POLITECNICO DI MILANO
School of Industrial and Information Engineering
Master of Science in Automation and Control Engineering

Data-driven attitude control design for multirotor UAVs

Advisor: Prof. Marco LOVERA
Co-Advisor: Ing. Pietro PANIZZA
Ing. Davide INVERNIZZI
Ing. Mattia GIURATO

Thesis by:
Thibaud Chupin Matr. 821382

Academic Year 2015–2016

Voici mon secret. Il est très simple: on ne voit bien qu'avec le cœur.
L'essentiel est invisible pour les yeux.
– Le Renard, Dans Le Petit Prince, Antoine de Saint Exupéry

Here is my secret. It is very simple: It is only with the heart that one can see rightly; what is essential is invisible to the eye.
– The Fox, In The Little Prince, Antoine de Saint Exupéry

Acknowledgements

Research is always a collaborative effort and it would be remiss of me if I did not thank all those who helped.

I would like to like to express my deep gratitude to Professor Lovera for offering me the opportunity to work on a project this interesting. His patience and guidance were essential to the completion of this work.

I must also thank Professor Ferrigno and Professor De Momi for their support during these last months. Without their encouragement I may never have graduated.

I would also like to thank Pietro Panizza, Davide Invernizzi and Mattia Giurato for their help. Without their understanding of control theory and quadrotors I doubt whether I would ever have been able to finish.

Finally I would like to thank my parents for always believing in me even when I was ready to give up.

Abstract

Small multirotor unmanned aerial vehicles (UAV) are a ground-breaking invention. They have made accessible to hobbyists and professionals alike technologies that, until recently, were prohibitively expensive. They are used by farmers to survey their fields and evaluate their fertility, by videographers looking to capture impressive images in remote and inaccessible areas or by power companies to monitor the state of high voltage power lines for example. These are but a sample of the infinite applications of such vehicles. These vehicles are, simple , sturdy and affordable. The downside is the relative complexity of the control schemes required to pilot them. A complex attitude control system is required to make the vehicle controllable.

The most complex part of a multirotor is the control software implementing the attitude and position control loops. In traditional controller synthesis a prerequisite is the availability of an accurate model of the system the development of which is non-trivial. An alternative approach is offered by so-called data-driven methods. These methods identify a suitable controller by solving a parameter identification problem and thus avoid the need to develop a model.

The goal of this thesis is to use the latter methods to obtain an attitude controller with performance at least as good as what has been previously achieved with traditional methods. In detail, the reasoning behind the choice of the specific controller tuning algorithm will be presented following which the methodology used to develop the controllers in practice will be exposed. Most importantly, the results obtained from both a simulation of the system and experimental tests will be shown.

Contents

List of Figures

List of Tables

Introduction

Throughout history the development of heavier than air vehicles can generally be split between fixed wing aircraft (*i.e.*aeroplanes) and rotorcraft (*i.e.*helicopters). The former use engines aligned along the longitudinal axis of the vehicle to provide forward thrust, fixed wings for lift and a system of control surfaces to generate the control torques and forces. The latter rely on a single large main rotor where the pitch angle of each blade can be controlled which provides both lift and thrust as well as a smaller rotor to provide a counter-torque. The vehicle is then controlled by varying the individual pitch of the rotor blades over the course of each rotation. With few exceptions, such as tilt-wing aircraft that have elements of both fixed and rotary wing aircraft, this separation has remained accurate until recently.

Multirotor vehicles have emerged as a radically different approach to heavier than air flight. These are rotary wing aircraft that, instead of depending on a large main rotor for lift and control, employ many smaller rotors. Whereas a single large rotor requires a powerful (and thus expensive) motor to power, smaller rotors can be powered with smaller motors whose cumulative cost is less than the single large motor.

The downside is that the number of motors significantly complicates the control scheme. Simple rotations and translations now require the combined use of several motors which, in turn, requires a complex on-board flight control unit (FCU). This FCU usually implements the different control loops using data from an on-board inertial measurement unit (IMU) that provides a continuously updated view of the aircraft's orientation.

In turn the reduction in costs has created new opportunities: multi rotor drones are used for low-cost aerial surveying and mapping, airborne video and photography or, more recently, delivery. This leads to a wide variety of payloads with different sizes and weights. This wide range of applications has also lead to a wide variety of vehicles: multirotors with three, four, six or even

eight rotors are quite common. Of these, the most common configuration is without a doubt the quadrotor: a machine with with four rotors. These have been widely used to fly small payloads in the neighbourhood of 1kg, are relatively low cost and simple to control.

Thesis Description

The problem of attitude control in quadrotors is the focus of this thesis. With such a wide variety of vehicles sizes and applications a *one-size-fits-all* approach cannot provide satisfactory results. The individual attitude controllers must be tuned manually to achieve acceptable levels of performance. This is usually done by first producing an accurate physical model of the quadcopter before applying classical control theory methods such as H_∞ tuning to synthesise a suitable controller. This procedure is long, complex but most of all, is only performed once over the lifetime of the vehicle and cannot account for the degradation of the components as they age.

Data-driven methods represent an alternate approach to the controller synthesis problem. Instead of requiring and accurate model of the system they rely entirely on the availability of a set of input-output data measured in open loop conditions and directly produce the parameters of a controller minimising a specified cost criterion.

These approaches are attractive firstly because they do not require an accurate model of the system to be controlled but most of all because they can be easily re-run to account for changes in the hardware of the system. One could for example periodically re-tune the controllers to account for the aging of the components or even temperature variations. It may also be possible to leverage this approach to automatically re-tune the controllers in flight to account for the specific size and shape of the payload.

This thesis represents a single early step on the way to such auto-tuning systems. In it the applicability of a specific data-driven controller synthesis technique: Virtual Reference Feedback Tuning to the pitch control loop and the performance levels that can be achieved are explored. The results were compared with a pre-existing H_∞ -tuned controller.

Thesis Structure

The first chapter of this thesis aims to provide the reader with a more in-depth understanding of multirotor systems in general and, considering quadrotors specifically, the design considerations behind their structure. The second half of the chapter concerns itself with an explanation of how a quadrotor moves and turns, which control inputs are required. This leads us to consider the individual pitch, roll and yaw control loops and their structure.

The second chapter looks at controller tuning, comparing the classical approach with model-reference based methods. In talking about the latter model-based and data-driven approaches were considered before several different data-driven controller synthesis methods were detailed. After explaining why VRFT is the better choice for this problem an in depth explanation of this method is provided.

The third chapter looks at how the requirements on the system were translated for the VRFT method and how a known model can be used to guide the VRFT procedure. It shows how the simulated model was built from the pre-existing knowledge of the machine and how this knowledge was used to inform the structure of the reference models used to specify the performance for the VRFT. Finally it shows how a set of input-output data was simulated to be used for the VRFT and the performance of the simulated model.

The fourth and final chapter presents the specifics of the actual quadrotor that was used for which the controller was tuned. It presents a brief overview of the hardware components and the test-bed. It then shows how the input-output data required for VRFT was collected and processed, how the reference models used were identified and finally it presents the results of the experiments testing campaign.

Chapter 1

Quadrotors & Attitude Control

Quadrotors are multi-rotor aerial vehicles that exploit differential thrust from their motors to move. Whilst quadrotors are the most common form of multi-rotor in use today they are not by any means the only one: configurations with three (tricopters), six (hexacopters) and eight rotors (octocopters) have also seen significant use.

Most multi-rotor designs strive to use relatively inexpensive commodity motors to keep the system affordable. To that end, when trying to increase the payload of the system it is often preferable to increase the number of motors instead of sourcing more powerful and more expensive motors capable of driving larger propellers. This naturally increases the effective payload of the system as well as its redundancy. Whereas the loss of one motor on a tricopter or a quadcopter would have disastrous consequences on the controllability of the system, the loss of one motor on an octocopter might be relatively unnoticeable. The downside is that increasing the number of motors leads to a lower global efficiency compared to simply increasing the swept area of the rotors.

Quadcopters today have emerged as the standard architecture for lightweight systems with payloads weighing in the neighbourhood of 1 kg. They provide acceptable levels of cost, redundancy, payload and performance and are used widely by hobbyists and professionals alike.

In Section 1.1 the general design considerations that must be taken into account when designing a quadcopter will be briefly discussed following which

the control inputs necessary to achieve the various motions of the system will be shown in Section 1.2. Finally, in Section 1.3 the various attitude control loops will be shown.

1.1 Design Considerations

Unlike fixed wing aircraft that employ a complex system of control surfaces to generate control forces and moments, multicopters can only vary the thrust provided by each propeller. This can be achieved in a number of ways each providing different trade-offs.

The most common way to vary the thrust of each propeller is to vary the speed of each motor. This is the simplest way from both a mechanical and control theory standpoint. By increasing or decreasing its speed, the amount of thrust produced by each propeller can be increased or decreased. This simplicity comes at a cost: the time constants of an electric motor spinning at several thousand rpm are comparatively high which limits the overall agility of the system.

A scheme that allows for faster reactions at the cost of complexity is to tilt the arms onto which the motors are attached. In this configuration small changes in vertical thrust can be achieved with relatively small rotations of the arms. Whilst these can be actuated rather quickly, it is not possible to obtain large changes to the vertical thrust without also varying the speed of the propellers.

The fastest but most complex way of varying thrust is to control each propeller like the main rotor of a helicopter. A mechanical linkage allows the control system to change the pitch angle of the blades of the propeller. This allows extremely fast changes in thrust but is significantly more complex mechanically. This configuration is generally reserved for larger quadcopters where the inertia of the propellers might make the previous methods too slow.

A second important design choice is whether to adopt an X or a $+$ structure. In the first configuration the motors are located on each side of the system, one at each tip of the X. In the latter configuration two motors are situated along the longitudinal axis of the system: one in front of and one behind the centre. The other two motors are placed along the lateral axis of the system: one right and one left of the centre.

The X configuration is often retained because it can provide more rotational acceleration and, if a camera is attached to the system it can be pointed forward without the landing gear being in its field of view.

The quad copter used for this work has an X configuration and fixed pitch rotors.

1.2 Control of a Fixed Pitch Quadcopter

In a fixed pitch quadrotor the only way to vary the attitude of the system is to intervene on the speed of the motors. Thus, with only four actuators to control the six spatial degrees of freedom the system is underactuated. Vertical movement as well as the pitch, roll and yaw dynamics is directly controlled. Forward and lateral translation movements however must be controlled with the dynamics of the whole system.

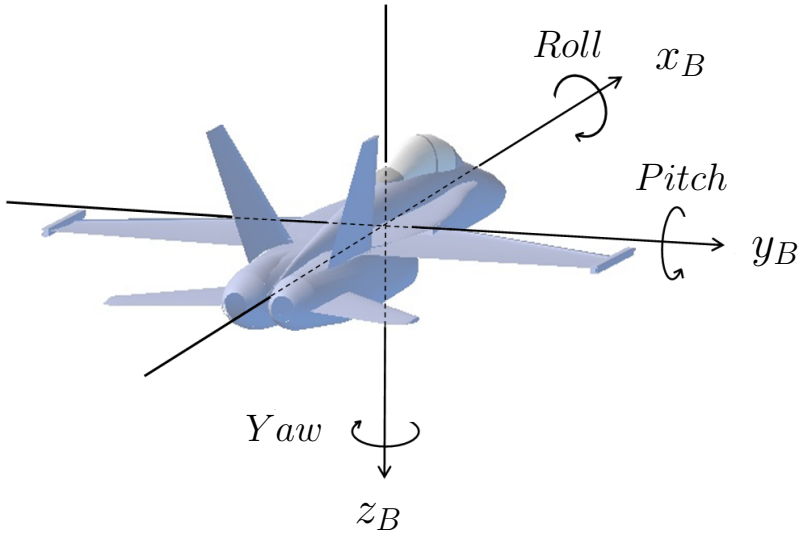


Figure 1.1: Illustartion of the Roll, Pitch & yaw Motions

The pitch, roll, yaw and elevation motions of the quadcopter are uncoupled. Changing the yaw angle does not affect the roll or pitch dynamics for example. From a practical standpoint this means that the problem of designing a controller can be reduced from a MIMO problem to a set of independent SISO problems. The outputs of the different controllers are

simply summed to obtain a combined motion.

Additionally, unlike helicopters, quadrotors do not require anti-torque tail rotors. This is achieved by having half of the propellers rotate in a clockwise direction and half rotate in a counterclockwise direction. As shown in Figure 1.2, on the vehicle used for this work, rotors 1 (front left) and 3 (rear right) rotate clockwise whilst rotors 2 (front right) and 4 (rear left) rotate counter clockwise.

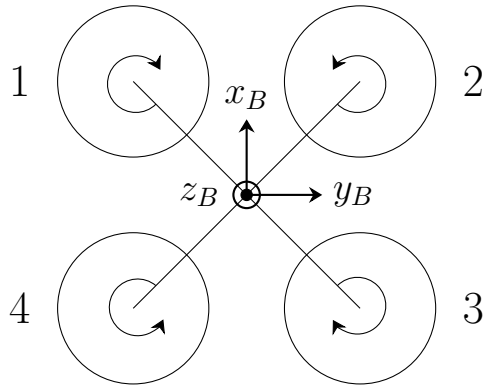


Figure 1.2: Rotation directions of the quadrotor propellers

1.2.1 Elevation

To maintain the quadcopter in stable hovering flight a base speed (Ω_H) corresponding to a nominal rpm must be applied to all the propellers to provide a thrust sufficient to counter gravitational effects. To fly upwards or downwards it is sufficient to vary the speed of all four rotors by the same amount $\delta\Omega_T$ such that

$$\Omega_{1,2,3,4} = \Omega_H + \delta\Omega_T \tag{1.1}$$

where Ω_1, Ω_2, Ω_3 and Ω_4 are respectively the speeds of rotors 1, 2, 3 and 4. This generates a vertical force along the z_B axis.

1.2.2 Roll

Roll motion (a rotation around the longitudinal axis) is achieved by increasing the speed of the rotors on the left of quadcopter (1, 4) by some amount

$\delta\Omega_R$ and decreasing the speed of the rotors on the right (2, 3) by the same amount. This generates a torque around the x_B axis of the quadcopter.

$$\begin{aligned} \Omega_{1,4} &= \Omega_H + \delta\Omega_R \\ \Omega_{2,3} &= \Omega_H - \delta\Omega_R. \end{aligned} \tag{1.2}$$

1.2.3 Pitch

Pitch motion (a rotation around the lateral axis) is obtained by increasing the speed of the rotors on the front of the quadcopter (1, 2) by some amount $\delta\Omega_P$ and decreasing the speed of the rotors on the rear (3, 4) by the same amount. This generates a torque around the y_B axis of the quadcopter.

$$\begin{aligned} \Omega_{1,2} &= \Omega_H + \delta\Omega_P \\ \Omega_{3,4} &= \Omega_H - \delta\Omega_P. \end{aligned} \tag{1.3}$$

1.2.4 Yaw

Yaw motion (a rotation around the vertical axis) is achieved by increasing the speed of the clockwise turning rotors by some amount $\delta\Omega_Y$ and decreasing the speed of the counterclockwise turning rotors by the same amount. This generates an imbalance of angular momenta around the z_B axis which causes the system to rotate around the vertical (z_B) axis .

$$\begin{aligned} \Omega_{1,3} &= \Omega_H + \delta\Omega_Y \\ \Omega_{2,4} &= \Omega_H - \delta\Omega_Y. \end{aligned} \tag{1.4}$$

1.2.5 Longitudinal & Lateral translation

Longitudinal and lateral motions cannot be controlled directly with just four control variables. Instead the dynamics of the entire system are used to obtain these motions. Specifically, to move the quadcopter forward the system is tilted forward (pitched forward). To move backward the opposite motion is applied. The same reasoning applies to lateral translations where instead of tilting around the pitch axis the system is rolled left and right.

1.3 The Attitude Control Loops

A closed loop control system is required in order to precisely control the orientation of the aircraft. In Section 1.2 we showed that the rotational degrees of freedom of the vehicle are uncoupled and that the attitude control problem can be viewed as a set of independent SISO problems.

When looking at a X-frame quadcopter, after removing any bodywork and aerodynamic fairings, it can be quite difficult to differentiate the front and the sides of the system. This symmetry is exploited when designing the control system: the pitch and roll control loops can generally be considered identical. Only the yaw regulation loop differs.

Rather counter-intuitively the pitch and roll loops are stable. They must however be extremely reactive in order to compensate for external disturbances such as wind gusts.Thus, to maintain the vehicle in stable flight it is essential that these control loops be as fast as possible. The performance of the yaw control loop by contrast is imposed not by physical imperatives but by the ability of the pilot. Whilst it would be possible to achieve extremely high turn rates a human pilot must be able to control the vehicle and as such the bandwidth of this control loop is kept artificially low. This difference is apparent when comparing the structure of the pitch, roll and yaw control loops. On the quadcopter used for this work the pitch and roll control are implemented with a nested controller scheme whilst the yaw control loop uses a simpler single-loop architecture.

1.3.1 The Pitch Control Loop

The pitch control loop employs two nested loops to achieve the best possible performance. This is due to the fact that the pitch rate dynamics of the system, controlled by modifying the speed of the rotors, are considerably faster than the pitch angle dynamics.

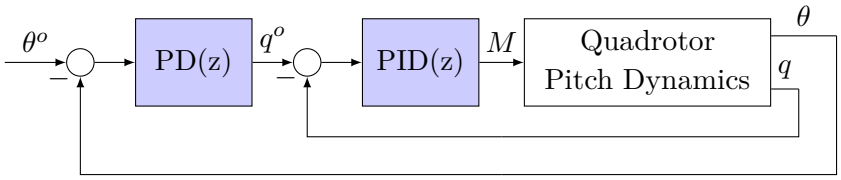


Figure 1.3: The pitch control loop

The controllers implemented in the firmware of our quadcopter have a fixed structure. The outer loop controller is a PD whereas the inner loop controller is a PID. These controllers are expressed in the ideal parallel form

$$\begin{aligned} PD(s) &= K_{p_i} + \frac{K_{d_i} s}{T_f s + 1} \\ PID(s) &= K_{p_o} + \frac{K_{i_i}}{s} + \frac{K_{d_i} s}{T_f s + 1} \end{aligned} \tag{1.5}$$

where K_{p_i}, K_{i_i} and K_{d_i} are respectively the proportional, integral and derivative gains of the the PID controller, K_{p_o} and K_{d_o} are respectively the proportional and derivative gains of the the PD controller and T_f is the filter time constant.

The control variable produced by the inner controller is the torque around the pitch axis. The real input to the plant however is the difference in speed between the front rotors (numbers 1 & 2) and the rear rotors (numbers 3 & 4). The transformation from a torque to a speed difference is the responsibility of the mixer matrix χ.

Consider the fact that each propeller generates a vertical thrust and, due to the distance between the centre of the propeller and the centre of mass of the vehicle, a torque. Consequently each propellers contributes to the vertical thrust T and the moments L, M and N around, respectively, the pitch, roll and yaw axes. It can be shown that the cumulative force generated by the propellers is

$$F_{Props} = \begin{bmatrix} 0 \\ 0 \\ K_T \left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \end{bmatrix} \tag{1.6}$$

where Ω_i is the rotational speed of the i-th motor and K_T is a known constant. It can also be shown that cumulative moments L, M and N around the x_B, y_B and z_B axes are

$$M_{props} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} K_T \frac{b}{\sqrt{2}} \left(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2\right) \\ K_T \frac{b}{\sqrt{2}} \left(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2\right) \\ K_Q \left(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2\right) \end{bmatrix} \tag{1.7}$$

where K_Q is another known constant. The forces and moments can be rearranged in order to isolate on one side L, M, N and T and on the other

Ω_1^2, Ω_2^2, Ω_3^2 and Ω_4^2

$$\begin{aligned}\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} &= \begin{bmatrix} K_T & K_T & K_T & K_T \\ K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} \\ K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} \\ K_Q & K_Q & K_Q & K_Q \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \\ &= \chi \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}\end{aligned} \tag{1.8}$$

where χ is the mixer matrix which relates the required thrusts and moments to around each axis to the rotational speed of each propeller.

1.3.2 The Roll Control Loop

The roll control loop as implemented on the quadcopter that was used for this work is identical to the pitch control described in Section 1.3.1. It differs only in the pairs of rotors it considers. The input to the plant is the difference in speed between the right rotors (numbers 1 & 4) and the left rotors (number 2 & 3).

1.3.3 The Yaw Control Loop

Due to its lower bandwidth requirements the Yaw control loop is the simplest of the attitude control loops.

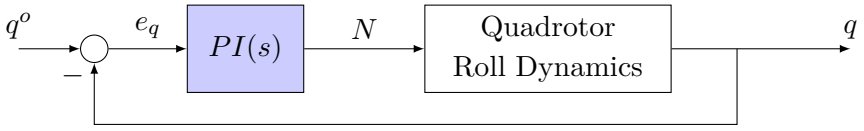


Figure 1.4: The yaw control loop

On this quadcopter the controller is implemented as a simple PI expressed in the ideal parallel form

$$PI(s) = K_p + K_i\frac{1}{s} \tag{1.9}$$

which controls the angular rate of the system around the yaw axis. Unlike the pitch and roll loops the yaw control loop groups the motors by the direction of rotation of the propellers. The real control input is the difference in speed between the clockwise turning and the counterclockwise turning propellers. The transformation from an angular momentum to a speed difference is assured by the mixer matrix.

Chapter 2

Data Driven Control Methods: State of the Art

Virtual Reference Feedback Tuning (VRFT) and its application to the problem of attitude control is the focus of this thesis. VRFT is a *data-driven* controller synthesis technique, part of a more recent branch of control theory that, instead of relying on the availability of a model, depend only on the existence of a suitable set of input output data.

In Section 2.1 we will discuss the traditional approach to controller design and specifically look at H_∞ synthesis. In Section 2.2 we will explore *model-reference* methods, comparing *model-based* and *data-driven* approaches. Finally in Section 2.2.2 I will detail the virtual reference feedback tuning approach.

2.1 The Classical Approach

In the traditional approach to controller synthesis the requirements on the closed loop behaviour of the system are expressed as simple conditions on, for example, the bandwidth of the system or its disturbance rejection properties. In addition some robustness requirements may be considered such as requiring a certain gain and phase margin.

One such method is H_∞ synthesis. In this framework the requirements on the closed loop system are expressed as constraints on the H_∞ norm of its transfer function. This framework can extend the classical synthesis

techniques to multi-loop and MIMO control architectures. However, practical considerations have slowed its adoption: whereas industrial controllers usually have a decentralised architecture employing many simple controllers the H_∞ synthesis produces a monolithic high-order controller.

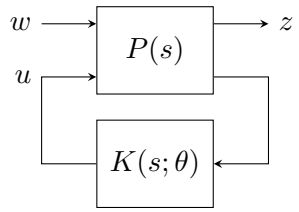


Figure 2.1: The standard form for structured H_∞ synthesis

Structured H_∞ synthesis is a solution to this problem. All the non tunable blocks of the system are combined into a single block $P(s)$ and all the tunable elements are merged into a single structured controller $K(s;\ \theta)$ parametrised in θ.

Solving the H_∞ problem consists in identifying the parameter vector θ that minimises

$$\|T_{w\to z}\left(P(s),\ K(s;\ \theta)\right)\|_\infty \tag{2.1}$$

subject to the constraint that $K(s;\ \theta)$ stabilises $P(s)$ where $T_{w\to z}(P(s),\ K(s;\ \theta))$ is the closed loop transfer function from w to z on which the requirements the requirements have been imposed.

2.2 Model-Reference Controller Tuning

Model-reference approaches to controller tuning differ from traditional methods in how the requirements for the controller are specified. Instead of providing explicit limits on overshoot, bandwidth or response time the requirements are provided in the form of a reference model for the closed loop behaviour of the system.

The objective is to design a controller such that the difference between the reference model and the actual closed-loop behaviour of the system is as small as possible.

Consider the closed loop system shown in Figure 2.2 with the stable linear SISO plant $G(z)$, the controller $C(z;\ \rho)$ parametrised in ρ and the

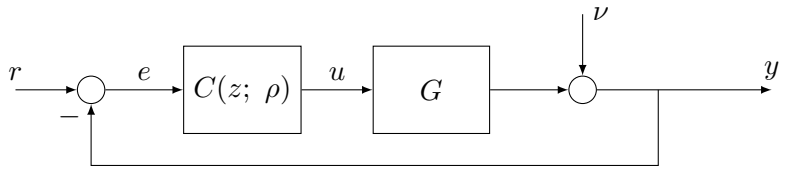


Figure 2.2: A generic closed loop system with a single degree of freedom controller and a disturbance on the output

stable strictly proper reference model $M_R(z)$. The objective of minimising the difference between the reference model and actual closed loop transfer functions can be formulated as

$$J_{mr}(\rho) = \left\| M_R(z) - \frac{C(z;\ \rho)G(z)}{1 + C(z;\ \rho)G(z)} \right\|_2^2. \tag{2.2}$$

This is not the only possible control objective. Indeed, different methods usually specify their own but they are usually similar in intent. In this specific case the ideal controller C_0 is

$$C_0(z) = \frac{M(z)}{G(z)\left(1 - M(z)\right)}. \tag{2.3}$$

We can identify two approaches to solving this problem:

- **Model-Based** approaches assume that a detailed and reliable model of the plant is available in order to directly compute the ideal controller.
- **Data-Driven** approaches attempt to minimise the control objective $J_{MR}(\rho)$ by solving a parameter optimisation problem without first estimating a model of the plant.

The following section will be dedicated to detailing both approaches and their limitations. We will also rapidly introduce several algorithms that implement data-driven approaches and explain their specific limitations in order to justify our decision to use VRFT.

2.2.1 Model-Based Control

As previously noted, model-based approaches solve the model-reference problem by assuming that a plant model is available. This model may be

derived from an understanding of the underlying physics or obtained with an identification procedure. This requires the collection of a sufficiently large set of input-output data from the plant in open-loop conditions. The choice of a high order model reduces the modelling error to such a level that it can be considered negligible in later steps. The identified model is then used to compute a full-order controller that minimises the control objective.

In many applications however the type of controller is pre-determined. Many industrial processes, for example, use pre-defined PID blocks and the control procedure is limited to tuning the PID gains. If the tuning method produces only high-order controllers an additional model-order reduction pass. This step is generally problematic since any stability guarantees that were formulated for the full-order controller may not transfer to the reduced-order controller. Furthermore whilst the optimality of the full-order controller can be guaranteed that is not the case for the reduced-controller. It may not even be the best controller in its class. For this reason, structured model-based control techniques have been developed using an approach similar to that of structured H_∞ control.

2.2.2 Data-Driven Control

Data-driven controller tuning methods skip the modelling phase entirely and instead reformulate the controller identification procedure as a parameter optimisation problem in which the optimisation is carried out directly on the controller parameters.

The principal advantage of data-driven methods compared to their model-based brethren is their ability to tune low-order controllers directly whereas model-based methods may produce n without having to first pass through a model order reduction stage. This ensures that any stability or optimality guarantees provided by these methods will not be lost. Furthermore, the achieved performance of the controllers is not linked to the techniques used to model the plant or the order of the identified model.

In practice there are many ways to achieve this objective. In this section we will detail several methods. In particular we will explain the functioning of Iterative FeedbackTuning (IFT), Correlation Based Tuning (CBT) and Virtual Reference Feedback Tuning (VRFT).

Iterative Feedback Tuning

IFT, originally described in [6], is an iterative method that uses a *gradient-descent* algorithm to compute a local minimum of the cost function. The key observation of IFT is that the chosen cost criterion can be estimated by using carefully designed experiments on the plant.

Consider a generic closed loop system with a single degree of freedom controller as in Figure 2.2. Given a reference model $M_R(z)$ the desired output of the plant in response to a reference input r is

$$y^d = M_R(z)r \tag{2.4}$$

and consequently, the error between the achieved and desired responses is

$$\tilde{y}(\rho) = y(\rho) - y^d \tag{2.5}$$

where ρ is the parameter vector and the (ρ)-argument indicates that the terms were collected whilst the controller $C(z;\ \rho)$ was in place. The control objective that IFT seeks to minimise is the following:

$$J(\rho) = \frac{1}{2N} E\left[\sum_{t=1}^{N} \left[L_y(z)\tilde{y}(\rho)\right]^2 + \lambda \sum_{t=1}^{N} \left[L_u(z)u(\rho)\right]^2\right] \tag{2.6}$$

where N is the number of points in the input-output dataset, $L_y(z)$ and $L_u(z)$ are frequency weighting terms and $u(\rho)$ is the control variable measured with the controller $C(z;\ \rho)$ in place.

This criterion considers the $\mathcal{L}_2$-norm of the frequency weighted error between the achieved and desired responses and penalises the frequency weighted control action. An algebraic solution to this problem would require knowledge of the partial derivatives $\frac{\partial \tilde{y}}{\partial \rho}(\rho)$ and $\frac{\partial u}{\partial \rho}(\rho)$ which are unknown.

IFT elegantly repurposes the plant itself to solve the problem. Indeed, through careful choice of the input signals and simple computations the derivatives can be estimated as described in [6].

The estimate of the gradient can be updated at each iteration using these simple experiments in order to asymptotically approach the local optimum of the control objective. At each iteration these experiments are re-run and the gradient is estimated.

Therein lies the main limitation of IFT and gradient-descent based methods in general: it can only guarantee a local optimum. Thus, the quality of the achieved controller depends in large part on the choice of the starting guess for the parameters.

A very attractive property of the method is that it is even applicable to non-linear systems. In this case the system should simply be linearised at every iteration.

Correlation Based Tuning

CBT [8] is a *one-shot* method to obtain a controller that minimises the control objective whilst only requiring that the input signal provided to the plant be persistently exciting. It re-frames the model-reference problem as a decorrelation problem using a single set of input-output data.

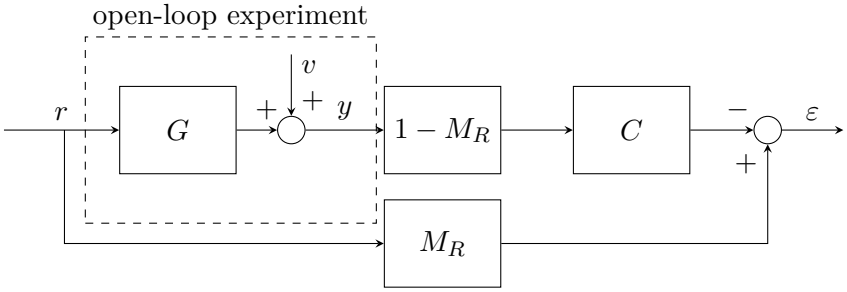


Figure 2.3: Tuning scheme for correlation based tuning

Consider the scheme in figure 2.3. If the model matching problem is perfectly solved then

$$M_R = \frac{C_0 G}{1 + C_0 G} \quad \Rightarrow \quad 1 - M_R = \frac{1}{1 + C_0 G} \tag{2.7}$$

and consequently

$$M_R = G(1 - M_R)C_0 \tag{2.8}$$

as in the block diagram. Thus, the error $\varepsilon(t;\ \theta)$ can be computed as

$$\begin{aligned} \varepsilon(t;\ \theta) &= M_R r - C(\theta)(1 - M_R)y \\ &= M_R r - C(\theta)(1 - M_R)r - C(\theta)(1 - M_R)v. \end{aligned} \tag{2.9}$$

The goal is to find the optimal parameter vector θ such that the error $\varepsilon(t;\ \theta)$ is uncorrelated from the reference signal r. To that end an extended instrumental variable $\zeta(t)$ correlated with r is introduced.

$$\zeta(t) = [r_L(t+l) \quad \dots \quad r_L(t) \quad \dots \quad r_l(t-l)]^T \tag{2.10}$$

where l is a sufficiently large integer and r_L is a suitably filtered version of r. The exact form of the filter is discussed in [8]. The correlation function is defined as

$$f_{N,l}(\theta) = \frac{1}{N}\sum_{t=1}^{N}\zeta(t)\varepsilon(t,\ \theta) \tag{2.11}$$

and the correlation criterion to minimise is

$$J_{N,\ l} = f_{N,l}^T(\theta) f_{N,l}(\theta). \tag{2.12}$$

It can be demonstrated that the optimal parameters for (2.12) asymptotically converge to the optimisers of the model-reference problem.

Virtual Reference Feedback Tuning

VRFT [2], like CBT is a *one-shot* method that only requires that the input to the plant be persistently exciting in order to minimise the control objective. In practice however the achieved parametrised controller will only be *near-optimal* since the probability that the ideal controller (*i.e.*the controller that perfectly solves the model-matching problem) belongs to the chosen controller family is vanishingly small.

Consider once again a parametrised single degree of freedom controller inserted into a closed loop system as in Figure 2.2. The key concept underlying VRFT is that if the error signal $e(t)$ and the plant input signal $u(t)$ are known then the model matching problem can be reformulated as a parameter identification problem on the controller. Specifically, the methods seeks to minimise the parameter vector θ such that the criterion

$$J(\theta) = E\left[(u_L(t) - C(z;\ \theta)e_L(t))^2\right] \tag{2.13}$$

where L is an appropriate pre-filter for the data is minimised. Conveniently this criterion is convex and can be expressed entirely in terms of the input-output data collected from the plant. Accordingly, the optimal parameter vector is

$$\hat{\theta}_N = \left[\sum_t \varphi_L(t)\varphi_L(t)^T\right]^{-1} \sum_t \varphi_L(t)u(t). \tag{2.14}$$

In the case of noisy measurements the method can be extended with the use of a second input-output dataset using the same input signal and instrumental variable based approach. This however may not always be possible. Alternatively the noise can be estimated with an appropriate ARX model however identifying the correct order for the model is non-trivial and an incorrect choice will lead to unsatisfactory performance.

2.2.3 Comparison of the data-driven methods

The three data-driven methods presented previously all have different limitations and require different trade-offs.

IFT, being an iterative method, is comparatively slow and requires several experiments on the plant at each iteration. Moreover it can only guarantee that the result is close to the local minimum of the cost function.

CBT is an extremely efficient method. It only requires one set of input-output data and some relatively simple calculations to to tune a globally optimal controller within the bounds of the chosen controller family.

VRFT is also very efficient computationally and data-wise. Furthermore, it has been extended in [4] to tune nested control loops of arbitrary depths with a one shot procedure. The principal drawback of VRFT is the issue of noise. None of the proposed solutions are ideal.

Whilst an iterative procedure like IFT would be technically acceptable, the turnaround time due to many required experiments disqualified the method from our consideration. In comparison CBT and VRFT both offer extremely fast turnaround times since many controllers can be tuned from the same set of input-output data by simply changing the reference model. This allowed us to interactively explore the performance limitations of the system. These fast turnaround times also make it easy to re-tune the controllers to compensate for payload changes or the degradation of the actuators due to

fatigue and temperature variations.

The final choice of VRFT was due to two predominant considerations:

- VRFT has already been successfully applied to the cascade control problem and a `MATLAB` toolbox implementing the method is available [3].
- significant pre-existing work exists documenting the application of VRFT to quad-rotors (*e.g.*, [7]).

Having detailed the reasons for the choice of VRFT to develop the controllers the method will now be presented in more detail.

2.3 VRFT In Detail

In VRFT the aim is to identify the parameters of a controller such that the complementary sensitivity of the system aligns with a user-specified *reference model* describing the desired behaviour of the closed loop system. This is achieved without requiring any knowledge of the structure of the system and using only open-loop measurements.

The method is direct, it does not require a prior identification of the plant, and searches for a global optimum of the design criterion. Additionally, if the controller complexity is restricted the produced controller is a good approximation of the restricted complexity global optimal controller.

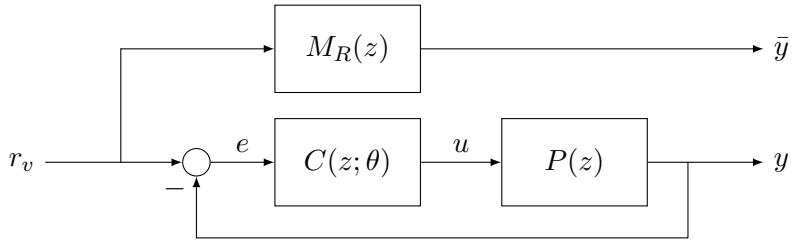


Figure 2.4: Tuning scheme for VRFT

The method transforms the control problem into an identification problem that minimises the $\mathcal{L}_2$-norm of the mismatch between the reference model and the actual closed loop system.

If both the input and the output of the controller are known then it is possible to find an optimal parameter vector that achieves this goal.

The key observation of this approach lies in the fact that this is possible when the input signal of the reference model is chosen with care. If the input we provide to the reference model is such that its output is equal to the measured output of the plant we can easily calculate the tracking error of the closed loop system.

2.3.1 A Rigorous Explanation

We assume that the plant under consideration is a linear SISO dynamical system described by an unknown, rational transfer function $P(z)$.

We also assume that a set of open-loop input-output data has been collected during an experiment on the plant. The only requirement for this experiment is that it must excite the system over the entirety of the frequency range of interest. This experiment provides us two vectors of data-points

$$u = \{u_0,\ u_1,\ u_2,\ \dots\ , u_n\}$$
$$y = \{y_1,\ y_2,\ y_3,\ \dots\ , y_n\}\,.$$

We must also determine a reference model that describes the desired closed-loop behaviour of the system. Great care should be taken here to choose a model that is physically achievable given the physical constraints of the system. If the reference model is unachievable the controller will be of very limited use. Let this reference model be $M_R(z)$.

Finally, we must decide on a family of controllers to tune. We restrict our attention to the class of controllers that can be expressed as a linear combination of linear, discrete time, transfer functions. This controller family takes the form

$$C(z;\ \theta) = \beta^T(z)\theta \tag{2.15}$$

where $\beta^T(z)$, the vector of linear transfer functions is

$$\beta^T(z) = [\beta_1(z),\ \beta_2(z)\ \dots\ \beta_n(z)]^T$$

and θ, the parameter vector is

$$\theta = [\theta_1,\ \theta_2\ \ldots\ \theta_n]^T .$$

The control objective is the minimisation of the following performance criterion:

$$\begin{aligned} J_{MR}(\theta) &= \left\| \left(M_R(z) - \frac{P(z)C(z;\ \theta)}{1 + P(z)C(z;\ \theta)} \right) W(z) \right\|_2^2 \\ &= \left\| \left(T(z) - M_R(z) \right) W(z) \right\|_2^2 \end{aligned} \tag{2.16}$$

where $T(z)$ is the complementary sensitivity function of the closed-loop system and $W(z)$ is a user-specified weighting function.

The control objective penalises the difference between the closed-loop transfer function and the reference model scaled by an appropriate weighting function. This allows us to emphasize or de-emphasize performance in certain frequency ranges.

The presence of the parameter vector in both the numerator and the denominator of this function makes the minimisation problem non-convex with respect to θ which significantly increases its difficulty.

In order to make the problem tractable an equivalent, convex optimisation criterion must be identified. Consider now the following cost function:

$$J_{VR}^N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \left(u_L(t) - C(z;\ \theta)\ e_L(t) \right)^2 \tag{2.17}$$

where $u_L(t)$ and $e_L(t)$ are respectively the plant input and the tracking error multiplied by a suitable pre-filter. Its form will be discussed later. Let this filter be $L(z)$ and defined filtered versions of the signals $u(t)$ and $e(t)$ as

$$\begin{aligned} e_L(t) &= L(z)e(t) \\ u_L(t) &= L(z)u(t). \end{aligned} \tag{2.18}$$

Through simple transformations it is possible to rewrite the criterion as

$$
\begin{aligned}
J_{VR}^{N}(\theta) &= \frac{1}{N}\sum_{t=1}^{N}\left(u_L(t) - \beta^T(z)\theta e_L(t)\right)^2 \\
&= \frac{1}{N}\sum_{t=1}^{N}\left(u_L(t) - \varphi_L^T(t)\theta\right)^2, \quad \varphi_L = \beta^T(z)e_L(t).
\end{aligned}
\tag{2.19}
$$

Since this criterion is quadratic in θ the optimal parameter vector $\hat{\theta}$ is an explicit function of the data

$$
\hat{\theta}_N = \left[\sum_t \varphi_L(t)\varphi_L(t)^T\right]^{-1}\sum_t \varphi_L(t)u(t). \tag{2.20}
$$

We will now show that it is possible, through an appropriate choice of the pre-filter, to make the two cost functions (equations (2.16) and (2.17)) equivalent.

Let $J_{VR}(\theta)$ be the asymptotic counterpart of $J_{VR}^{N}(\theta)$. If $u(t)$ and $y(t)$ can be considered realisations of stationary stochastic processes then as the amount of data grows (*i.e.*, $N \to \infty$) the minimum $\hat{\theta}_N$ of $J_{VR}^{N}(\theta)$ converges to the minimum of $J_{VR}(\theta)$, $\hat{\theta}$.

$$
\lim_{N\to\infty} J_{VR}^{N}(\theta_N) = J_{VR}(\theta).
$$

As such, the asymptotic criterion is

$$
\begin{aligned}
J_{VR}(\theta) &= E\left[\left(u_L(t) - C(z;\ \theta)e_L(t)\right)^2\right] \\
&= E\left[\left(L(z)\left(u(t) - C(z;\ \theta)e(t)\right)\right)^2\right].
\end{aligned}
\tag{2.21}
$$

The dependency on $e(t)$ should be removed in order to make the criterion depend only on the measured data. To that end we introduce $r_v(t)$, the reference signal that would feed the closed loop system when the closed loop transfer function is $M_R(z)$

$$
y(t) = M_R(z)r_v(t) \tag{2.22}
$$

from which we can deduce

$$
\begin{aligned}
r_v(t) &= \frac{1}{M_R(z)} y(t) \\
&= \frac{1}{M_R(z)} P(z) u(t).
\end{aligned}
\tag{2.23}
$$

The signal $r_v(t)$ is not a physical signal, it was not used in the physical experiments to generate $y(t)$ rather, $y(t)$ was used to synthesise this signal. It is a tool used in the identification of the optimal controller. It is the *virtual reference* signal, the namesake of the method.

The tracking error of the closed loop system is

$$
\begin{aligned}
e(t) &= r_v(t) - y(t) \\
&= r_v(t) - M_r(z) r_v(t) \\
&= (1 - M_R(z)) r_v(t)
\end{aligned}
\tag{2.24}
$$

and, by substituting the expression of $r_v(t)$ provided by equation (2.23) into the tracking error (equation (2.24)) we obtain

$$
e(t) = \frac{1 - M_R(z)}{M_R(z)} P(z) u(t) \tag{2.25}
$$

By substituting this new expression for $e(t)$ into the cost function J_{VR} (equation 2.21) we can rewrite the cost function as

$$
\begin{aligned}
J_{VR}(\theta) &= E\left[\left(u_L - C(\theta)\frac{1 - M_R}{M_R} P u_L\right)^2\right] \\
&= E\left[\left(L\left(1 - C(\theta)\frac{1 - M_R}{M_R} P\right) u\right)^2\right].
\end{aligned}
\tag{2.26}
$$

Under our current hypothesis, the reference model $M_R(z)$ is equal to the closed loop transfer function of the system

$$
M_R(z) = \frac{P(z) C_0(z)}{1 + P(z) C_0(z)} \tag{2.27}
$$

where $C_0(z)$ is the *ideal* controller *i.e.*, the controller that perfectly solves the model matching problem. Note that in general $C_0(z)$ might not belong to the family of parametrised controllers $\{C(z;\ \theta)\}$, be a proper rational transfer function or stabilise the closed loop system.

Additionally, observe that

$$\begin{aligned} 1 - M_R(z) &= \frac{1 + P(z)C_0 - P(z)C_0(z)}{1 + P(z)C_0(z)} \\ &= \frac{1}{1 + P(z)C_0(z)} \end{aligned} \tag{2.28}$$

from which

$$\begin{aligned} 1 - C(z;\ \theta)\frac{1 - M_R(z)}{M_R(z)}P(z) &= \frac{1}{M_R(z)}\left[M_R(z) - P(z)C(z;\ \theta)\left(1 - M_R(z)\right)\right] \\ &= \frac{1}{M_R(z)}\left(\frac{P(z)C_0(z)}{1 + P(z)C_0(z)} - \frac{P(z)C(z;\ \theta)}{1 + P(z)C_0(z)}\right) \\ &= \frac{1}{M_R(z)}\left(P(z)\frac{C_0(z) - C(\theta)}{1 + P(z)C_0(z)}\right). \end{aligned} \tag{2.29}$$

If we substitute the previous result into the expression of the control objective derived in equation (2.26) we can rewrite it in a more convenient form

$$J_{VR}(z;\ \theta) = E\left[\left(P(z)\frac{C_0(z) - C(z;\ \theta)}{1 + P(z)C_0(z)}\frac{L(z)}{M_R(z)}u\right)^2\right]. \tag{2.30}$$

An alternative frequency domain representation of this criterion is

$$\begin{aligned} J_{VR}(\theta) &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|P(e^{\mathrm{j}\omega})\frac{C_0(e^{\mathrm{j}\omega}) - C(e^{\mathrm{j}\omega};\ \theta)}{1 + P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}\frac{L(e^{\mathrm{j}\omega})}{M_R(e^{\mathrm{j}\omega})}\right|^2 \Phi_u(\omega)\ d\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\left(\frac{PC_0(e^{\mathrm{j}\omega})}{1 + P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})} - \frac{P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}{1 + P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}\right)\frac{L(e^{\mathrm{j}\omega})}{M_R(e^{\mathrm{j}\omega})}\right|^2 \Phi_u(\omega)\ d\omega \end{aligned} \tag{2.31}$$

where $\Phi_u(\omega)$ is the power spectral density of the plant input $u(t)$.

Observe that if $C_0(z) \in \{C(z;\ \theta)\}$ and $J_{VR}(\theta)$ has a unique minimum then minimising $J_{VR}(\theta)$ gives $C_0(z)$ for any choice of the filter $L(z)$. Generally however this is not the case and $C_0(z) \notin \{C(z;\theta)\}$.

The expression of the initial control objective $J_{MR}(z)$ as given in equation (2.16) can be re-written in the following form:

$$J_{MR}(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|M_R(e^{\mathrm{j}\omega}) - \frac{P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}{1+P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}\right|^2 \left|W(e^{\mathrm{j}\omega})\right|^2 d\omega$$
$$= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\frac{P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}{1+P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})} - \frac{P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}{1+P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}\right|^2 \left|W(e^{\mathrm{j}\omega})\right|^2 d\omega. \tag{2.32}$$

There is a striking similarity between J_{MR} and the new form of J_{VR}. If the filter were such that

$$\left|L(e^{\mathrm{j}\omega})\right|^2 = \left|\frac{M_R(e^{\mathrm{j}\omega})W(e^{\mathrm{j}\omega})}{1+P(e^{\mathrm{j}\omega})C(e^{\mathrm{j}\omega};\ \theta)}\right|^2 \frac{1}{\Phi_u(\omega)}, \quad \forall\omega\in[-\pi,\ \pi] \tag{2.33}$$

then we would have $J_{VR}(\theta) = J_{MR}(\theta)$ and, as a consequence, minimising J_{VR} would be equivalent to minimising J_{MR}. Unfortunately, this choice of $L(z)$ is not practical since $P(z)$ is not known. VRFT solves this problem by using the following pre-filter on the measured data

$$\left|L(e^{\mathrm{j}\omega})\right|^2 = \left|\left(1-M_R(e^{\mathrm{j}\omega})\right)M_R(e^{\mathrm{j}\omega})W(e^{\mathrm{j}\omega})\right|^2 \frac{1}{\Phi_u(\omega)}, \quad \forall\omega\in[-\pi,\ \pi] \tag{2.34}$$

With the exception of the power spectral density of the input signal Φ_u, all the quantities on the right hand side of this filter are known. Φ_u can be considered known only when the input signal has been selected by the control system designer. In the general case it must be estimated.

Whilst on first approach the two different expressions of $L(e^{\mathrm{j}\omega})$ in equations (2.33) and (2.34) may have little in common, their parentage becomes obvious if, exploiting (2.28), we re-write the latter as

$$\left|L(e^{\mathrm{j}\omega})\right|^2 = \left|\frac{M_R(e^{\mathrm{j}\omega})W(e^{\mathrm{j}\omega})}{1+P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}\right|^2 \frac{1}{\Phi_u(\omega)}, \quad \forall\omega\in[-\pi,\ \pi] \tag{2.35}$$

which makes it clear that this choice of $L(e^{\mathrm{j}\omega})$ is equivalent to substituting $|1+PC(\theta)|^2$ for $|1+PC_0|^2$. This is a sensible choice if, as required, $C(\theta)$ is a good approximation of C_0.

In conclusion, by a judicious choice of the pre-filter $L(z)$ it is possible to render the optimisation problem convex and purely quadratic in θ. This, in turn, makes the optimisation problem trivial to solve.

2.3.2 The Method Step By Step

In summary, the VRFT procedure requires the following steps:

1. Compute the virtual reference signal $r_v(t)$ such that

$$y(t) = M_R(z) r_v(t)$$

and the corresponding tracking error

$$e(t) = r_v(t) - y(t).$$

2. Compute the pre-filter

$$|L|^2 = |(1 - M_R)\, M_R W|^2 \, \frac{1}{\Phi_u}$$

and apply it to $u(t)$ and $y(t)$ in order to obtain the filtered signals $u_L(t)$ and $y_L(t)$.

3. Compute the optimal parameter vector $\hat{\theta}^N$ that minimises the cost function J_{VR}^N.

$$J_{VR}^N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \left[u_L(t) - C(z;\ \theta)\ e_L(t) \right]^2$$

2.3.3 The Problem of Noisy Data

The method as presented so far considers noiseless signals. In practice it is rare that a signal can be considered such. Consider the case where the plant output $y(t)$ is affected by additive noise $d(t)$. The measured signal $\tilde{y}(t)$ is thus

$$\tilde{y}(t) = P(z)u(t) + d(t) \tag{2.36}$$

where $P(z)$ is the (unknown) transfer function of the plant. It is assumed that the input signal $u(t)$ and the disturbance $d(t)$ are uncorrelated. This is

generally the case in open loop configurations. If we were to apply the VRFT method to this noisy data the performance would be significantly affected by noise. This is clearly evidenced by the frequency domain of the asymptotic criterion $J_{VR}(\theta)$. Considering the effect of the additive noise it becomes

$$J_{VR}(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|P(e^{\mathrm{j}\omega})\frac{C_0(e^{\mathrm{j}\omega}) - C(e^{\mathrm{j}\omega};\ \theta)}{1 + P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}\frac{L(e^{\mathrm{j}\omega})}{M_R(e^{\mathrm{j}\omega})}\right|^2 \Phi_u(\omega)\ d\omega + \underbrace{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\frac{C(e^{\mathrm{j}\omega};\ \theta)}{P(e^{\mathrm{j}\omega})C_0(e^{\mathrm{j}\omega})}\right|^2 \Phi_d(\omega)d\omega}_{Bias\ due\ to\ disturbance} \tag{2.37}$$

where $\Phi_d(\omega)$ is the power spectral density of the disturbance. The disturbance creates a bias in the criterion that needs to be overcome.

This is achieved with an instrumental variable method. We introduce the quantity

$$\tilde{\varphi}_L(t) = \beta(z)L(z)\left(M(z)^{-1} - 1\right)\tilde{y}(t) \tag{2.38}$$

as a replacement for $\varphi_L(t)$ introduced in equation (2.19). We also introduce the instrumental variable $\zeta(t)$ and compute the optimal parameter vector as

$$\hat{\theta}_N^{IV} = \left[\sum_{t=1}^{N}\zeta(t)\tilde{\varphi}_L(t)^T\right]^{-1}\left[\sum_{t=1}^{N}\zeta(t)u_L(t)\right] \tag{2.39}$$

Two different choices for the instrumental variable have been proposed in the literature.

Instrumental Variable From Repeated Experiments

Instead of performing a single experiment on the plant, two experiments must be carried out with the same input signal. This yields three vectors of data points

$$\begin{aligned} u &= \{u_0,\ u_1,\ u_2,\ \ldots\ , u_n\} \\ \tilde{y} &= \{\tilde{y}_1,\ \tilde{y}_2,\ \tilde{y}_3,\ \ldots\ , \tilde{y}_n\} \\ \tilde{y}' &= \{\tilde{y}'_1,\ \tilde{y}'_2,\ \tilde{y}'_3,\ \ldots\ , \tilde{y}'_n\} \end{aligned}$$

where $\tilde{y}$ and $\tilde{y}'$ will be different since they depend on different realisation of the disturbance. If the experiments are sufficiently spaced apart in time it is reasonable to assume that the noise measured in both experiments is uncorrelated. The instrumental variable is computed as

$$\zeta(t) = \beta(z)L(z)\left(M(z)^{-1} - 1\right)\tilde{y}'(t). \tag{2.40}$$

This method has the benefit of guaranteeing that asymptotically $\hat{\theta}^{IV} = \hat{\theta}$ at the cost of more experiments.

Instrumental Variable From Plant Identification

If performing multiple experiments with the same input signal is not possible or acquiring new measurements is not cost effective, an alternative, identification based, approach has been developed.

Identify a high-order model $\hat{P}(z)$ of the plant from the collected input output data and use it to generated a simulated output

$$\hat{y}(t) = \hat{P}(z)u(t) \tag{2.41}$$

and use this dataset to construct the instrumental variable as in the previous scenario

$$\zeta(t) = \beta(z)L(z)\left(M(z)^{-1} - 1\right)\hat{\tilde{y}}(t). \tag{2.42}$$

This method does not provide the same optimality guarantees as the previous one since its accuracy depends on the quality of the identified model, however it does not required any extra data.

The identification of the plant required for the computation of the instrumental variable does not make the method any less data-driven since the model is not used directly to design the controller.

2.3.4 Extension to Cascade Control

The VRFT algorithm can be extended to cascade control systems with a little extra work. A naive approach is to perform the tuning of the inner

and outer loop sequentially. First the inner loop is tuned using the standard approach. The inner loop is then closed and VRFT is performed on the outer loop.

However, the VRFT method can in fact be extended to tune both the inner and the outer controller at the same time using the same dataset as described in [4].

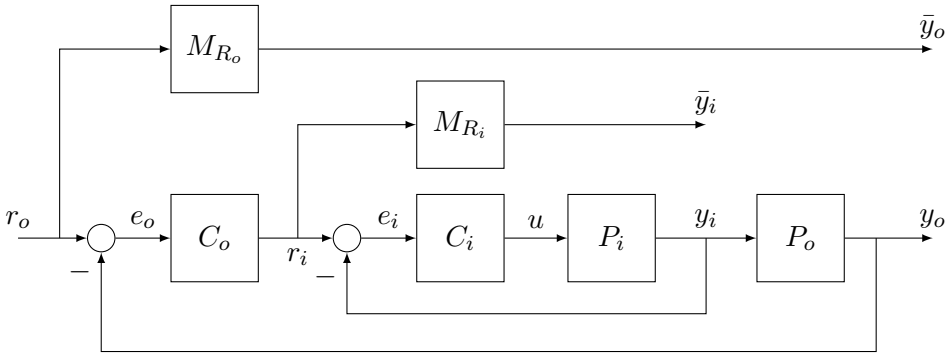


Figure 2.5: Control and reference model

We assume that a set of open loop measurements have been taken during an experiment on the plant. The plant input signal $u(t)$ must be measured as well as the inner and outer plant outputs $y_i(t)$ and $y_o(t)$. We should have three vectors of data-points available

$$
\begin{aligned}
u &= \{u_0,\ u_1,\ u_2,\ \dots\ , u_n\} \\
y_i &= \{y_{i_1},\ y_{i_2},\ y_{i_3},\ \dots\ , y_{i_n}\} \\
y_o &= \{y_{o_1},\ y_{o_2},\ y_{o_3},\ \dots\ , y_{o_n}\}
\end{aligned}
$$

We must also choose two reference models to describe the desired performance of the inner and outer loops. Let $M_{R_i}(z)$ be the reference model of the inner loop and $M_{R_o}(z)$ be the reference model of the outer loop.

We must also choose two families of controllers $\{C_i(z;\ \theta_i)\}$ and $\{C_o(z;\ \theta_o)\}$ for the inner and outer loops.

The requirements on the reference models and the controller families are unchanged with regards to the requirements of the standard VRFT method.

The inner controller can be tuned immediately using standard VRFT since the input-output data required for this is available. The only requirement

imposed on the inner loop is that it be as close as possible to the reference model.

The outer loop is slightly more complex. In this case, to apply the VRFT algorithm we must compute a new plant model that encompasses the entirety of the inner loop and the outer plant as well as a signal analogous to the inner plant input $u(t)$.

Calculating a new, extended, plant model is rather simple

$$G(z) = \frac{C_i(z)P_i(z)}{1 + C_i(z)P_i(z)}P_o(z)$$

Considering this enlarged plant, the input is simply $r_i(t)$, the reference signal of the inner loop. In contrast with the inner loop however this signal takes on a physical meaning, it is no longer a *virtual* reference signal. Additionally, since the original dataset was acquired in open-loop conditions it is not part of our original dataset. Fortunately it can be computed quite simply as

$$r_i(t) = e_i(t) + y_i(t)$$

where the output $y_i(t)$ is part of our initial dataset and is known. The inner tracking error $e_i(t)$ is entirely fictitious but since it derives directly from the design of the inner plant it is rather simple to compute. If the inner controller $C_i(z)$ is invertible then

$$e_i(t) = C(z;\ \theta_i)^{-1}u(t).$$

This calculation only yields useful results if the inner controller is minimum-phase. If that is not the case it will have zeros located outside of the limit cycle. When the controller is inverted these will become unstable poles. In turn they will cause the tracking error to diverge, making the signal quite useless for our purposes.

Unfortunately, VRFT provides no guarantees that the controllers it produces will be minimum phase but in practice non-minimum phase controllers only appear when the reference model is unachievable. Reducing the requirements on the reference model is usually sufficient to obtain a suitable inner controller.

Thus, the reference input for the outer loop is

$$r_i(t) = C(z;\ \theta_i)^{-1}u(t) + y_i(t).$$

Once the extended plant input $r_i(t)$ is known the outer controller can be easily found with the classic VRFT synthesis using as input-output data the set of data-points $\{r_i(t),\ y_o(t)\}$.

2.3.5 The Cascade VRFT Method Step By Step

To summarise, in order to tune both the inner and outer controllers of a cascade control system using a single set of data the procedure is:

1. Compute the inner virtual reference $r_{iv}(t)$ such that

   $$y_i(t) = M_i(z)r_{iv}(t)$$

   and the corresponding tracking error

   $$e_{iv}(t) = r_{iv}(t) - y_i(t).$$

   Next, compute the outer virtual reference $r_{ov}(t)$ such that

   $$y_o(t) = M_o(z)r_{ov}(t)$$

   and the corresponding tracking error

   $$e_{ov}(t) = r_{ov}(t) - y_o(t).$$

2. compute the pre-filter for the inner loop

   $$|L_i|^2 = |(1 - M_i)\, M_i W_i|^2 \, \frac{1}{\Phi_u}$$

   and apply it to the signals $u(t)$ and $e_i(t)$ to obtain the filtered signals $u_L(t)$ and $e_{iL}(t)$.

3. Compute the optimal parameter vector for the inner loop $\hat{\theta}_i^{\,N}$ that minimises the cost function

   $$J_{VR}^N(\theta_i) = \frac{1}{N}\sum_{t=1}^{N}\left[u_L(t) - C_i(z;\ \theta_i)\ e_{iL}(t)\right)^2.$$

4. If $C_i(z)$ is non-minimum phase change the reference model or the sample time. Otherwise, if $C_i(z)$ is minimum phase calculate the reference input $r_i(t)$ for the inner loop as

$$r_i(t) = C_i(z;\theta_i)u(t) + y_i(t)$$

5. Compute the pre-filter for the outer loop

$$|L_o|^2 = |(1 - M_o)\, M_o W_o|^2 \, \frac{1}{\Phi_{r_i}}$$

and apply it to the signals $r_i(t)$ and $e_o(t)$ to obtain the filtered signals $r_{iL}(t)$ and $e_{oL}(t)$.

6. Finally, estimate the optimal parameter vector for the outer loop $\hat{\theta_o}^N$ that minimises the cost function

$$J_{VR}^N(\theta_o) = \frac{1}{N}\sum_{t=1}^{N} \left(r_{iL}(t) - C_o(z;\ \theta_i)\ e_{oL}(t)\right)^2 .$$

Chapter 3

Simulation Results

One of the main selling points of data-driven controller tuning methods is that they do not require a detailed mathematical description of the plant. However, if is a model is available, it can be used as a guide to evaluate the performance of the tuned controllers without having to run extensive tests on the test-bed.

In Section 3.1 a model of the quadcopter will be introduced. In Section 3.2 how this model was used to refine the structure of the reference models will be shown. In Section 3.3 the controller families for the inner and outer loops used for VRFT will be shown and, finally, in Section 3.4 some controllers generated with VRFT and the performance indicated by the simulations will be shown.

3.1 Simulated Model

3.1.1 Pitch Dynamics

The quadcopter used in this work has been extensively characterised and modelled in [5]. For the purpose of simulation the pitch dynamics of the system can be reduced to two blocks: a simple model of the pitch dynamics $G_q(s)$ and an integrator used to compute the current pitch angle. This structure is shown in Figure 3.1.

The model of the pitch dynamics obtained in the cited work

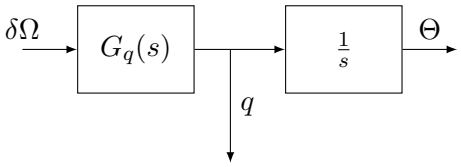


Figure 3.1: Simplified model of the quadcopter

$$G_q(s) = \frac{0.423}{s + 1.33} \tag{3.1}$$

is a linearisation, in the origin, of the transfer function from $\delta\Omega$, the requested change in the speed of the propellers, to q, the pitch rate.

This transfer function does not attempt to account for any non linear behaviours of the system such as saturations. It does not consider the dynamics of the motors or any aerodynamic drag effects. In practice this is not an issue as long the dynamics of the simulated system are slow enough not to interfere with the motor dynamics.

The calculation of the pitch angle as the integral of the pitch rate represents a divergence from the real system. In the real system the pitch angle is not directly available but is estimated as part of a sophisticated sensor fusion algorithm by the IMU the dynamics of which are not accounted for in this approximation. The signal Θ as computed in Figure 3.1 is the real pitch angle.

This approximation is used for the simulations as it makes the two nested control loops explicit.

Using a set of previously acquired input-output data this simplified model was simulated by passing to the input ($\delta\Omega$) a **P**seudo **R**andom **B**inary **S**equence (PRBS). The results of this simulation are show in Figure 3.2. The measured and simulated pitch rates (q) are in agreement even if the simulated system tends to be slower than the actual quadcopter. The pitch angle was not included in the dataset but the simulated pitch angle was included in the results for completeness.

3.1.2 Mixer Matrix

As discussed in Section 1.3.1, the pitch rate controller is a PID whose output is the desired pitch moment of the quadcopter. The input of the above model

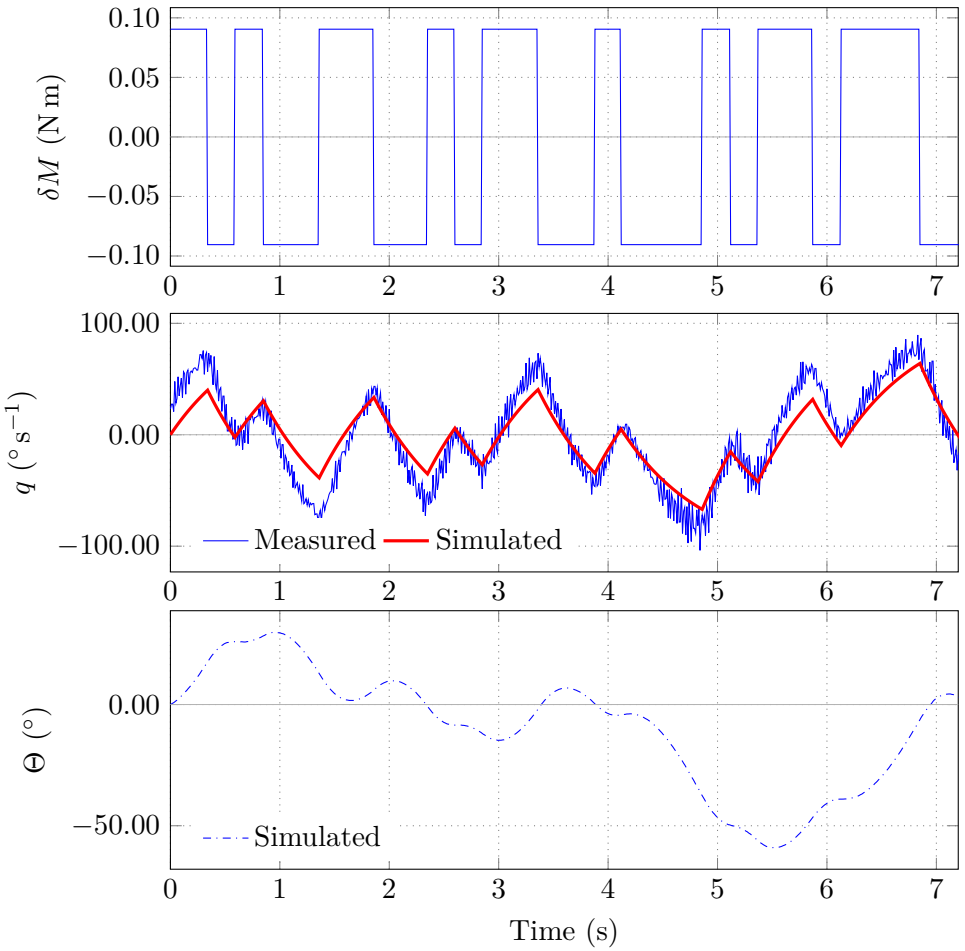


Figure 3.2: Comparison of the measured and simulated open loop responses of the quadcopter

however is the difference in speed between the front and rear propellers. The translation from one to the other is devolved to the mixer matrix.

Recall that the mixer matrix relates the vertical thrust T and the moments L, M and N around, respectively, the pitch, roll and yaw axes to the speeds of all four rotors Ω_1, Ω_2, Ω_3, Ω_4 in the following manner

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \chi^{-1} \begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix}. \tag{3.2}$$

Considering just the pitch control loop, the only non-null angular momentum term is M. We also know that the real input to the system is the difference in speed between the front and the rear rotor. This allows us to reduce the mixer to a single scalar:

$$\chi = 66.67 \tag{3.3}$$

3.1.3 Complete System

With this last piece of missing information the models obtained so far can be inserted into the pitch control scheme previously presented in Figure 1.3. The mixer matrix and the plant model combined describe the quadcopter dynamics from the controller output to the pitch rate and the pitch angle is produced by the integrator. The resulting control scheme is shown in Figure 3.3.

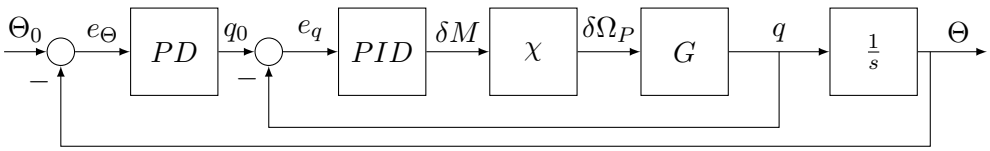


Figure 3.3: Control Scheme used to simulate the pitch dynamics

The task now is to choose appropriate reference models.

3.2 Structure of the Reference Models

The choice of the reference model is critical to VRFT, the model should be fast enough to obtain the best possible bandwidth without being unachievable, but there is no need to rely on complex models to obtain satisfying levels of performance. A second order model has been a good starting point. This is the simplest class of models that allows us to tune both its static gain and its bandwidth and it can also be easily extended to account for known properties of the system such as delays.

If we were to choose the reference models without any knowledge of the plant we would be reduced to semi-random guessing. The knowledge of the plant that we have can be used to guide our decisions and hopefully choose a reference model that better represents the actual closed loop transfer function of the plant.

VRFT also provides for a user-defined weighting function to emphasize performance in certain bands of interest. For simplicity all bands were considered of equal importance and the inner and outer weighting functions were defined as, respectively, $W_i(z) = 1$ and $W_o(z) = 1$.

3.2.1 Inner Reference Model

The reference model for the inner loop derives from a second order model. It was observed algebraically that, considering a PID controller and a first order plant model G such that

$$PID(s) = \frac{K_i}{K_p}\frac{1+s}{s}, \qquad G(s) = \frac{a}{b+s} \tag{3.4}$$

the resulting closed loop transfer function would be of the form

$$F(s) = \mu\frac{1+s}{s^2+cs+d} \tag{3.5}$$

and, to compensate for this additional zero, it was decided to add a zero to the reference model. This zero was chosen to be in the neighbourhood of the zero naturally emerging from the PID controller for a reasonable range of PIDs. The resulting form of the inner reference model is thus

$$M_i(s) = \frac{\omega_{n_i}^2}{s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2}\frac{s + z_0}{z_0} \tag{3.6}$$

where ω_{n_i} is the bandwidth of the reference model, ζ_i is the damping ratio and z_0 is the position of the zero.

Initially the specific bandwidth and damping ratio of the reference model were set to achieve similar performance to what had been observed with the pre-existing H_∞ controller. During the experimental phase the reference model would be further tuned to extract better performance from the system.

3.2.2 Outer Reference Model

The structure of the outer reference model was assigned in a similar fashion. It was observed that the closed loop transfer function of the system could be approximated with a second order system without any additional poles or zeroes.

$$M_o(s) = \frac{\omega_{n_o}^2}{s^2 + 2\zeta_o\omega_{n_o}s + \omega_{n_o}^2} \tag{3.7}$$

where ω_{n_i} is the bandwidth of the reference model and ζ_i is the damping ratio.

3.3 Controller Families

The final choice to be made is the choice of the family of controllers to tune. As discussed in Section 1.3.1 the pitch rate controller is a PID whilst the pitch angle controller is a PD.

$$\begin{aligned} PD(s) &= K_{p_o} + \frac{K_{d_o}s}{T_f s + 1} \\ PID(s) &= K_{p_i} + \frac{K_{i_i}}{s} + \frac{K_{d_i}s}{T_f s + 1} \end{aligned} \tag{3.8}$$

VRFT requires that the controller family expressed as a vector of linear transfer functions $\beta(z)$ such that the parametrised controller is

$$C(z;\ \theta) = \beta^T(z)\theta \tag{3.9}$$

where θ is the parameter vector. This requirement that the controller family be linear in the parameters means that T_f, the filter time constant of the derivative terms cannot be tuned by the VRFT but must be set beforehand. In the previous H_∞ controller the filter time constant was manually set to $T_f = 0.01$ and this value was left unchanged.

The controllers written as vectors of linear transfer functions are

$$\begin{aligned} PID(s) &= \begin{bmatrix} 1 & \frac{1}{s} & s \end{bmatrix} \theta_i \\ PD(s) &= \begin{bmatrix} 1 & s \end{bmatrix} \theta_o \end{aligned} \tag{3.10}$$

where θ_i and θ_o are, respectively, the parameter vectors for the inner and outer controllers.

3.4 Simulation Results

Before starting the testing campaign it was decided to validate the applicability of the VRFT approach to the problem of pitch control with a set of simulations. Initially, reference models for the inner and outer loops were developed that mimicked the behaviour of the pre-existing H_∞ controllers.

One such pair on inner and outer reference models is

$$\begin{aligned} M_{Ri}(s) &= \frac{64s + 320}{s^2 + 72s + 320} \\ M_{Ro}(s) &= \frac{16}{s^2 + 7.2s + 16} \end{aligned} \tag{3.11}$$

where the inner reference model (M_{Ri}) has a bandwidth of 8 $\mathrm{rad\,s^{-1}}$ and a damping ratio of 0.9 and the outer reference model (M_{Ro}) has a bandwidth of 4 $\mathrm{rad\,s^{-1}}$ and a damping ratio of 0.9.

The VRFT algorithm, applied with the above reference models produced controllers with the parameters shown in Table 3.1.

A closed loop simulation of the controllers was performed to ensure that the closed loop performance of the system is sufficiently close to the the

	Inner Controller (PID)				Outer Controller (PD)		
	K_{p_i}	K_{i_i}	K_{d_i}	T_f	K_{p_o}	K_{d_o}	T_f
VRFT Tuned	0.2978	0.514	0	0.01	1.6057	0.0	0.01
H_∞ Tuned	0.298	0.304	0.0499	0.01	2.0	0.00522	0.01

Table 3.1: Controller parameters produced by VRFT with the reference models (3.11).

reference models. This results of the simulation are shown in Figure 3.4. The simulated system appears to be slightly slower than the reference model but overall the two are in accordance.

For completeness the Bode plots of both the inner and outer loops and their associated reference models are reported, respectively, in Figures 3.6 and 3.7.

As shown in Figure 3.5 the synthesised controller provides similar behaviour to that of the previously obtained H_∞ controller even though the parameters of the controllers differ significantly.

Having shown, using the simulated model, that performance similar to that of the pre-existing H_∞ controller could be achieved with a VRFT tuned controller we looked into improving the performance of the controllers. New inner and outer reference models with bandwidths of, respectively, 15 and 10 $\mathrm{rad\,s^{-1}}$ were generated

$$\begin{aligned} M_{Ri}(s) &= \frac{225s + 1125}{s^2 + 135s + 1125} \\ M_{Ro}(s) &= \frac{100}{s^2 + 18s + 100}. \end{aligned} \tag{3.12}$$

Using these reference models for the VRFT yielded the parameters in table 3.2. Whilst the controller parameters seem extremely high the simulation shown in Figure 3.8 bears out their effectiveness on paper. However, this assumes that the system is accurately described by the model. Since the model of the vehicle does not impose any limits to the closed loop bandwidth this is not the case when the bandwidth of the reference model is high. A test with similar controller parameters run on the actual quadrotor had to be prematurely terminated due to uncontrolled oscillations of the system.

To achieve the requested speed an unrealistic control effort is required. The real world performance is limited by the saturations on the controller

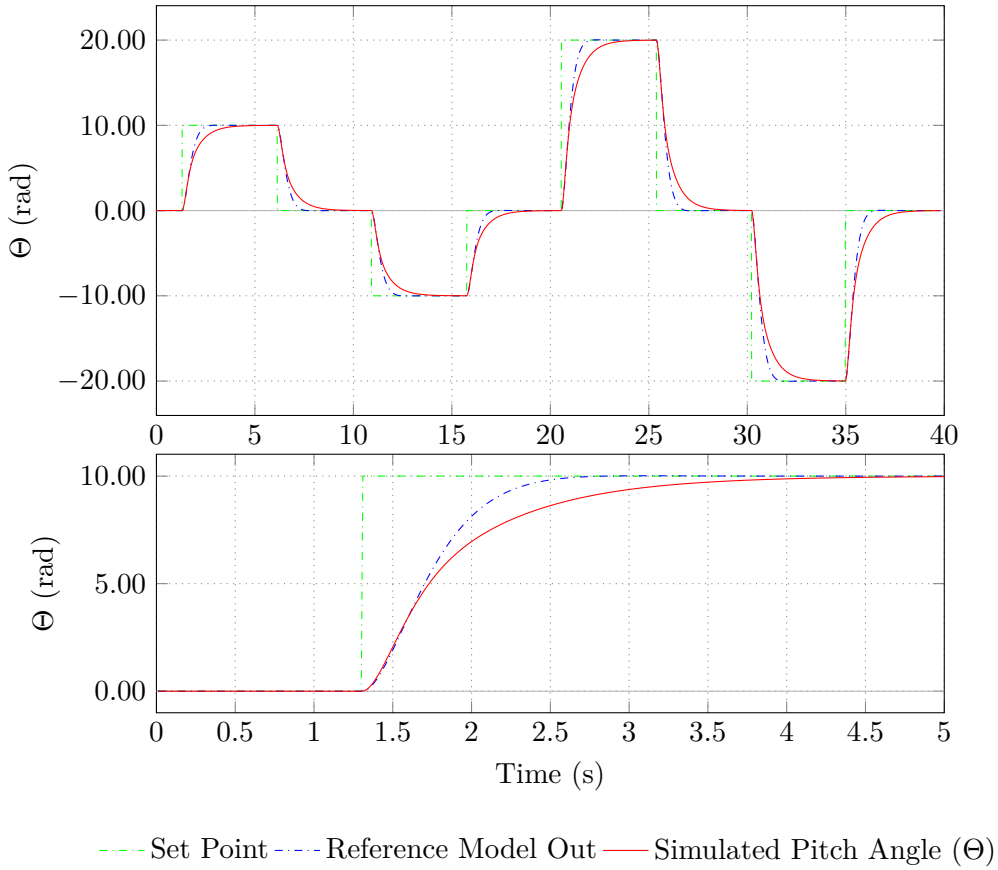


Figure 3.4: Comparison of the responses of the outer reference model M_{Ro} and the closed loop transfer function of the system with the controllers of Table 3.1. The upper plot shows the complete simulation whilst the lower plot shows a zoomed-in view of the first step.

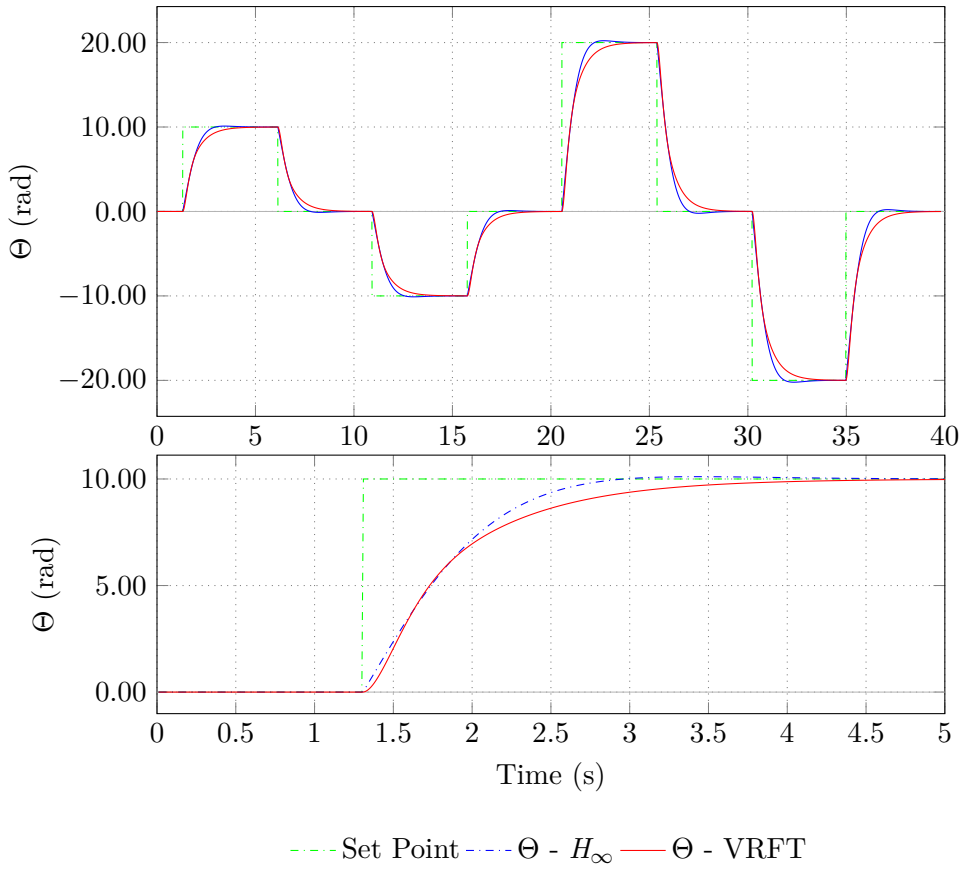


Figure 3.5: Comparison of the responses of the closed loop transfer function of the system with the VRFT tuned controllers and the pre-existing H_∞ controllers with the parameters of Table of Table 3.1. The upper plot shows the complete simulation whilst the lower plot shows a zoomed-in view of the first step.

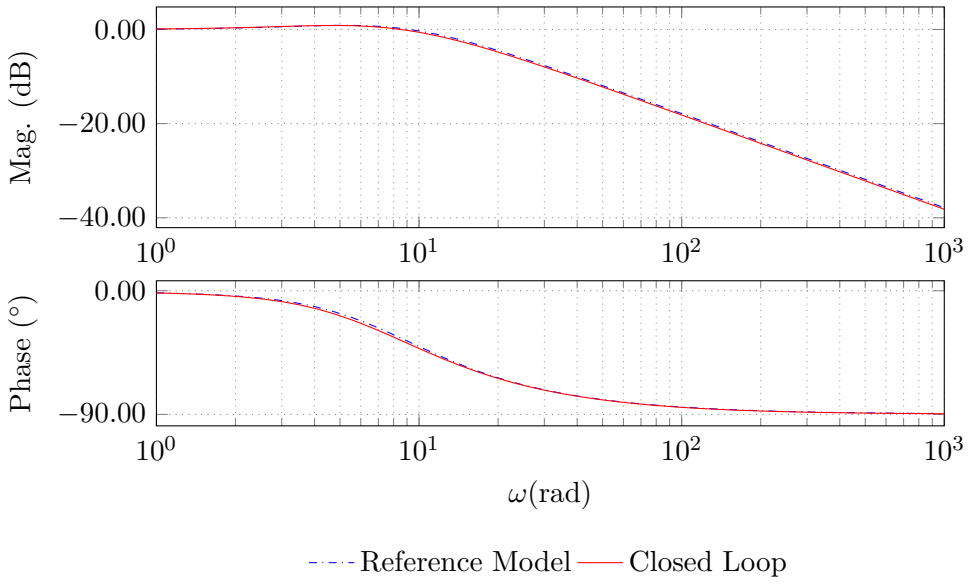


Figure 3.6: Comparison of the Bode plots of the inner reference model M_{Ri} and the closed loop transfer function for the pitch rate regulation loop with the controllers of Table 3.1.

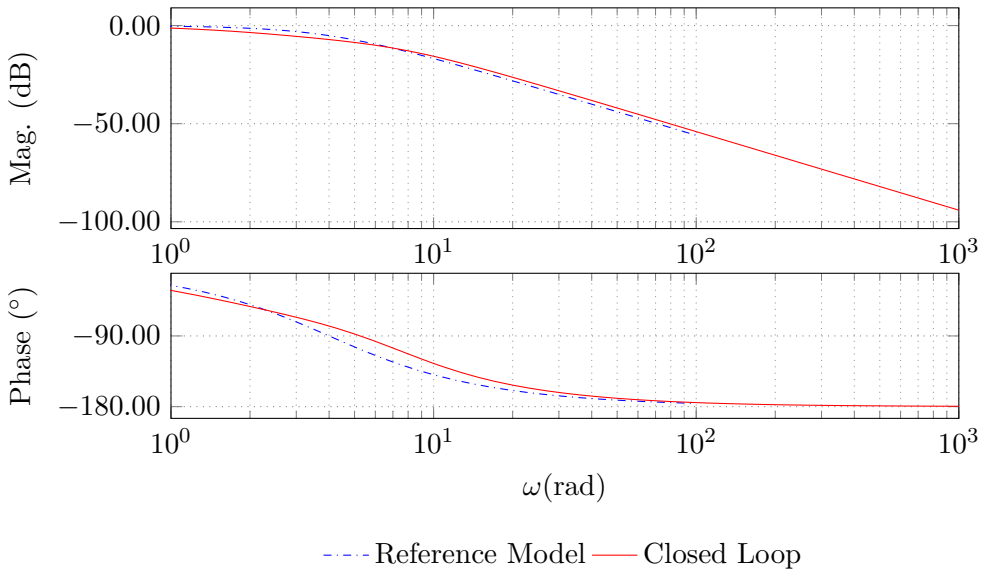


Figure 3.7: Comparison of the Bode plots of the outer reference model M_{Ro} and the closed loop transfer function for the pitch angle regulation loop with the controllers of Table 3.1.

outputs and, consequently, the maximal real-world performance was much lower than this.

Clearly, relying on only the simulated model is not sufficient to accurately tune the controllers. An comprehensive set of experiments must be performed on the system to identify the actual upper bound for performance.

	Inner Controller (PID)				Outer Controller (PD)		
	K_{p_i}	K_{i_i}	K_{d_i}	T_f	K_{p_o}	K_{d_o}	T_f
VRFT Tuned	0.7826	9.4900	0.00162	0.01	4.1507	0.0	0.01

Table 3.2: Controller parameters produced by VRFT with the reference models (3.12).

The bode plots of the inner and outer loops with their associated reference models are reported, respectively, in Figures 3.9 and 3.10. Note the the bode diagrams of the inner loop and its reference model show a significant divergence between the two at higher frequencies. This is an indication that the required performance is difficultly achievable.

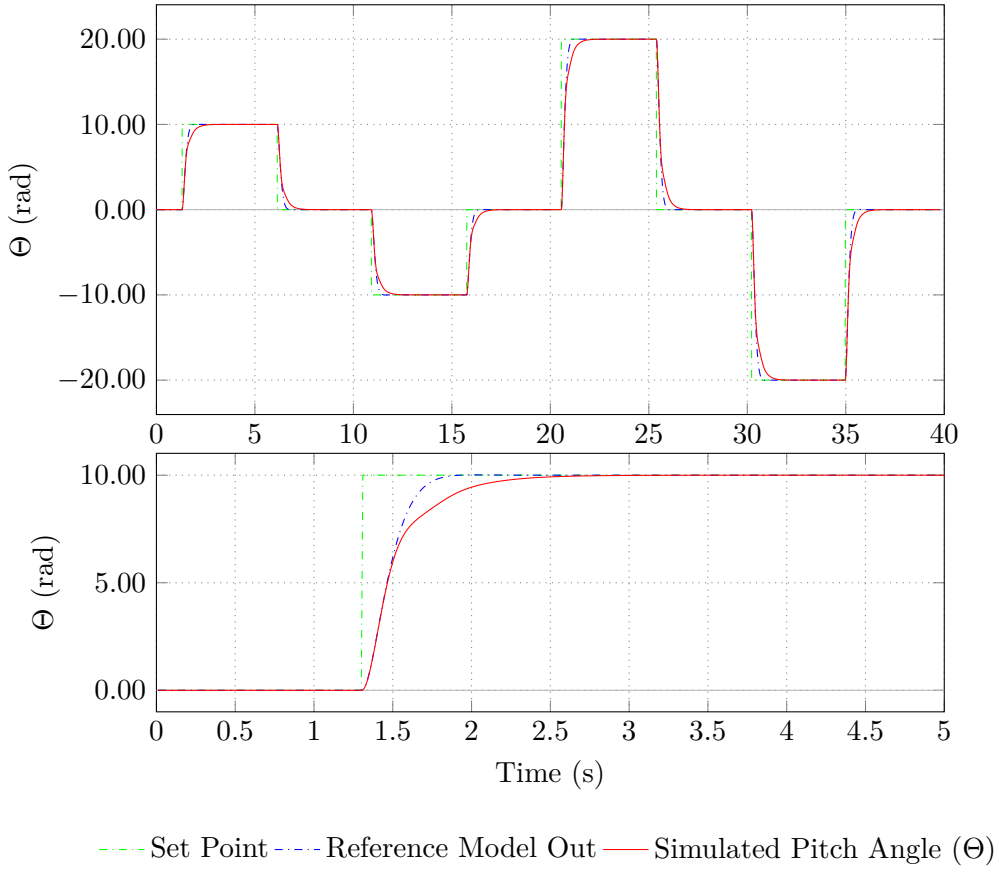


Figure 3.8: Comparison of the responses of the outer reference model M_{Ro} and the closed loop transfer function of the system with the controllers of Table 3.2.

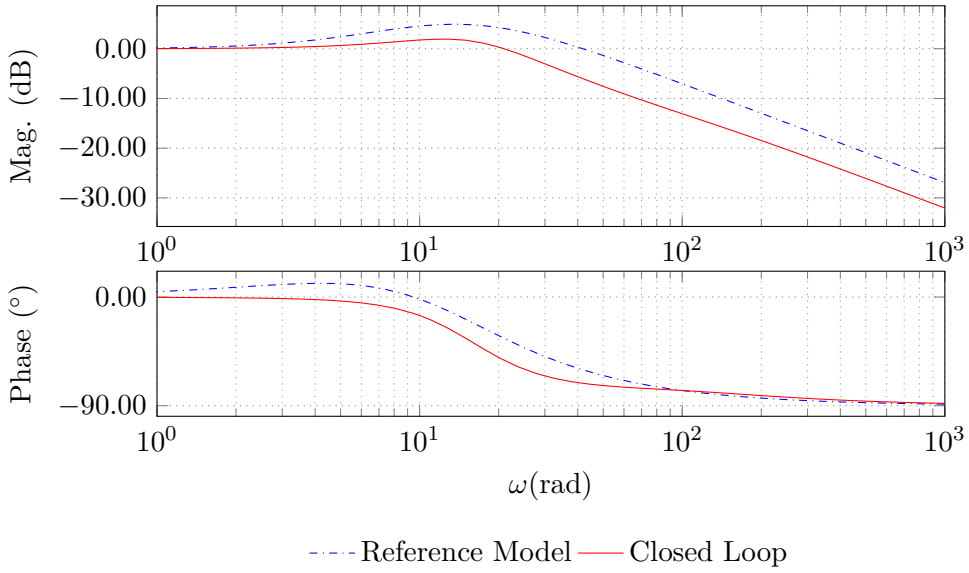


Figure 3.9: Comparison of the Bode plots of the inner reference model M_{Ri} and the closed loop transfer function for the pitch rate regulation loop with the controllers of Table 3.2.

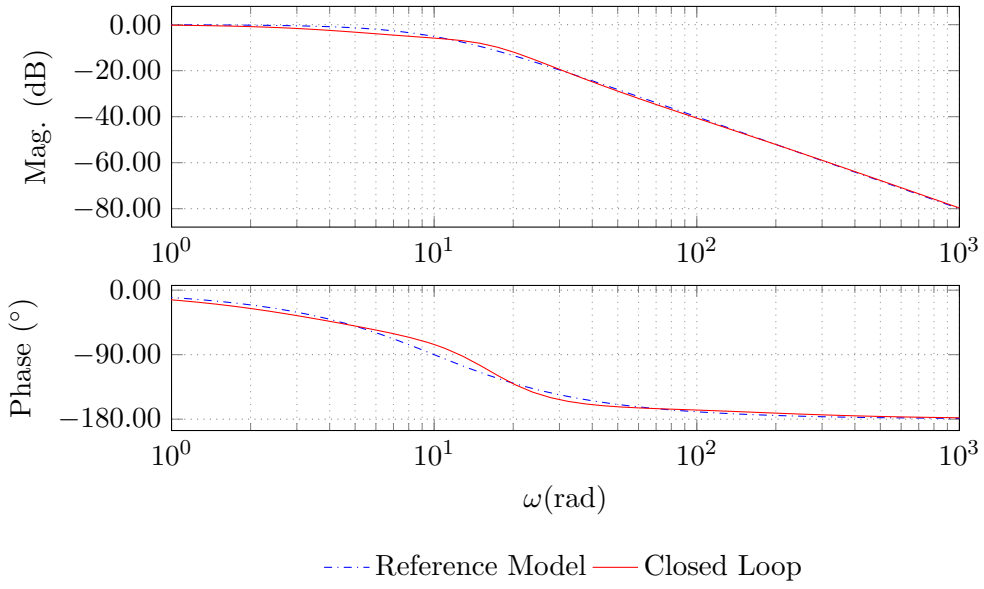


Figure 3.10: Comparison of the Bode plots of the outer reference model M_{Ro} and the closed loop transfer function for the pitch angle regulation loop with the controllers of Table 3.2.

Chapter 4

Experimental Results

In chapter 3 the performance of the simulated system was explored and it was showed that, since the model does not account for non-linearities and other complex effects, it is possible (at least theoretically) to achieve unrealistic levels of performance. Tests on the real hardware were conducted in order to find an actual bound to the achievable performance.

In Section 4.1 the actual components of the quadcopter will be detailed. In Section 4.2 the fashion in which the required input-output data was collected will be explained. VRFT has provisions for noise mitigation procedures. Their use will be detailed in Section 4.3. In Section 4.4 the reference models used for the VRFT will be shown and, finally, in Section 4.5 the results of the actual experiments on the system will shown and compared to a pre-existing manually tuned H_∞ controller.

4.1 Quadcopter Hardware & Firmware

The quadcopter used for this work was entirely developed at Politecnico di Milano based on the following requirements:

- X frame of medium size (450 to 550 mm distance between opposite rotors)
- Overall weight of less than 2 kg
- Payload of at least 0.5 kg
- Flight time of at least 10 minutes

The specifications of the components chosen in order to achieve these goals are documented in the following sections. A fixed-pitch actuation scheme was retained since the performance requirements of the system are relatively low. For a more detailed explanation behind these choices see [5]

4.1.1 Frame

The frame (Figure 4.1a) is the central building block of the quadcopter. The Talon V2.0 frame, built out of carbon fibre and aluminium was retained due to its low weight and high strength. The distance between opposite motors on this frame is 500 mm.

4.1.2 Motors, ESC & Propellers

The motors (Figure 4.1b) are brushless DC motors (BLDC) from the RCTimer High Performance series (HP2814). Since brushless motors are synchronous they cannot be driven by a simple DC current but require an inverter to provide a switching electric signal. This is the **E**lectronic **S**peed **C**ontroller (ESC). In this case, the RCTimer NFS ESC 30A (Figure 4.1c). This combination of motor and driver is able to push the propeller to close to 8000 rpm

The propellers mounted on the motors have two 12 inch blades with a 4.5° pitch angle (Figure 4.1d).

4.1.3 Battery

The batteries used on the quadrotor are Turnigy nano-tech 4000 mA h. These are LiPo 3 cell batteries with a nominal tension of 3.7 V per cell for a total tension of 11.1 V per battery. The batteries are capable of supplying a constant current of 100 A with peaks of up to 200 A for short periods of time.

4.1.4 Vibration Damping

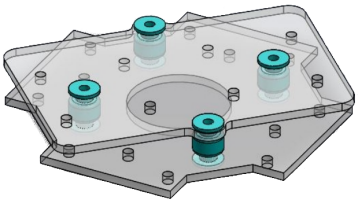
To reduce the mechanical vibrations transmitted through the frame from the motors a support plate onto which the electronics are mounted was realised

(a) Hobby King Talon V2.0 Frame

(b) RCTimer HP2814 Motor

(c) RCTimer NFS ESC 30A

(d) 12 inch, $4.5\ ^{\circ}$ pitch propellers



(e) Electronic boards support

(f) Vibration dampers

Figure 4.1: Quadrotor Components

(Figure 4.1e). This plate is then fixed to the frame with vibration damping rubber feet (Figure 4.1f).

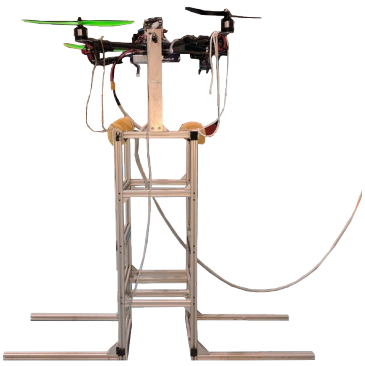
4.1.5 Flight Control Unit & Firmware

The flight control unit is the brains of the quadrotor. It was implemented on a **R**apid **R**obot **P**rototyping (R2P) boards [1]. This is an open source hardware and software framework that enables the rapid development of robotic applications. These boards implement an IMU, serial communication, control of the RC motors with PWM signals and the actual flight control. The modules use a publish/subscribe architecture to communicate.

The control portion is implemented in Simulink and compiled to C++ code. The firmware manages the communication between the sensors and the generated code in order to control the quadcopter.

4.1.6 Test Bed

All the tests on the quadrotor were performed on a test bed that constrains all translational degrees of freedom as well as the roll and yaw motions. Only the pitch rotation is left unconstrained. This ensures that the tests are repeatable and safe and that an erroneous choice of the controller parameters will not send the system crashing through the room.



(a) Test Bed, Front View

(b) Test Bed, Top view

The test bed is built out of *x-frame* aluminium rods and weighted with sacks of concrete. The upper part of the frame has a smooth rod resting on ball bearings at each extremity for frictionless rotation. The quadrotor is

then securely fastened to this rod.

In the current mounting scheme the rod passes as close as possible to the centre of mass of the system in order to interfere as little as possible with the dynamics of the quadrotor. Because of the physical configuration of the system there is small distance between the rod and the actual centre of mass. In turn, this causes the system to act like a very small pendulum and the test bed adds some damping when the system quadrotor achieves higher pitch angles. In practice this damping is negligible for small oscillations.

The test bed holds the quadcopter high enough that ground effect disturbances are avoided however, since the test takes place in a closed space some recirculation of rotor wakes occurs. This represents a discrepancy when compared to outdoor flight conditions where the rotor wakes develop free from obstacles. Even so, it has been shown in previous work that such a test bed is representative of actual attitude dynamics in flight.

4.1.7 Additional Flight Hardware

The quadcopter also holds a RaspberryPi 2 board used to interface the R2P modules with a ROS network but this functionality was unused for the tests. In addition a small ultrasonic sensor has been mounted on the drone to measure the distance from the ground when landing.

4.2 Tuning Experiment

The most important pre-requisite for data-driven algorithms such as VRFT is the collection of a sufficiently long persistently exciting input-output dataset.

Since the aim of this work is to tune the controllers on the pitch regulation loop all the tests were performed on the test bed described in Section 4.1.6. The test bed constrains all the degrees of freedom of the vehicle except the pitch rotation.

The control scheme for the pitch control loop is shown in Figure 4.3. See Section 1.3.1 for a detailed description of each component and the symbols used.

In the cascade VRFT setting the system should be divided into four parts: an *inner* and an *outer* parametrised controller and and *inner* and

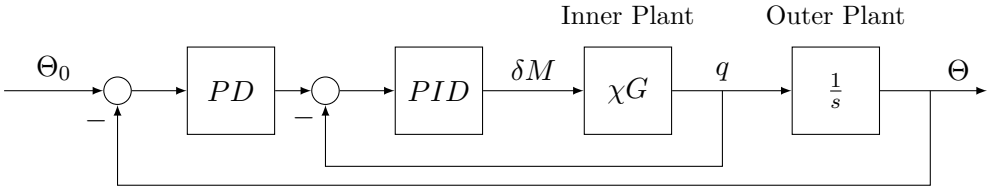


Figure 4.3: Control scheme for VRFT as applied to the pitch control loop where q, the pitch rate of the quadcopter, is analogous to the inner output signal y_i, Θ, the pitch angle, is analogous to the outer *output signal y_o and $\delta\Omega_P$, the change in speed of the propellers, is analogous to the plant input signal u.*

an*outer* unknown plant. The controllers are quite obviously the PD and PID blocks in Figure 4.3. It follows that the χG blocks representing the dynamics from the requested torque around the pitch axis to the pitch rate is the inner plant and the integrator $\frac{1}{s}$ computing the pitch angle from the pitch rate is the outer plant.

The experiments used to gather the input-output data should be performed in *open loop*, providing the torque δM as an input and reading the pitch rate q and the pitch angle Θ as the *inner* and *outer* outputs y_i and y_o. Unfortunately the firmware does not allow us to provide δM as an input directly or read the pitch angle Θ when operating in open loop conditions. Instead we were only able to specify as the input $\delta\Omega_P$ the difference in the speed of the front and read propellers and read he pitch rate q as the single output.

This is not an issue however since, as shown in Section 3.1.2, the mixer, considering only the pitch control loop, is a known scalar. Consequently, the *inner* plant input could be computed quite simply from the available data as

$$\delta M = \chi^{-1}\delta\Omega_P. \tag{4.1}$$

The pitch angle , Θ, the output of the *outer* plant, can also be calculated algebraically. Whilst in the general case the outer plant model is unknown, in our case the it is an integrator. The pitch angle Θ is simply the integral of the pitch rate q over time. Consequently we were able to calculate the pitch angle as

$$\Theta = \int_0^t q\,dt \tag{4.2}$$

With the quadcopter firmware operating in open loop, a **P**seudo **R**andom **B**inary **S**equence (PRBS) was generated for the input $\delta\Omega_P$ and fed into the system with a time step of 0.01s. This sequence was generated in such way as to reliably excite the dominant dynamics of the system. This input signal and the computed torque around the pitch axis δM are show in Figure 4.4.

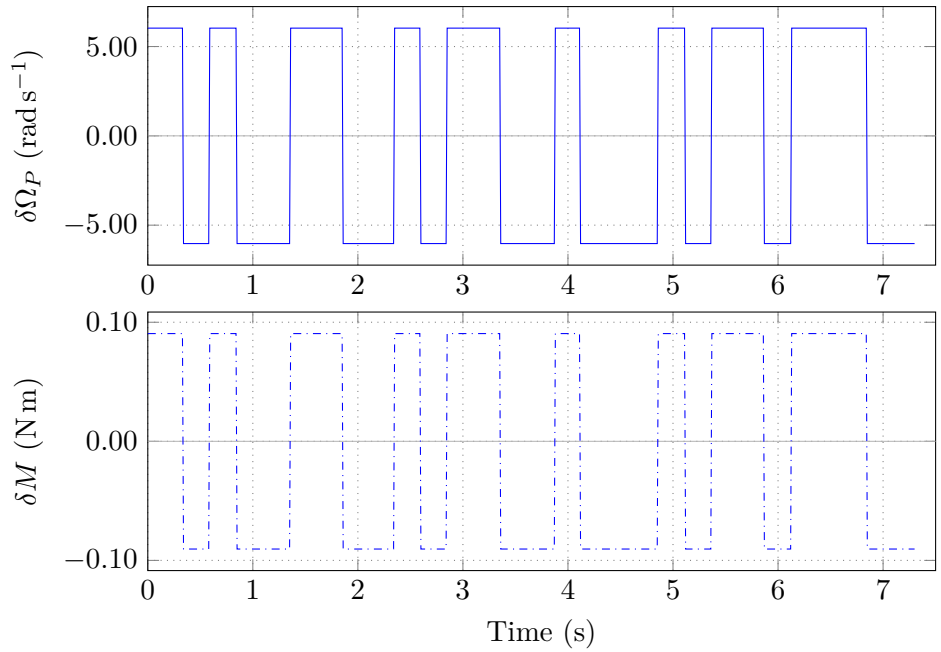


Figure 4.4: Measured $\delta\Omega_P$ and computed control variable δM. Measured signals are represented with a continuous line, computed signals are shown with a dotted line

The output measurement, the pitch rate signal q, was acquired and logged by the quadcopter itself using its on-board IMU with a time step of 0.01s. The pitch angle Θ was computed from the pitch rate measurements as shown in equation (4.2). The measured pitch rate and the computed pitch angle are shown in Figure 4.5.

4.3 Noise Considerations

Like all real world signals the input-output signals measured on the quadcopter are noisy. As detailed in Section 2.3.3, VRFT solves the problem of reducing the noise induced bias in the criterion it minimises with an instrumental variable approach. This requires either the collection of a second input-output dataset or the identification of a high-order model of the plant

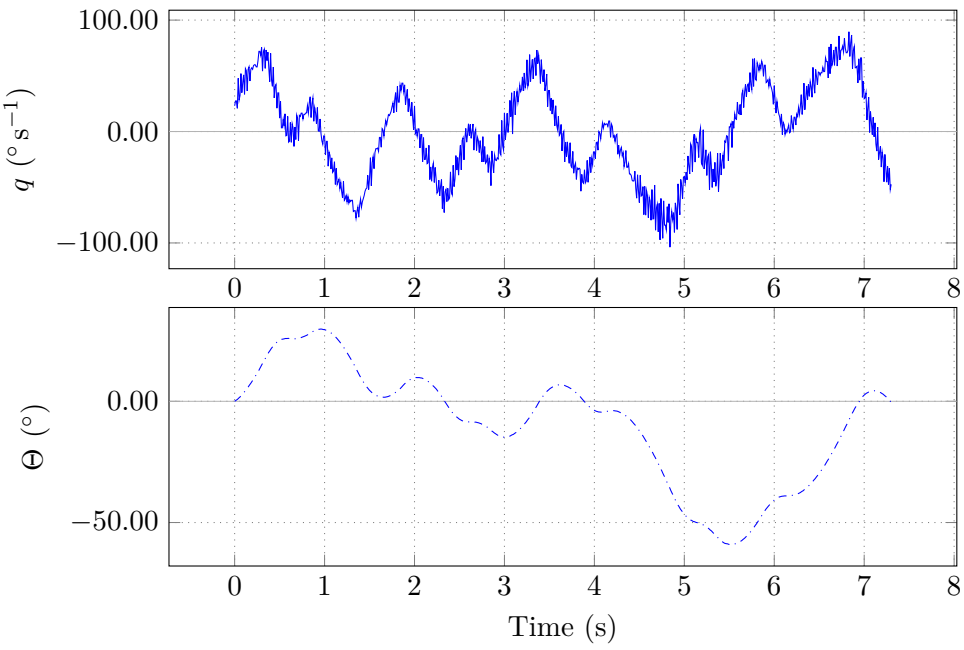


Figure 4.5: Measured pitch rate (q) and computed pitch angle (Θ). Measured signals are represented with a continuous line, computed signals are shown with a dotted line

that can be used to generate such a dataset.

The toolbox used in this work to perform the VRFT [3] uses the second method. It accepts as an input the order of an ARX model that will be used to generate the second dataset.

To identify the order of the ARX model of the plant to be used the measured input data was loaded into the `MATLAB` System Identification tool and de-biased. Using the polynomial estimation function It was observed that the model that provided the best fit was an $ARX(17, 7)$.

4.4 Reference Models

The structure of the reference models was previously defined in Section 3.2 based on the knowledge provided by the available models. What is left is to identify an *inner* and an *outer* reference model that maximises the achievable performance.

This was done with a series of tests. VRFT makes this incredibly fast; a

single input-output dataset can be used to tune any number of controllers. Once programmed onto the quadcopter the system was put through a series of step inputs and any controllers that unexpectedly yielded unstable control loops were immediately discarded to avoid what Elon Musk famously referred to as *rapid unscheduled disassembly* events. The final choice of parameters represents the best trade-off between the rise time and the settling time.

This also illustrates one of the limitations of VRFT: it does not provide any kind of guarantee that the closed loop transfer function will be stable.

4.4.1 Inner Reference Model

The structure of the *inner* reference model decided in Section 3.2.1 is

$$M_i(s) = \frac{\omega_{n_i}^2}{s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2}\frac{s + z_0}{z_0}$$

where ω_{n_i} is the bandwidth of the reference model, ζ_i is the damping ratio and z_0 is the position of the zero.

At the conclusion of the testing campaign I observed that the best tradeoff between the rise time and the settling time was obtained with a bandwidth $\omega_{n_i} = 8\text{rad}\,\text{s}^{-1}$, a damping-ratio $\zeta_i = 0.9$ and a zero placed in $z_0 = 5$

$$M_i(s) = \frac{64s + 320}{5s^2 + 72s + 320}\frac{s - 5}{5} \tag{4.3}$$

The VRFT procedure is defined only for discrete time systems but the quadcopter firmware considers only continuous time models. This mismatch was overcome by discretising the models prior to performing the VRFT and converting the output back to a continuous time model.

The discretised form of the inner reference model, considering a time step $T_s = 0.01\text{s}$ is

$$M_{Ri}(z) = \frac{0.1221z - 0.1162}{z^2 - 1.86z + .8659}. \tag{4.4}$$

4.4.2 Outer Reference Model

In Section 3.2.1 it was decided that the best form for the *outer* reference model is a simple second order model. At the conclusion of the testing campaign the model providing the best traedoff between the rise time and the settling time had a bandwidth of $\omega_{n_o} = 4\text{rad}\,\text{s}^{-1}$ and a damping ration of $\zeta_o = 0.9$

$$M_{Ro}(s) = \frac{16}{s^2 + 7.2s + 16} \tag{4.5}$$

and the discretised form of this reference model is

$$M_{Ro}(z) = \frac{7.81z + 7.625}{z^2 - 1.929z + 0.9305}10^{-4} \tag{4.6}$$

4.5 Results & Comparison

Running the VRFT procedure on both the *inner* and *outer* loops with the input-output data shown in Figures 4.4 and 4.5, the reference models (4.4) and (4.6) and considering an $ARX(17, 7)$ model for the noise mitigation leads to the controller parameters shown in table 4.1. Also shown are the parameters for the pre-existing H_∞ controller to be used as a point of reference.

	Inner Controller (PID)				Outer Controller (PD)		
	K_{p_i}	K_{i_i}	K_{d_i}	T_f	K_{p_o}	K_{d_o}	T_f
VRFT Tuned	0.2978	0.514	0	0.01	1.6057	0.0	0.01
H_∞ Tuned	0.298	0.304	0.0499	0.01	2.0	0.00522	0.01

Table 4.1: Controller parameters for both the VRFT and H_∞ tuned controllers. Note that T_f, the filter time constant, was set manually for both controllers and is not a direct product of the tuning procedures.

4.5.1 Set-Point Tracking

To validate the controllers a test sequence with steps of increasing amplitude was generated an fed as a set a set-point to the quadcopter. For safety the

tests were performed with the quadcopter securely fastened to the test-bed. The result of one such run with the VRFF tuned controllers of Table 4.1 are shown in Figure 4.6.

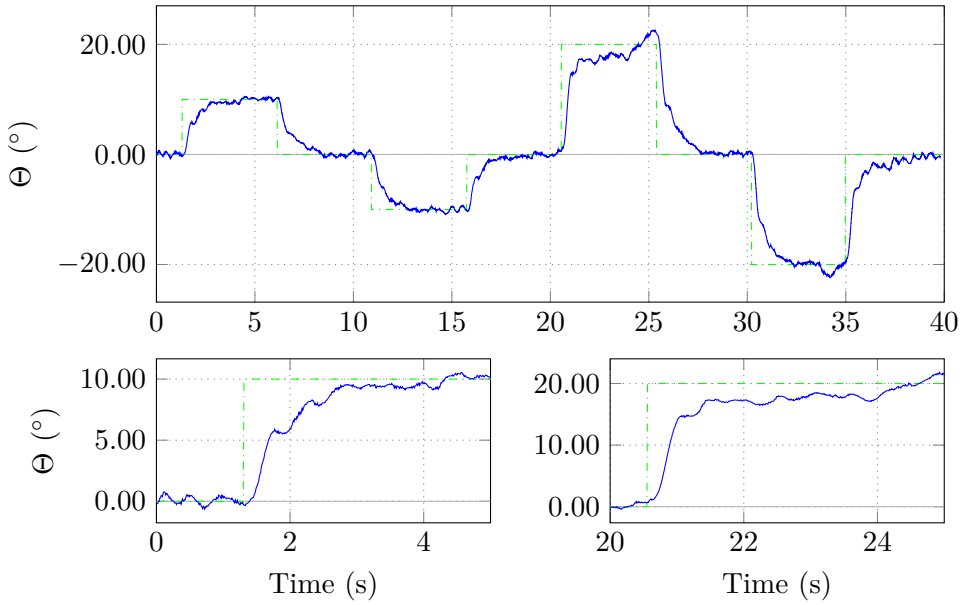


Figure 4.6: Pitch angle set point tracking using the VRFT tuned controller

The test was repeated 10 times using . the both the VRFT and H_∞ tuned controllers and the average mean square error was computed.

The set-point tracking performance of the quadcopter with, respectively, the VRFT and H_∞ controllers during one, randomly chosen, run is shown in Figures 4.6 and 4.7.

The VRFT and H_∞ tuned controllers were very similar in the simulation runs with the VRFT controller being marginally slower than the H_∞ controller. In these experiments this trend is inverted. The VRFT tuned controller is marginally faster than the H_∞ controller. This can be explained quite easily: the H_∞ tuned controller is based on an identified model of the system and can only ever be as good as the identified model. The VRFT tuned controller has the advantage of being tuned on real data collected on the system and is thus exempt from modelling errors.

The mean square error, comparing the measured pitch angle to the set-point, was computed for each test and averaged. The values for the VRFT and H_∞ tuned controllers are shown in the first column of table 4.2. The

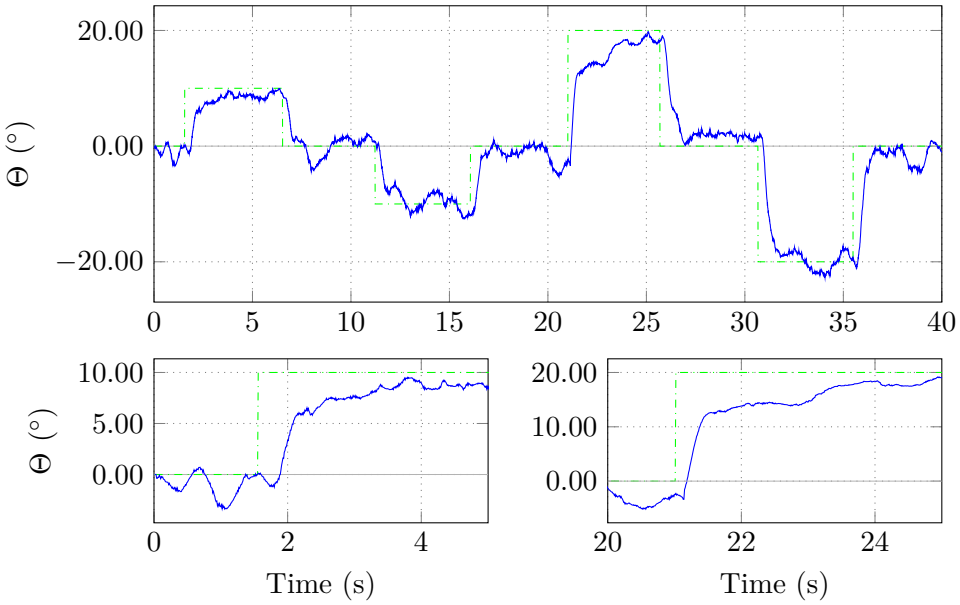


Figure 4.7: Pitch angle set point tracking using the pre-existing H_∞ controller

mean square error of the VRFT controller, whilst a little higher than that of the H_∞ controller is still within acceptable bounds. It can be explained by the slightly more oscillatory nature of the VRFT tuned controller.

4.5.2 Disturbance Rejection

The disturbance rejection properties of the firmware were considered using a similar method. The firmware of the quadcopter provides a method to introduce a step disturbance on the speed of the motors. This was used to repeatedly reduce the lift generated by the motors on the front of the quadcopter (motors 1 & 2) by 10 %. The test was repeated 10 times with both the VRFT and H_∞ tuned controllers and the average of the mean square error was computed.

The disturbance rejection performance of the quadcopter with, respectively, the VRFT and H_∞ controllers during one, randomly chosen, run are shown in Figures 4.7 and 4.7.

It is immediately apparent that the VRFT tuned controller offers significant improvements to the steady state error. The H_∞ tuned controller settles with a steady state error of several degrees whereas the VRFT tuned

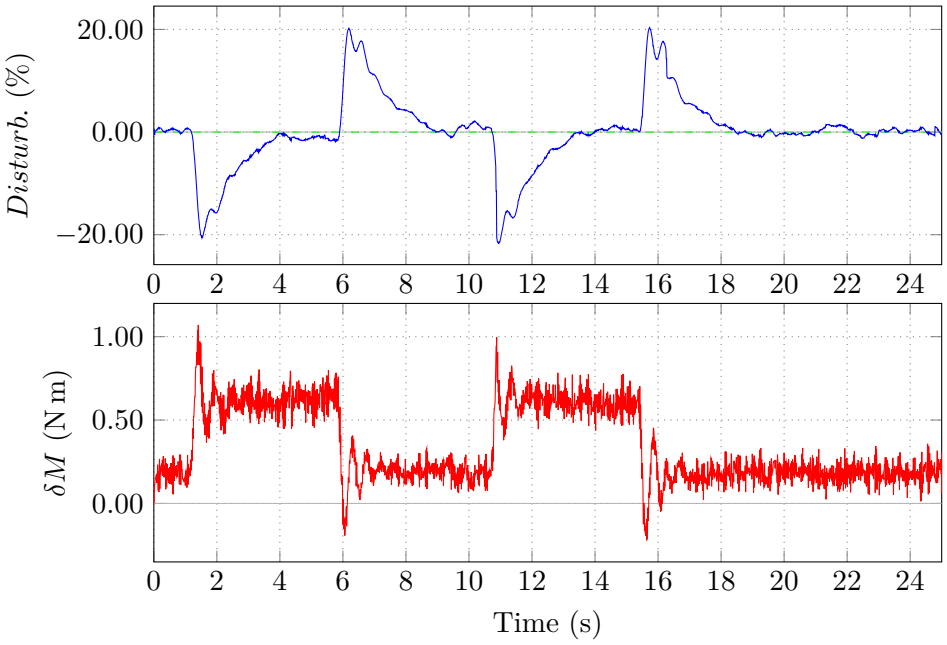


Figure 4.8: Disturbance rejection properties of the VRFT Tuned controller. The disturbance is a 10% drop in the speed of rotors 1 & 2.

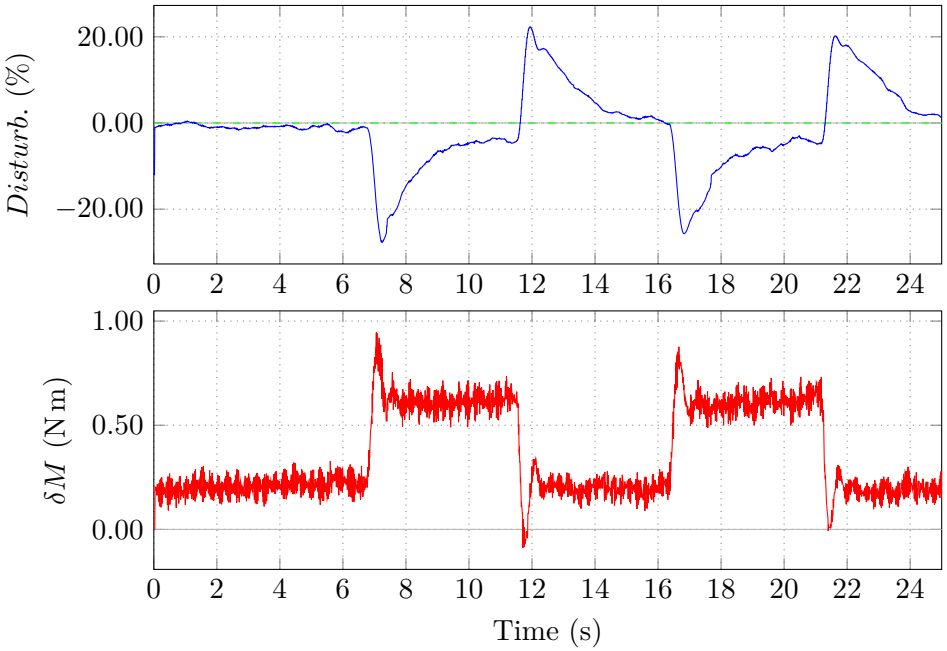


Figure 4.9: Disturbance rejection properties of the H_∞ Tuned controller. The disturbance is a 10% drop in the speed of rotors 1 & 2.

controller achieves zero state error. In addition the control effort required by both controllers is quite similar even if the VRFT tuned controller has slightly higher peaks.

The improvement is reflected in the steady state error of the two controllers as shown in Table 4.2. The mean square error of the VRFT tuned controller is slightly lower than that of the H_∞ controller.

	Mean Square Error	
	VRFT Tune Controller	H_∞ Tuned Controller
Undisturbed	10.719	6.3384
Disturbed	4.7908	5.5686

Table 4.2: Mean of MSE for validation experiments considering both VRFT and H_∞ tuned controllers

Bibliography

[1] Andrea Bonarini, Matteo Matteucci, Martino Migliavacca, and Davide Rizzi. "R2P: An open source hardware and software modular approach to robot prototyping". In: *Robotics and Autonomous Systems* (2014). ISSN: 09218890. DOI: 10.1016/j.robot.2013.08.009.

[2] M C Campi, A Lecchini, and S M Savaresi. "Virtual reference feedback tuning: a direct method for the design of feedback controllers". In: *Automatica* 38 (2002), pages 1337–1346. URL: www.elsevier.com/locate/automatica.

[3] Marco Campi and Sergio Savaresi. *VRFT Toolbox for MATLAB*. URL: http://marco-campi.unibs.it/VRFTwebsite/.

[4] Simone Formentin, Alberto Cologni, Damiano Belloli, Fabio Previdi, and Sergio M. Savaresi. "Fast tuning of cascade control systems". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)*. 2011. DOI: 10.3182/20110828-6-IT-1002.02761.

[5] Mattia Giurato. "Design, integration and control of a multirotor UAV platform". Master's thesis. Politecnico Di Milano, 2015. URL: http://hdl.handle.net/10589/115191.

[6] Htikan Hjalmarsson, Michel Gevers, Svante Gunnarsson, and Olivier Lequin. "Iterative feedback tuning: Theory and applications". In: *IEEE Control Systems Magazine* (1998). DOI: 10.1109/37.710876.

[7] Pietro Panizza, Davide Invernizzi, Fabio Riccardi, Simone Formentin, and Marco Lovera. "Data-driven attitude control law design for a variable-pitch quadrotor". In: *Proceedings of the American Control Conference*. 2016. DOI: 10.1109/ACC.2016.7525620.

[8] Klaske Van Heusden, Alireza Karimi, and Dominique Bonvin. "Data-driven model reference control with asymptotically guaranteed stability". In: *International Journal of Adaptive Control and Signal Processing* (2011). ISSN: 08906327. DOI: 10.1002/acs.1212.