Ingeniería de Control
Práctica 2

Aircraft vertical takeoff and landing

Consider the simplified planar model of the system for vertical takeoff and landing of an aircraft represented in Figure 1, in which the aircraft is represented by a bar. The position of the center of mass of the aircraft, $\mathbf{c} = (x, y)^T$, the roll angle of the aircraft, θ, and their time derivatives are the state variables of the system. The thrust force S, applied to the center of mass of the aircraft, and the forces F, applied to the wing tips, are the control inputs u_1 and u_2 of the system, respectively. The thrust force S keeps the aircraft flying. The forces F, which always act in opposite directions, control the roll of the aircraft.

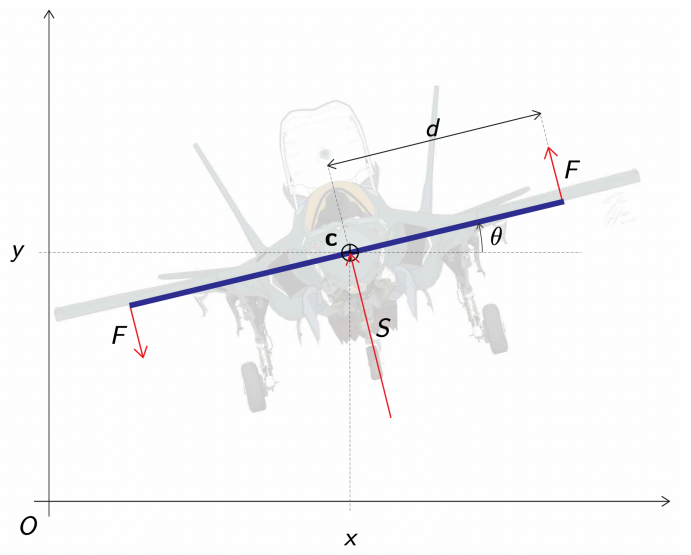


Figure 1: Sketch of the system for aircraft vertical takeoff and landing.

The dynamic model of this system is

$$\begin{aligned} \ddot{x} &= -\frac{1}{m}\sin(\theta)S, \\ \ddot{y} &= -g + \frac{1}{m}\cos(\theta)S, \\ \ddot{\theta} &= \frac{2d}{J}F, \end{aligned}$$

with the following parameters

- barycentric moment of inertia of the aircraft: $J = 10000$ [kg m^2],
- mass of the aircraft: $m = 30000$ [kg],
- $d = 5.5$ [m],
- gravity acceleration: $g = 9.81$ [m/s^2].

1) Demonstrate the equations of the dynamic model using the Lagrange method. (Copy the solution from Problem 1)
2) Calculate the state space representation of the system, assuming that $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (x_1, x_2, x_3, x_4, x_5, x_6)^T$, where distances are measured in [m], angles in [rad], linear velocities in [m/s], and angular velocities in [rad/s]. (Copy the solution from Problem 1)
3) Calculate all the operating points of the system and explain the obtained result.
4) Find the operating point that corresponds to $\overline{u}_1 = mg, \overline{u}_2 = 0$. Linearize the system around this operating point.
5) Is the linearized system controllable using both control inputs u_1 and u_2? Is the linearized system controllable using only the control input u_1?
6) Using the pole placement or the linear quadratic regulator method, design a state feedback controller to control the landing of the aircraft. We want to steer the aircraft from the state $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (1, 5, 0.0174533, -0.1, -0.2, 0.00174533)^T$ to the state $\mathbf{x} = (0, 2.4, 0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation.
7) Using the pole placement or the linear quadratic regulator method, design a state feedback controller to control a lateral displacement of the aircraft. We want to steer the aircraft from the state $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (0, 5, 0.0174533, -0.1, -0.2, 0.00174533)^T$ to the state $\mathbf{x} = (10, 5, 0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation.

Write a detailed report answering each question in a different section.

Originality and completeness of the answers will be the aspects that will be taken into account in the grading of the report. Include the Matlab code in the report.

Additionally, upload the Matlab code to Aula Virtual. Upload the code of each answer in a separate folder.

Solution of Práctica 2

1) Demonstrate the equations of the dynamic model using the Lagrange method. (Copy the solution from Problem 1)

the Lagrange method is based on the difference between total kinetic energy T and the total potential energy V of the system, that is, $L = T - V$

The total kinetic energy of this system can be written as:

$$T = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2 + \frac{1}{2}J\dot{\theta}^2$$

The term

$$\frac{1}{2}m\dot{x}^2$$

represents the kinetic energy of the aircraft due to the horizontal component of its velocity.

The term

$$\frac{1}{2}m\dot{y}^2$$

represents the kinetic energy of the aircraft due to the vertical component of its velocity.

The term

$$\frac{1}{2}J\dot{\theta}^2$$

represents the rotational kinetic energy of the aircraft due to its angular velocity.

The total potential energy of the system can be written as:

$$V = mgy$$

Therefore, the Lagrangean is given by:

$$L = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2 + \frac{1}{2}J\dot{\theta}^2 - mgy$$

The Lagrange equations are

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = f_1$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f_2$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = f_3$$

From

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = f_1$$

since

$$\begin{aligned} \frac{\partial L}{\partial \dot{x}} &= m\dot{x}, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= m\ddot{x}, \\ \frac{\partial L}{\partial x} &= 0, \\ f_1 &= -S\sin\theta \end{aligned}$$

the first Lagrange equation is

$$m\ddot{x} - 0 = -S\sin\theta$$

Likewise, from,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f_2$$

since

$$\begin{aligned} \frac{\partial L}{\partial \dot{y}} &= m\dot{y}, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{y}} &= m\ddot{y}, \\ \frac{\partial L}{\partial y} &= -mg, \\ f_2 &= S\cos\theta \end{aligned}$$

the second Lagrange equation is

$$m\ddot{y} + mg = S\cos\theta$$

Likewise, from,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = f_3$$

since

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta}} &= J\dot{\theta}, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} &= J\ddot{\theta}, \\ \frac{\partial L}{\partial \theta} &= 0, \\ f_3 &= 2dF \end{aligned}$$

the third Lagrange equation is

$$J\ddot{\theta} - 0 = 2dF$$

Thus, in this case the Lagrange equations are

$$\begin{aligned} m\ddot{x} &= -S\sin\theta, \\ m\ddot{y} + mg &= S\cos\theta, \\ J\ddot{\theta} &= 2dF \end{aligned}$$

Finally,

$$\begin{aligned} \ddot{x} &= -\frac{1}{m}S\sin\theta, \\ \ddot{y} &= -g + \frac{1}{m}S\cos\theta, \\ \ddot{\theta} &= \frac{2d}{J}F \end{aligned}$$

2) Calculate the state space representation of the system, assuming that $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (x_1, x_2, x_3, x_4, x_5, x_6)^T$, where distances are measured in [m], angles in [rad], linear velocities in [m/s], and angular velocities in [rad/s]. (Copy the solution from Problem 1)

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix}$$

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix}$$

The state space representation of the system is thus

$$\begin{aligned}
\dot{x_1} &= x_4, \\
\dot{x_2} &= x_5, \\
\dot{x_3} &= x_6, \\
\dot{x_4} &= -\frac{1}{m} S \sin\theta, \\
\dot{x_5} &= -g + \frac{1}{m} S \cos\theta, \\
\dot{x_6} &= \frac{2d}{J} F
\end{aligned}$$

For the output, I select the x and y, so:

$$g(x, u) = (x_1, x_2)$$

3) Calculate all the operating points of the system and explain the obtained result.

A point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is an operating point if $\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \mathbf{0}$. Thus

$$\begin{aligned}
0 &= x_4, \\
0 &= x_5, \\
0 &= x_6, \\
0 &= -\frac{1}{m} S \sin\theta, \\
0 &= -g + \frac{1}{m} S \cos\theta, \\
0 &= \frac{2d}{J} F
\end{aligned}$$

By the fourth equation, $\theta \in \{0, \pi\}$, but the only valid value in the fifth equation is 0. (the aircraft can't fly upside down). Also, $\theta = 0 \implies S = gm$.

Finally, by the sixth equation, $F = 0$. Thus, the operating points are

$$\begin{aligned} x &= a, \\ y &= b, \\ \theta &= 0, \\ \dot{x} &= 0, \\ \dot{y} &= 0, \\ \dot{\theta} &= 0, \\ S &= gm, \\ F &= 0, \\ \forall a, b \in \mathbb{R} & \end{aligned}$$

As can be seen, the operating points do not depend on a particular x or y, since the aircraft can hover on any x, y position

File `answer_3.m`

```
clear all; close all; clc;

syms x_1 x_2 x_3 x_4 x_5 x_6 S F
syms m g d J

f_1 = x_4 == 0;
f_2 = x_5 == 0;
f_3 = x_6 == 0;
f_4 = -(sin(x_3)*S) / m == 0;
f_5 = -g + (cos(x_3)*S) / m == 0;
f_6 = 2*d*F / J == 0;

result = solve([f_1 f_2 f_3 f_4 f_5 f_6], [x_1 x_2 x_3 x_4 x_5 x_6 S F], ...
    ReturnConditions=true, IgnoreAnalyticConstraints=true)
```

4) Find the operating point that corresponds to $\overline{u}_1 = mg, \overline{u}_2 = 0$. Linearize the system around this operating point.

Around the operating point calculated in the last exercise, the system is approximated by the following state equations:

$$\begin{cases} \dot{x} = f(\bar{x}, \bar{u}) + A(x - \bar{x}) + B(u - \bar{u}), \\ y = g(\bar{x}, \bar{u}) + C(x - \bar{x}) \end{cases}$$

with

$$\begin{aligned} A &= \frac{\partial f}{\partial x}(\bar{x}, \bar{u}), B = \frac{\partial f}{\partial u}(\bar{x}, \bar{u}), \\ C &= \frac{\partial g}{\partial x}(\bar{x}, \bar{u}) \end{aligned}$$

If we take $\tilde{u} = u - \bar{u}, \tilde{x} = x - \bar{x}$ and $\tilde{y} = y - \bar{y}$, we get

$$\begin{cases} \frac{d}{dt}\tilde{x} = A\tilde{x} + B\tilde{u}, \\ \tilde{y} = C\tilde{x} \end{cases}$$

Computing A, B, C

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{2\,d}{J} \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

File answer_4.m

```
clear all; close all; clc;

syms x_1 x_2 x_3 x_4 x_5 x_6 S F
syms m g d J

f_1 = x_4;
f_2 = x_5;
f_3 = x_6;
f_4 = -(sin(x_3)*S) / m;
f_5 = -g + (cos(x_3)*S) / m;
f_6 = 2*d*F / J;
y_1 = x_1;
y_2 = x_2;

A = jacobian([f_1 f_2 f_3 f_4 f_5 f_6], [x_1 x_2 x_3 x_4 x_5 x_6]);
B = jacobian([f_1 f_2 f_3 f_4 f_5 f_6], [S F]);
C = jacobian([y_1 y_2], [x_1 x_2 x_3 x_4 x_5 x_6]);

A = subs(A, [x_3 S], [0 g*m])
B = subs(B, x_3, 0)
C
```

5) Is the linearized system controllable using both control inputs u_1 and u_2? Is the linearized system controllable using only the control input u_1?

The system is controllable if and only if

$$rank(B|AB|A^2B|...|A^{n-1}B) = n$$

where n is the dimension of x. With matlab:

```
Co = ctrb(A,B);
unco = length(A) - rank(Co)
```

unco gives the number of uncontrollable states. Executing it with the original B gives 0, so it is controllable. However, if it is executed with the second

column of B with all ceros, it gives 4, so it is not controllable only with the control input u_1.

File answer_5.m

```
clear all; close all; clc;

syms x_1 x_2 x_3 x_4 x_5 x_6 S F
syms m g d J

f_1 = x_4;
f_2 = x_5;
f_3 = x_6;
f_4 = -(sin(x_3)*S) / m;
f_5 = -g + (cos(x_3)*S) / m;
f_6 = 2*d*F / J;

A = jacobian([f_1 f_2 f_3 f_4 f_5 f_6], [x_1 x_2 x_3 x_4 x_5 x_6]);
B = jacobian([f_1 f_2 f_3 f_4 f_5 f_6], [S F]);

A = subs(A, [x_3 S], [0 g*m]);
A = subs(A, g, 9.81);
B = subs(B, x_3, 0);
B = subs(B, [m d J], [30000 5.5 10000])

disp('number of uncontrollable states with original B:');
Co = ctrb(A, B);
unco = length(A) - rank(Co)
B(end, end) = 0
disp('number of uncontrollable states with edited B:');
Co = ctrb(A, B);
unco = length(A) - rank(Co)
```

6) Using the pole placement or the linear quadratic regulator method, design a state feedback controller to control the landing of the aircraft. We want to steer the aircraft from the state $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (1, 5, 0.0174533, -0.1, -0.2, 0.00174533)^T$ to the state $\mathbf{x} = (0, 2.4, 0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation.

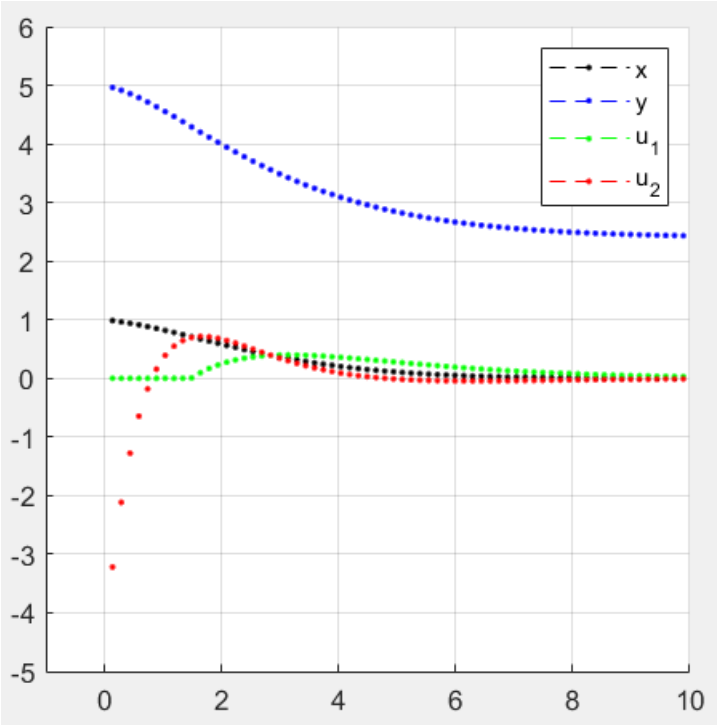
In this folder I have videos: https://urjc-my.sharepoint.com/:f:/g/personal/j_delacanoniga_2019_alumnos_urjc_
720Oa6DdMuC2XDb4-UBUBnXr3uQmQaprBm-RJ0gw24Q?e=byhFXR

Since it is a very simple move, I use pole placement.

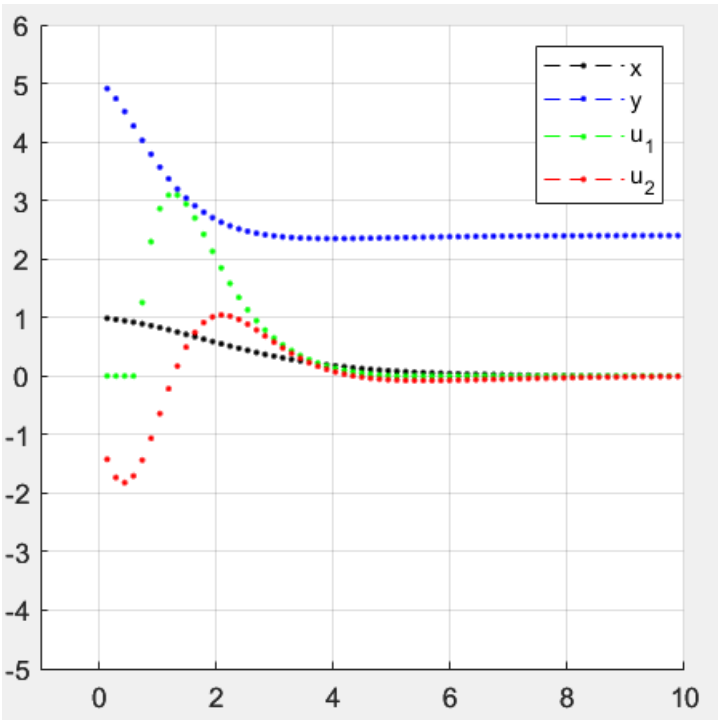
The state feedback controller has the form $\tilde{u} = -K\tilde{x}$. In order to be able to use it in our non linear system, u must be isolated: $u = -K(x - \bar{x}) + \bar{u}$.

To calculate k I use the matlab function place. Since there are 2 inputs and 6 state variables, k is a 2x6 matrix. I tried 3 ranges for the eigenvalues, which are the following:

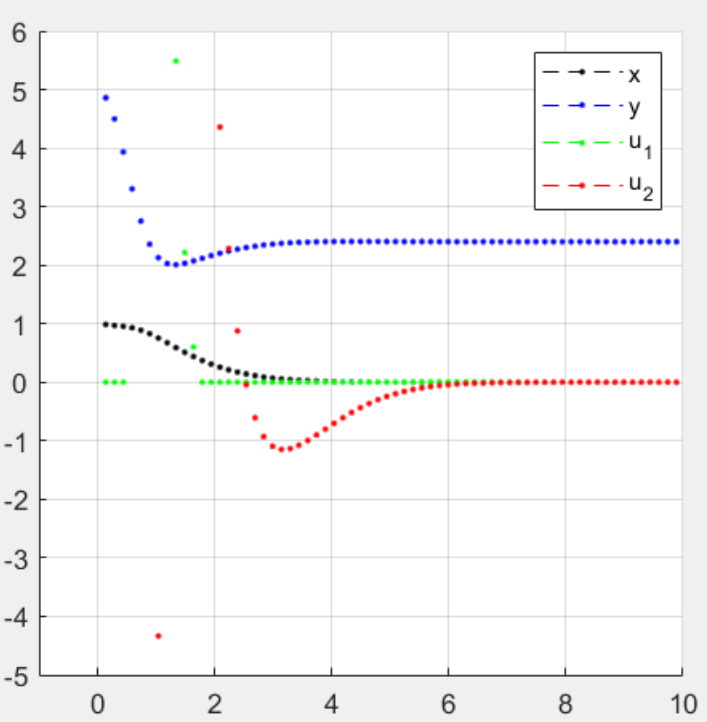
Slow poles: [-1, -0.9, -0.8, -0.7, -0.60, -0.5]


6
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
2
4
6
8
10
x
y
u1
u2

Intermediate poles: [-1.5, -1.4, -1.3, -1.2, -1.1, -1]



Fast poles: [-2.4, -2.3, -2.2, -2.1, -2, -1.9]



As can be seen, the faster poles are not valid, because the state variable y overshoots, so the plane would crash. Ultimately, I would choose the intermediate over the slow poles, because the settling time is half of the slow, so it would burn less fuel.

File answer_6_init.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

close all;
clear all;
clc;

f = figure();
f.Position = [0 0 1920/2 1080/2];
subplot(1, 2, 1);
axis equal;
hold on;
xmin = -1;
xmax = 6;
ymin = -5;
ymax = 6;
axis([xmin xmax ymin ymax]);
grid on

subplot(1, 2, 2);
axis equal;
hold on;
xmin = -6;
xmax = 8;
ymin = -1;
ymax = 8;
axis([xmin xmax ymin ymax]);
grid on
%axis ('square');
```

File answer_6_f.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

function  xdot  = answer_6_f(x, u)
    x_1 = x(1);
    x_2 = x(2);
    x_3 = x(3);
    x_4 = x(4);
    x_5 = x(5);
    x_6 = x(6);

    S = u(1);
    F = u(2);

    m = 30000;
    g = 9.81;
    d = 5.5;
    J = 10000;

    xdot = [x_4; x_5; x_6; -(sin(x_3)*S) / m; -g + (cos(x_3)*S) / m; 2*d*F / J];
end
```

File answer_6_draw.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

function answer_6_draw(x, u)
persistent draws;
    x_1 = x(1);
    x_2 = x(2);
    x_3 = x(3);
    x_4 = x(4);
    x_5 = x(5);
    x_6 = x(6);

    m = 30000;
    g = 9.81;
    d = 5.5;
    u(1) = max(u(1), 0);
    u(2) = u(2)/50;


    delete(draws);

    subplot(1, 2, 2)
    p_1 = [x_1 - d*cos(x_3), x_1 + d*cos(x_3)];
    p_2 = [x_2 - d*sin(x_3), x_2 + d*sin(x_3)];
    draws = [draws plot(p_1, p_2, '-black', LineWidth=3)];

    if u(1) < g*m
        mult_factor = u(1) / (g*m);
        draw_u_1_x = [x_1 - 0.2*cos(x_3), x_1 + 0.2*cos(x_3), x_1 + 3*mult_factor*sin(x_3)];
        draw_u_1_y = [x_2 - 0.2*sin(x_3), x_2 + 0.2*sin(x_3), x_2 - 3*mult_factor*cos(x_3)];
    else
        draw_u_1_x = [x_1 - 0.2*cos(x_3), x_1 + 0.2*cos(x_3), x_1 + 3*sin(x_3) + abs((u(1)-g*m)/25000)*cos(x_3), x_1 +
        draw_u_1_y = [x_2 - 0.2*sin(x_3), x_2 + 0.2*sin(x_3), x_2 - 3*cos(x_3) + abs((u(1)-g*m)/25000)*sin(x_3), x_2 -

    end
    draws = [draws patch(draw_u_1_x, draw_u_1_y, 'red')];
    set(draws(end) ,'FaceColor', [0.8500 0.3250 0.0980])
    set(draws(end) ,'EdgeColor', [0.8500 0.3250 0.0980])



    draws = [draws quiver(p_1(1), p_2(1), u(2)*sin(x_3), -u(2)*cos(x_3), 'red', LineWidth=2)];
    draws = [draws quiver(p_1(2), p_2(2), -u(2)*sin(x_3), u(2)*cos(x_3), 'red', LineWidth=2)];
end
```

File answer_6_main.m

```
%Adapted from https://www.ensta-bretagne.fr/jaulin/
answer_6_init;

x = [1; 5; 0.0174533; -0.1; -0.2; 0.00174533];
x_tilde = [0; 2.4; 0; 0; 0; 0];

t = 0;

m = 30000;
g = 9.81;
d = 5.5;
J = 10000;

answer_6_draw(x, [0; 0]);


A = [0 0 0 1 0 0;
     0 0 0 0 1 0;
     0 0 0 0 0 1;
     0 0 -g 0 0 0;
     0 0 0 0 0 0;
     0 0 0 0 0 0;];
B = [0 0;
     0 0;
     0 0;
     0 0;
     1/m 0;
     0 2*d/J;];

%poles = [-1:0.1:-0.5];
poles = [-1.5:0.1:-1];
%poles = [-2.4:0.1:-1.9];
k = place(A, B, poles);

frame_counter = 0;
dt = 0.01;
for t=0:dt:6
    u = -k * (x - x_tilde);
    u(1) = max(u(1) + g*m, 0);

    %x = x + answer_6_f(x, u)*dt; % Euler
    x=x+dt*(0.25*answer_6_f(x,u)+0.75*(answer_6_f(x+dt*(2/3)*answer_6_f(x,u),u))); % Runge-Kutta

    pause(dt);

    frame_counter = frame_counter+1;

    % Frame sampling
    if frame_counter == 15
       answer_6_draw(x, u);
       subplot(1, 2, 1)
       plot(t, x(1),'black--.');
       plot(t, x(2),'blue--.');
       plot(t, max((u(1)-g*m)/10000, 0),'green--.');
       plot(t, u(2)/10,'red--.');
       legend('x', 'y', 'u_1', 'u_2');
       frame_counter = 0;
    end
end
```

Your figures go here

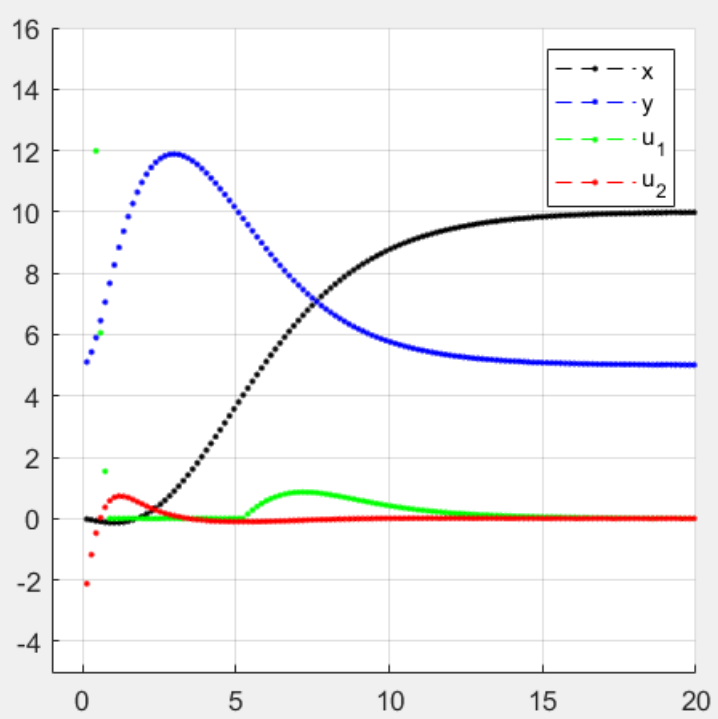
7) Using the pole placement or the linear quadratic regulator method, design a state feedback controller to control a lateral displacement of the aircraft. We want to steer the aircraft from the state $\mathbf{x} = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T = (0, 5, 0.0174533, -0.1, -0.2, 0.00174533)^T$ to the state $\mathbf{x} = (10, 5, 0, 0, 0, 0)^T$. Give the eigenvalues that have been assigned to the controlled system and illustrate the behaviour of the controller by plotting the relevant state and control variables and by a graphical animation.

The state feedback controller has the form $\tilde{u} = HW - K\tilde{x}$. where w represents two inputs: one to control the x position and the other to control the y position. H is the precompensator. In order to be able to use it in our non linear system, u must be isolated: $u = HW - K(x - \bar{x}) + \bar{u}$.

In order to select the x and y variables, $E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$

For the values of k, first I try pole placement.

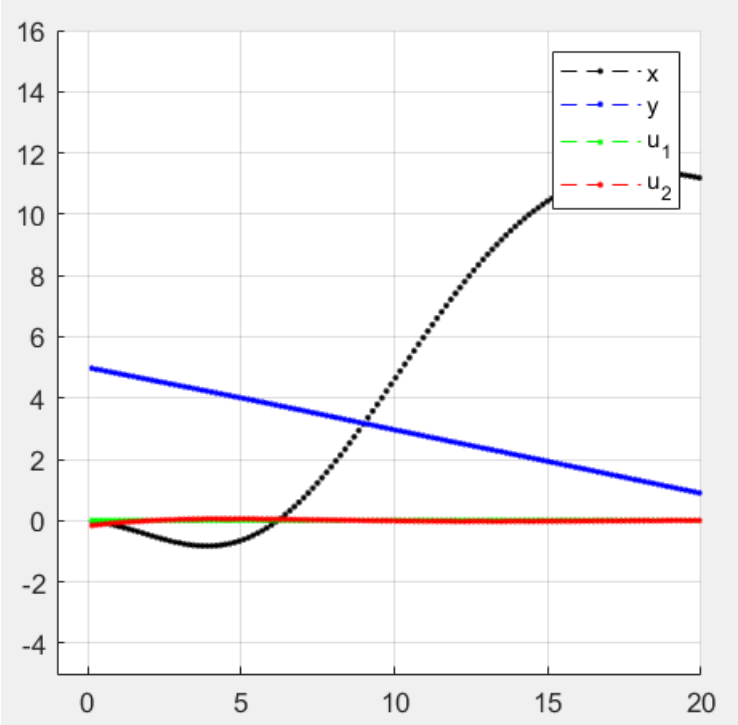
[-1, -0.9, -0.8, -0.7, -0.6, -0.5].



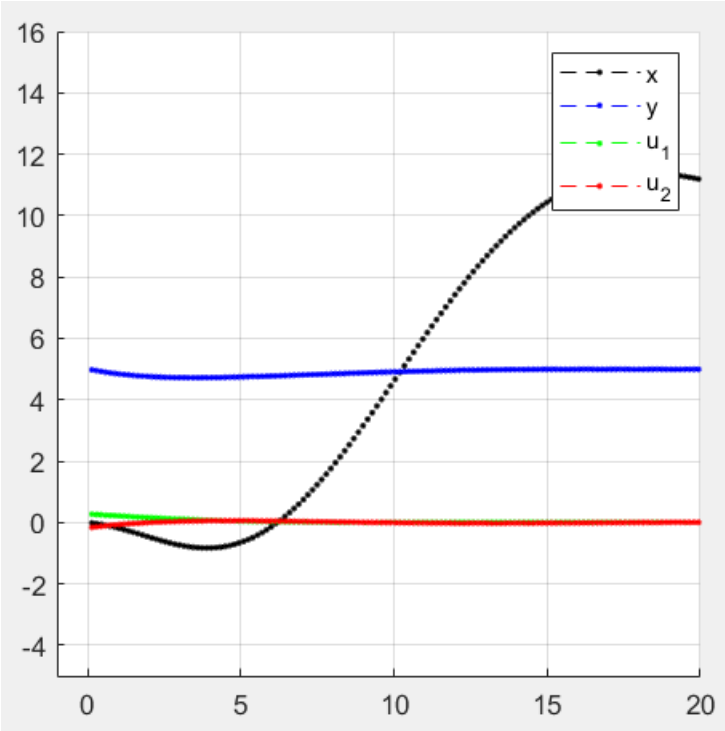
As can be seen, it is very inefficient, since it gains altitude in order to change its x position. Uses the main engine too much, and the settling time is too slow. Seeing this I test lqr.

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

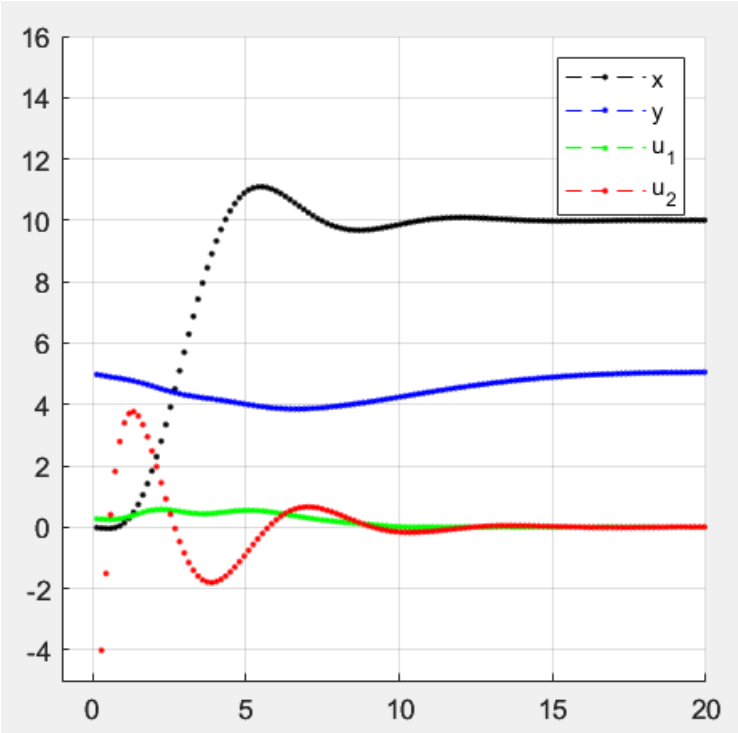
With this initial values, the aircraft couldn't hold altitude as can be seen in the graph and in the video.



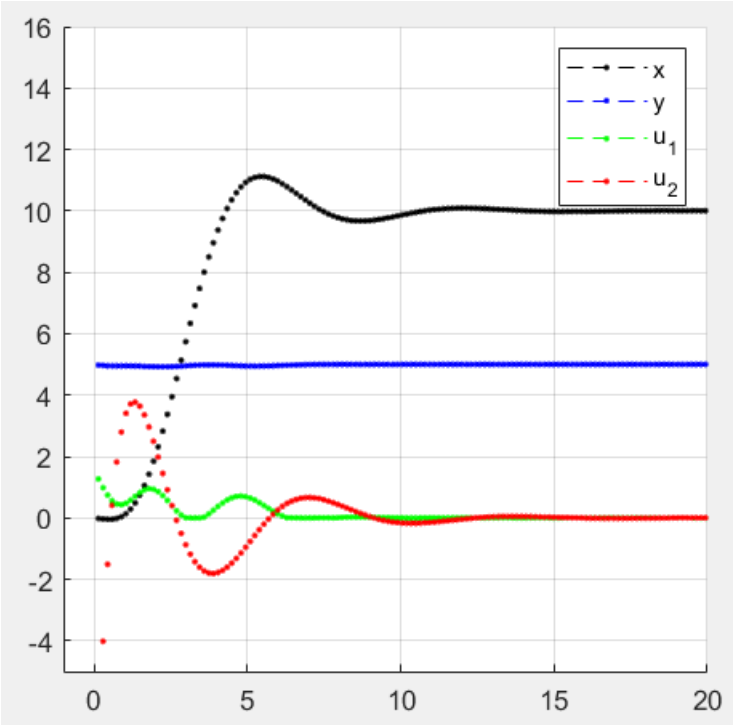
It looks like it cannot use its main engine, so I lower its cost.



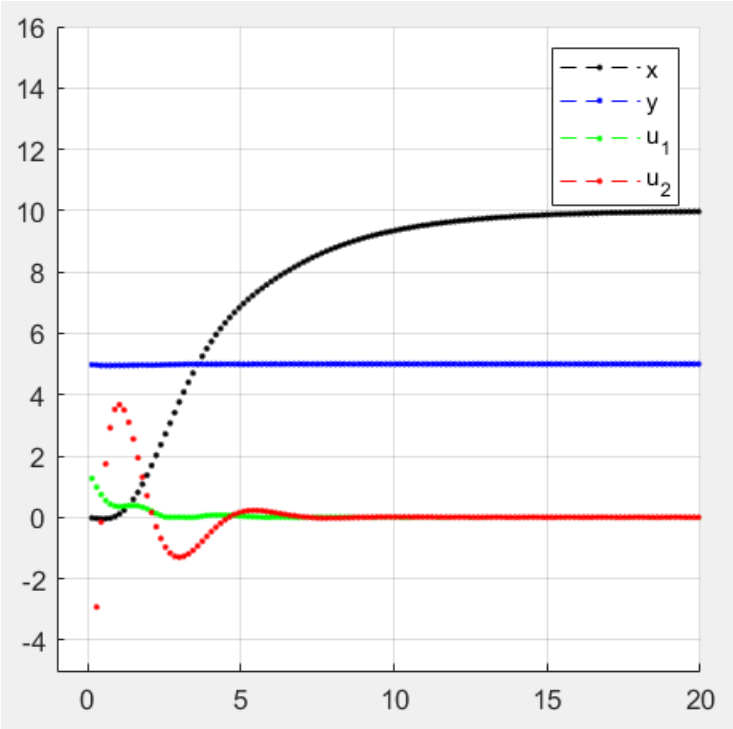
Now it can hold altitude, but it is very slow, so I lower the cost of the second input.



better, but it loses altitude, so I increase the cost of the y.

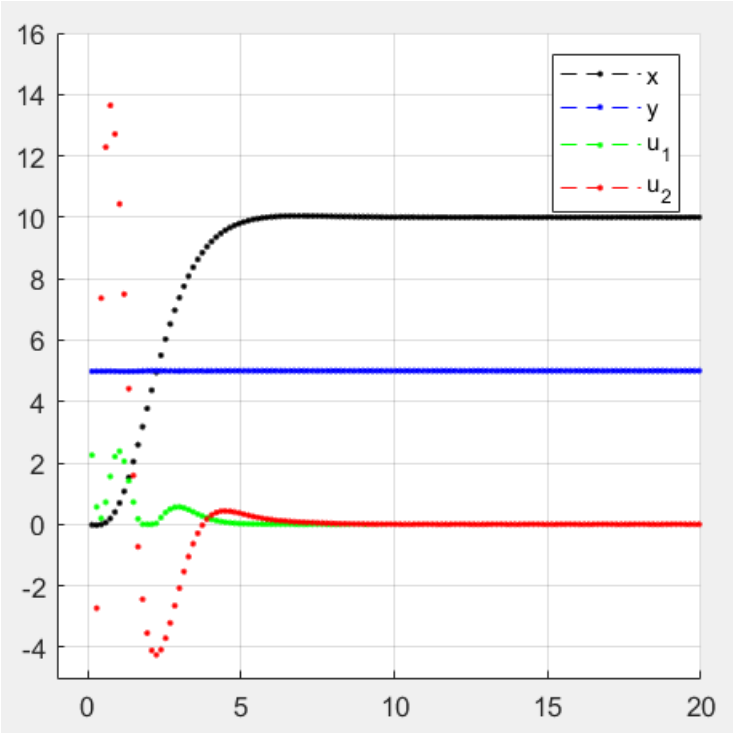


Now I increase the cost of the x velocity in order to avoid overshoot.

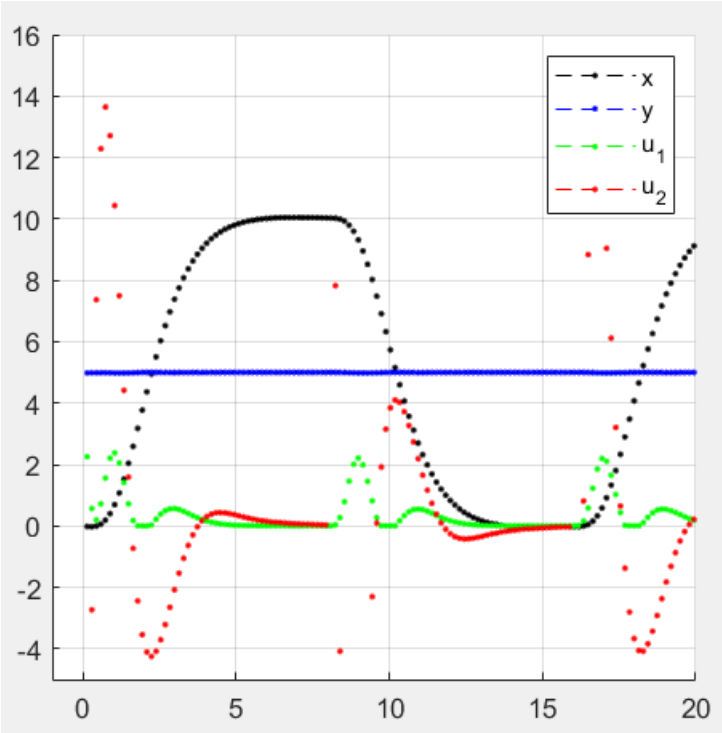


Tweaking some more parameters, I get the following

$$Q = \begin{pmatrix} 100000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10000000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, R = \begin{pmatrix} 0.0000000001 & 0 \\ 0 & 0.6 \end{pmatrix}$$



I have also taken into consideration the angle, to avoid hitting the floor with the tip of the wings.



I also made tests constantly changing the position.

File answer_7_init.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

close all;
clear all;
clc;

f = figure();
f.Position = [0 0 1920/2 1080/2];
subplot(1, 2, 1);
axis equal;
hold on;
xmin = -1;
xmax = 8;
ymin = -5;
ymax = 16;
axis([xmin xmax ymin ymax]);
grid on

subplot(1, 2, 2);
axis equal;
hold on;
xmin = -6;
xmax = 16;
ymin = -1;
ymax = 16;
axis([xmin xmax ymin ymax]);
grid on
%axis ('square');
```

File answer_7_f.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

function  xdot  = answer_7_f(x, u)
    x_1 = x(1);
    x_2 = x(2);
    x_3 = x(3);
    x_4 = x(4);
    x_5 = x(5);
    x_6 = x(6);

    S = u(1);
    F = u(2);

    m = 30000;
    g = 9.81;
    d = 5.5;
    J = 10000;

    xdot = [x_4; x_5; x_6; -(sin(x_3)*S) / m; -g + (cos(x_3)*S) / m; 2*d*F / J];
end
```

File answer_7_draw.m

```
% Adapted from https://www.ensta-bretagne.fr/jaulin/

function answer_7_draw(x, u)
persistent draws;
    x_1 = x(1);
    x_2 = x(2);
    x_3 = x(3);
    x_4 = x(4);
    x_5 = x(5);
    x_6 = x(6);

    m = 30000;
    g = 9.81;
    d = 5.5;
    u(1) = max(u(1), 0);
    u(2) = u(2)/1000;


    delete(draws);

    subplot(1, 2, 2)
    p_1 = [x_1 - d*cos(x_3), x_1 + d*cos(x_3)];
    p_2 = [x_2 - d*sin(x_3), x_2 + d*sin(x_3)];
    draws = [draws plot(p_1, p_2, '-black', LineWidth=3)];

    if u(1) < g*m
        mult_factor = u(1) / (g*m);
        draw_u_1_x = [x_1 - 0.2*cos(x_3), x_1 + 0.2*cos(x_3), x_1 + 3*mult_factor*sin(x_3)];
        draw_u_1_y = [x_2 - 0.2*sin(x_3), x_2 + 0.2*sin(x_3), x_2 - 3*mult_factor*cos(x_3)];
    else
        draw_u_1_x = [x_1 - 0.2*cos(x_3), x_1 + 0.2*cos(x_3), x_1 + 3*sin(x_3) + abs((u(1)-g*m)/25000)*cos(x_3), x_1 +
        draw_u_1_y = [x_2 - 0.2*sin(x_3), x_2 + 0.2*sin(x_3), x_2 - 3*cos(x_3) + abs((u(1)-g*m)/25000)*sin(x_3), x_2 -

    end
    draws = [draws patch(draw_u_1_x, draw_u_1_y, 'red')];
    set(draws(end) ,'FaceColor', [0.8500 0.3250 0.0980])
    set(draws(end) ,'EdgeColor', [0.8500 0.3250 0.0980])



    draws = [draws quiver(p_1(1), p_2(1), u(2)*sin(x_3), -u(2)*cos(x_3), 'red', LineWidth=2)];
    draws = [draws quiver(p_1(2), p_2(2), -u(2)*sin(x_3), u(2)*cos(x_3), 'red', LineWidth=2)];

    %circle(p_x, p_y, 0.15);

    %plot(t, x(1),'black--.')
    %plot(t, x(2),'red--.')
end
```

File answer_7_main.m

```
%Adapted from https://www.ensta-bretagne.fr/jaulin/
answer_7_init;

x = [0; 5; 0.0174533; -0.1; -0.2; 0.00174533];
%x = [0; 7; 0; 0; 0; 0];
x_tilde = [0; 2.4; 0; 0; 0; 0];
%x = [5; 5; 0; 0; 0; 0];

t = 0;

m = 30000;
g = 9.81;
d = 5.5;
J = 10000;

answer_7_draw(x, [g*m; 0]);

A = [0 0 0 1 0 0;
     0 0 0 0 1 0;
     0 0 0 0 0 1;
     0 0 -g 0 0 0;
     0 0 0 0 0 0;
     0 0 0 0 0 0;];
B = [0 0;
     0 0;
     0 0;
     0 0;
     1/m 0;
     0 2*d/J;];
%poles = [-2.4:0.1:-1.9];
poles = [-1.5:0.1:-1];
k = place(A, B, poles);

Q = [100000 0 0 0 0 0;
     0 100 0 0 0 0;
     0 0 10000000 0 0 0;
     0 0 0 100000 0 0;
     0 0 0 0 0 0;
     0 0 0 0 0 0;];

R = [0.0000000001 0;
     0 0.6;];

k = lqr(A, B, Q, R);

E = [1 0 0 0 0 0;
     0 1 0 0 0 0;];

H = -inv(E * inv(A - B*k) * B);
w = [10; 5];

frame_counter = 0;
dt = 0.01;
for t=0:dt:7
    u = -k * x + H*w;
    u(1) = max(u(1) + g*m, 0);

    %x = x + answer_7_f(x, u)*dt; % Euler
    x=x+dt*(0.25*answer_7_f(x,u)+0.75*(answer_7_f(x+dt*(2/3)*answer_7_f(x,u),u))); % Runge-Kutta

    pause(dt);

    frame_counter = frame_counter+1;

    % Frame sampling
    if frame_counter == 15
       answer_7_draw(x, u);
       subplot(1, 2, 1)
       plot(t, x(1),'black--.');
       plot(t, x(2),'blue--.');
       plot(t, max((u(1)-g*m)/10000, 0),'green--.');
       plot(t, u(2)/100,'red--.');
       legend('x', 'y', 'u_1', 'u_2');
       frame_counter = 0;
    end
end
```

Your figures go here