref*0.88)
            % actualizar el ángulo de elevación objetivo
            directive = wrapTo2Pi(atan(sqrt((set_x-px(iter-1))^2 + (set_y-py(iter-1))^2)/(set_z-pz(iter-1)) ));
        else
            landing = 1;
        end
    else % controlador en base a velocidad en coordenada cartesiana z
        if (x0(end,2) < 50) % se calcula con este setpoint cuando la velocidad es baja
            set_z = set_z*0.997*(1 + 0.00011*(-x0(end,2)) + 0.005*( ref/((pz(iter-1)+1)*1000) ) );
        end
        % actualizar el ángulo de elevación objetivo
        directive = wrapTo2Pi(atan(sqrt((set_x-px(iter-1))^2 + (set_y-py(iter-1))^2)/(1e10-pz(iter-1)) ));
    end
    % transformar a coordenadas de elevación dadas en el enunciado del
    % proyecto
    if directive >= 0
        directive = pi/2-directive;
    end
    if directive < 0
        directive = directive+pi/2;
    end
    PhiSP = directive;

    % actualizar setpoints en z y en theta (el de z es solo
    % referencial, no se usa en el cálculo del error)
    ThetaSP = wrapTo2Pi(atan((set_y-py(iter-1))/(set_x-px(iter-1))));
    AlturaSP = sqrt((set_y-py(iter-1))^2 + (set_y-py(iter-1))^2 + (set_z-pz(iter-1))^2);
end
% ***************CONTROLADOR**********************
```

Figura 14: Cálculo de setpoints en coordenadas esféricas para aterrizar con velocidad cercana a cero y posición vertical en 90 grados recpecto a la horizontal del cohete

IV. Análisis y supuestos

- El modelo no lineal del cohete es asintóticamente estable
- Es posible encontrar uno o más controladores PID que logren controlar gruesamente la magnitud de la respuesta en cada lazo de coordenadas esféricas y cartesianas
- Los controladores PID se pueden extender por medio la adición de términos regularizadores", en el sentido de que pueden hacer que respondan de forma más suave, o que se los pueda prealimentar para acelerar su convergencia hacia un valor de setpoint
- El lazo que causa mayor intestabilidad en altura es el lazo de elevación
- El control del lazo de altura varía la magnitud de la respuesta de los lazos que dependen de los ángulos θ y ϕ
- En teoría se podrían adicionar filtros pasabajos al lazo antes de llegar a los controladores, pero en la práctica esto implica una nueva sintonitonización de todas las constantes, y una mayor tendencia a inestabilizarse debido a la gran cantidad de combinaciones posibles que pueden generar una respuesta de ese tipo
- Se intentó en varios experimentos de prueba y error añadir antiwindup, pero esto nuevamente causó inestabilidad del sistema debido a la cantidad de combinaciones erróneas del antiwindup combinado con todos los controladores en uso

- El controlador está diseñado para que en la mayor parte de los casos de uso no sature las señales de control en los límites absolutos permitidos por la planta
- El controlador está diseñado para operar con un tiempo de muestreo de 0.1 segundos. Un tiempo de muestreo mayor, por ejemplo de 1 segundo, probó que hace imposible que un controlador PID sintonizado por prueba y error logre llegar al cohete a las referencias. Esto se debe a lo rápido que se mueve el cohete respecto al tiempo de respuesta mínimo (igual al de muestreo) que puede tener el controlador. Este punto fue comprobado empíricamente al sintonizar controladores en Simulink.

V. Discusión de los resultados

El controlador del cohete lleva a cabo un control fino y completamente no lineal para alcanzar los setpoints que se le propongan. Su limitación es la falta de combustible para aterrizar suavemente desde alturas de cientos de kilómetros, debido a su pequeño tamaño comparado con un cohete Falcon 9 Heavy, de peso promedio de 34000 kg.
El controlador toma en cuenta la no linealidad del problema por medio de corrección de elevación y altura por medio de prealimentación, disminuye el error en régimen permanente con decaimiento exponencial del término proporcional del controlador PID en el lazo de altura, y responde rápidamente a cambios de setpoints en coordenadas esféricas.
Para este modelo de cohete, es muy importante controlar las sobreoscilaciones. Se previnieron completamente por medio de del control con setpoints incrementales y rápidos tiempos de respuesta de cada controlador. Habría sido interesante filtrar el ruido de sensores ruidosos, pero debido a la naturaleza incremental de los controladores, el ruido se habría cancelado rápidamente por los cambios oportunos en señales de control. Esto último además se debe al bajo tiempo de muestreo, el que es posible de lograr gracias a la Ley de Moore. Esta ha permitido disminuír el tamaño de los transistores en procesadores de los computadores, por lo que es posible muestrear señales a mayor frecuencia.
Se podría sintonizar cada controlador con mayor precisión al estimar la respuesta del sistema con simulación de su diagrama de Nyquist, y otros métodos en frecuencia. Habría bastado perturbar a cada controlador con una señal representativa del promedio de las perturbaciones que no saturaran en los límites máximos permitidos para ángulo de tobera y empuje dado por el thruster.
En conclusión, los resultados son aceptables, pero mejorables con mayor esfuerzo de cómputo de un simulador, y análisis del sistema controlado con controladores de la forma presentada en el presente informe.

VI. Código de los experimentos realizados

Junto a este informe, se adjunta un archivo comprimido con los archivos de las pruebas realizadas para hallar el controlador descrito en el presente informe. En resumen, hay cuatro

experimentos:

- Experimento Simulink Control en Coordenadas Cartesianas (Simple): Se encuentra el proyecto de Simulink "*cohete_modelo_simulink.slx*" con el primer modelo sintonizado automáticamente por el bloque PID de Simulink. Se hizo un lazo de control por cada coordenada, con un solo controlador PID. Se controlan solo variables cartesianas. También hay un control limitador de velocidad con PID, pero este no se podría usar en la práctica. En este modelo se agregó antiwindup como un bloque saturador que da feedback.

- Experimento Simulink Control en Cascada de Coordenadas Cartesianas: Está el proyecto "$func\_modelo\_iter2.slx$", donde se hace un control en cascada de varios controladores PID, tomando las coordenadas cartesianas del cohete para calcular el error. En este caso, no se controla la velocidad directamente. En este modelo se agregó antiwindup como un bloque saturador que da feedback en el loop interior del controlador en cascada.

- Experimento controladores PID y PD en archivo .m: Este es el presentado en el presente informe. Los archivos asociados son "$ayudantia1\_simulacion\_simple.m$", "$MI\_CODIGO\_FINAL.m$" para las simulaciones del modelo, y los archivos "*cohete_tester.m*" y "*cohete_tester_simple.m*" para probar las constantes en coordenadas cartesianas para finalmente decidir utilizar solamente coordenadas esféricas. El archivo a ejecutar para reproducir los resultados presentados es "$MI\_CODIGO\_FINAL.m$".

Bibliografía

[1] Tacconi, Mantz, Solsona Puleston. (2005). Controladores basados en estrategias PID. Recuperado de: http://www2.udec.cl/ vladimirfriz/Apunte-PID.pdf