Model-based control design for the mini-drone TELLO

Hugues Garnier

November 21, 2024

Contents

Lab 1

Model-based control design for the mini-drone TELLO

The work done during this mini-project will be noted in a report and marked. You are asked to follow the instructions given below to write your report.

Instructions for writing your lab report

A lab report is a scientific document. It should be self-contained: that is, someone who has never seen the lab instructions should be able to understand the problems that you are solving and how you are solving them. All the choices you have made should be clearly motivated. Comment and explain all your plots so as to make it easy to follow your way of thinking.

Your lab report can be written in French or in English but **do not mix both languages**. It should be organized as follows:

- a general introduction specifying the objectives of the lab
- For each assignment or exercise:
  - ▷ a brief presentation of the expected outcomes
  - ▷ a description of the obtained results in graphical and or numerical form
  - ▷ a critical analysis of the results
  - ▷ a short conclusion
- a general conclusion explaining what has been understood during the lab and any difficulties encountered.

When working in an experimental environment, results are always important, but SO is your ability to communicate the information with others. Any work you will submit should be neat, well-organized and easy to understand. Your report is a demonstration of your ability to understand the course you are studying as well as a reflection of your organizational skills.

Sending your report to the tutor

The report should be written in groups of two students and sent by email to the instructor in the form of a single pdf format file attached to the message by the deadline given by your instructor during the lab. Please indicate the following "subject" for your email when you send your report to the instructor: `Tello_Lab_Names`

This lab provides an introduction to drone control, particularly focusing on altitude and yaw control using both Proportional and Proportional-Derivative (PD) controllers.
The drone we will be using is the TELLO mini-drone controlled via WiFi.

The objectives are the following:

1. to learn about the basics of a drone including its components and how the drone flies.

2. to identify a linear low-order models from real data for the vertical or yaw dynamics.

3. to design both P and PD controllers for the altitude of the mini-drone which are tuned from an identified linear model and evaluate its performance when the setpoint is a square wave.

4. to design PID-based controllers for the yaw of the mini-drone which are tuned from an identified linear model and evaluate its performance when the setpoint is a square or triangular wave.

5. to implement and evaluate the designed controllers on the physical TELLO mini-drone from a Python program.

Safety warning message: during this lab, you must exercise extreme caution when handling your mini-drone. Ensure that you follow all safety instructions. Unauthorized maneuvers or negligence can lead to injuries or damage to equipment.

1.1 The TELLO mini-drone

Prior to operating any experimental system, it is imperative to familiarize yourself with the various components of the system. Part of the process involves the identification of the individual hardware components, including its sensors and actuators.

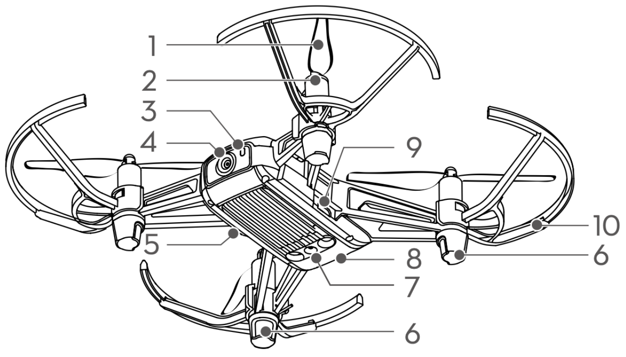
The TELLO mini-drone is a small quadcopter that features a vision positioning system and an onboard camera as shown in Figure 1.1.

Figure 1.1: Tello mini-drone from DJI

Using its vision positioning system and advanced flight controller, it can hover in place and is suitable for flying indoors. Its maximum flight time is about 13 minutes. Whenever you can, connect the TELLO drone to the USB port of your PC to recharge the battery for the next test.

1.1.1 Main components

The main components of the mini-drone are presented in Table 1.1.



1. Propellers	6. Antennas
2. Motors	7. Vision positioning system
3. Aircraft status indicator	8. Flight battery
4. Camera	9. Micro USB port
5. Power button	10. Propeller guards

Table 1.1: Main components of the TELLO mini-drone

1.1.2 Identification of the physical system components

Visit the official website `www.ryzerobotics.com/fr/tello` and answer the following questions from the TELLO User guide.

1. Present the onboard actuators, explaining their role in flight control.
2. Present the onboard sensors for altitude measurement and IMU (Inertial Measurement Unit) sensors for orientation and acceleration measurements.
3. Is there a GPS ?
4. How much does the mini-drone weigh?
5. Overview the communication protocol used to interface with the TELLO drone from a PC or a MAC, focusing on sending control commands and receiving sensor data.

1.2 First open-loop control of the TELLO mini-drone by using its App

1. Search for "Tello" on the App Store or Google Play to download the TELLO App on your mobile device.
2. Go to the room C335.
3. Press once the power button to turn the mini-drone on.
4. Place your TELLO on the black cross in the middle of the room with the in-front camera facing the window.
5. Go outside the cage.
6. Launch the TELLO App on your mobile device.

7. Enable the Wi-Fi on your mobile device and connect to the TELLO XXXXXX network. Connection has been established when the LED indicator blinks yellow slowly and the live camera view is shown on your mobile device.

8. Select Auto Takeoff.

9. Use the virtual joysticks in the App to test your ability to control the mini-drone. Try to control the drone to make a square trajectory of 50 cm.

10. Select Auto landing.

1.3 Understanding the basics of quadcopter control

There are many types of Unmanned Aerial Vehicles (UAVs), but the TELLO mini-drone used is a quadcopter. A quadcopter is a type of helicopter which has 4 rotors. Each one of them produces thrust and torque at the same time.
To understand how a quadcopter flies, it is very important to have the concept of 6-degrees of freedom (6-DOF).
The 6 degrees of freedom is a representation of how an object moves through 3D space by either translating linearly or rotating axially as shown in Figure 1.2.

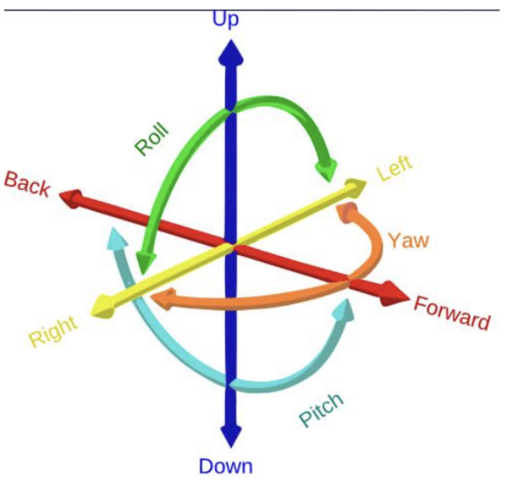


Figure 1.2: The 6 degrees of freedom

Specifically, the object can move in three dimensions, on the X, Y and Z axes (left/right, forward/back, up/down), as well as change orientation between those axes though rotation usually called yaw, roll and pitch which are defined below

- **Yaw (lacet)**: the drone rotates in place, flat, around its center of gravity. This can be likened to making a "No" sign with our head.
- **Roll (roulis)**: the drone behaves as if it wanted to roll. This can be likened to tilting our head left or right.
- **Pitch (tangage)**: the drone behaves as if it wanted to rear up or dive forward. This can be likened to making a "Yes" sign with our head.

Impact on the motors

A quadcopter is equipped with four rotors, two of which rotate clockwise and two of which rotate counterclockwise.

Hovering

When all rotors spin at exactly the same speed, the forces on the multi-rotor are equal, and the quadcopter will hover in place (fly and stabilize itself in midair).

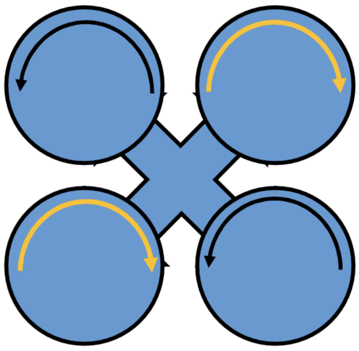


Figure 1.3: Principle of howering

For the quadcopter to be able to fly and stabilize itself in midair, the total thrust produced by the 4 rotors should be equal to the gravitational force subjected to the system.

Ascending or descending

To make the quadcopter ascend, the thrust is increased on all four rotors equally and conversely to descend.

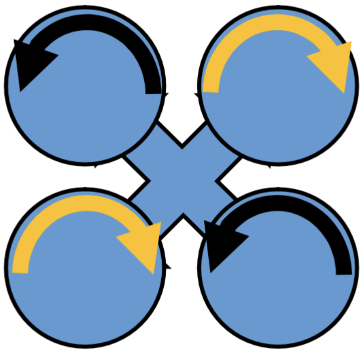


Figure 1.4: Principle of ascend and descend

Yaw

To rotate the quadcopter, we need to increase the thrust of two of the four motors. So, if we want to turn clockwise (right), we would increase the thrust towards the two rotors turning counterclockwise (left) and vice versa to turn in the other direction.

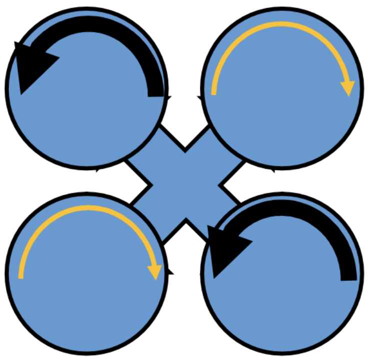


Figure 1.5: Principle of Yaw

Roll and Pitch

Finally, if we need to move forward/backward or go left/right, we will decrease the thrust of the rotors in the direction we want to move and increase the thrust towards the opposite rotors.

For the pitch movement, the rotation speed of the front motors is increased while that of the rear motors is decreased.

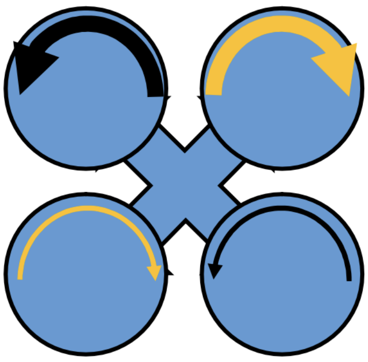


Figure 1.6: Principle of Pitch

1.3.1 Videos to understand the basics of drone control design

To known more about the basics of drone control, you can watch Brian Douglas's introduction videos:
Drone Simulation and Control, Part 1: Setting Up the Control Problem

Drone Simulation and Control, Part 2: How Do You Get a Drone to Hover?

1.3.2 Manual control

From the first video, we know that by manipulating the four motor speed in specific ways through the Motor Mixing Algorithm (MMA), thrust, roll, pitch, and yaw can be controlled directly and so we are able to control the drone in 3D space. This is the open-loop control configuration as shown in Figure 1.7 that you have used for controlling the TELLO with the App on your mobile device.

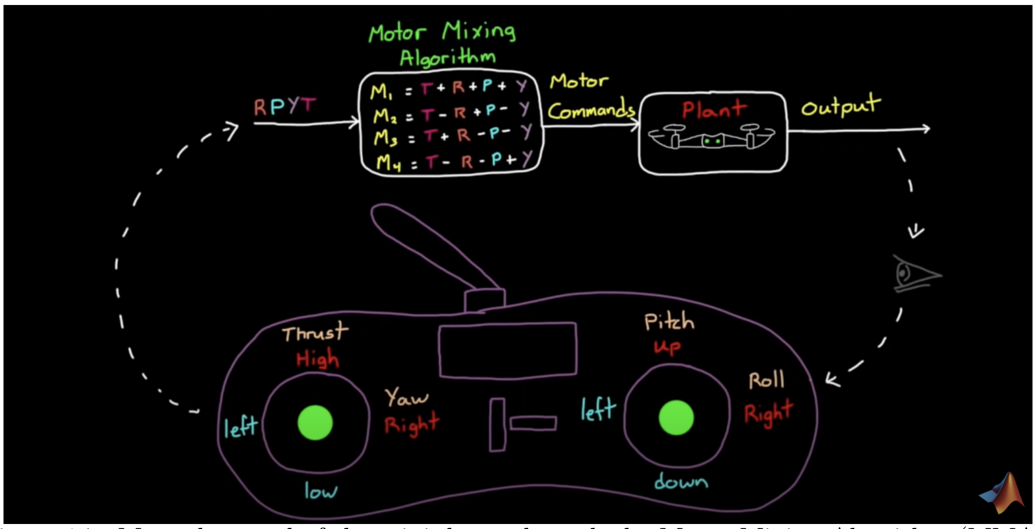


Figure 1.7: Manual control of the mini-drone through the Motor Mixing Algorithm (MMA).
From Brian Douglas's video: Drone Simulation and Control, Part 2: How Do You Get a Drone to Hover?

1.3.3 Full closed-loop control

From the second video, we know that the full closed-loop control architecture is quite complex and looks like the one shown in Figure 1.8 with several different control loops and 6 different PID controllers.

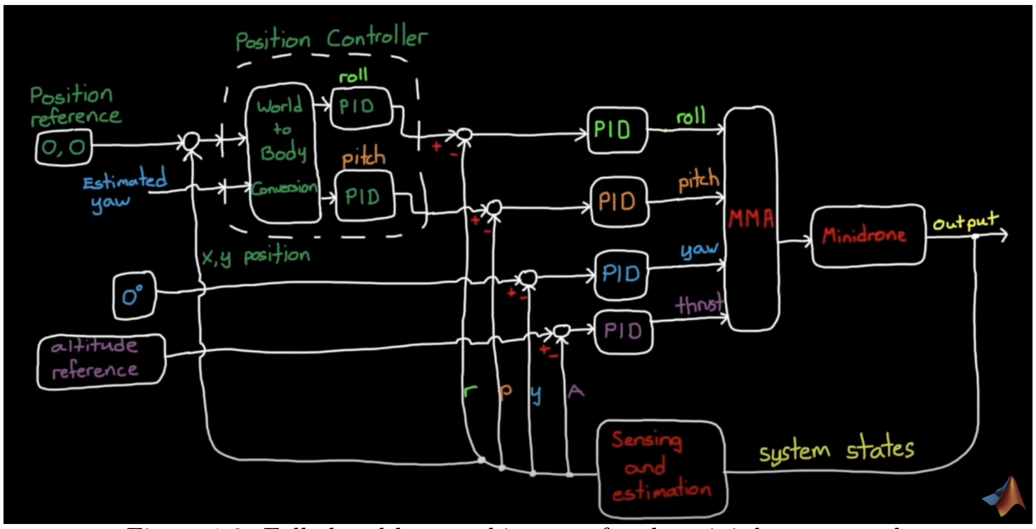


Figure 1.8: Full closed-loop architecture for the mini-drone control.
From Brian Douglas's video: Drone Simulation and Control, Part 2: How Do You Get a Drone to Hover?

1.4 Altitude control of the TELLO mini-drone

During this lab, the goal is first to consider a single loop: the altitude loop. Remember, this is independent of the other loops so we can tweak and adjust altitude without affecting roll, pitch, or yaw. To make sure they are out of the equation completely, we will just set the commands to 0 for roll, pitch, and yaw as shown in Figure 1.9.

The goal is therefore first to design a control for the mini-drone that uses thrust to adjust the altitude. If we are able to measure the drone altitude then we can feed it back to compare it to an altitude reference. The resulting error is then fed into a PID controller that is using that information to determine how to increase or decrease thrust.

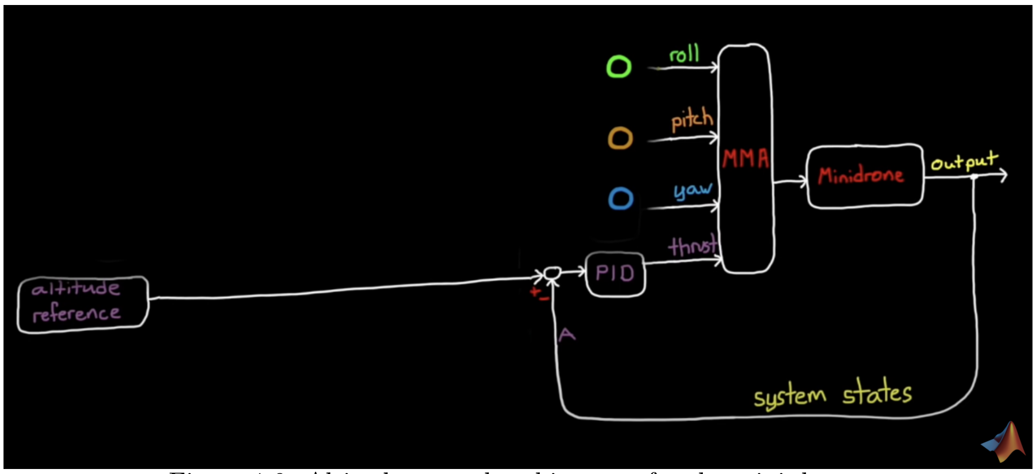


Figure 1.9: Altitude control architecture for the mini-drone.
From Brian Douglas's video: Drone Simulation and Control, Part 2: How Do You Get a Drone to Hover?

The input and output of the mini-drone altitude control system are:

- $u(t)$: motor speed in % (related in some way to the thrust generated by the 4 motors);
- $y(t)$: altitude in cm or vertical position along the Z axis of the mini-drone.

1.4.1 Altitude control performance requirements

The performance requirements for the mini-drone altitude control are described in Table 1.2.

Requirement	Assessment criteria	Level
Control of the	Square wave reference tracking	No steady-state error
drone altitude	Peak overshoot	$D_{1\%} = 4.3\%$
	Settling time at 5 %	$t_s^{5\%} = 2$ s

Table 1.2: Performance requirements for the mini-drone altitude control

1.4.2 Download of the files required for the lab

1. Download the zipped file `Lab`$_{T}ello.zip fromthecoursewebsiteandsaveandunzipitinyour$`Documents/Matlab/C`

2. Important ! By clicking on the browse for folder icon, **change the current folder of Matlab so that it becomes your** `Documents/Matlab/Control_Lab/` **folder** that contains the files needed for this lab.

3. In the Current Folder window of Matlab, click right on the folder `Documents/Matlab/Control_Lab/Lab`$_{T}elloandselect$`addtopaththeselectedfolder`$.Double-$
$clickonthefolder$`Lab`$_{\mathtt{T}}$`ello`$sothatitbecomesyourcurrentfolder.Youshouldseethedifferent$`.slx`, $.mand$`.py`$filesneed$

1.4.3 Vertical dynamic model identification

The determination of a model is a first crucial step for the design of a feedback control system. Unfortunately, DJI did not provide a physics-based model of the TELLO mini-drone. For this reason, the Tello mini-drone dynamics will be approached by black-box models that will be identified from input and output data.

1.4.4 Transfer function model of the vertical dynamic

The motor speed-to-altitude transfer function can be modelled as a first-order plus pure integrator model

$$\frac{Y(s)}{U(s)} = \frac{K}{s(1+Ts)} \tag{1.1}$$

where $Y(s) = \mathcal{L}[y(t)]$ and $U(s) = \mathcal{L}[u(t)]$.
K is the model gain and T is the model time-constant.

As the vertical velocity on the Z axis $v_z(t)$ is the time-derivative of the vertical position or altitude, both variables are linked in the Laplace domain by an integrator so that (1.1) can be expressed as:

$$\frac{Y(s)}{U(s)} = \frac{Y(s)}{V_z(s)} \times \frac{V_z(s)}{U(s)} = \frac{1}{s} \times \frac{K}{1+Ts} \tag{1.2}$$

where $V_z(s) = \mathcal{L}[v_z(t)]$ is the vertical velocity of the mini-UAV on the Z axis.
Identifying a system having a pure integrator is tricky and it is easier when the measure is available to identify the response between the vertical velocity and the input (the maximum motor speed here) since the transfer function takes the form of a simple first-order model:

$$\frac{V_z(s)}{U(s)} = \frac{K}{1+Ts} \tag{1.3}$$

1.4.4.1 Model identification from square wave response experiments

Recording of the square wave response

This experiment is carried out in open-loop/manual mode, i.e. there is no feedback control to the altitude. As the system is marginally stable, the input command should be designed with special care. As shown in Figure 1.10, the input command is a square wave input (a series of positive and negative steps) around a working operating point. The duration of each step should be carefully chosen so that the drone vertical position increases and decreases of about 50 cm without hitting the ceiling.

Warning. Remember that the experiment is conducted in open loop. If there are disturbances, like small, invisible airflow, they will induce a little roll or pitch changes into the system, the thrust will then not only adjust altitude but also create some horizontal motions and the mini-drone will start to move away from the centre of the room and possibly crashes into the net. This is why it is recommended **to close the door** of the classroom when the open-loop control test is carried out.

1. Go to classroom C335 with your TELLO mini-drone.
2. Log in to the PC with the following account: `.\admin_IA2R`
3. Ask the instructor to enter the password.
4. Go to the `C:/temp/Tello` folder.
5. Open the file `Tello_OL_control.py`.
6. Press once the power button to turn the mini-drone on.
7. Connect your mini-drone to the WIFI.
   Each mini-drone can be connected to the computer in room C335 when it is turned on via WIFI. You can identify your mini-UAV by removing the battery and looking inside the battery case. You should see written on a white paper
   WIFI: *TELLO-**XXXXXX*** where the six ***X*** represent the ID of your mini-UAV.

8. Place your Tello in the middle of the room with the in-front camera facing you.

9. Go outside of the cage.

10. Close the door of the room.

11. Run the python file. Choose `Select Run without debbuging` and then click on `Trust Workspace & Continue` or `Python debbuger and Yes`.

12. The mini-drone should auto-take off at about 1 m and waits for 3 seconds. Then a square wave input will be sent to the rotor speed. The drone should go up to 1.50m and then go down to about 1m four times and finally auto-land.

13. If everything went well, a figure will be plotted showing the experimental response data that are also recorded in the file `DataOL_Tello.txt`.

14. Copy the .txt file on a USB key or send it to you by e-mail.

15. Delete the .txt file.

16. Go back to classroom C311.

Identify your transfer function model

1. Open Matlab and execute the `dataOL_plot.m` file to get a plot similar to Figure 1.10.

2. Determine the quantization step size for both the altitude measure and vertical velocity measure.

3. Open and run the `test_your_model.mlx` file. This file will help you to adjust the first-order model parameters by using the data you recorded from initial default parameter values set at the top of the .mlx file.

4. Run the file and observe the FIT percentage between the measured and simulated model vertical speed response.

5. Adjust the parameter values for K and T at the top of the .mlx file to get the best fit possible between the two responses.

Note that from the measured data we can observe a small time-delay on the vertical speed response. We are however going to ignore this time-delay (as it is very small) for the design and tuning of the P and PD controllers.

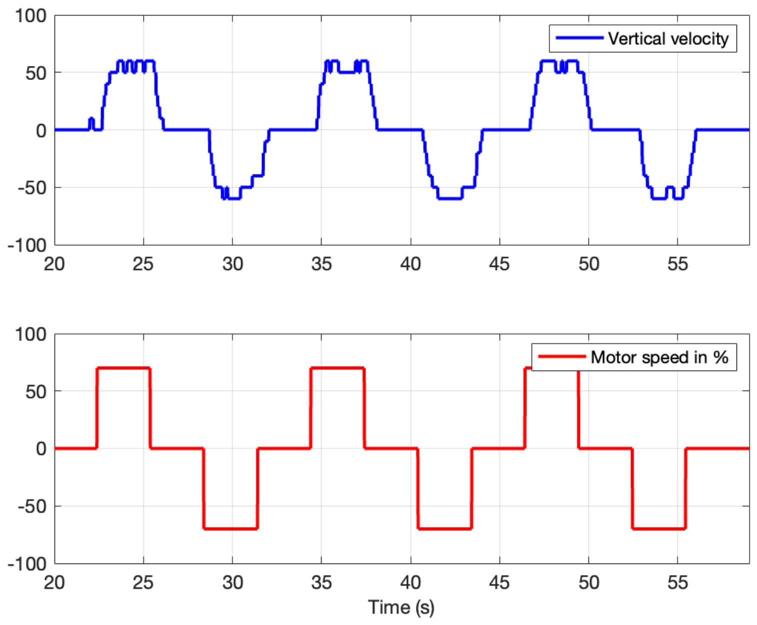


Figure 1.10: Vertical velocity response of the mini-drone to a square wave input

1.4.5 P control for altitude tracking of a square wave signal

Now we have a relatively good model of the vertical dynamic of the drone, we can design a model-based PID control for altitude tracking.
For systems represented by a first-order transfer function plus a pure integrator, a simple P controller or a PD controller should be enough to get good tracking performances.

We start by first tuning and testing the performance of a simple P controller using Simulink before implemented the P control in Python code on the TELLO mini-drone.

1.4.5.1 P controller tuning and test in Simulink

Figure 1.11 shows a simple proportional feedback configuration of the mini-drone altitude tracking system.

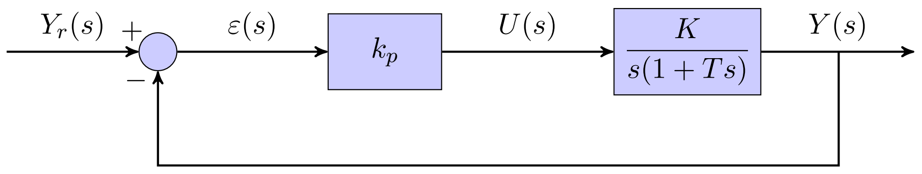


Figure 1.11: Block-diagram of the simple proportional feedback configuration of the mini-drone altitude tracking system

1. Determine the closed-loop transfer function $F_{CL}(s)$.
2. From the requirements specified in Table 1.2, determine the proportional gain value k_p.
3. Open the Simulink file `Simul_P_control_altitude.slx`. Enter the parameters of your identified model K and T and set the gain of the P controller.
4. Run the file. Are the specification requirements satisfied in simulation ? If not adjust the gain of the P controller.

5. Analyze the motor command response. Does it reach the saturation ?

6. Give the few Python lines to implement the P control.

1.4.5.2 P controller implementation in Python and test on the TELLO drone

1. Go to classroom C335 with your mini-drone.

2. Go to the `C:/temp/Tello` folder.

3. Open the file `Tello_CL_control.py`.

4. Set the P controller gain below the commented line: `Enter the P gain value below:`

5. Press once the power button to turn the mini-drone on.

6. Connect your mini-drone to the WIFI.

7. Place your TELLO in the middle of the room with the in-front camera facing you.

8. Go outside the cage.

9. Close the door of the room.

10. Run the python file. Choose `Select Run without debbuging` and then click on `Trust Workspace & Continue` or `Python debbuger and Yes`.

11. The mini-drone should auto-take off at about 1 m and waits for 3 seconds. Then a square wave will be applied to the altitude setpoint. The drone should track the square wave setpoint and finally auto-land.

12. If everything goes well, a figure will be plotted showing the experimental response data that is also recorded in the file `DataCL_Tello.txt`.

13. Copy the .txt file on a USB key or send it to you by e-mail to include it later in your report.

14. Delete the .txt file.

15. Go back to classroom C311.

16. Open Matlab and execute the `dataCL_plot.m` file to get a plot of the closed-loop control performance.

17. Analyze the motor command response. Does it reach the saturation ?

18. Analyze the altitude tracking with the simple P controller. Are the specification requirements satisfied in practice.

1.4.6 PD control for altitude tracking of a square wave signal

The proposed strategy is to use a variation of the standard PD control, where unlike the standard PD where the derivative term is usually applied to the error, it is applied to the output, as shown in Figure 1.12. This PD variation presents the advantage of obtaining a closed-loop transfer function (for a first-order +pure integrator plant model) without any zero as in the desired closed-loop transfer function $G_{ref}(s)$.

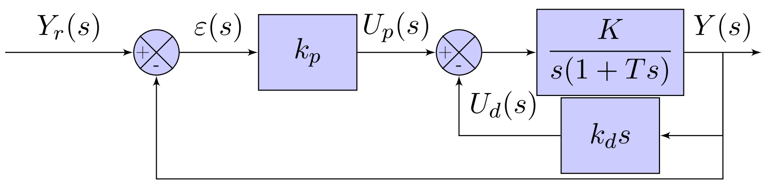


Figure 1.12: Block-diagram of the PD feedback configuration of the altitude control system

1.4.6.1 PD controller tuning and test in Simulink

1. Determine the closed-loop transfer function $F_{CL}(s)$.
2. From the requirements specified in Table 1.2, determine the proportional and derivative gains k_p and k_d of the analog PD controller.
3. Modify your Simulink file `Simul_`P`_control_altitude`.$slxtotesttheperformanceofthePDcontroller.Saveit$
4. Run the file. Are the specification requirements satisfied in simulation ? If not adjust the gains of the PD controller.
5. Analyze the motor command response. Does it reach the saturation ?
6. Give the few Python lines to implement the digital PD control. The code line example given below could be useful:
```
LastZ = 0.0
while (experimentIsRunning):
error = reference - altitude
errorDerivative = (Z - LastZ) / Ts
Up=kp*error
Ud=kd*errorDerivative
U=Up - Ud
LastZ = Z
```

where Ts is the sampling period.

1.4.6.2 PD controller implementation in Python and test on the TELLO drone

1. Go to room C335 and test the PD control implemented in Python on the TELLO mini-drone.
2. If everything goes well, a figure will be plotted showing the experimental response data that is also recorded in the file `DataCL_Tello.txt`.
3. Copy the .txt file on a USB key or send it to you by e-mail to include it later in your report.
4. Delete the .txt file.
5. Go back to the C311 room.
6. Open Matlab and execute the `dataCL_plot.m` file to get a plot of the closed-loop control performance.
7. Analyze the motor command response. Does it reach the saturation ?
8. Analyze the altitude tracking with the PD controller. Are the specification requirements satisfied in practice. Are the PD control performance better than those of the simple P control ?

1.5 Yaw control of the TELLO mini-drone

1.5.1 Yaw control performance requirements

The performance requirements for the mini-drone yaw control are described in Table 1.3.

Requirement	Assessment criteria	Level
Control of the drone yaw	Square wave setpoint tracking Peak overshoot Settling time at 5 %	No steady-state error $D_{1\%} = 4.3\%$ $t_s^{5\%} = 2$ s
Control of the drone yaw	Triangular wave setpoint tracking	No steady-state error

Table 1.3: Performance requirements for the mini-drone yaw control

1.5.2 Yaw control for a square wave signal

Based on what you have done for altitude control, develop a strategy for controlling the yaw of the TELLO mini-drone when the reference signal to follow is a square wave signal.

Test your new controller. Were you able to get 0 steady-state error?
Plot the reference signal superimposed with the yaw angle and plot also the yaw control for your best control design.

1.5.3 Yaw control for a triangular wave signal

Design a controller so that the mini-drone is able to follow a reference triangular wave signal that has period of length 20 seconds and an amplitude of 180°. Remember that a triangular wave is just a series of ramps.

To implement your controller, the code lines below could be useful:

```
integratedError = 0.0
while (experimentIsRunning):
error = reference - yaw
integratedError = integratedError + Ts*error
YAWcontrol = kp*error + ki*integratedError
```

Test your new controller. Were you able to get 0 steady-state error? Plot the reference signal superimposed with the yaw angle and plot also the yaw control for your best control design.

1.6 Summary and reflections

Summarize and reflect on what you have done and learned during this lab.

1.7 Troubleshooting

This section provides solutions to some issues you might encounter when using the TELLO drone during this lab.

1.7.1 Connection issues

If you try to connect to your drone via WIFI and it takes too long with the computer you are using, just run the Python code. If the battery level is printed, then your drone is connected to the computer.

Note that your drone will automatically switch off after 2 or 3 mn. You will need to switch it on and then connect again to the WIFI before executing the Python code.

1.7.2 Calibration issues

If the Python code returns an error message after printing the battery level, it either means that your drone battery is too low (under 10 or 20 %) or needs to be calibrated.
To calibrate your drone, you first need to download the TELLO app on your phone from this following link: `www.dji.com/uk/downloads/djiapp/tello`

Then, when you open the App, it will automatically try to connect your phone to a TELLO drone which will open your Wi-Fi interface and show you the available devices. Now, choose your drone. As it is not an internet device, your phone might ask you if you still want to connect to it. Confirm and when you are connected, go back until you see the App again.

Once you are connected to your drone and at the main interface of the App, you should see a gear on the top left side of your screen. Click on it then to "More" and to the "..." on the left side. The first option to appear should be "IMU Status". Click on "Calibrate" on the right side of this option and follow the instructions.

If you try to calibrate your drone and the App indicates that it has been calibrated without the need to place the drone in 6 different positions, close the App. Restart your drone and do the whole process again. If it is still the same, then it means your drone cannot be calibrated. Mark it and use another one.

In this calibration instructions, it is said that you should remove the propellers. There is a metallic and flat stick in the plastic bags inside the drone box. It is the Propeller Removal Tool they are talking about. The hollow part is to be used between the propeller and the motor to which it is attached.

Before removing them, mark that there is a pair with marks near the axis. The marked propellers should be at the top left and bottom right of the drone while the unmarked fill the other slots.

Note that this calibration is necessary for the Python code to be uploaded to the drone.

English to French glossary

bandwidth	:	bande passante
crane	:	grue
closed-loop system	:	système bouclé
cut-off frequency	:	fréquence (ou pulsation) de coupure
damped frequency	:	pulsation amortie
damping ratio	:	coefficient d'amortissement
drag	:	traînée
feedback	:	contre-réaction
feedback system	:	système à contre-réaction
hoisting device	:	dispositif de levage
impulse response	:	réponse impulsionnelle
integral wind-up	:	emballement (de l'action) intégral
input	:	entrée
gain	:	gain
heading angle	:	angle de cap
linear time-invariant (LTI)	:	linéaire invariant dans le temps
motor shaft	:	arbre moteur
output	:	sortie
overdamped	:	sur-amorti
overshoot	:	dépassement
pitch	:	tangage
rise time	:	temps de montée
road grade	:	inclinaison de la route
robot arm joint	:	articulation d'un bras de robot
rool	:	roulis
root locus	:	lieu des racines
setpoint	:	consigne
settling time	:	temps de réponse
steady-state gain	:	gain statique
steady-state response	:	réponse en régime permanent
steering	:	direction
step response	:	réponse indicielle
stream	:	courant
yaw	:	lacet

time-delay	:	retard pur
time-invariant	:	invariant dans le temps
transient response	:	réponse transitoire
throttle	:	accélérateur
undamped	:	non amorti
undamped natural frequency	:	pulsation propre non amortie
underdamped	:	sous-amorti