PROJECT REPORT

Quadcopter Modelling

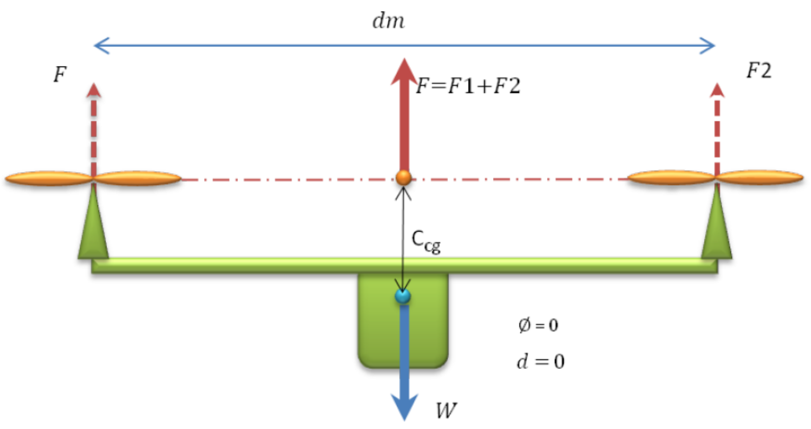
INDEPENDENT RESEARCH PROJECT, FALL 2017

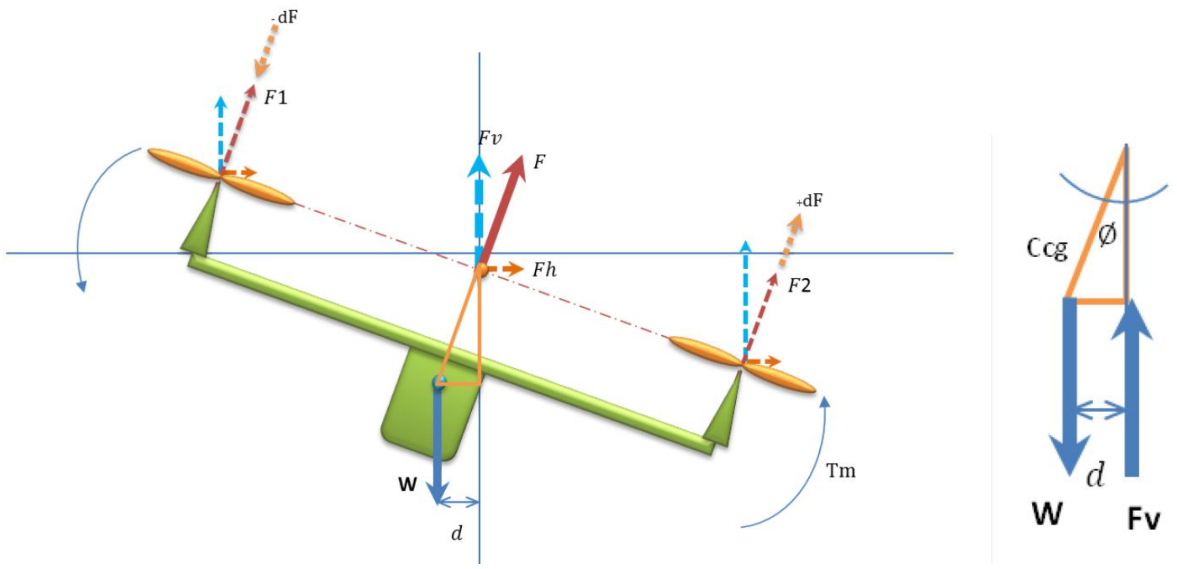
Siddharth Rajagopal
Mechanical Engineering, University of Washington

Contents

1. Modelling of quadcopter in 2D

Mathematical modelling of the quadcopter dynamics was done in a 2D vertical plane. The model consisted of the diagonal cross-section of the quadcopter with a thrust motor attached at each of the ends. A Free Body Diagram of the quadcopter about a small tilt angle was analyzed using Newtonian mechanics to get the governing equations for the position (x, y) and attitude ($\emptyset$) of the system. The actuator dynamics of the dc motors and drag coefficients were added to the system later. This model was simulated in MATLAB using ode45, generating an animation of the quadcopter dynamics based on the inputs of angular velocity of each motor (*w1*, *w2*).

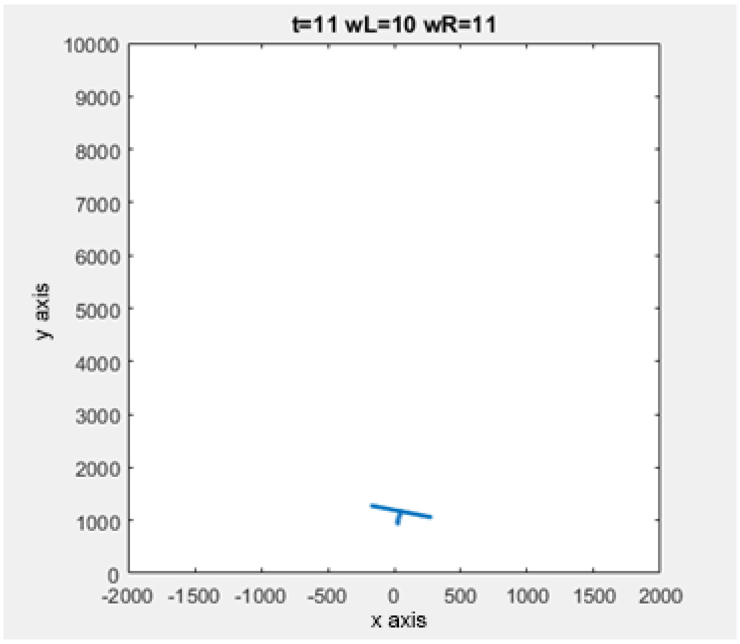




$$m\ddot{x} = (F1 + F2)\ \cos\emptyset$$

$$m\ddot{y} = \ (F1 + F2)\ \sin\emptyset$$

$$J\ddot{\emptyset} = (F1 - F2)\frac{d_m}{2}$$



Animation of the quadcopter shifting towards right when $w_{left} > w_{right}$

Program:

```
clc, clear all, close all
m=3;  %kg mass of quad
g=9.8;
d=.3;   %length in meters
h=.05;
J = (.5*d)^2*(2/5)*m + ((1/12)*m*(((.5*d)^2)*3 + h^2));

% w_i is considered to be the input for modelling presently.

wL = [10 10 10 10 10 10.003 10 10 10 10 10 10 10 10 10]; % left motor
wR = [10 10 10 10 10 10 10 10 10 10 10 10 10 10 10]; %right motor

% fi = k * w_i^2; where f_i is the thrust by each motor and w_i is its
%angular velocity.
f1 = wR.^2 * .5; % [ k=. ] and f_i is the thrust by individual motor
f2 = wL.^2 * .5;

% Torque generated from motors can thus be calculated
T = 0.5*d*(f2-f1); % net motor torque

% Animation
h = animatedline;
h=plot(0,0,'MarkerSize',3,'Marker','.','LineWidth',2);
range=2000; axis([-range range 0 5*range]); axis square;

%ODE :

% state vector : [phi theta v xposition w yposition] for ODE45
% t shows the outputs at the described time samples, the time interval
% is not t, time interval is taken by ode45 as per the trajectory.
```

```
q0 = [0 0 0 0 0 0]; % initial condition

dt = 1;
tspan = 1:dt:(length(wR));
% ode45 is given by [t,y] = ode45(func,tspan,y0)
[t,q]=ode45(@(t,q)quadode(t,q,T,f1,f2,d,J,m,g),tspan,q0);


X = q(: ,4);
Y = q(: ,6);
A = q(: ,2);

% Animation Display
Xcoord = [X-0.5*(500)*cos(A),X+0.5*(500)*cos(A)];
Ycoord = [Y+0.5*(500)*sin(A),Y-0.5*(500)*sin(A)];
%annotation('doublearrow',Xcoord,Ycoord);
for i=1:(length(wR))
set(h,'XData',Xcoord(i,:),'YData',Ycoord(i,:));
title(num2str(i));
title(['t=' num2str(i) ' wL=' num2str(wL(i)) ' wR=' num2str(wL(i))])
    drawnow;
    pause(.35);
 end

function qdot = quadode(t,q,T,f1,f2,d,J,m,g)
%q is the initial condition state vector : [phi theta v xposition w
yposition] for ODE45

%fix(t) takes the integer value from decimal value
    omprime = (1/J)*T(fix(t));
    thprime = q(1); %from initial condition
    vprime = (1/m)*(f1(fix(t))+f2(fix(t)))*sin(q(2));
    xprime = q(3);
    wprime = (1/m)*((f1(fix(t))+f2(fix(t)))*cos(q(2))-m*g);
    yprime = q(5);

qdot = [omprime; thprime; vprime; xprime; wprime; yprime];
```

2. Modelling of a pendulum attached to quadcopter in 2D

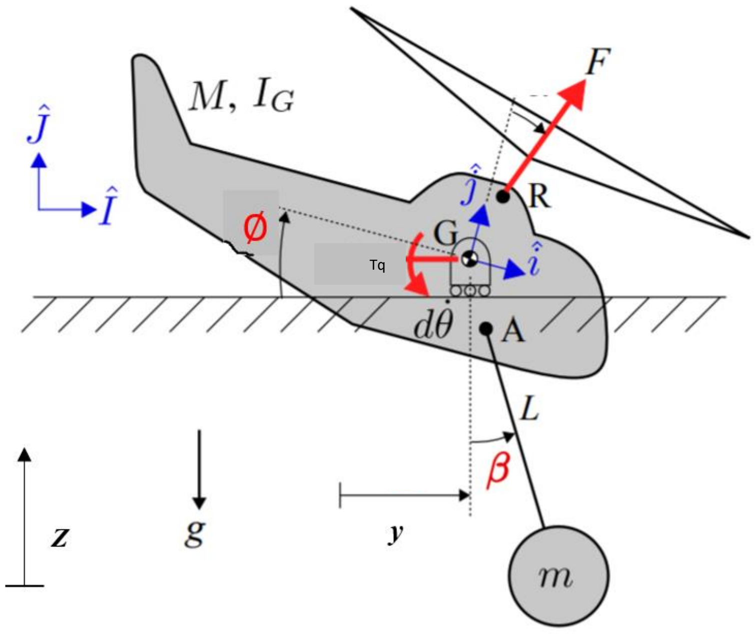
The next task was mathematically modelling the dynamics of the quadcopter with a suspended pendulum at its center in a 2D plane. I used Lagrangian mechanics to solve this problem with the Free Body Diagram having the pendulum at an orientation of '*phi*' degrees from vertical axis. The kinetic and potential energies of the system was used to calculate the Lagrangian equation to solve for the governing equations of 4 generalized coordinates of *x, y, theta* and *phi*. An animation of the nonlinear system dynamics from inputs *w1* and *w2* was generated in MATLAB using ode45.

The Lagrange's equation is given by:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial V}{\partial q_j} = Q_j, \qquad j = 1, 2, \ldots, N$$

$$T = \frac{1}{2}M\left(\vec{v}_G \cdot \vec{v}_G\right) + \frac{1}{2}I_G\dot{\theta}^2 + \frac{1}{2}m\left(\vec{v}_m \cdot \vec{v}_m\right) + \frac{1}{2}I_m\dot{\beta}^2$$

$$V = mg\left(-L\cos\beta - x_A\sin\theta + y_A\cos\theta\right)$$

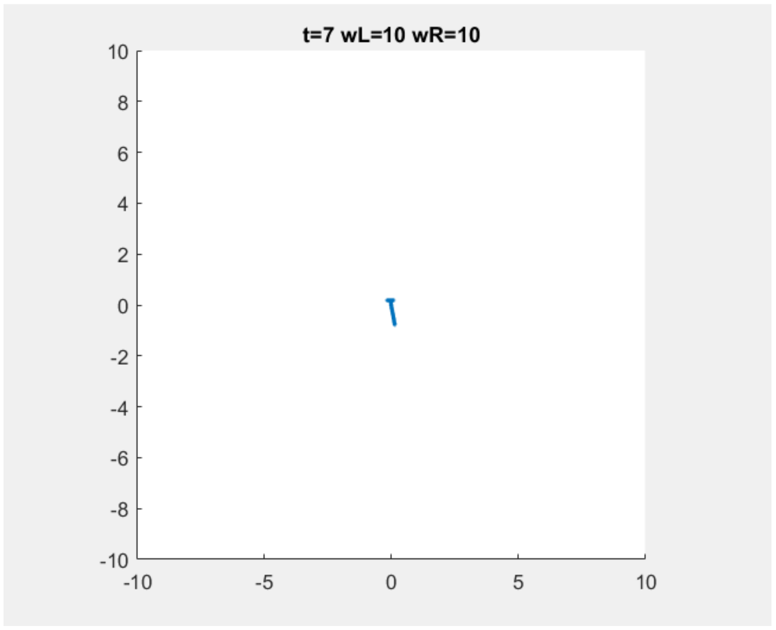


$$(M+m)\ddot{y} - ml\ddot{\beta}\cos(180-\beta) - ml\dot{\beta}^2\sin(180-\beta) = (F_2 + F_1)\cos\emptyset$$

$$(M+m)\ddot{z} + (M+m)g = (F_2 + F_1)\sin\emptyset$$

$$ml^2\ddot{\beta} - ml\cos(\beta - 180)\ddot{y} = 0$$

$$\ddot{\emptyset} = \frac{1}{2}d(F_2 - F_1)$$



Animation of the quadcopter shifting towards left with deflection in pendulum

<u>Program:</u>

```
clc, clear all, close all;
syms M m yq zq zp phi bet dphi dbet dyq dzq  In dphi l g F Tq

dyp = dyq-(l*dbet);
dzp = dzq;
%T = 0.5*M*(dyq^2+dzq^2) + 0.5*m*(dyp^2+dzp^2) +0.5*I*dphi^2;
T = 0.5*M*(dyq^2+dzq^2) + 0.5*m*(dyq^2-(2*l*dbet*dyq*cos(180-
bet))+(l*dbet)^2+dzq^2) +0.5*In*dphi^2;

%zp = zq-l*cos(bet);
U = M*g*zq+m*g*(zq-l);

L=T-U;
Ly=diff(L,yq)
% Ly = 0
Ldy = diff(L,dyq)
eq1 = M*dyq + (m*(2*dyq - 2*dbet*l*cos(bet - 180)))/2

Lz=diff(L,zq)
%Lz =  - M*g - g*m
Ldz=diff(L,dzq)
%Ldz = M*dzq + dzq*m
%
Lbet=diff(L,bet)
%Lz = dbet*dyq*l*m*sin(bet - 180)
Ldbet=diff(L,dbet)
Ldbet =(m*(2*dbet*l^2 - 2*dyq*cos(bet - 180)*l))/2

Lphi=diff(L,phi)
%Lphi =0
Ldphi=diff(L,dphi)
%Ldphi =I*dphi
```

```
eA=[(M+m) 0 -m*l*cos(180-bet) 0;
0 (M+m) 0 0;
-m*l*cos(bet-180) 0 m*(l^2) 0;
0 0 0 In]
% %
 eC=[F*sin(phi)+m*l*dbet^2*sin(180-bet);
 F*cos(phi)-(M+m)*g;
 0;
 Tq]

eB=inv(eA)*eC

d2yq = eB(1)
d2zq = eB(2)
d2bet = eB(3)
d2phi = eB(4)


m_q=2.3;  %kg mass of quad
m_p=.51;
l=.1; %length of pendulum
g=9.8;
d=.15;   %length in meters
h=.2;
In = 0.00804;%(.5*d)^2*(2/5)*m_q + ((1/12)*m_q*(((.5*d)^2)*3 + h^2));

% w_i is considered to be the input for modelling presently.

wL = [10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10]; % left motor
wR = [10 10 10 10.05 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10]; %right motor

% fi = k * w_i^2; where f_i is the thrust by each motor and w_i is its
%angular velocity.
k=1.175*1.0011111
f1 = wR.^2 * .1175920*k; % [ k=. ] and f_i is the thrust by individual motor
f2 = wL.^2 * .1175920*k;
%kbal=.1175920
%[nd2y;ndy;nd2z;ndz;nd2bet;ndbet;nd2phi;ndphi]
q0 =[0;0;0;0;0;0;0;0];
dt = 1;
tspan = 1:dt:(length(wR));
% ode45 is given by [t,y] = ode45(func,tspan,y0)
[t,q]=ode45(@(t,q)quadpenheli2(t,q,f1,f2,d,In,m_q,m_p,g,l),tspan,q0);
%aero drag
for i=1:15
T_d(i) = 0.5*1.225*0.234*(sqrt(q(i,3)*q(i,3)+q(i,5)*q(i,5)));
end


Y = q(: ,2);
Z = q(: ,4);
P = q(: ,6);
A = q(: ,8);
```

```
h1=line(0,0,'MarkerSize',3,'Marker','.','LineWidth',2);
h2=line(0,0,'MarkerSize',3,'Marker','.','LineWidth',2);

range=10; axis([-range range -range range]); axis square;

% Animation Display
Xcoord = [Y-0.5*(.3)*cos(A),Y+0.5*(.3)*cos(A)];
Ycoord = [Z+0.5*(.3)*sin(A),Z-0.5*(.3)*sin(A)];
XPcoord = [Y,Y+l*(10)*sin(P)];
YPcoord = [Z,Z-l*(10)*cos(P)];

for i=1:(length(wR))
set(h1,'XData',Xcoord(i,:),'YData',Ycoord(i,:));
set(h2,'XData',XPcoord(i,:),'YData',YPcoord(i,:));
title(num2str(i));
title(['t=' num2str(i) ' wL=' num2str(wL(i)) ' wR=' num2str(wL(i))])
    drawnow;
    pause(.35);
 end

function qdot = quadpenheli2(t,q,f1,f2,d,In,M,m,g,l)
%q is the initial condition state vector :
F=f1(fix(t))+f2(fix(t));
Tq= 0.5*d*(f2(fix(t))-f1(fix(t)));
% F_d = 0.5*1.225*0.234*(sqrt(q(3)*q(3)+q(5)*q(5))); % aerodynamic Drag force
% T = T_m;
%fix(t) takes the integer value from decimal value

%[ dy  y   dz    z    dbet   bet   dphi  phi] for ODE45
%[ q1  q2  q3    q4    q5     q6    q7   q8 ]
dyq=q(1);
yq=q(2);
dzq=q(3);
zq=q(4);
dbet=q(5);
bet=q(6);
dphi=q(7);
phi=q(8);


nd2y =(- l*m*sin(bet - 180)*dbet^2 + F*sin(phi))/(M + m - m*cos(bet -
180)^2);
ndy = q(1);
nd2z = (F*cos(phi) - g*(M + m))/(M + m);
ndz= q(3);
nd2bet = (cos(bet - 180)*(- l*m*sin(bet - 180)*dbet^2 + F*sin(phi)))/(M*l +
l*m - l*m*cos(bet - 180)^2);
ndbet = q(5);
nd2phi = Tq/In;
ndphi = q(7);

qdot = [nd2y;ndy;nd2z;ndz;nd2bet;ndbet;nd2phi;ndphi];
```

3. Conversion of model to State-Space

The nonlinear governing equations were linearized about an equilibrium position with the net thrust force nullifying the effect of weight of the system. The linearized Jacobian matrix was calculated to get the A and B matrices of the system. An animation of the dynamics of this state space system was simulated in MATLAB based on inputs *u1*(thrust) and *u2*(torque).

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}\vec{u}$$

$$\vec{x} = [\dot{y}\,, y\,, \dot{z}\,, z\,, \dot{\beta}\,, \beta, \dot{\emptyset}\,, \emptyset]$$

$$\vec{u} = [F, Tq]$$

Program:

```
A = simplify(jacobian([d2yq,dyq,d2zq,dzq,d2bet,dbet,d2phi,dphi],
[dyq,yq,dzq,zq,dbet,bet,dphi,phi]))

B = simplify(jacobian([d2yq,dyq,d2zq,dzq,d2bet,dbet,d2phi,dphi], [F,Tq]))

clc, clear all, close all
d=.3;   %length in meters
h=.05;
m_q= .5
J = (.5*d)^2*(2/5)*m_q + ((1/12)*m_q*(((.5*d)^2)*3 + h^2));
l=.05;

wL =3* [8 9 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10]; % left motor
wR =3* [8 9 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10]; %right motor

% fi = k * w_i^2; where f_i is the thrust by each motor and w_i is its
%angular velocity.
f1 = wR.^2 * .5; % [ k=. ] and f_i is the thrust by individual motor
f2 = wL.^2 * .5;

dt = 1;
tspan = 1:dt:(length(wR));


A =[ 0         0         0         0   -0.0000    0.0020         0   10.4833;
     1         0         0         0         0         0         0         0;
     0         0         0         0         0         0         0   -0.0098;
     0         0    1.0000         0         0         0         0         0;
     0         0         0         0    0.0002   -0.0956         0  -62.8221;
     0         0         0         0    1.0000         0         0         0;
     0         0         0         0         0         0         0         0;
     0         0         0         0         0         0    1.0000         0]

B =[0.0004         0;
         0         0;
    0.3559         0;
         0         0;
   -0.0023         0;
         0         0;
         0  124.3781;
         0         0]
```

```matlab
X0 = [0;50;0;50;0;0;0;0]

F=f1+f2;
Tq= 0.5*d*(f2-f1);

U = [F' Tq']

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

D=0;

sys=ss(A,B,C,D);

co = ctrb(sys);
controllability = rank(co)

[q,t,x]=lsim(sys,U,tspan,X0);

Y = q(: ,2);
Z = q(: ,4);
P = q(: ,6);
A = q(: ,8);

h1=line(0,0,'MarkerSize',3,'Marker','.','LineWidth',2);
h2=line(0,0,'MarkerSize',3,'Marker','.','LineWidth',2);
%range=10; axis([-range range -range range]); axis square;

% Animation Display
Xcoord = [Y-0.5*(.3)*cos(A),Y+0.5*(.3)*cos(A)];
Ycoord = [Z+0.5*(.3)*sin(A),Z-0.5*(.3)*sin(A)];
XPcoord = [Y,Y+l*(10)*sin(P)];
YPcoord = [Z,Z-l*(10)*cos(P)];
%annotation('doublearrow',Xcoord,Ycoord);
for i=1:(length(wR))
set(h1,'XData',Xcoord(i,:),'YData',Ycoord(i,:));
% hold on;
%set(h,'XPData',Xcoord(i,:),'YPData',Ycoord(i,:));
set(h2,'XData',XPcoord(i,:),'YData',YPcoord(i,:));
title(num2str(i));
title(['t=' num2str(i) ' wL=' num2str(wL(i)) ' wR=' num2str(wL(i))])
    drawnow;
    pause(.35);
 end
```

4. <u>**Applying Control to the System**</u> (Ongoing)

An LQR controller is being designed based on the linear state space model to make the pendulum follow a trajectory. This controller will be implemented on the non-linear model in stage-2 for analysis. A transfer function of the controller will be generated which can be added to the physical quadcopter controller.

Eventually a pendulum will be attached to the physical quadrotor with its motion restricted to a 2D plane for further analysis.