2D-MAV Simulation Exercise

João Filipe Silva
MP-282: Dynamic Modeling and Control of MAVs

March 23, 2020

Abstract

This document describes the first simulation exercise for the Dynamic Modeling and Control of MAVs class. Based on the content given in the lectures, it was requested to formulate an algorithm capable of representing a two-rotor MAV (Multirotor Aerial Vehicle) with one degree of freedom, in a two-dimensional environment. The vehicle should follow a route passing through previously defined waypoints and avoiding two obstacles of known geometry. At the end of the simulation, plots were generated to illustrate the vehicle's behavior, concerning the actual path traveled, the forces exerted by the rotors, the pitch angle for horizontal movement and the angular velocity of the MAV.

1 Introduction

A Multirotor Aerial Vehicle (MAV) is an unmanned flying aircraft. Its applications are distributed over different markets, such as mapping farmlands, cargo delivery, film-making, search and rescue, surveillance and others. To function in a three-dimensional scenario, MAV's usually are composed of 2 or more pairs of rotors, enabling them to move in all directions stably. But for this report, a simplified model is going to be used, with only one pair of rotors, and flying in a two-dimensional environment. The fictitious MAV is illustrated in Figure 1.

Initially, it's necessary to assume a Cartesian Coordinate System (CCS) on the ground, referred here as S_G, and another on the MAV's center of mass,

called S_B. This allows the MAV translation and rotation movements to be described referring to a stationary observer on the ground.

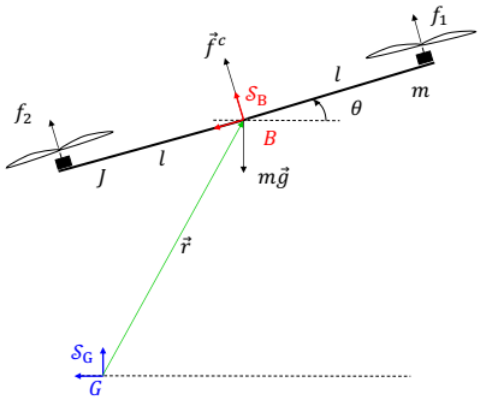


Figure 1: Fictitious MAV in 2D [1]

According to Figure 1, $f1$ and $f2$ are the rotors individual forces, l is the MAV arm length, J is its moment of inertia, m its mass, θ its pitch angle, and $\vec{f^c}$ the resulting force vector. It is important to note that, as the MAV is a rigid body and the axes of the rotors have zero degrees of freedom, the resulting force vector $\vec{f^c}$ will always be perpendicular to the MAV's body. In other words, $\vec{f^c}$ is always located over the vertical upward component of S_B.

From the aforementioned data, it is possible to describe the MAV's equations of motion.

1.1 Translational Motion

Adopting r as the position of the MAV in the 2D simulation space, which will be from now on referred to as the arena, it is possible to describe the vehicle's velocity as $\dot{r}$ and its acceleration as $\ddot{r}$. Using Newton's Second Law, the MAV's translational motion can be described in S_G by:

$$\ddot{r_G} = -ge_2 + \frac{1}{m}\mathbf{D}^{G/B}(f_1 + f_2)e_2 \tag{1}$$

where g is the gravity acceleration, $e_2 \triangleq [0\ 1]^T$ is the unitary vertical vector, and $\mathbf{D}^{G/B}$ is the attitude matrix of S_G with relation to S_B [1].

1.2 Rotational Motion

Adopting θ as the pitch angle of the MAV in the arena, it is possible to describe the vehicle's angular velocity as $\dot{\theta}$ and its angular acceleration as $\ddot{\theta}$. Using Euler's Second Law, the MAV's rotational motion can be described by:

$$\ddot{\theta} = \frac{1}{J}\tau^c \tag{2}$$

where τ^c is the resulting torque acting on the vehicle.

The relation between θ and the attitude matrix $\mathbf{D}^{G/B}$ is given by:

$$\mathbf{D}^{G/B} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \tag{3}$$

With the MAV's equations of motion described, it is time to design the vehicle's controller.

2 Flight Control

The flight control is composed of three main blocks: Position Controller (PC), Attitude Controller (AC), and Control Allocation (CA), as illustrated by Figure 2. Besides, the Path Planning (PP) and Guidance (GD) blocks are added to define the task the MAV will have to perform. Each of the blocks will be detailed in its own subsection. It is important to assume that the AC block has a much faster performance than the PC block, to guarantee a stable and efficient flight.

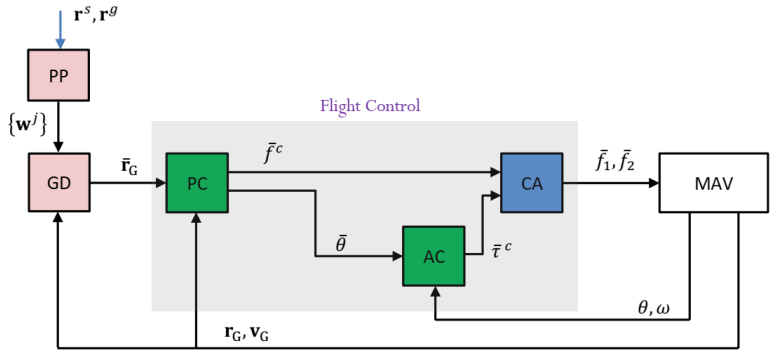


Figure 2: Block Diagram of the MAV Control Scheme [1]

2.1 Attitude Controller (AC)

Initially, it helps to set the physical limits of the MAV's rotors. Assume that $f^{min} \leq f_1, f_2 \leq f^{max}, f^{max} > f^c$ and $f^{min} > 0$. In order to respect the actuation symmetry of f_1 and f_2 with respect to $f^c/2$, to ensure that there is no reaction torque to spin the vehicle, it is vital to assume that the rotor forces are bounded by $[f^{min}, f^c - f^{min}]$. Consequently, the maximal torque bound is

$$\tau^{max} = (f^c - 2f^{min})l \tag{4}$$

By adopting a proportional-derivative control law to approximate equation 2 to the closed-loop attitude dynamics in $\ddot{\theta} + K_2\dot{\theta} + K_1\theta = K_1\bar{\theta}$, where $\bar{\theta}$ is the pitch angle command, the torque command can be defined as

$$\bar{\tau^c} = sat_{[-\tau^{max}, \tau^{max}]}(JK_1(\bar{\theta} - \theta) - JK_2\dot{\theta}) \tag{5}$$

2.2 Position Controller (PC)

In order to prevent large inclinations and actuator saturation, the resulting force vector f^c will be bounded by

$$\mathbf{f}^{min} = \begin{bmatrix} -2f^{min}tan(\theta^{max}) \\ 2f^{min} \end{bmatrix} \quad \mathbf{f}^{max} = \begin{bmatrix} 2f^{min}tan(\theta^{max}) \\ 2f^{max} \end{bmatrix} \tag{6}$$

By adopting a proportional-derivative control law to approximate equation 1 to the closed-loop attitude dynamics in $\ddot{r}_G + K_4\dot{r}_G + K_3 r_G = K_3\bar{r}_G$, where $\bar{r}_G$ is the position command, the resulting force command can be defined as

$$\bar{\mathbf{f}}_G^c = sat_{[\mathbf{f}^{min}, \mathbf{f}^{max}]}(mK_3(\bar{r}_G - r_G) - mK_4\dot{r}_G + mge_2 \tag{7}$$

The resulting force command will also be used to calculate the pitch angle command with the following equation

$$\bar{\theta} = atan\frac{e_1^T\bar{f}_G^c}{e_2^T\bar{f}_G^c} \tag{8}$$

where $e_1 \triangleq [1\ 0]^T$.

2.3 Control Allocation (CA)

With the outputs of the PC and AC blocks, the individual rotor forces can be described by

$$\begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/(2l) \\ 1/2 & -1/(2l) \end{bmatrix} \begin{bmatrix} \bar{f}^c \\ \bar{\tau}^c \end{bmatrix} \tag{9}$$

where $\bar{f}^c = ||\bar{\mathbf{f}}^c_G||$.

2.4 Guidance (GD) and Path Planning (PP)

The Guidance block generates a desired trajectory $\bar{r}_G$ from a sequence of given waypoints $\{\mathbf{w}^j\}$. The equation that calculates the position command is given by

$$\bar{r}_G = r_G + K^g(w^j - r_G) \tag{10}$$

where K^g is a gain diagonal matrix.

The Path Planning block, in this simple scenario, will be modelled by manually choosing waypoints that trace the apparently shortest path between the starting point r^s and goal point r^g, minding that the vehicle will still be avoiding the obstacles.

3 Simulation and Results

To begin, the simulation parameters must be defined.

Table 1: Simulation parametrization.

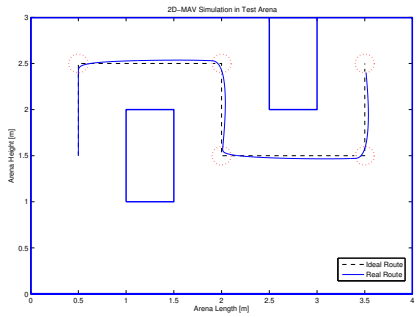
Par.	Def.	Value	Par.	Def.	Value
m	Mass	$1kg$	l	Arm Length	$0.2m$
J	Moment of Inertia	$0.02kgm^2$	g	Gravity	$9.81m/s^2$
T_s	Sampling Time	$0.01s$	K^g	Guidance Gain	$1.5I_2$
K_1	AC Prop. Gain	6	K_2	AC Deriv. Gain	3
K_3	AP Prop. Gain	$0.2I_2$	K_4	PC Deriv. Gain	$0.8I_2$
f_i^{max}	Max Indv. Force	mg	f_i^{min}	Min Indv. Force	$0.375mg$
r^s	Starting Point	$(0.5, 1.5)\ m$	r^g	Goal Point	$(3.5, 2.5)\ m$

The chosen waypoints are $\{w^j\} = (0.5, 2.5), (2, 2.5), (2, 1.5), (3.5, 1.5), (3.5, 2.5)$. The chosen Proportional and Derivative gains result in a settling

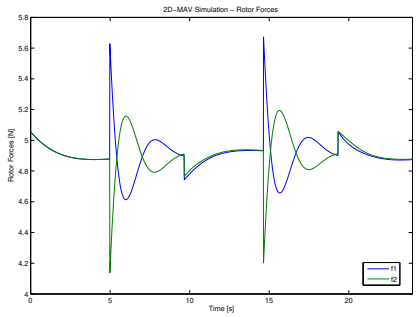
time of $st_{AC} = 1.9972s$ for the Attitude Control and $st_{PC} = 7.4893s$ for the Position Control, satisfying the condition presented in the beginning of Section 2.

The algorithm is enclosed in a loop to iterate every period of T_s seconds until the MAV gets inside a circular vicinity of radius $0.1m$ from the goal point r^g. Besides all the aforementioned equations being converted to a MATLAB algorithm, it is necessary to use a numerical method to solve the differential equations 1 and 2. The method chosen was the classic Runge-Kutta method, or commonly known as RK4 [2]. The full algorithm is added in the end of this report, in the Appendix section.

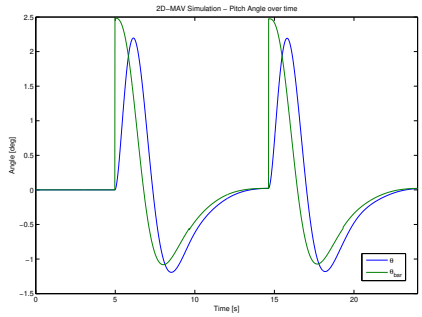
The plots obtained by running the simulation are given in Figure 3



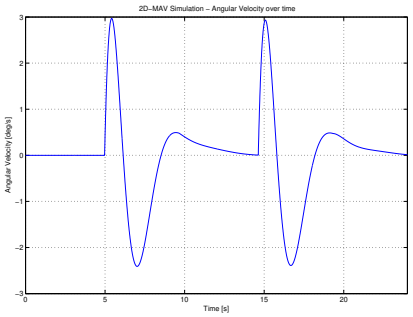
(a) Ideal and Real path in the arena



(b) Individual rotor forces



(c) Command and real Pitch Angles



(d) MAV Angular velocity

Figure 3: 2D-MAV Simulation Results

As Figure 3a illustrates, the simulated vehicle performed its task with adequate behavior, considerably close to the ideal path. As the rotor dy-

namics were not considered in this simulation exercise, it was assumed that the individual rotor forces are the same as the commands applied to them, which explain the sudden leaps in Figure 3b. The chosen gains resulted in a stable flight control, in which the resulting force and torque vectors never reached saturation and the pitch angle maximum value was of only 3 degrees, as shown by Figure 3c. Figure 3d shows the angular velocity of the MAV, to demonstrate its efforts to recover a stable position after a horizontal translation.

4 Conclusion

The objective of this report was to describe development of a simulation algorithm of a fictitious 2D-MAV in a rectangular arena. Based on the equations of motion and the desired outcome of the simulation, control laws were established and the algorithm created. The simulation was successful and the vehicle demonstrated decent performance throughout the task.

References

[1] "MP-282: Dynamic Modeling and Control of MAVs - Chapter 1: Introduction - Lecture Notes"

[2] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*, Cambridge University Press, USA. 1992. ISBN:978-0-521-43108-8

Appendix

Main Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       INSTITUTO TECNOLOGICO DE AERONAUTICA       %
% MP-282: Dynamic Modeling and Control of MAVs %
%             Simulation Exercise 1               %
%                2D-MAV Control                   %
%         Joao Filipe R. P. de A. Silva           %
%                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Getting Ready
clc
close all
clear all

%% General Parameters

g = 9.81; %Gravity [kgm/s^2]
Ts = 0.01; %Sampling Time [s]
e1 = [1;0]; %Unitary x vector
e2 = [0;1]; %Unitary y vector
wx = [0.5 2 2 3.5 3.5];
wy = [2.5 2.5 1.5 1.5 2.5];
w = [wx;wy]'; %waypoints

%% MAV Model

mav.m = 1; %MAV mass [kg]
mav.l = 0.2; %MAV arm length [m]
mav.v = [0;0]; %MAV speed [m/s]
mav.W = 0;  %MAV angular speed [deg/s]
mav.J = 0.02; %MAV moment of Inertia [kgm^2]
mav.max_theta = 30; %Maximum angle of maneuver [deg]
mav.r0 = [0.5;1.5]; %Initial position [m]
mav.r = mav.r0;
mav.theta0 = 0; %Initial angle [deg]
mav.theta = mav.theta0;
```

```
mav.Fmax = 2*mav.m*g;
mav.Fmin = 0.75*mav.m*g;

%% Control Loop

cont = 1;
Fmin = [-mav.Fmin*tand(mav.max_theta);mav.Fmin];
Fmax = [mav.Fmin*tand(mav.max_theta);mav.Fmax];
Tmax = (mav.m*g-(mav.Fmin))*mav.l;
x = [mav.r;mav.v;mav.theta;mav.W];
K1 = 6;
K2 = 3;
K3 = eye(2)*0.2;
K4 = eye(2)*0.8;
Kg = 1.5*eye(2);

waycount = 1;
while waycount < 6

    mav.r_= mav.r + Kg*((w(waycount,:)'-mav.r));
    %mav.r_= mav.r + Ts*0.5*((w(waycount,:)'-mav.r)/norm(w(
        waycount,:)'-mav.r));
    mav.D = [cosd(mav.theta) -sind(mav.theta);sind(mav.theta
        ) cosd(mav.theta)]';
    mav.Fc = mav.m*K3*(mav.r_-mav.r)-mav.m*K4*mav.v+mav.m*g*
        e2;
    mav.Fc = sat(mav.Fc,Fmin,Fmax);
    mav.theta_ = atand(e1'*mav.Fc)/(e2'*mav.Fc);
    mav.Tc = (mav.J*K1*(mav.theta_ - mav.theta) - mav.J*K2*
        mav.W);
    mav.Tc = sat(mav.Tc,-Tmax,Tmax);
    mav.f = CA(mav.Fc,mav.Tc,mav.l);

%integration using RK4

    k1 = Ts*dyn(x,mav);
    k2 = Ts*dyn(x+k1/2,mav);
    k3 = Ts*dyn(x+k2/2,mav);
    k4 = Ts*dyn(x+k3,mav);
    x  = x + k1/6 + k2/3 + k3/3 + k4/6;
```

```
73
74 %updated states
75
76     mav.r = x(1:2);
77     mav.v = x(3:4);
78     mav.theta = x(5);
79     mav.W = x(6);
80
81     rp(cont,1) = mav.r(1);
82     rp(cont,2) = mav.r(2);
83     fp(cont,1) = mav.f(1);
84     fp(cont,2) = mav.f(2);
85     tp(cont) = mav.theta;
86     tbp(cont) = mav.theta_;
87     Tp(cont) = mav.Tc;
88     Fp(cont,1) = mav.Fc(1);
89     Fp(cont,2) = mav.Fc(2);
90     wp(cont) = mav.W;
91     cont = cont + 1;
92
93     if bounded(mav.r(1),mav.r(2),w(waycount,1),w(waycount,2)
           ) == 1;
94         waycount = waycount + 1;
95     end
96
97     mav.r  %Show updated position
98 end
99 stencil()
100 time = 0:Ts:Ts*(cont-2);
101 plot(rp(:,1),rp(:,2));
102 figure
103 plot(time,fp(:,1),time,fp(:,2));
104 figure
105 plot(time,tp,time,tbp);
106 figure
107 plot(time,wp)
```

Functions

```
%%% Arena and goal drawing
function [] = stencil()
    wx = [0.5 0.5 2 2 3.5 3.5];
    wy = [1.5 2.5 2.5 1.5 1.5 2.5];
    arenax = [0 0 4 4 0];
    arenay = [0 3 3 0 0];
    obs1x = [1 1 1.5 1.5 1];
    obs1y = [1 2 2 1 1];
    obs2x = [2.5 2.5 3 3 2.5];
    obs2y = [2 3 3 2 2];
    plot(arenax, arenay, 'b-', 'LineWidth', 3);
    hold on;
    plot(obs1x, obs1y, 'b-', 'LineWidth', 2);
    plot(obs2x, obs2y, 'b-', 'LineWidth', 2);
    plot(wx, wy, 'k--', 'LineWidth', 1.5);
    bound1 = viscircles([wx(2),wy(2)],0.1,'LineStyle',':');
    bound2 = viscircles([wx(3),wy(3)],0.1,'LineStyle',':');
    bound3 = viscircles([wx(4),wy(4)],0.1,'LineStyle',':');
    bound4 = viscircles([wx(5),wy(5)],0.1,'LineStyle',':');
    bound5 = viscircles([wx(6),wy(6)],0.1,'LineStyle',':');
```

```
%%% Verify if inside circular boundary
function b = bounded(rx,ry,wx,wy)
    taux=0:.01:2*pi;
    x=cos(taux)*0.1+wx;
    y=sin(taux)*0.1+wy;
    b = inpolygon(rx,ry,x,y);
```

```
%%%Control Allocation
function f1f2 = CA(f_c,tau_c,l)
    f1f2 = zeros(2,1);
    f1f2 = [0.5 1/(2*l);0.5 -1/(2*l)]*[norm(f_c);tau_c];
```

```
%%%Saturation
function tosat = sat(val,LI,LS)
    tosat = val;
    for i=1:size(val,1)
        if val(i) < LI(i)
            tosat = LI(i);
```

```
7           end
8           if val(i) > LS(i)
9               tosat = LS(i);
10          end
11      end
12  end
```

```
1  %%%MAV Dynamics
2  function xnew = dyn(x,mav)
3      v = x(3:4);
4      W = x(6);
5      g = 9.81;
6      e2 = [0;1];
7      xnew =  [v;
8               (1/mav.m)*mav.D*e2*(mav.f(1)+mav.f(2)) - e2*g;
9              W;
10             (1/mav.J) * mav.Tc];
11     mav.a = xnew(3:4);
12     mav.Wdot = xnew(6);
13 end
```