UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Seminario de Problemas de Modelado y Simulación de Sistemas

Reporte Proyecto

Robot Móvil 3DOF Helicóptero Quanser.

Alumno: Pacheco Quintero Marco Antonio

Profesor: Gómez Anaya David Alejandro

Fecha: 11 de diciembre de 2019

Semestre 2019B Sección D02

Objetivos

- Simular tridimensionalmente el robot móvil mostrando el espacio de trabajo del robot.
- Modelo matemático de la dinámica del sistema.
- Simulación del modelo dinámico en Simulink.
- Generar un bloque de control, dadas dos entradas (altura y desplazamiento deseado), generando entradas al sistema para cumplir dichas posiciones.
- Comparar el comportamiento simulado y real del experimento bajo mismas condiciones.

Marco Teórico

Robot móvil 3DOF helicóptero

El aparato consta de dos motores de corriente directa montados en un extremo sobre un armazón rectangular que propulsan dos hélices, una para cada motor respectivamente. Dicho armazón posee un eje horizontal que le permite girar libremente a las hélices. El cuerpo principal del aparato esta suspendido por la mitad y posee dos ejes uno que le permite elevarse verticalmente y otro mas que le permite trasladarse 360 grados sobre dicho eje.

Figura 1 - Robot móvil 3DOF helicóptero

En el extremo contrario a las hélices se encuentra un cubo que sirve de contrapeso para equilibrar el aparato. Aplicar un voltaje igual a ambos motores provoca una elevación, Una diferencia de voltajes entre los motores provocara un giro en el armazón del extremo sobre el que están montadas las hélices, y por consecuente una traslación.

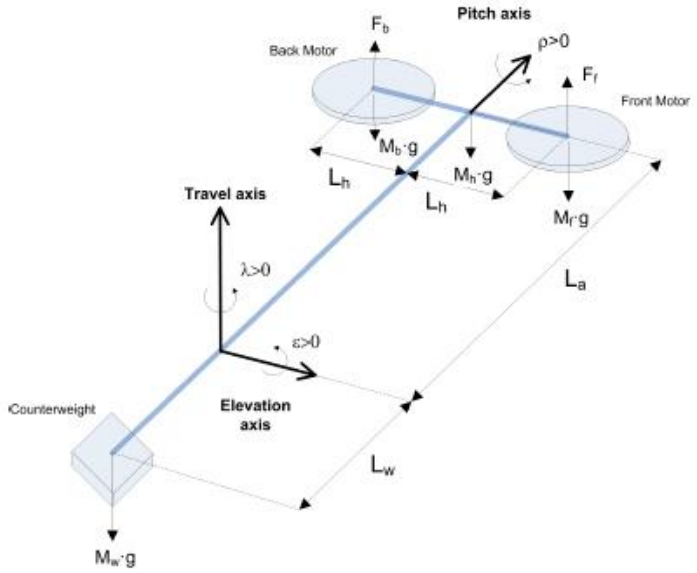

Figura 2 Medidas del robot

Las medidas del aparato mostradas en el diagrama anterior se pueden describir como:

L_w: la distancia que hay desde el eje de elevación a el centro de gravedad del contrapeso.

L_a: la distancia del eje de elevación al centro del marco donde están montados las hélices.

L_h: la distancia del centro del marco justo donde esta el eje de tono hasta el centro de gravedad de cualquiera de las dos hélices.

M_w: la masa del contrapeso.

M_h: la masa total que conforman los motores, las hélices y todo el armazón del extremo superior

$M_b\ y\ M_f$: las masas de cada hélice (incluyendo motor), trasera y frontal, respectivamente.

$F_b\ y\ F_f$: las fuerzas de impulso generadas por las hélices para cada una respectivamente.

λ: el ángulo que representa la posición de traslación del sistema.

ρ: el ángulo que representa la posición de tono del sistema.

ε: el ángulo que representa la posición de elevación del sistema.

Cinemática de un robot

Esta estudia el movimiento de un robot con respecto a un sistema de referencia, se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, relacionando la posición y orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Dado que un robot puede considerarse como una cadena cinemática formada por eslabones unidos entre si mediante articulaciones, estableciendo un sistema fijo en la base del robot se puede describir la localización de cada uno de los eslabones con respecto a este sistema. Así la cinemática directa del robot se reduce a encontrar la matriz homogénea de transformación T que relacionan posición y orientación del extremo del robot.

Dicha matriz se obtiene como el producto de las matrices homogéneas de transformación que describen la cinemática de cada eslabón representadas como ${}^{i-1}A_i$ donde i es el numero de eslabones que presenta el robot. Así para un robot de tres grados de libertad la matriz de transformación de T está formada por:

$$T = {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3$$

Algoritmo de Denavit-Hartenberg

Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. El método utiliza una matriz de transformación homogénea para describir la relación espacial entre los dos elementos rígidos adyacentes.

El algoritmo consta de 16 pasos, que dan como resultado una tabla de varios parámetros que servirán para construir las matrices de transformación ${}^{i-1}A_i$ que describirán la cinemática del sistema. Los parámetros son: θ_i es un parámetro variable en articulaciones giratorias, d_i es un parámetro variable en articulaciones prismáticas, a_i es la distancia que hay de un eje x_i a la intersección con el eje z_{i-1}, α_i es un ángulo de separación entre los ejes z_{i-1} y z_i.

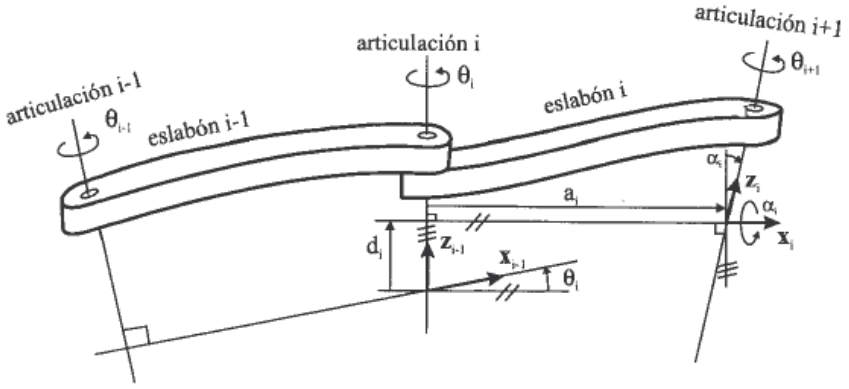

Figura 3 - Parámetros de Denavit-Hartenberg para un eslabón giratorio

Una vez se obtienen estos parámetros para cada eslabón se construye su respectiva matriz de transformación homogénea con el siguiente modelo:

$$^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & \mathrm{a}_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & \mathrm{a}_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Espacio de estados

El método de espacio de estados está basado en la descripción del sistema mediante n ecuaciones diferenciales de primer orden, que se agrupan en una ecuación vectorial matricial. Un sistema está representado en espacio de estados por las siguientes ecuaciones.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

para $t \geq t_0$ y las condiciones iniciales, $x(t_0)$, donde

$x =$ vector de estado

$\dot{x} =$ derivada del vector de estado con respecto al tiempo

$y =$ vector de salida

$u =$ vector de entrada o de control

$A =$ matriz del sistema

$B =$ matriz de la entrada

$C =$ matriz de la salida

$D =$ matriz de la prealimentación

Desarrollo

Comenzamos desarrollando la cinemática del helicóptero de tres grados de libertad, las matrices de transformación homogéneas usadas para realizar la simulación del espacio de trabajo del sistema fueron las siguientes:

De la base al eje de traslación:

$$HTM\_Base\_To\_Travel = \begin{bmatrix} C(\lambda(t)) & S(\lambda(t)) & 0 & 0 \\ -S(\lambda(t)) & C(\lambda(t)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Del eje de traslación al contrapeso:

$$HTM\_Travel\_To\_CW = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(\varepsilon(t)) & -S(\varepsilon(t)) & -C(\varepsilon(t))L_w \\ 0 & S(\varepsilon(t)) & C(\varepsilon(t)) & -S(\varepsilon(t))L_w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Del eje de traslación al centro de las hélices:

$$HTM\_Travel\_To\_HB = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(\varepsilon(t)) & -S(\varepsilon(t)) & C(\varepsilon(t))L_a \\ 0 & S(\varepsilon(t)) & C(\varepsilon(t)) & S(\varepsilon(t))L_a \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Del centro de las hélices al motor de enfrente:

$$HTM\_HB\_To\_FM = \begin{bmatrix} C(\rho(t)) & 0 & -S(\rho(t)) & C(\rho(t))L_h \\ 0 & 1 & 0 & 0 \\ S(\rho(t)) & 0 & C(\rho(t)) & S(\rho(t))L_h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Del centro de las hélices al motor de atrás:

$$HTM\_HB\_To\_BM = \begin{bmatrix} C(\rho(t)) & 0 & -S(\rho(t)) & -C(\rho(t))L_h \\ 0 & 1 & 0 & 0 \\ S(\rho(t)) & 0 & C(\rho(t)) & -S(\rho(t))L_h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De la base al contrapeso:

$$HTM\_BASE\_TO\_CW = HTM\_BASE\_TO\_TRAVEL * HTM\_TRAVEL\_TO\_CW;$$

De la base al centro de las hélices:

$$HTM\_BASE\_TO\_HB = HTM\_BASE\_TO\_TRAVEL * HTM\_TRAVEL\_TO\_HB;$$

De la base al motor frontal:

$$HTM\_BASE\_TO\_FM = HTM\_BASE\_TO\_HB * HTM\_HB\_TO\_FM;$$

De la base al motor de atrás:

$$HTM\_BASE\_TO\_BM = HTM\_BASE\_TO\_HB * HTM\_HB\_TO\_BM;$$

Comenzamos nuestro código declarando las distancias necesarias del helicóptero, las posiciones angulares de los ejes de movimiento y un par de ciclos for, uno para recorrer el eje de traslación (de 0 ° a 360 °) y el otro para el eje de elevación (aproximadamente de -40 ° a 90°, considerando que el helicóptero completamente horizontal tiene 0 ° de elevación y esta sobre una mesa).

```
L_w=0.470;
L_a=0.660;
L_h=0.178;

Elevacion = 0;
Pitch = 0;
Traslacion = 0;
i=1;

for Traslacion = 0:20:360
    for Elevacion = -40:20:100
```

Debido a que las funciones seno y coseno que usaremos para las matrices son manejadas en radianes por Matlab tendremos que hacer la conversión en cada pasada para las siguientes posiciones angulares.

```
Elevacion_nueva = Elevacion*(pi/180);
Pitch_nuevo = Pitch*(pi/180);
Traslacion_nueva = Traslacion*(pi/180);
```

Las matrices descritas anteriormente son declaradas en Matlab

```
HTM_BASE_TO_TRAVEL = [cos(Traslacion_nueva) sin(Traslacion_nueva) 0 0;
                     -sin(Traslacion_nueva) cos(Traslacion_nueva) 0 0;
                      0 0 1 0;
                      0 0 0 1];

HTM_TRAVEL_TO_CW = [1 0 0 0;
0 cos(Elevacion_nueva) -sin(Elevacion_nueva) -cos(Elevacion_nueva)*L_w;
0 sin(Elevacion_nueva) cos(Elevacion_nueva) -sin(Elevacion_nueva)*L_w;
                    0 0 0 1];

HTM_TRAVEL_TO_HB  = [1 0 0 0;
0 cos(Elevacion_nueva) -sin(Elevacion_nueva) cos(Elevacion_nueva)*L_a;
0 sin(Elevacion_nueva) cos(Elevacion_nueva) sin(Elevacion_nueva)*L_a;
                      0 0 0 1];

HTM_HB_TO_FM =[cos(Pitch_nuevo) 0 -sin(Pitch_nuevo) cos(Pitch_nuevo)*L_h;
                0 1 0 0;
               sin(Pitch_nuevo) 0 cos(Pitch_nuevo) sin(Pitch_nuevo)*L_h;
                0 0 0 1];

HTM_HB_TO_BM=[cos(Pitch_nuevo) 0 -sin(Pitch_nuevo) -cos(Pitch_nuevo)*L_h;
              0 1 0 0;
             sin(Pitch_nuevo) 0 cos(Pitch_nuevo) -sin(Pitch_nuevo)*L_h;
             0 0 0 1];

      HTM_BASE_TO_CW = HTM_BASE_TO_TRAVEL*HTM_TRAVEL_TO_CW;
      HTM_BASE_TO_HB = HTM_BASE_TO_TRAVEL*HTM_TRAVEL_TO_HB;
      HTM_BASE_TO_FM = HTM_BASE_TO_HB* HTM_HB_TO_FM;
      HTM_BASE_TO_BM = HTM_BASE_TO_HB* HTM_HB_TO_BM;
```

Para graficar el cuerpo del bicoptero haremos uso de la función de matlab plot3 que permite graficar en tres dimensiones, usaremos las matrices respectivas para graficar la base del bicoptero que ira del punto de origen a el eje de traslación. Para el cuerpo trazaremos desde el contrapeso a el centro del armazón donde se encuentran las hélices, y para las hélices, trazaremos desde dicho centro a el centro de cada hélice.

```
%Base
plot3([0 HTM_BASE_TO_TRAVEL(1,4)], [0 HTM_BASE_TO_TRAVEL(2,4)], [-1
HTM_BASE_TO_TRAVEL(3,4)],'Color', [0.6 0.6 0.6], 'LineWidth', 3);

%Cuerpo del bicoptero
plot3([HTM_BASE_TO_CW(1,4),HTM_BASE_TO_HB(1,4)],
[HTM_BASE_TO_CW(2,4),HTM_BASE_TO_HB(2,4)],
[HTM_BASE_TO_CW(3,4),HTM_BASE_TO_HB(3,4)], 'k', 'LineWidth', 3);

%helice derecha
plot3([HTM_BASE_TO_HB(1,4),HTM_BASE_TO_FM(1,4)],[HTM_BASE_TO_HB(2,4),HTM_
BASE_TO_FM(2,4)],[HTM_BASE_TO_HB(3,4),HTM_BASE_TO_FM(3,4)], 'k',
'LineWidth', 3);

%helice izquierda
plot3([HTM_BASE_TO_HB(1,4),HTM_BASE_TO_BM(1,4)],[HTM_BASE_TO_HB(2,4),HTM_
BASE_TO_BM(2,4)],[HTM_BASE_TO_HB(3,4),HTM_BASE_TO_BM(3,4)], 'k',
'LineWidth', 3);
```

El siguiente boque de código crea un sistema de coordenadas de referencia que nos sirve para cambiar la escala de lo graficado y poder apreciar mejor el modelo 3d.

```
plot3([-1,1],[-1,-1],[-1,-1], 'k', 'LineWidth', 1);
plot3([-1,-1],[-1,1],[-1,-1], 'k', 'LineWidth', 1);
plot3([-1,-1],[-1,-1],[-1, 1], 'k', 'LineWidth', 1);
```

Guardaremos las posiciones de los extremos de las hélices para después graficarlas y así representar el alcance del espacio de trabajo que tiene el sistema.

```
Xb(i)=HTM_BASE_TO_FM(1,4);
Yb(i)=HTM_BASE_TO_FM(2,4);
Zb(i)=HTM_BASE_TO_FM(3,4);
Xf(i)=HTM_BASE_TO_BM(1,4);
Yf(i)=HTM_BASE_TO_BM(2,4);
Zf(i)=HTM_BASE_TO_BM(3,4);

plot3(Xb,Yb,Zb,':b');
plot3(Xf,Yf,Zf,':r');
```

El código completo y a detalle del modelado del espacio de trabajo del bicoptero se muestra en el apéndice al final de este reporte.

Para el modelado de la dinámica del sistema nos hemos ayudado de las ecuaciones que se encuentran en el archivo de maple que se facilita en la documentación de laboratorio del experimento. En especifico hemos hecho uso de la representación en espacio de estados del modelo ya linealizado, debido a que tenemos un sistema con 3 grados de libertad y se obtiene un modelo con tres ecuaciones diferenciales de segundo orden el espacio de estados contara con 6 estados dando una matriz A de 6x6, la representación matricial se muestra a continuación:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\dfrac{(L_w M_c - 2L_a M_f)g}{M_c L_w^2 + 2M_f L_h^2 + 2M_f L_a^2} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \dfrac{L_a K_m}{M_c L_w^2 + 2M_f L_a^2} & \dfrac{L_a K_m}{M_c L_w^2 + 2M_f L_a^2} \\ \dfrac{1}{2}\dfrac{K_m}{M_f L_h} & -\dfrac{1}{2}\dfrac{K_m}{M_f L_h} \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Para su implementación en simulink hemos hecho uso del bloque especifico para ello, primero hemos inicializado las matrices en un script y una vez hecho esto aplicadas en simulink.

```
L_w=0.470;
M_c=1.87;
L_a=0.660;
M_f=0.713;
g=9.81;
```

```
L_h=0.178;
K_m=0.1188;
Elevacion=50;
Traslacion=60;

a=(((L_w*M_c)-
(2*L_a*M_f))*g)/((M_c*L_w^2)+(2*M_f*L_h^2)+(2*M_f*L_a^2));
b=(L_a*K_m)/((M_c*L_w^2)+(2*M_f*L_a^2));
c=(1/2)*(K_m/(M_f*L_h));

A=[0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1;
   0 0 0 0 0 0;
   0 0 0 0 0 0;
   0 -a 0 0 0 0];

B=[0 0;
   0 0;
   0 0;
   b b;
   c -c;
   0 0];

C=[1 0 0 0 0 0;
   0 1 0 0 0 0;
   0 0 1 0 0 0];

D=[0 0;
   0 0;
   0 0];
```

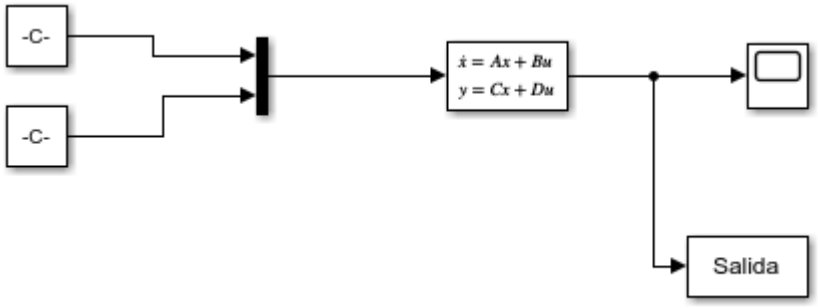

Figura 4 - Diagrama del espacio de estados en simulink

Las entradas al sistema son los dos respectivos voltajes de cada motor que impulsan a las hélices. Las salidas son las posiciones angulares de la elevación, el pitch y la traslación del bicoptero.

Para el bloque de control usamos el bloque Matlab function que permite incluir código, tal como se crea una función en un script de Matlab.

Las entradas de la funcion son la elevación que se desea lograr y la elevación retroalimentada desde la salida del espacio de estados, de igual manera se hace lo mismo con la Traslacion. Las salidas de la función son los dos voltajes ya procesados que entraran a cada motor respectivamente.

```
function [V_F,V_B]= fcn(Elevacion_actual, Elevacion_nueva,
Traslacion_actual, Traslacion_nueva)
```

Lo primero que se hace es obtener la diferencia entre la posición retroalimentada y la elevación que se desea tener.

```
delta_altura=Elevacion_actual-Elevacion_nueva;
```

Una vez hecho esto, se usan dos condicionales if dependiendo del resultado obtenido de la diferencia de elevaciones. Si esta es negativa quiere decir que la elevación es menor a la deseada por lo que hay que aplicar igual voltaje a ambos motores. Si la diferencia es positiva quiere decir que la elevación es mayor a la deseada por lo que hay que considerar la gravedad que hace bajar al bicoptero, esto último se aplica fuera del bloque con ayuda un switch. La razón de porque el voltaje asignado por la función cuando la elevación es mayor es negativo es para que dado este caso se aplique la gravedad y no el voltaje de salida. El bloque switch esta configurado para que cambie de canal cuando lo recibido sea menor a cero.

```
if delta_altura<0
    V_nuevoB=10;
    V_nuevoF=10;
end
if delta_altura>0
    V_nuevoB=-15;
    V_nuevoF=-15;
end

V_B=V_nuevoB;
V_F=V_nuevoF;
```

Una lógica similar se trato de aplicar al problema de la traslación, pero no se logro nada aceptable por lo que se descarto esa parte del control.

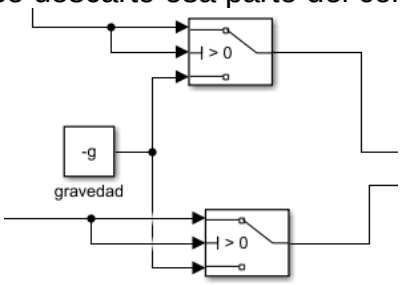

Figura 5 - Sección del diagrama en simulink que se encarga de la gravedad

Pruebas y resultados

A continuación, se muestran capturas de los resultados obtenidos del modelado en 3d de la cinemática del sistema (espacio de trabajo), la comparación del sistema simulado y real con entradas iguales (voltaje de 15 a cada motor) sin controlador y la comparación del sistema simulado y real con el controlador desarrollado.

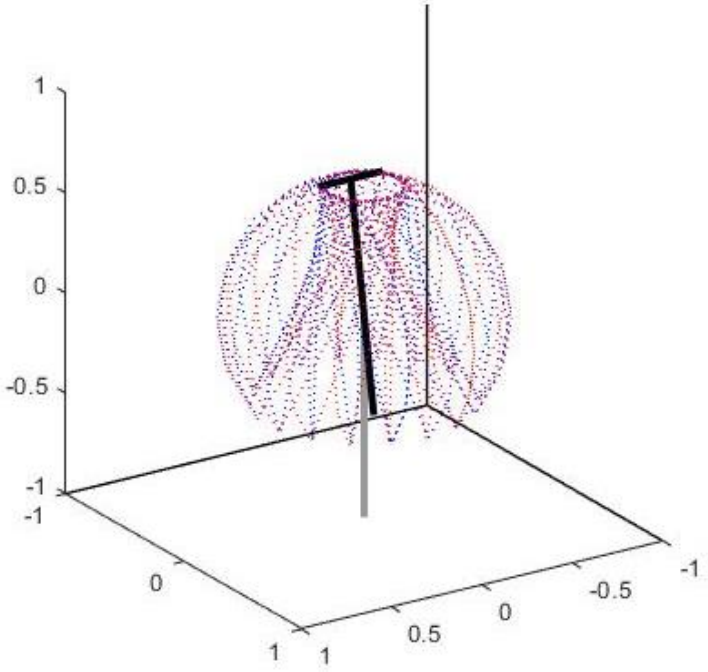

Figura 6 - Simulación tridimensional del espacio de trabajo del robot

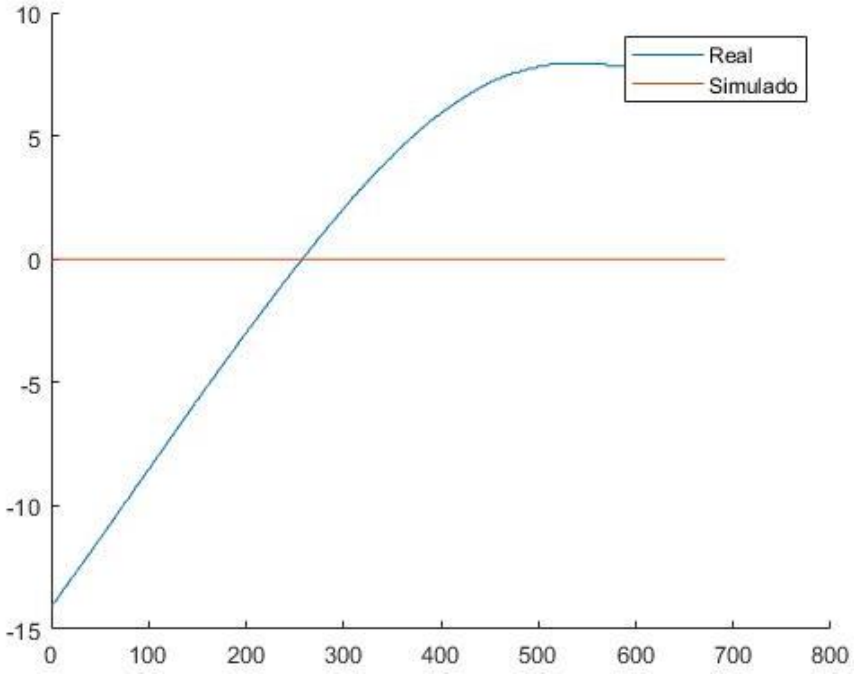

Figura 7 - Comparación del sistema simulado y real con entradass iguales sin control

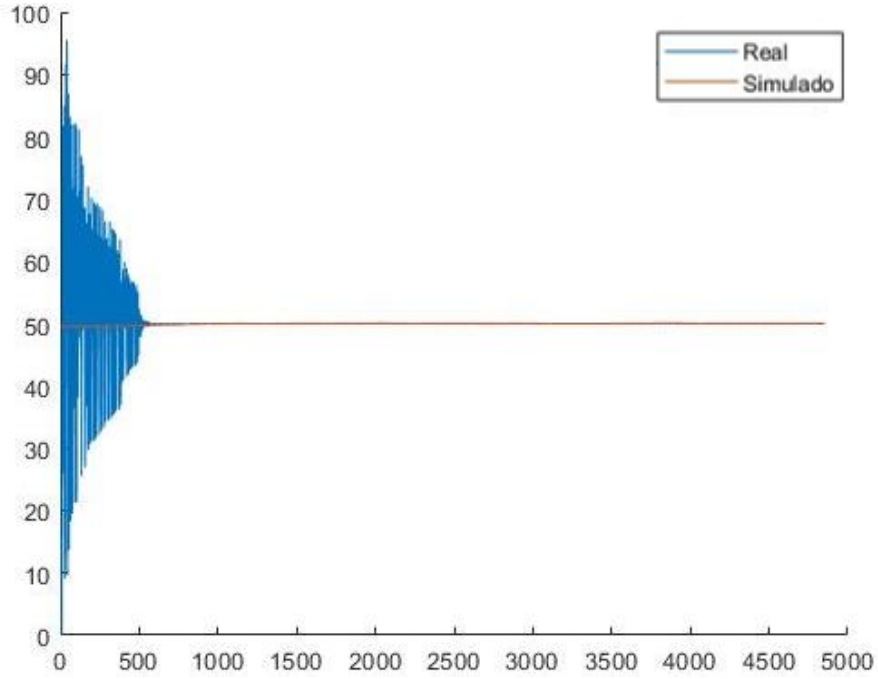

Figura 8 Comparación del sistema simulado y real con una elevación deseada de 50 grados

Conclusiones

La realización de esta actividad resulto complicada en varios aspectos sobre todo en el diseño del controlador, al final no logramos que el bicoptero tomara el desplazamiento deseado. En la simulación del sistema si se logro que se estabilizara en la elevación o altura deseada, pero una vez probado en la realidad con el experimento no funciono.

En cuanto a la cinemática del sistema, resulto más fácil ya que se tenia experiencia previa en este tema por lo visto en la catedra. Lo más valioso que nos llevamos de realizar esta actividad es la experiencia y la lección de que tratar con sistemas más complejos y útiles en la realidad que son mas cercanos a los que se manejan como profesional son bastante complejos y te das cuenta de la importancia del control como una de las herramientas mas importantes para el ingeniero en robótica.

Apéndices

Código usado para modelar el espacio de trabajo del robot

```
clear; clc;

L_w=0.470; %distancia del contrapeso al eje de elevacion
L_a=0.660; %distancia del eje de rotacion al extremo de las helices
L_h=0.178; %distancia del eje pitch a un motor.

Elevacion = 0;
Pitch = 0;
Traslacion = 0;
i=1;
for Traslacion = 0:20:360
    for Elevacion = -40:20:100

Elevacion_nueva = Elevacion*(pi/180);
Pitch_nuevo = Pitch*(pi/180);
Traslacion_nueva = Traslacion*(pi/180);

%Matrices de coordenadas cartesianas de los cuerpos en movimiento.
HTM_BASE_TO_TRAVEL = [cos(Traslacion_nueva) sin(Traslacion_nueva) 0
0; %de la base al eje de traslacion
                      -sin(Traslacion_nueva) cos(Traslacion_nueva) 0 0;
                       0 0 1 0;
                       0 0 0 1];

HTM_TRAVEL_TO_CW = [1 0 0
0;                                              %del eje de traslacion al
contrapeso
                    0 cos(Elevacion_nueva) -sin(Elevacion_nueva) -
cos(Elevacion_nueva)*L_w;
                    0 sin(Elevacion_nueva) cos(Elevacion_nueva) -
sin(Elevacion_nueva)*L_w;
                    0 0 0 1];

HTM_TRAVEL_TO_HB  = [1 0 0
0;                                              %del eje de traslacion al
centro de las helices
                     0 cos(Elevacion_nueva) -sin(Elevacion_nueva)
cos(Elevacion_nueva)*L_a;
                     0 sin(Elevacion_nueva) cos(Elevacion_nueva)
sin(Elevacion_nueva)*L_a;
                     0 0 0 1];

HTM_HB_TO_FM = [cos(Pitch_nuevo) 0 -sin(Pitch_nuevo)
cos(Pitch_nuevo)*L_h;  %del centro de las helices al motor de enfrente
                0 1 0 0;
                sin(Pitch_nuevo) 0 cos(Pitch_nuevo) sin(Pitch_nuevo)*L_h;
                0 0 0 1];

HTM_HB_TO_BM = [cos(Pitch_nuevo) 0 -sin(Pitch_nuevo) -
cos(Pitch_nuevo)*L_h; %del centro de las helices al motor de atras
              0 1 0 0;
```

```
            sin(Pitch_nuevo) 0 cos(Pitch_nuevo) -sin(Pitch_nuevo)*L_h;
            0 0 0 1];

HTM_BASE_TO_CW = HTM_BASE_TO_TRAVEL*HTM_TRAVEL_TO_CW;     %de la base al
contrapeso

HTM_BASE_TO_HB = HTM_BASE_TO_TRAVEL*HTM_TRAVEL_TO_HB;     %de la base al
centro de las helices

HTM_BASE_TO_FM = HTM_BASE_TO_HB* HTM_HB_TO_FM;            %de la base al
motor frontal

HTM_BASE_TO_BM = HTM_BASE_TO_HB* HTM_HB_TO_BM;            %de la base al
motor de atras

hold off
plot3([0 HTM_BASE_TO_TRAVEL(1,4)], [0 HTM_BASE_TO_TRAVEL(2,4)], [-1
HTM_BASE_TO_TRAVEL(3,4)],'Color', [0.6 0.6 0.6], 'LineWidth', 3); %eje de
base del bicoptero
hold on

plot3([HTM_BASE_TO_CW(1,4),HTM_BASE_TO_HB(1,4)],
[HTM_BASE_TO_CW(2,4),HTM_BASE_TO_HB(2,4)],
[HTM_BASE_TO_CW(3,4),HTM_BASE_TO_HB(3,4)], 'k', 'LineWidth', 3); %cuerpo
del bicoptero

plot3([HTM_BASE_TO_HB(1,4),HTM_BASE_TO_FM(1,4)],[HTM_BASE_TO_HB(2,4),HTM_
BASE_TO_FM(2,4)],[HTM_BASE_TO_HB(3,4),HTM_BASE_TO_FM(3,4)], 'k',
'LineWidth', 3); %helice derecha
plot3([HTM_BASE_TO_HB(1,4),HTM_BASE_TO_BM(1,4)],[HTM_BASE_TO_HB(2,4),HTM_
BASE_TO_BM(2,4)],[HTM_BASE_TO_HB(3,4),HTM_BASE_TO_BM(3,4)], 'k',
'LineWidth', 3); %helice izquierda

%tres ejes (x,y,z) para reescalar lo ya graficado
plot3([-1,1],[-1,-1],[-1,-1], 'k', 'LineWidth', 1);
plot3([-1,-1],[-1,1],[-1,-1], 'k', 'LineWidth', 1);
plot3([-1,-1],[-1,-1],[-1, 1], 'k', 'LineWidth', 1);

i=i+1;
Xb(i)=HTM_BASE_TO_FM(1,4);
Yb(i)=HTM_BASE_TO_FM(2,4);
Zb(i)=HTM_BASE_TO_FM(3,4);
Xf(i)=HTM_BASE_TO_BM(1,4);
Yf(i)=HTM_BASE_TO_BM(2,4);
Zf(i)=HTM_BASE_TO_BM(3,4);

plot3(Xb,Yb,Zb,':b');
plot3(Xf,Yf,Zf,':r');

axis equal;

pause(0.002);

    end
```

```
end
```

Código usado para la simulación de la dinámica del sistema en Simulink

```
clear; clc;

L_w=0.470;
M_c=1.87;
L_a=0.660;
M_f=0.713;
g=9.81;
L_h=0.178;
K_m=0.1188;
Elevacion=50;
Traslacion=60;

a=(((L_w*M_c)-(2*L_a*M_f))*g)/((M_c*L_w^2)+(2*M_f*L_h^2)+(2*M_f*L_a^2));
b=(L_a*K_m)/((M_c*L_w^2)+(2*M_f*L_a^2));
c=(1/2)*(K_m/(M_f*L_h));

A=[0 0 0 1 0 0;
   0 0 0 0 1 0;
   0 0 0 0 0 1;
   0 0 0 0 0 0;
   0 0 0 0 0 0;
   0 -a 0 0 0 0];

B=[0 0;
   0 0;
   0 0;
   b b;
   c -c;
   0 0];

C=[1 0 0 0 0 0;
   0 1 0 0 0 0;
   0 0 1 0 0 0];

D=[0 0;
   0 0;
   0 0];
```

Codigo de la funcion del bloque de Simulink encargado del controlador

```
function [V_F,V_B]= fcn(Elevacion_actual, Elevacion_nueva,
Traslacion_actual, Traslacion_nueva)

V_nuevoB=0;
V_nuevoF=0;

delta_altura=Elevacion_actual-Elevacion_nueva;
delta_traslacion=Traslacion_actual-Traslacion_nueva;

if delta_altura<0
```

```
    V_nuevoB=10;
    V_nuevoF=10;
end
if delta_altura>0
    V_nuevoB=-15;
    V_nuevoF=-15;
end

V_B=V_nuevoB;
V_F=V_nuevoF;

End
```

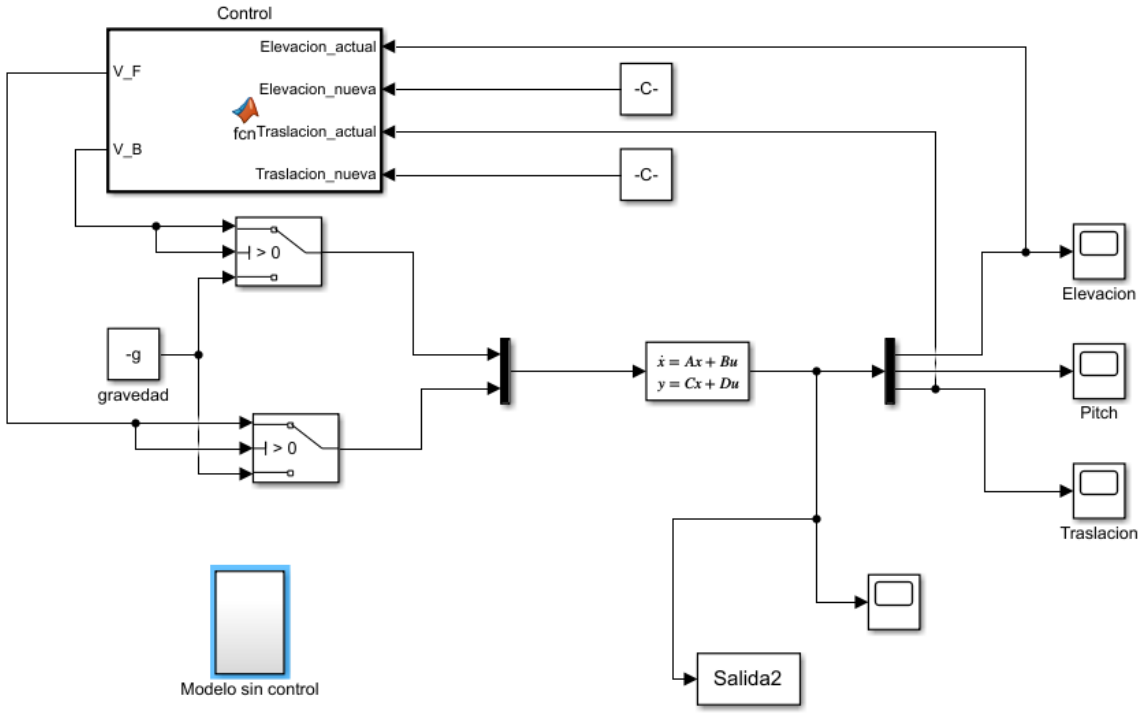

Figura 9 - Diagrama general de Simulink

Referencias

BARRIENTOS, ANTONIO *et al*. Fundamentos de Robótica (Segunda edición). McGraw-Hill. Mexico. 2007

Craig, John J. Robótica (Tercera edición). Pearson-Prentice Hall. Mexico. 2006