Julius-Maximilians-Universität Würzburg
Fakultät für Mathematik und Informatik

Informationstechnik für Luft- und Raumfahrt
Lehrstuhl für Informatik 8 Prof. Dr. Sergio Montenegro

Bachelorthesis

Adaptive Altitude Controller for a Quadrocopter

Vorgelegt von

Jasper Tobias Zevering

Matr.-Nr.: 2172534

Prüfer: Prof. Dr. Sergio Montenegro

Betreuende wissenschaftliche Mitarbeiter: M.Sc. Michael Strohmeier

Würzburg, 07. 11. 2018

Erklärung

Ich versichere, dass ich die vorliegende Arbeit einschließlich aller beigefügter Materialien selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Werken entnommen sind, sind in jedem Einzelfall unter Angabe der Quelle deutlich als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.
Mir ist bekannt, dass Zuwiderhandlungen gegen diese Erklärung und bewusste Täuschungen die Benotung der Arbeit mit der Note 5.0 zur Folge haben kann.

Würzburg, 07. 11. 2018

Jasper Tobias Zevering

Abstract

An adaptive control system adjusts its parameters during the control process. This happens according to defined rules. This thesis introduces and explains the following rules and systems: gain scheduling, high-gain control, self-oscillating adaptive system, model reference adaptive control, self-tuning regulators, model identification regulator and stochastic control. This provides an overview of the adaptive approaches and shows the underlying considerations, as well as the development and necessity, of increasingly complex controllers. The Model Reference Adaptive Controller (MRAC) adapts by means of comparison to a chosen reference system. This adaptation can be done according to the rule of the Massachusetts Institute of Technology (MIT rule) or the Stability Proofed Rule (SPR rule) - MIT with better empiric results and SPR with a mathematical proof of stability. Both methods are explained and mathematically derived. The Modified MRAC is a combination of an adaptive control and PID-controller, which leads to better results when rapidly changing the system. In this thesis a Modified Model Reference Adaptive Controller for controlling the altitude of a quadrocopter will be set up, implemented and evaluated. The requirements of the implemented system are to restore or even hold the flight dynamics when changing a system's mass rapidly, and to compensate the impact of the change. This will be done with a reference system of the second order.

Zusammenfassung

Eine Adaptive Regelung beschreibt eine Regelung, die bestimmte Parameter im Laufe des Reglungsprozesses anpasst. Dies folgt bestimmten Regeln. Von diesen Reglern, beziehungsweise Systemen, werden die folgenden vorgestellt: gain scheduling, high-gain control, self-oscillating adaptive system, model reference adaptive control, self-tuning regulators, model identification regulator und stochastic control. Dies gibt einen Überblick über die verschiedenen Methoden und zeigt den gedanklichen Weg, die Entwicklung und Notwendigkeit von komplexeren adaptiven Systemen. Der Model Reference Adaptive Controller adaptiert mittels Vergleich zu einem gewählten Refernzsystem. Der eigentliche Schritt der Adaption nutzt entweder das Voergehen von dem Massachusetts Institute of Technology (MIT regel) oder die Stability Proof Rule (SPR Regel). MIT mit den empirisch besseren Ergebnissen, SPR mit dem mathematischen Beweis der Stabilität. Beide Wege werden erklärt und mathematisch hergeleitet. Modified MRAC ist die Kombination aus einem adaptiven Regler und einem PID-Regler, was zu besseren Ergebnissen führt bei plötzlichen Änderungen des Systems. In dieser Arbeit wird ein Modified Model Reference Adaptive Controller für die Höhenregelung eines Quadrocopters aufgestellt, implementiert und evaluiert. Als Anforderung für die Implementation gilt: Die Wiederherstellung beziehungsweise Beibehaltung eines Flugverhaltens und die höhentechnische Kompensation von plötzlichen Systemänderungen in Bezug auf die Masse. Als Referenzsystem wird ein System zweiter Ordnung gewählt.

Contents

List of Figures

1 Introduction

The advantage of an open-loop controller over a closed loop feedback controller is the capability to regulate not well-known systems, or systems whose parameters are not set as needed for a good open-loop controller. But even a controller with feedback has its limitations as to changing the environment or system settings. Even if its technical means are capable of handling these parameter changes, the controller will mostly not be able to handle them while maintaining the same characteristics the system had prior to this change. The Julius Maximilian University of Würzburg is doing research on a defense system against micro drones, called *MIDRAS*. This system uses drones carrying a net in order to catch smaller drones, as shown in 1.1. Varying characteristics display a problem of the system. The decreased mass, the force for deceleration of the caught drones, or even a caught drone still having on its motors apply extra force to the net. This short-term forces could be five times greater than the normal gravity force the drones face. In the long-term they should be able to handle carrying up to 70% of the initial mass of the system. So even if the normal control would be able to somehow handle this non-negligible change of system, the flight characteristics would not be anywhere near as they would without it. With a sudden change, but one that is schedulable, it is possible to change the predefined control parameters for a flight without payload to a flight with payload. This is possible if the payload is well known. But with processes like weight loss of planes due to the kerosene consumption, there was an early need for adaptive

Figure 1.1: Animation of the principle of the MIDRAS project. Source: University of Würzburg

controllers. Therefore, in the early 1950s, several methods were developed to adapt controllers, such as the sensitivity rule or the MIT-rule, which will be used and explained in this thesis. In 1958, R. Kalman developed the first self-tuning controller which was an optimal LQR with identification of parameters, but it was not taken seriously due to the lack of high-speed computers and insufficient setup of the theory [4] [5]. The crash of the X-15A-3 in 1967 due to a stable but non-robust adaptive controller led to a delay in further research. It was not until the early 1990s that NASA and the air force started their research on "reconfigurable flight-control". Later in the 90s the number of methods for adaptive controllers increased rapidly, described by M. Steinberg as "an explosion in the number and types of approaches applied to the reconfigurable flight-control problem" [19]. Controlling with fuzzy-logic and neural-networks also emerged at this time. The test flights with these controllers faced several uncertainties; the most advanced was the Lyapunov-based proof of stability, which today is still the concept of proof of stability [19] [5].

There have been several approaches to take an adaptive controller as a controller for a Unmanned Aerial Vehicle (UAV).

In "Modified Model Reference Adaptive Control of UAV with Wing Damage" by Yahui Xiao, Yufei Fu, Chengfu Wu and Pengyuan Shao an Modified Model Reference Adaptive Controller (M-MRAC) was implemented for stabilizing a Quadrocopter when taking wing damage, simulated by the loss of 40% of the thrust on one side. This was done only by simulation and no real system. The M-MRAC was able to stabilize the damaged quadrocopter, but not in a practical relevant amount of time of nearly 20 seconds. As further research, the flight dynamics of the quadrocopter after damage with and without system change was suggested, to fulfill an emergency landing[20].

Zachary Thompson Dydek implements in "Adaptive Control of Unmanned Aerial Systems" a Modified Reference Adaptive Controller (MRAC) for a quadrocopter attitude control. One test setup is the sudden removal of a tip of one rotor. Due to practical and stability reasons the implementation uses a deadband in which the adaptions is deactivated when the error between reference model and real value is to small. So the adaptation only takes place with big enough change to the system. This is not done in the implementation in this thesis. Furthermore, a comparison of a so-scalled Combined Model Reference Controller (CMRAC), which is a combination of the direct and indirect, model identifying, approach of the MRAC, to normal MRAC is done

with normal flight behavior. As system change time delay of the loops of different duration are taken[3].

In "Adaptive Control of Unmanned Aerial Vehicles: Theory and Flight Tests" by Suresh K. Kannan, Girish Vinayak Chowdhary, and Eric N. Johnson an MRAC was implemented on a real helicopter and showed a good behavior when flying autonomously. With the altitude control, there could be seen non-negligible oscillations of up to more then 3 meters. This was due to the limitations of the original controller. The behavior was recognized and even described as "expected" but neglected due to other priorities of the implementation. Therefore, for the implementation in this thesis, where the altitude is the main focus, an approach which does not rely on inner limitations of the controller is applied [8].

2 Basics of Control-Theory

In this chapter two basic concepts of control theory will be explained, which are essential for this thesis. The PID controller, as the chosen controller for all implementations and evaluations, and the stability theorem of Lyapunov, as it is taken as proof for the stability of adaptive approaches. With the PID controller, the cascaded controller will be explained, too. Also the characterization of transfer systems is explained, to be able to compare different systems in this thesis.

2.1 PID Controller

A Proportional-Integral-Derivative-controller (PID-controller) is a closed loop feedback controller that aims to eliminate the error between the feedback and control signal. Figure 2.1 shows the block schema of the basic system.

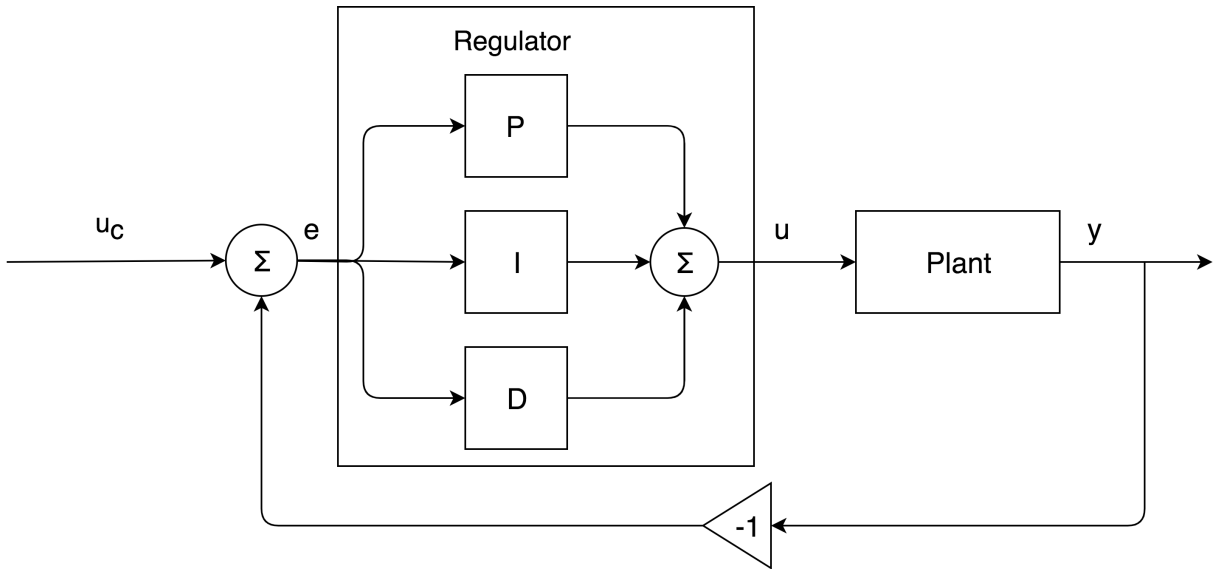

Figure 2.1: Block schema of basic PID-controller

The generalized time domain functions are:

$$P(t) = k_p \cdot e(t) \tag{2.1}$$

$$I(t) = k_i \cdot \int_0^t e(t)dt \tag{2.2}$$

$$D(t) = k_d \cdot \dot{e(t)} \tag{2.3}$$

P represents the current error, I the past error and D the prediction of the future error. $e(t)$, which is the error between the command u_c and the output y, is going to three separate blocks representing P, I and D. The transfer functions in the frequency domain are:

$$P(s) = k_p \tag{2.4}$$

$$I(s) = \frac{k_i}{s} \tag{2.5}$$

$$D(s) = k_d \cdot s \tag{2.6}$$

In Figure 2.1 the *P*, *I* and *D* block can be summed up by one block called *Regulator*. The transfer function of this *Regulator* Block $G(s)$ is:

$$G(s) = P(s) + I(s) + D(s) \tag{2.7}$$

Plant is the whole process from a mathematical controller output u to the real output y. Speaking for the real system, knowing y for the feedback implicates some sort of measurement, because *plant* is the real world process. Thus plant is the part where the real world disturbance comes into the system. However, this is not shown due to the abstraction process.
This all leads to the general closed loop feedback controller design, seen in Figure 2.2, which is taken as basis for all controllers in this thesis. For better visualization, the *Regulator* and *Plant* are sometimes combined as block *Process*. [6]
A development of the closed loop feedback controller is the cascaded controller. It contains an inner and outer loop. The inner loop controls the value of the secondary process. The outer loop the value of the primary process. It is necessary, that these two processes correlate and the change of the secondary output leads to a change of the primary output in a fixed relation.

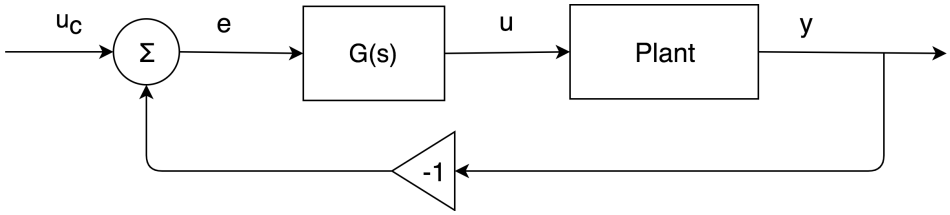

Figure 2.2: Block schema of a general closed loop feedback controller

The corresponding block schema can be seen in Figure 2.3. This structure requires two feedback channels. One of the value after the secondary process and one after the primary process [9].

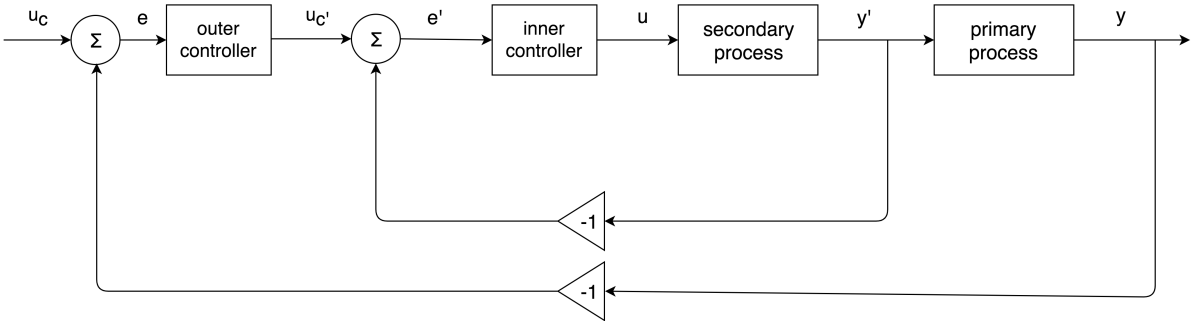

Figure 2.3: Block schema of a cascaded closed loop controller

A simplified option of this general cascaded controller is the approach of having only one process, so there is only one output. Therefore the feedback for the inner controller is the derivation of the output of the process. So in the general model the primary and secondary process are summarized, with the primary process being an integration. Practically this means, that the outer controller gives the desired change of the output, like a velocity, and the inner controller tries to match the change of the output to the desired change and not controlling the specific output itself. The two feedbacks can be either realized by two different measurements, one of the value and one of its change (like altitude and vertical velocity), or by only one measurement of the value and the mathematical derivation of it. This shown in Figure 2.4.

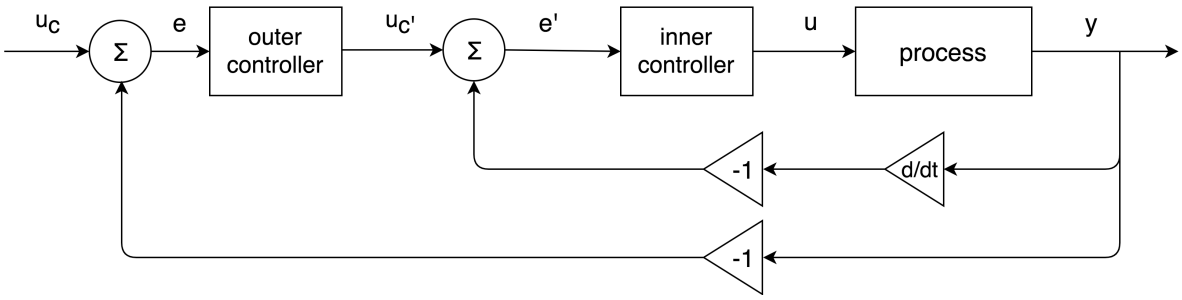

Figure 2.4: Block schema of cascaded closed loop controller with one process

2.2 Characterization

To characterize different transfer systems, which can be a transfer function or a whole system with one input and output, three values are taken as characteristics. Rise time, settle time and overshoot. Therefore a system gets an input change from a start value to a new desired value without any transition. The rise time is defined as the time from the point where the input changed, to the time the output first reaches 95% of the desired value. The settle time is the time from the start to the first point where the output does not leave the range between 95% and 105% of the desired value. The Overshoot is the percentage of the difference between the maximum reached value and the desired value (or minimum to the desired when the desired value is under the start value) to the difference of the desired value to the start value. Most likely is a step response from 0 to 1.

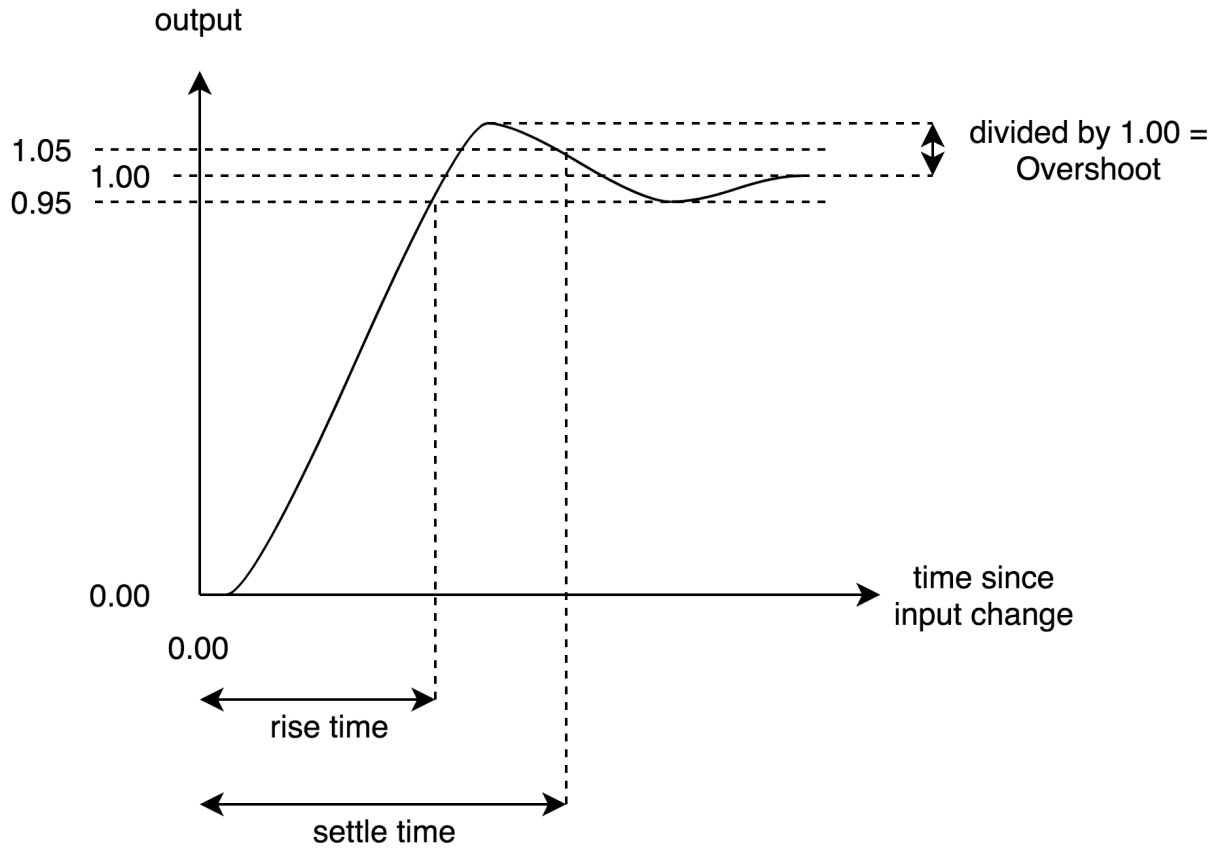

Figure 2.5: Visualisation of the characteristics of a transfer system

2.3 The Stability Theorem of Lyapunov

Consider a system:

$$\dot{x}(t) = f(x,t) \quad \text{and} \quad x(t_0) = x_0 \quad x \in \mathbb{R}^n \tag{2.8}$$

Unlike asymptotic stability, which is defined by a point x_α being asymptotic stable if the point itself is stable and locally attractive, meaning at t_0 there exists $\theta(t_0)$ so that:

$$\big\|x(t_o)\big\| < \theta \quad \Longrightarrow \quad \lim_{t\to\infty} x(t) = x_\alpha \tag{2.9}$$

the stability in sense of Lyapunov is weaker, saying a point x_α is stable at $t = t_0$ if for any $\epsilon > 0$ there is a $\theta(t,\epsilon)$ such that:

$$\big\|x(t_o)\big\| < \theta \quad \Longrightarrow \quad \big\|x(t)\big\| < \epsilon \quad \forall t > t_0 \tag{2.10}$$

This means that for every value ϵ, there is another value θ, from which, once reached, $\|x\|$ cannot become ϵ or larger.

Now the direct, also called second, method of Lyapunov is introduced. It provides theorems about a continuous function $V(x,t)$. The method provides criteria when the V in sense of Lyapunov is locally stable, uniformly locally stable, uniformly locally asymptotically stable and globally uniformly asymptotically stable. Only the second will be needed in this thesis. Lyapunov's criteria for that is:

"If $V(x,t)$ is locally positive definite and decrescent, and $\dot{V}(x,t) \leq 0$ locally in x and for all t, then the origin of the system is uniformly locally stable (in the sense of Lyapunov) ".[12]

3 Adaptive Controllers

Prof. Karl Johan Åström defines an adaptive controller as follows[16]:

> *An adaptive controller is a controller with adjustable parameters and a mechanism of adjusting the parameters.*

So there are three parts to be considered, the sort of controller, which parameters to be adaptively updated and the sort of mechanism that leads to adaptation of the parameters. Because the different mechanisms of the adaptations are designed to work with all kinds of controllers, as long as they provide the right adjustable parameter, like a gain as the last step, the different mechanisms will be the subject of the further thesis and not the different kinds of controllers. The PID-controller was explained in Chapter 2.1 because of its use in the practical implementation. The following approaches will be treated because Åström, Sastry and Bodson claim them as the common adaptive schemata, although Sastry and Bodson list the stochastic control approach(3.6) as common and Åström does not and Åström lists self-oscillating adaptive systems(3.3), which Sastry and Bodson do not list. Moreover, Åström lists high-gain control (3.2) and Sastry and Bodson only list it as a schema of the model-reference control itself. For the sake of completeness, and as the self-oscillating adaptive systems(3.3) is a developed on high-gain control (3.2) they are all listed separately here.

3.1 Gain Scheduling

The most basic and intuitive idea of adaptive control is implemented in gain scheduling. It adapts parameters of the regulator according to chosen conditions. This could be measured by parameters such as the level of kerosene in a tank, altitude or temperature. Though, it can be any operative condition. The only requirement is that these measured parameters correlate with

the change of dynamics of the system. The higher the correlation, the better the possible adaptation becomes. This method can be as fast as the change of the measured parameters, and for

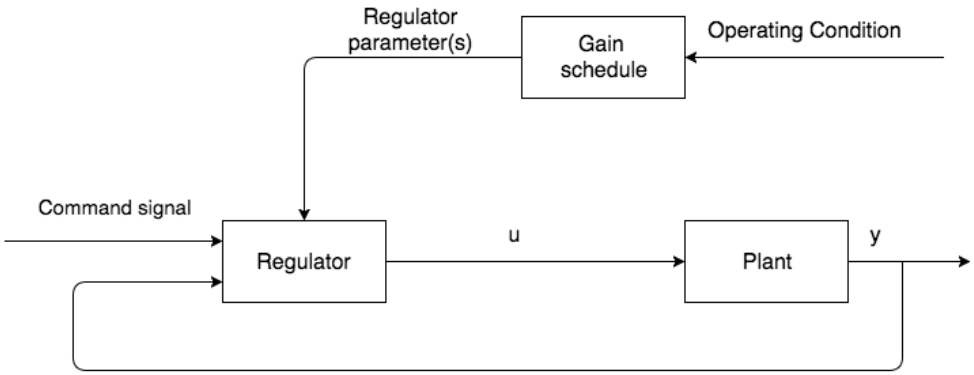

Figure 3.1: Block schema of the gain scheduling system with reference to [16]

well-known correlations this is a popular method for adaptive controllers. The disadvantage is that there is no real learning, as the process from the operating condition to the adapted regulator parameters itself is an open-loop controller, so it cannot handle changes of the correlation between system dynamics and the measured parameters. Due to this behavior as a result of the open-loop structure, it is controversial whether gain scheduling even belongs to adaptive controllers. However, it can be classified as an adaptive controller when taking the definition at the begin of this chapter into consideration [18] [16].

3.2 High-Gain Control

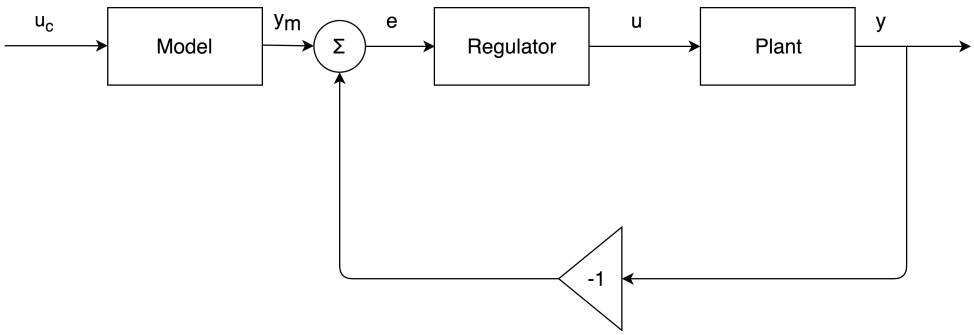

Figure 3.2: Block schema of the high-gain control method with reference to [16]

High-gain control is a second quite simple adaptive controller. The input of the adaptive mechanism itself is a reference model, thus Systry and Bodson [18] describe it as a section of the MRAC. Åström [16] does not, as in his opinion the high-gain control approach only uses the

reference model as input and thus it does not belong to the MRAC approach.
The idea is to set the regulator as a high-gain amplification for the following reasons: With the given model on the right, the transfer function of the plant considered as $P(s)$, and the regulator transfer function as gain k, the transfer function $G(s)$ from y_m to y is

$$G(s) = \frac{k \cdot P(s)}{1 + k \cdot P(s)}$$

where, without further mathematical proof, it can be seen that with a high gain k the transfer function becomes

$$\lim_{k \to \infty} G(s) = 1$$

which leads to $y = y_m$, the aim of this adaptive controller. The whole system is very insensitive to changes in the dynamics because of the very high gain. However, there are some downsides to this method.
First, it is not possible to run this approach without oscillations. In a real system the plant is not able to handle an infinite u without oscillations. In this case u would be watched by a limit cycle detector. This stops the increment of the gain when reaching a critical cycle of u for the real setup and it stops the decreasement if this cycle exceeds a certain level. The limit cycle detection is most likely to be a rectifier and low-pass filter.
Second, because of noise in the frequency-band it leads to a decrease in the gain even if it is not near the critical value.
Third, the reference input may lead to saturation of the system because of the high gain.
Fourth, the saturation may mask oscillations of the limit cycle, which lets the gain reach values above the limit that causes instability.
Although 3. and 4. may seem easy to handle practically, 4. was the reason an experimental X-15 aircraft crashed. It was found that it had saturation issues due the high-gain control. The saturation left instabilities undetected and led to the crash [10][18].

3.3 Self-Oscillating Adaptive Systems

The self-oscillating adaptive systems (SOAS) build on the same idea as the high-gain control 3.2 but the gain is not automatically set to be as high as possible. This is implemented by a

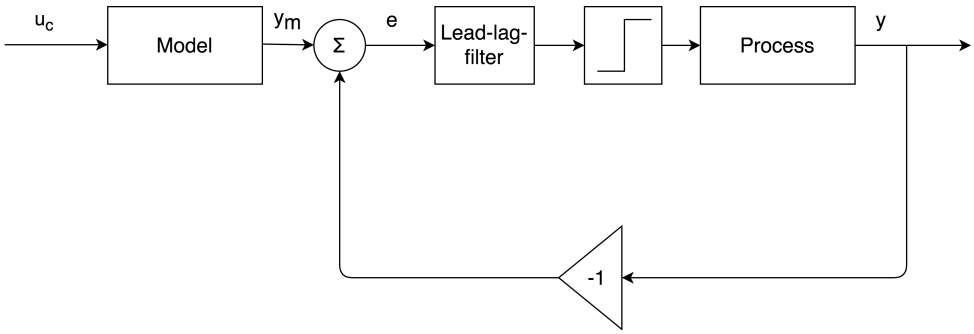

Figure 3.3: Block schema of the self-oscillating adaptive systems with reference to [16]

relay that leads to maximum cycle oscillations. The system will always be excited because of these oscillations. The frequency can be adjusted by a lead-lag filter, which is put right after the error calculation, and the amplitude can be in- or decreased by the value relay switching on or off. Like the high-gain method, the behavior of a SOAP is also not sensitive to changes in the dynamics of the system because of the high gain. Although this is a well-known method, it is not considered to be implemented in aircraft, because the limit cycle oscillations will be always noticed by pilots. For applications where noticeable oscillations are acceptable, this is a robust approach. The problem of oscillations can be improved by creating a second loop, where the gain of the relay is adaptively customized in consideration of e. Another method is to adapt the limit cycle [16].

3.4 Model Reference Adaptive Control

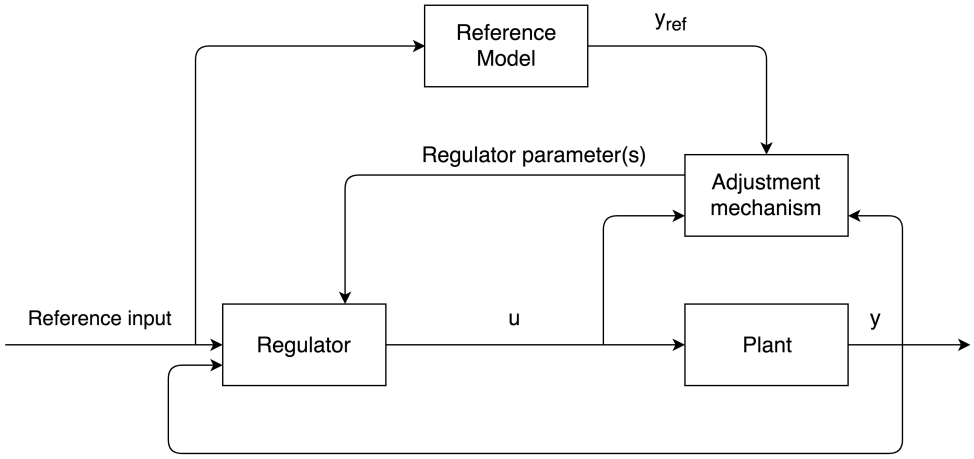

Figure 3.4: Block schema of the model reference adaptive controller with reference to [18]

The Model Reference Adaptive Control (MRAC) is an approach with a desired behavior of the real system, set by a reference model, which can represent an optimal model as well as one that is only corresponding to certain parameters. The aim of MRAC is the minimization of the error e defined as:

$$e = y - y_{ref}$$

To accomplish this criterion there are certain methods like the MIT-rule, which belong to the *Gradient Approaches*, or the SPR-method that is based on the *Stability Theory* which is based on the *Lyapunov's Second Theorem*. In 4.2 both will be further explained. MRAC (sometimes called MRAS as Model Reference Adaptive System) can be thought of as two loops, an inner loop that presents the normal feedback control and an outer loop that adapts the regulator parameter to minimize e. Hence the outer loop can be seen as the regulator itself.

3.4.1 MIT-Rule

The MIT-rule is basically the so-called *Gradient Approach*, but because most of the research and work was carried out at the instrumentation laboratory at MIT (Massachusetts Institute of Technology), it is nowadays called the MIT-rule.
First, there is the definition of a cost function $J(\theta)$ as follows:

$$J(\theta) = \frac{1}{2} \cdot e^2 \tag{3.1}$$

e, as shown in Figure 3.4, is the error between the reference model output and the reality. θ is the adjustable parameter. Because the cost function has to be minimized as the aim of the controller, the change of θ, $\dot{\theta}$, has to be kept in the negative gradient of J.

$$\dot{\theta} = -\gamma \frac{\delta J}{\delta \theta} \tag{3.2}$$

γ is therefore a positive quantity that indicates the adaptation gain. Taking Equation 3.1 into Equation 3.2:

$$\dot{\theta} = -\gamma \cdot e \frac{\delta e}{\delta \theta} \tag{3.3}$$

It has to be considered that θ could be a vector of parameters and not just one. Then $\frac{\delta e}{\delta \theta}$ becomes the gradient of the parameters. $J(\theta)$ could theoretically be chosen arbitrarily, although there

are some common equations, which will be not further explained, with Equation 3.1 being the most common one.

For our requirements, building a basic adaption, like a feedforward gain system, the error is described as:

$$e = y - y_{ref} = u_c \cdot \theta \cdot G(p) - u_c \theta^0 \cdot G(p) \tag{3.4}$$

with $G(s)$ being the transfer function of controller and plant, p being the differentiation operator $\frac{d}{dt}$, θ^0 being a known gain for $G(s)$ of the model and u_c being the controller input. The control law is

$$u(t) = \theta \cdot u_c(t) \tag{3.5}$$

So the criterion of minimizing e is reached with getting the gain of the real controller becoming the gain of the adaptive model $\theta = \theta^0$. For this assumption $\frac{\delta e}{\delta \theta}$ becomes the following:

$$\frac{\delta e}{\delta \theta} = G(p) \cdot u_c \tag{3.6}$$

and $G(p) \cdot u_c$ can be easily described as

$$G(p) \cdot u_c = y_{ref}/\theta^0 \tag{3.7}$$

The MIT rule using Equation 3.3 then gives

$$\dot{\theta} = -\gamma' \cdot e \cdot y_{ref}/\theta^0 \tag{3.8}$$

This basic approach for only feed forward gain can be easily generalized by setting $\theta^0 = 1$, which leads to:

$$\dot{\theta} = -\gamma' \cdot e \cdot y_{ref} \tag{3.9}$$

which is the common known basic system description for MRAC with MIT-rule [7][16].

The resulting block-diagram, shown in Figure 3.5, shows the characteristic *parallel schema* because of two parallel loops. One is $G(s)$ itself because it is the transfer function for the controller and plant with a feedback loop, and the outer adaptive loop of adjusting θ. There is also a so-called *series schema*, one implementation was introduced with the high-gain control in 3.2, which had a reference model and then a loop for adjusting the controller. This schema does not

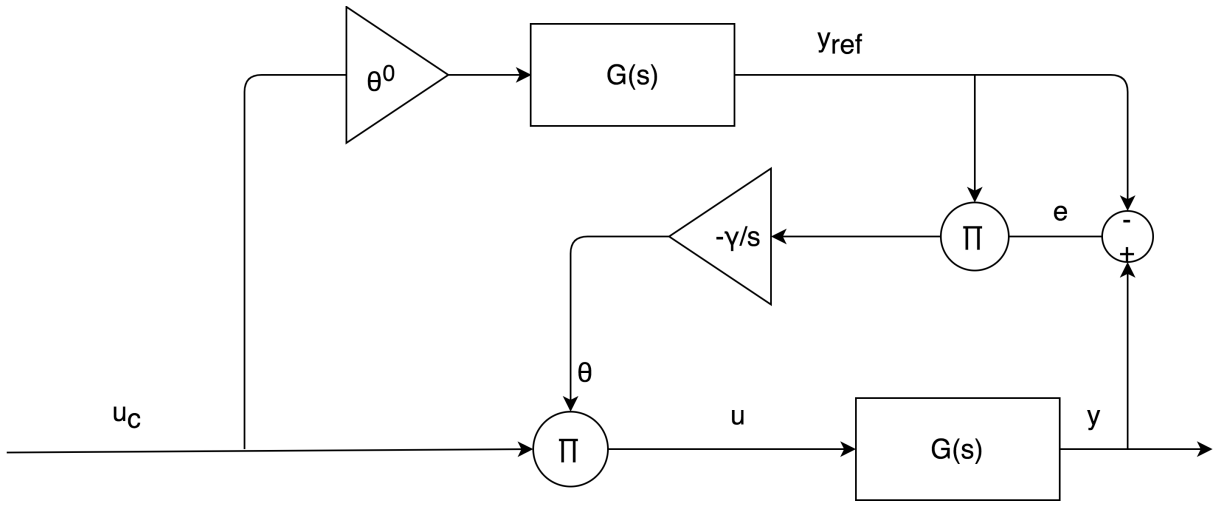

Figure 3.5: Block diagram of the mathematical model of the MIT rule with reference to [7]

consist of two loops, ignoring the possibility of the reference model consisting of a loop itself. That is why Åström[16] does not count it as MRAC, because for him MRAC always represents a parallel schema, although in Sastry and Bodson[18] this is not the case.

As a very basic rule, the MIT-rule faces some problems. A main problem is that the stability cannot be proven absolutely. It can only be proven as stable with u_{ref} and γ being not "sufficiently large"[16]. This is a well-known problem of large gains, leading to instability and/or oscillations [7]. However, it is still the most used approach with MRAC. There is a development of the MIT-rule, namely the modified MIT-rule, which normalizes the algorithm. This overcomes the problem of instability with large inputs because the gain θ depends on the $e \cdot u_{ref}$. It sets up $\dot{\theta}$ as follows:

$$\dot{\theta} = \frac{-\gamma \cdot e \cdot \lambda}{\alpha + \lambda^2} \tag{3.10}$$

and $\lambda = u_{ref}/\theta^0$ so with θ^0 simplified as 1 it becomes

$$\dot{\theta} = \frac{-\gamma \cdot e \cdot u_{ref}}{\alpha + u_{ref}^2} \tag{3.11}$$

α is introduced to overcome the problem of dividing by 0 when λ is small.[7]

3.4.2 SPR rule with Lyapunov stability

The second major approach of designing the MRAC adaption mechanism is the so-called SPR (Stability Proof Rule) which is proven with Lyapunov stability. Therefore, e is written just like with the MIT-rule in Equation 3.4 as:

$$e = y - y_{ref} = u_c \cdot \theta \cdot G(s) - u_c\theta^0 \cdot G(s) = G(s)(\theta - \theta^0) \cdot u_c \tag{3.12}$$

Now the state space model of G is introduced. The general state space model of a system is

$$\dot{x} = Ax + Bu \qquad\qquad y = Cx \tag{3.13}$$

Putting this model of G to Equation 3.12, leads to a relation between θ and e of:

$$\dot{x} = Ax + B(\theta - \theta^0)u_c \qquad\qquad e = Cx \tag{3.14}$$

If the homogeneous system $\dot{x}$ is stable in purpose of the asymptotical stability there are positive definite matrices P and Q with

$$A^T P + PA = -Q \tag{3.15}$$

being solved. As a possible solution of the Lyapunov function the function used by Åström[16] is used, which is:

$$V = 0.5(\gamma x^T P x + (\theta - \theta^0)^2) \tag{3.16}$$

Realizing γ, P and θ^0 being constant over time, the derivation of V becomes

$$\dot{V} = 0.5 \cdot \gamma(\dot{x}^T P x + x^T P \dot{x}) + (\theta - \theta^0) \cdot \dot{\theta} \tag{3.17}$$

Using the $\dot{x}$ and the resulting $\dot{x}^T$of Equation 3.14 in the Equation 3.17 considering Equation 3.15 $\dot{V}$ becomes:

$$\dot{V} = -\gamma \cdot 0.5 \cdot x^T Q x + (\theta - \theta^0)(\dot{\theta} + \gamma u_c B^T P x) \tag{3.18}$$

Searching for a law prohibiting the derivation of the Lyapunov function becoming positive, the adjustment law

$$\dot{\theta} = -\gamma u_c B^T P x \tag{3.19}$$

can be found under the condition of x not being 0. Now it is assumed the Lyapunov function could be chosen, such that

$$B^T P = C \tag{3.20}$$

with C being the output matrix of the system. Than e in Equation 3.14 leads to

$$B^T P x = Cx = e \tag{3.21}$$

Now the adjustment rule can be written as

$$\dot{\theta} = -\gamma u_c e \tag{3.22}$$

It is remarkable that the resulting adaption law of the MIT rule and SPR rule looking nearly the

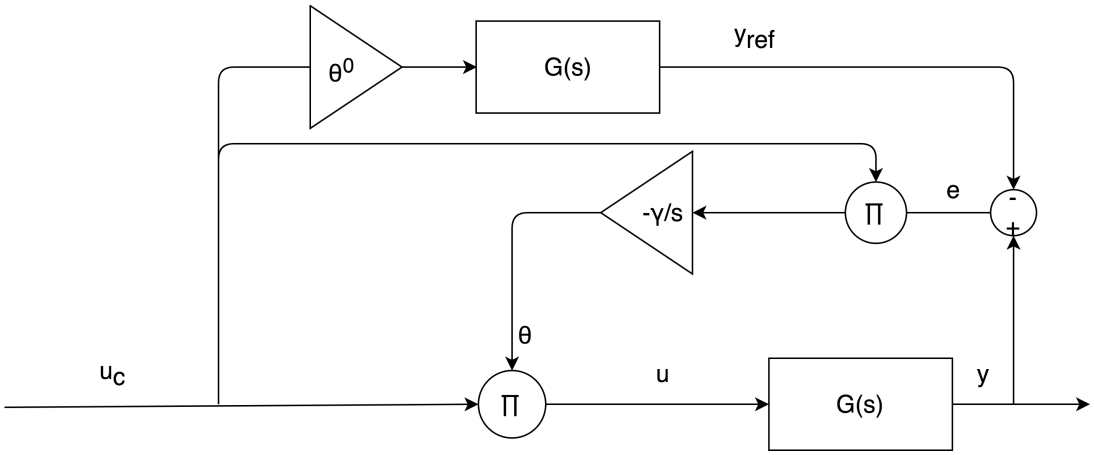

Figure 3.6: Block diagram of the mathematical model of the SPR rule with reference to [16]

same (see Equation 3.23)and one being fully proven stable and the other not. Also the SPR rule also only works for much smaller classes of input than MIT. The MIT rule provides with less gain than SPR a better result of following the reference model than SPR.[15]

$$\dot{\theta}_{MIT} = -\gamma y_{ref} e \qquad \dot{\theta}_{SPR} = -\gamma u_c e \tag{3.23}$$

With the simulation-setup of a quadrocopter explained in chapter 4 with the exact same setup for MIT and SPR , which only differ in one multiplying u_c and one y_{ref} in the adjustment rule, the system was stable with MIT at a γ of 12 and crashed with one at 13. SPR, on the other hand, could handle a γ up to 32 and crashed with 33. Fig 3.7 shows that even with the SPR system

handling the high gain, the error between real system and reference system is significantly larger than MIT, although it is oscillating. SPR put to a gain of 12, just like MIT, still has a significantly larger error at the point where the simulated UAV changes its velocity (40 s). In this setup, SPR always has the higher error than MIT, even at a gain of 4 which from the tested natural numbers is the gain with the least error. Therefore, the MIT rule is chosen in this thesis for the use in MRAC over SPR.

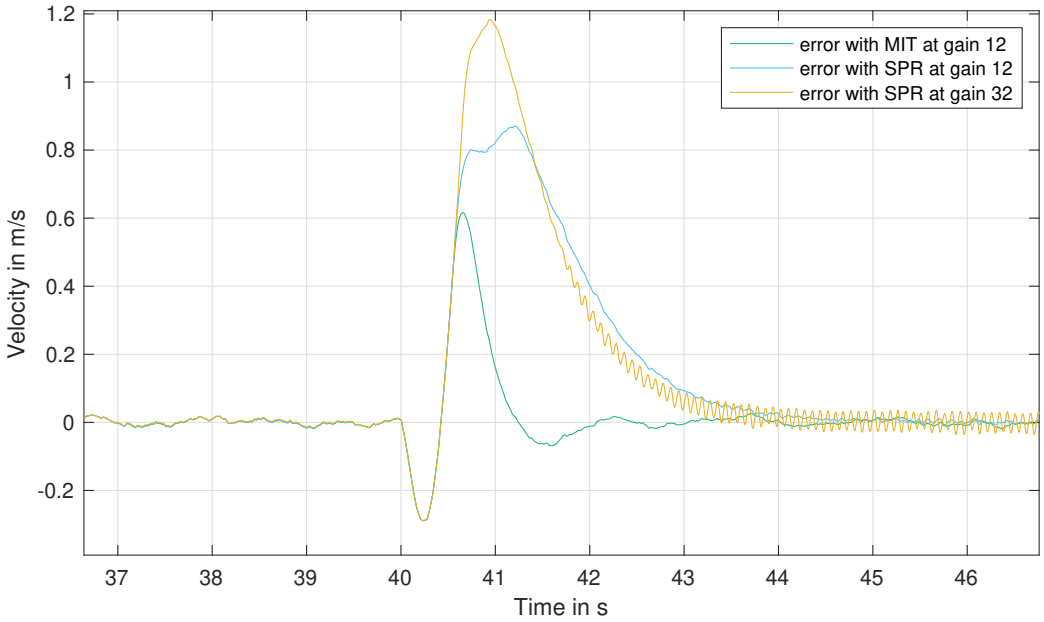

Figure 3.7: Error between velocity of the reference model and real velocity. At 40s the quadrocopter rises.

3.4.3 Modified MRAC

Figure 3.8 shows how MRACs with the MIT-rule of different gains minimize the error between the reference model and reality. It is conspicuous that all gains seem unable to handle the immediate error at 40s, where the altitude is increased. The internal integration cannot adapt with the required speed to avoid that first peak, or even see noticeable differences between the different gains. Therefore, an expansion of the system for more directness is needed. One expansion fulfilling this aim is the so-called Modified Model Reference Adaptive Regulator (M-MRAC). It is a combination of conventional MRAC and a PID-controller.

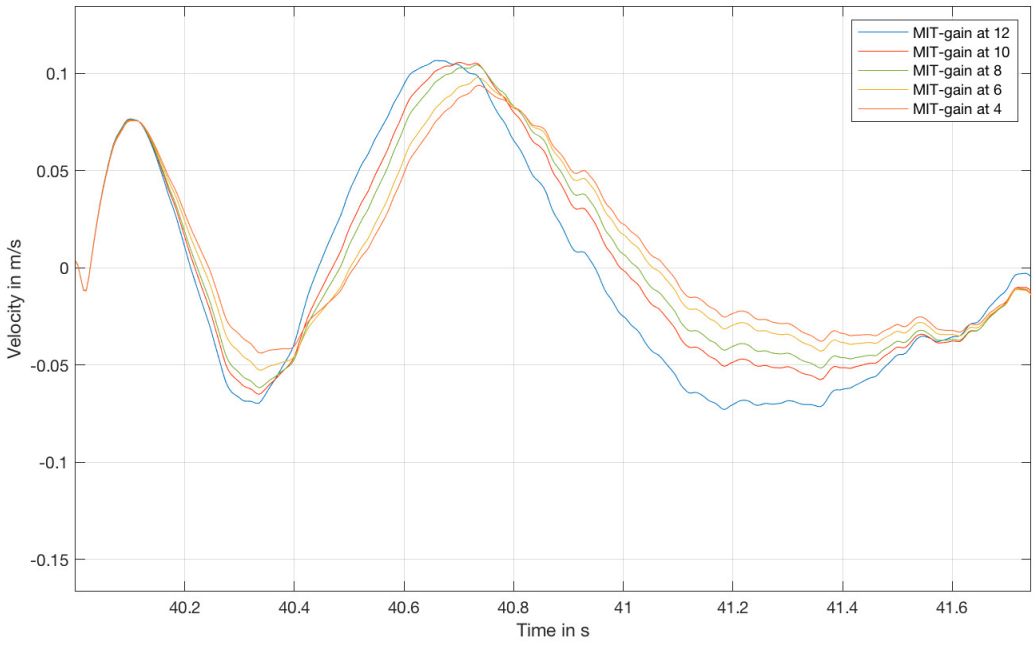

Figure 3.8: Error between reference velocity and real with MRAC with MIT rule adapted with gains from 4 to 12. The height was increased at 40 s which leads to acceleration and deceleration.

Thus, controller output u is no longer only described as

$$u = \theta G(s) \cdot (u_c - y) \tag{3.24}$$

with θ being adapted by the MIT-rule with the reference model. With M-MRAC u is described as

$$u = \theta G(s) \cdot (u_c - y) + (k_p \cdot e + k_i \cdot \int e dt + k_d \cdot \dot{e}) \tag{3.25}$$

meaning a PID-control of e was added. This can also be done with MRAC based on MIT as well as the ones based on SPR. The values of this PID can be estimated with the Ziegler-Nichols tuning method if the system is open-loop stable. If the system is unstable, another PID-tuning method may be required. [14][13] The comparison of MRAC and M-MRAC is done in Chapter 5.1.

3.5 Self-tuning Regulator and Model Identification Adaptive Controller

Self-tuning regulators and model identification adaptive controllers essentially perform the same function. In MIAC with the control signal u and the output y a model is estimated. This is given together with the uncertainty of the identification to the adjustment mechanism just like MRAC.

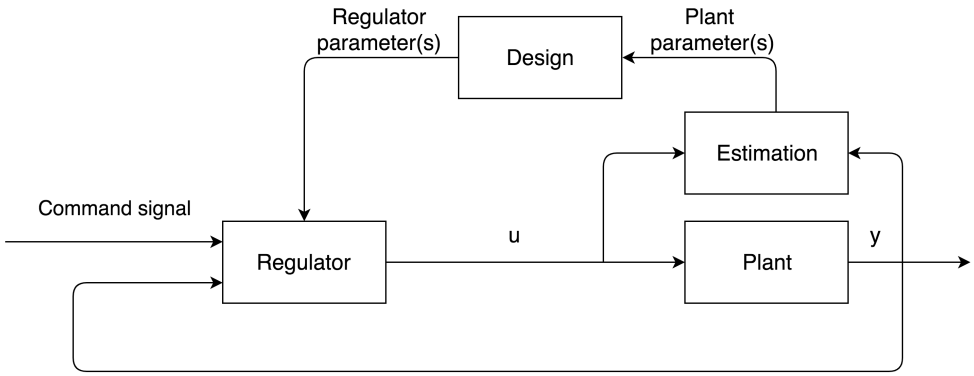

Figure 3.9: Block schema of the self-tuning regulator with reference to [16]

A self-tuning regulator (STR) also uses u and y for an estimation; however, it does not really estimate a model but "plant parameters", which are given to a design, where the regulator parameters are calculated with these plant parameters.[16] Uncertainty has to be included in the

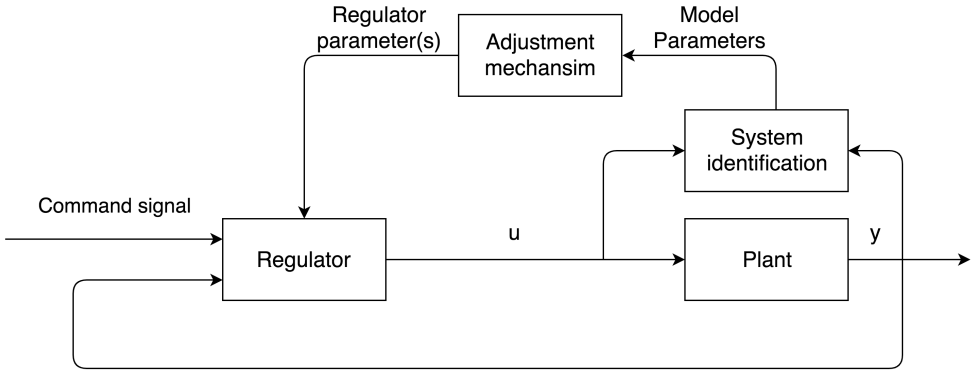

Figure 3.10: Block schema of MIAC regulator with reference to [11]

calculation.[11] Any system-identification-method can be used, and there is no real standard. Common methods are the least squares method, maximum likelihood and extended Kalman filtering. Most of the identifications can be described as pole placement designs, so the optimal

design is found through a variation of poles of the transfer function.[18][16] The direct identification of the transfer function parameters is "the most straightforward approach"[16]. In difference to the MIAC it is not necessarily the only correct way. It could only estimate single parameters of the system, and then like in gain scheduling 3.1 only adapt in regard to this parameter. In contrast to MIAC the parameters are considered as true, so there are no considered uncertainties of the calculation. This is called the certainty equivalence principle. [16]

MIAC was introduced in 1958 by R. Kalman, as mentioned in the introduction of this thesis. As he chose the approach of this method, which rests on identification of the system, the computers were not potent enough to handle these kinds of algorithms in a justifiable amount of time. [16]

The biggest advantage of MIAC and the self tuning regulator is that they can be applied to a very wide range of systems and parameter-configurations, because the systems do not have to be known a priori. However, the stability analysis is more difficult in comparison to other methods, because of the wide possible variation of the parameters. [18] [16] There are several ways to solve this problem, such as a given initial transfer function where there is only need for a small change of the parameters, or the numbers of poles and zeros of the transfer function are given. All these simplifications lead to a better stability analysis, but fewer variations of possible systems handled by this method.

With this knowledge it is noticeable that MRAC could be considered as highly concertized SOAS or MIAC, with an uncertainty of the model of 0. To define MIAC as direct or indirect, first there has to be found a definition for both. While Sastry and Bodson point out that MRAC is the direct version of SOAS and SOAS is always indirect[18], Åström and Gregory share the opinion of a weaker definition, of SOAS being able to be indirect and direct without being a full MRAC.[16][5] Gregory even points out MRACs ability to be indirect, but the definition of an indirect MRAC estimating its model leads to the definition of as SOAP or MIAC. Here the definition Åström is taken: direct adaptive controllers update the controller parameter directly, without first calculating a completely new model. This is also the case when the model is only given in a certain way and to a certain degree and not completely re-estimated but only updated, not fundamentally continuously, but in discrete time intervals. Indirect adaptive controllers only adapt indirectly and always include a step of designing in their procedure. Åström does not give concrete information about where to draw a line between being direct or indirect.[16] Sastry and Bodson do so by defining MRAC as direct and everything being SOAS as indirect.[18]

3.6 Stochastic Control

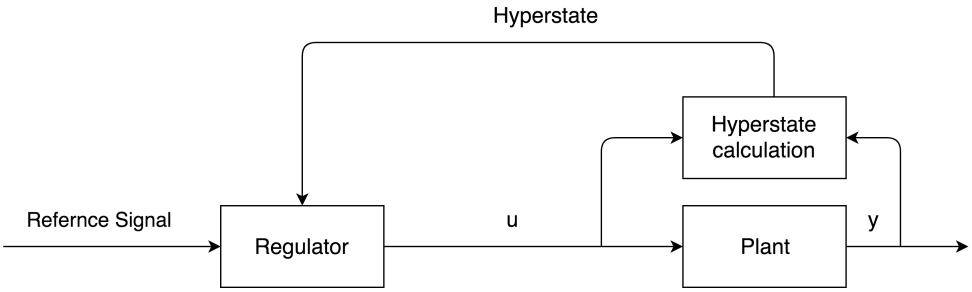

Figure 3.11: Block schema of the stochastic control with reference to [18]

Stochastic control is the only non-heuristic but pure theoretically-based method. The system and environment are described as stochastic models and the aim of this controller is to minimize the expected value of the loss function. The only stochastic problem to be considered as solvable is the linear quadratic Gaussian problem, which can be used to tune an LQR-controller. As seen in 3.11, what was assumed as estimator in MIAC 3.5 can still be seen as such, but now it generates the conditional probability distribution of the state, called the hyperstate, which usually belongs to an infinite dimensional vector space, and shows the complexity and the reason why it is a common but a not often used approach. If there is a possible solution for the minimization the loss function it can be found with the Bellman algorithm, which is based on dynamic programming. Stochastic control is likely to drive the output to the desired behavior. If the estimation has uncertainties it will add perturbations which lead to a better estimation and the future control will be improved.

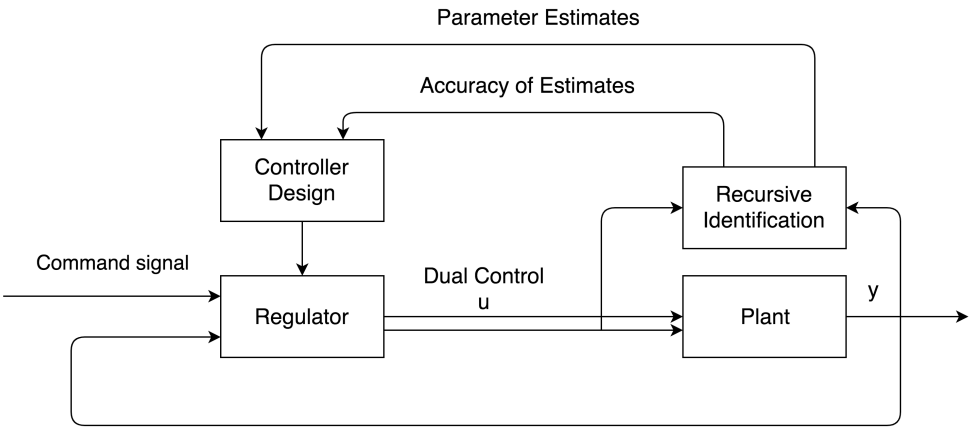

Figure 3.12: Block schema of the dual adaptive control with reference to [2]

This leads to the definition of the *dual control*, first introduced by Feldbaum in 1960, of the balance of maintaining good control and small estimation errors. Dual control theory is also represented partly by MIAC 3.5, because it is able to hand over the accuracy of the identification to the adjustment mechanism. But that is only one value because MIAC identifies only one system. Self tuning regulators do not give estimation accuracy of the parameters by definition. Contrary, stochastic control does so, as it is the nature of a stochastic estimation of state(s). As seen in Figure 3.12, the dual adaptive control transfers parameters and accuracy to the control design; stochastic control accomplishes this all in the calculation of the hyperstate which is the combination of both, hence it is considered to be the prime example of the dual adaptive control [2][18][16].

4 Implementation

In this chapter, the requirements to the adaptive controller based on the requirements of the MIDRAS project are pointed out and on that base the choose and the specific implementation of the M-MRAC is explained. Furthermore two reference models for the M-MRAC are shown and compared.

4.1 Requirements

Because the design of an M-MRAC is part of MIDRAS-project, the requirements for the simulation and the real system are adapted here to this. The structure for controlling the z-value (which can be interpreted as altitude and will be called that from now on) is a cascaded loop. A first P-controller runs hat 40Hz for altitude and a second PI controller runs at 200Hz for the resulting demanded vertical velocity.
In the future it is considered to make it a triple controller with a third acceleration controller.

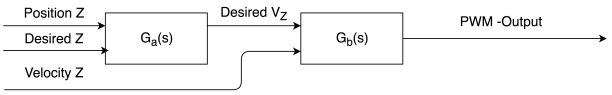

Figure 4.1: Block schema showing the double controller of the altitude control of MIDRAS

The requirements for the MRAC are:

- Minor optimization of the current second controller, but without endangering robustness or stability.
- Intercept fast abrupt changes in the system most likely due to an increase of the gravity force by adding mass.
- No need for changing the current controller, which has limitations.

This leads to the following implementation:
Because of the first requirement, the MRAC is implemented around the inner controller. So as input of the reference model the output of the outer controller is taken as u_c . This leads to the reference model representing the inner controller and the plant. The second requirement shows the need for implementing the modified version of MRAC, so M-MRAC, because as shown in 3.4.1, merely a gain adaptation is not able to quickly handle abrupt changes. As input u_c is able to change rapidly, the MIT rule is chosen. Another reason is its good adaption rate. As the quadrocopter is an unstable system, it could be considered to use SPR because of its proven stability. The third requirement leads to the adaptive gain θ being multiplied by the output of the second controller and not by the input.

4.2 M-MRAC

With the second controller (PID-controller) seen in the discrete time domain, the calculations are linear

$$e = u_{innerloop} - y \tag{4.1}$$

$$u_{outerloop} = e_n \cdot k_p + \sum_{i=1}^{n} e_i \cdot dt + (e_n - e_{n-1})/dt \tag{4.2}$$

with n being the time steps since start of the controller and dt being the time since the last step. In this case it does not matter if we write

$$e_\theta = (u_{innerloop} - y) \cdot \theta \tag{4.3}$$

and

$$u_{outerloop} = e_{\theta n} \cdot k_p + \sum_{i=1}^{n} e_{\theta i} \cdot dt + (e_{\theta n} - e_{\theta n-1})/dt \tag{4.4}$$

or

$$u_{outerloop} = \theta \cdot (e_{\theta n} \cdot k_p + \sum_{i=1}^{n} e_{\theta i} \cdot dt + (e_{\theta n} - e_{\theta n-1})/dt) \tag{4.5}$$

As the versions are mathematically indistinguishable, the second version is chosen for practical reasons, because in the real system there will be limitations for all characteristics and behaviors of the controller. This could lead to restrictions in the adaptation. The adaptation will have its

own restriction rules, which are not applied in the simulation, but to the real system for safety reasons.

For the simulation the software MATLAB/Simulink was used with the aerospace toolbox, which provides a quadrocopter simulation. It uses the *Parrot* Mambo drone, a small-sized drone with a weight of 63 g. The real system test will be flown on small drones of 194 g. It would be possible to identify the plant transfer function of the real quadrocopter, but the simulation provides an in-depth simulation of plant, environment characteristics, sensor simulation and the influence of the regulation of the other degrees of freedom. Default altitude control of the toolbox is a simple PD controller, with the D-part not calculating the derivation of the altitude. On the real system, the vertical velocity is taken from the integration of an accelerometer, which is also provided by the simulation. The default structure of the simulation is replaced by the cascaded control (P and PI) architecture of the real system. As the simulation uses a kick-start the motors run at full speed for a certain amount of time. There are two take-off helpers, one overwriting output signal with the kick-start and one setting the real velocity as reference velocity for overcoming the adaption mechanism to integrate an error, without having an impact on the system, while the kick-start is active. The take-off helper works deeply in the model, so could not be shut off, but there is no need for it, as we will not analyze the start but mid-flight. The resulting MATLAB/Simulink workspace can be seen in Figure 9.1. The C++ code can be seen in the appendix.

4.3 Design of the Reference-Model

Because of the special requirements of the MIDRAS project seen in Chapter 1, the criterion is to minimize error e between the system without extra forces of the net, due to extra mass and absorption of the velocity of caught drones, and the system with these extra forces. Based on this approach, the system of the drone without the later attached weight is taken as reference system. A function with one zero and two poles is chosen as transfer function so it represents a transfer function without many perturbations. One zero and two poles lead to less overshoot and shorter settling time than a two-zero one-pole system (PID-controller). A second order system is a commonly known transfer function for simulating an abstracted optimal drone.[17] According to MATLAB, the transfer function of the real system was more likely to have a zero,

due to unknown factors. The transfer function found by the MATLAB Identification toolbox for one zero and two poles is $G_1(s)$, which is rated with an accuracy of 89.34%.

$$G_1(s) = \frac{2.995 \cdot s + 55.78}{s^2 + 8.042 \cdot s + 55.78}$$

Because of the desire to improve the performance of the default system, a second function, a two-pole null-zero function $G_{2'}(s)$ with an accuracy of 88.72%, was calculated.

$$G_{2'}(s) = \frac{45.69}{s^2 + 6.65 \cdot s + 45.69}$$

$G_{2'}(s)$ has an overshoot of 15.44%, rise time of 0.33 s and settle time of 0.77 s. $G_{2'}(s)$ was modified by hand, so it now had an overshoot of less than 0.1% and therefore the same rise and settle time of 0.42s. This function is called $G_2(s)$.

$$G_2(s) = \frac{90}{s^2 + 17 \cdot s + 90}$$

The accuracy of MATLAB-system-identification is defined as follows:

$$\beta = 100 \cdot (\frac{\|y - \hat{y}\|}{\|y - mean(y)\|})$$

with β as accuracy, y as system output and $\hat{y}$ as the output of the transfer function. Because the reference system of an adaptive controller neither has to be the exact same as the original system, nor does it have to be necessarily optimal, but could even be considered to impair the system response or not fit the original system, the impact of the mathematics on the accuracy with which MATLAB rates the systems will not be discussed. It is sufficient that the found transfer function $G_1(s)$, when considering the requirements of 4.1, is an improvement of the system, and has characteristics that seem feasible for the drone.
$G_1(s)$ originally was

$$\frac{2.995 \cdot s + 55.78}{s^2 + 8.042 \cdot s + 55.77}$$

but with that, an infinite lasting input of 1 would not lead to the output 1 but to the output 1.0018. Therefore the 55.77 in denominator was changed to 55.78, because only when the constant part of denominator and nominator are the same, an input of 1 for an infinite amount of time will

lead to an output of 1. So it was set 0.01 higher, to solve this desired characteristic of the real system. And as pointed out, there is no need to take the exact same model as calculated by MATLAB, as long as it solves the requirements it is made for.

The result of $G_1(s)$ applied to the value of the velocity desired by the outer controller can be seen in Figure 4.2, and of $G_2(s)$ in Figure 4.3.

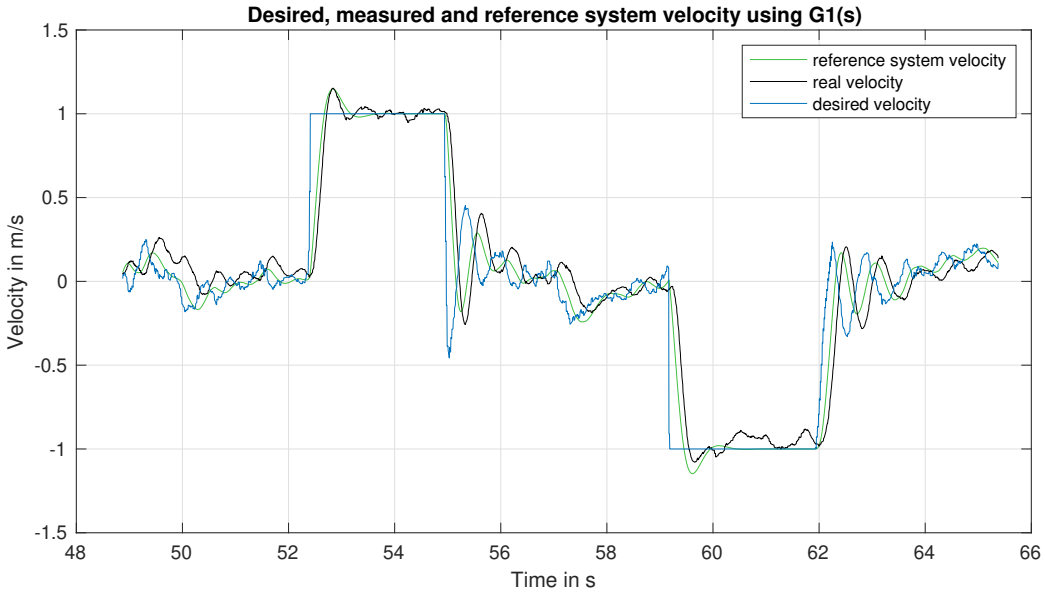

Figure 4.2: Comparison of the real velocity of the test-quadrocopter with the estimated velocity of the reference model $G_1(s)$ with the desired Velocity

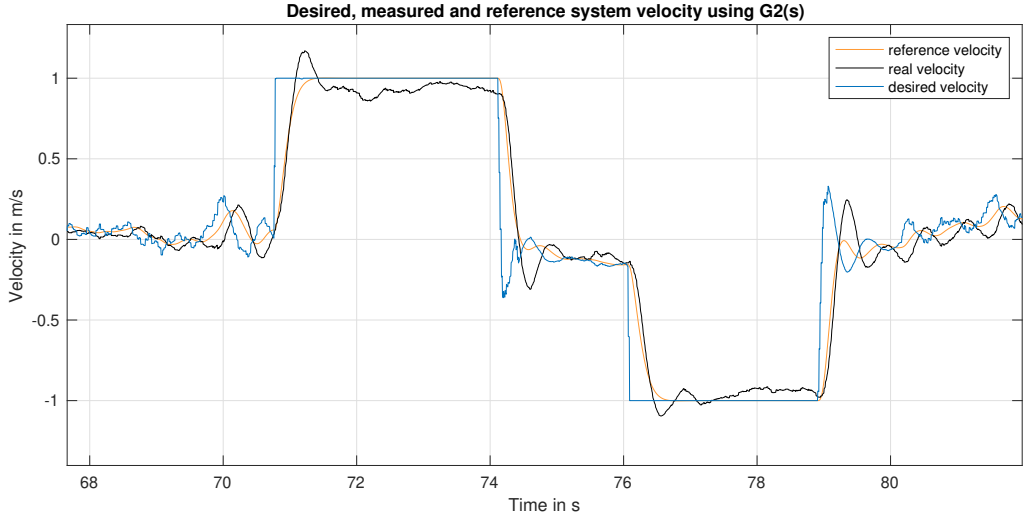

Figure 4.3: Comparison of the real velocity of the test-quadrocopter with the estimated velocity of the reference model $G_2(s)$ with the desired Velocity

The reference output of $G_1(s)$ compared to the real output is a little bit faster and with fewer perturbations. So with an adaptive controller this should have a stabilizing effect on the real output and increase its reaction time somewhat, but also keep the overshoot, with the danger of even increasing it. $G_2(s)$ has nearly no perturbations, starts quicker but becomes slightly slower halfway. However, it does not overshoot. To characterize the transfer function, rather than taking the time-continuously calculated characteristics, it is more reasonable to take the characteristics of the time-discrete implementation, which will also be run later on the real system. This implementation in C++, which can be seen in Chapter 9, shows the following characteristics of the transfer function:

Characteristics of the reference models $G_1(s)$, $G_2(s)$ and the real system for a step response from 0 to 1:

	rise time	settling time	overshoot
$G_1(s)$	0.24s	0.645s	16.1991 %
$G_2(s)$	0.42s	0.42s	0.0008 %
real system	0.30s	1.4s	16.972%

Finding the characteristics of the real system is more complex because of the input velocity and real velocity not being really 0 at the start of the rise.

For taking the transfer function from the frequency domain to the discrete domain, $G_1(s)$ will be implemented as a combination of P, I and D controllers.

Therefore the general function $G_g(s)$ of the form

$$G_g(s) = \frac{a \cdot s + b}{s^2 + c_1 \cdot s + c_2}$$

will be discussed. The numerator side can be seen as a summation of a D-controller with gain a, and a P-controller with gain b like shown in Figure 4.4. For reasons of stability, the derivation in the c++ implementation in Chapter 9 is averaged over the last 10 values which are the last 50 milliseconds with the code running at 200Hz.

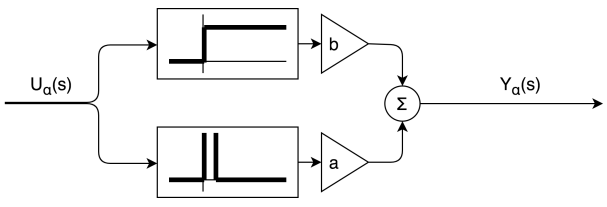

Figure 4.4: Block schema of the numerator side of the function $G_g(s)$

$1/(s^2 + c_1 \cdot s + c_2)$ can be considered as a double integration with a feedback loop amplified with the gain c_2. Furthermore, the value of the first integral-controller is amplified by the factor c_1 and added to the feedback.

This is illustrated in Figure 4.5. To get the multiplication of these two parts, they are connected in series resulting in $Y_\alpha(s) = U_\beta(s)$. As the commutative law applies to the multiplication of two functions of the frequency domain, it does not matter if $Y_\alpha(s) = U_\beta(s)$ or $U_\alpha(s) = Y_\beta(s)$. This is the basis for the implementation in C++ in 9 .

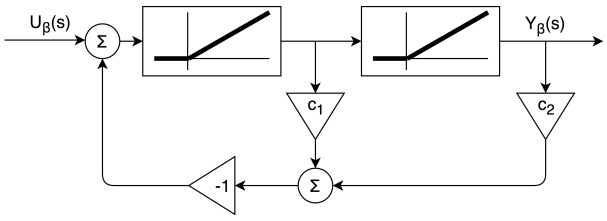

Figure 4.5: Block schema of the denominator side of the function $G_g(s)$

4.4 Practical Issues

MRAC with the MIT rule works best in a pure positive (excluding 0) system. A zero-symmetrical system is also possible, but has some issues. For example, if a reference system has the value 0, the θ-gain of the adaptation stays the same, whatever the real value varies or not, because the error is multiplied with the value of the reference system. Consequently, when the real value rises linearly from 1 to 3, but the reference system and the desired value sink linearly from 1 to -1, the error e will increase linearly. However, θ, because of the multiplication (see Equation 3.9), does not rise linearly and even stops rising when the reference value reaches exact 0. Because a controller will sink further than 0, this does not lead to failure of the MRAC, but the adaptation is not optimal. With the MIDRAS-copters, the range of velocity is theoretically from $-\infty$ to $+\infty$ and y_{out} from to controller from -50 to $+50$. The unit of the output will not be discussed, as it has no consequence for the explanation and therefore implementation. The output 0 stands for the amount of throttle, which leads to hovering with no vertical velocity. Considering this, the MRAC works fine with the default setup. But as soon as the system changes, 0 is not the hovering throttle anymore. Normally, this is not a problem, because θ is the gain for the input, where 0m/s is still the symmetric middle, even with a changed system. However, because θ is

taken as the gain of the output of the controller and not the input (see requirement 3 in 4.1), this then becomes a problem.

A quick example. Consider a drone with an attached weight, which hovers at a output of 20. When the weight is detached, v_{real} becomes positive and v_{ref} negative. This leads to e (with $e = v_{real} - v_{ref}$) being positive, therefore $e \cdot V_{ref}$ becomes negative and because $\dot{\theta} = e \cdot v_{ref} \cdot -\gamma$, θ will increase. The increasing θ multiplied with the desired velocity smaller than zero will lead to a lower desired velocity and therefore to a faster decreasing u. But multiplied to the (still) positive output, because it started at 20 and not 0, this becomes bigger, which works against the desired change of letting the drone sink. Practically, this leads to the system oscillating stronger and stronger. To overcome this, the output is no longer considered to have a range from -50 to $+50$ but from 0 to 100. So with a too fast system the gain should decrease, and with a too slow system it should increase. Therefore $e \cdot v_{ref}$ is just changed to $e \cdot \left|v_{ref}\right|$, since with a time-discrete PID-controller $G_{PID}(a \cdot x) = a \cdot G_{PID}(x)$ and therefore the same behavior is kept as multiplying by the input of a linear system.

Alternatively the maximum possible sink rate of $-9.81 \frac{m}{s^2}$ could be defined as velocity 0, for the step of the adaptation, and therefore the whole vertical velocity becoming positive. With the limitation of the first controller, it would be enough to define the minimum limit as 0.

5 Evaluation

In this chapter, the M-MRAC is tested on simulated and real drones. Therefore the altitude, vertical velocity and vertical acceleration of several flight maneuvers in combination with changes of system parameters and changes of controller parameters are evaluated.

5.1 Simulation

For all the simulations $\gamma = 7$ and for M-MRAC $k_p = 4$, $k_i = 2$ and $k_d = 0,3$ were chosen.

5.1.1 Flight Dynamics

The first evaluation is the one of the behavior of the simulation without changes to the system, with and without M-MRAC with MIT rule, for a change of the altitude from the reference 1.50 m to 1.90 m.

Figure 5.1 shows the altitude with and without M-MRAC. The characteristics of rise and settling time and overshoot are not much different. It can be seen that the system with M-MRAC has less oscillation because the reference system of the velocity is much smoother than the real system. This applies to $G_1(s)$ and $G_2(s)$ and is just as expected because in this case, the first controller is the restricting part. Therefore the normal system, as well as both M-MRACs, can easily follow the desired altitude given by the first controller, but the M-MRACs can pursue this without as many perturbations as the default system.

In the next simulation, seen in Figure 5.2, the k_p of the first controller has risen from 0.5 to 2 which leads to a high overshoot and oscillation of the inner controller - the parameters of which were not designed for this quick change. Now it becomes obvious that the M-MRAC is stabilizing the system by adapting the PI controller, which is parameterized for much slower changes. It is still not changing the rise time, although now overshoot and settle time are clearly

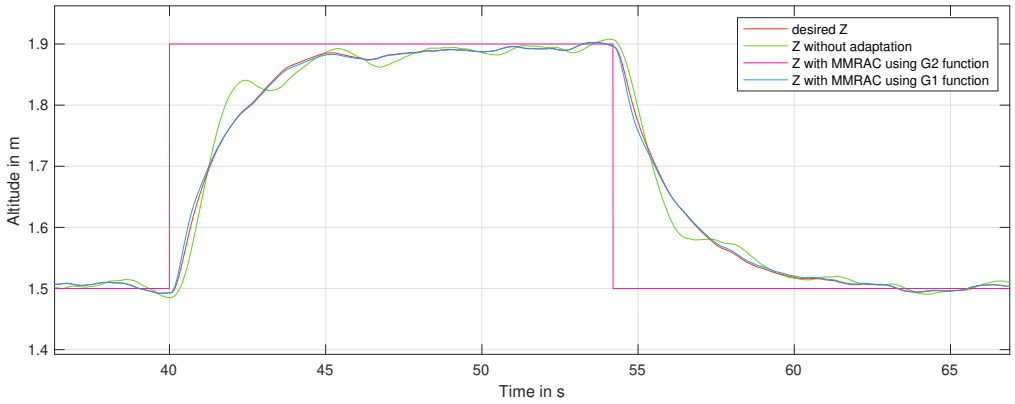

Figure 5.1: Results of the simulation of rising to 1.90 m from 1.50 m with and without M-MRAC $G_1(s)$ and $G_2(s)$ at the default PID values of the first controller

better because of the missing oscillating of the system. M-MRAC is a good way of adjusting poorly designed controllers, although the results will always be better when using an optimally designed controller. [1]

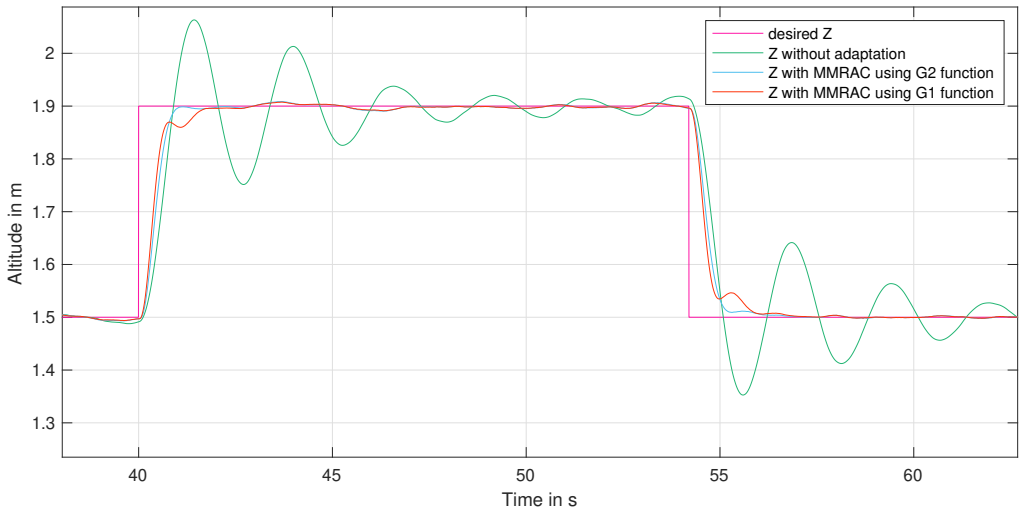

Figure 5.2: Results of the simulation of rising to 1.90 m from 1.50 m with and without M-MRAC using $G_1(s)$ and $G_2(s)$ at the higher PID values of the first controller

The comparison of the velocity of the default system to $G_1(s)$ can be seen in Figure 5.3 and to $G_2(s)$ in Figure 5.4. It shows that the velocity follows the specific reference velocity. This leads to $G_1(s)$ having some oscillation just before reaching 1.90 m, just as expected. This clarifies the oscillation seen in Figure 5.2 just before reaching 1.90 m and the faster rise time in comparison to the use of $G_2(s)$. The small perturbations, mainly seen in the velocity of the system with

MRAC, are not different from the perturbations when the default system is steady at the target altitude.

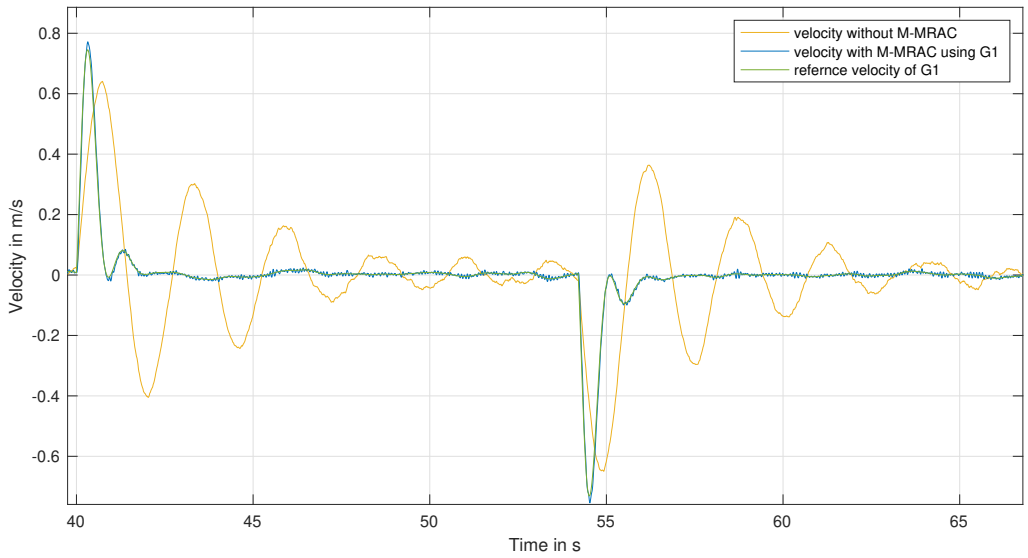

Figure 5.3: Results of the simulation showing the vertical velocity of rising to 1.90 m from 1.50 m. with and without M-MRAC using $G_1(s)$ at higher PID values of the first controller

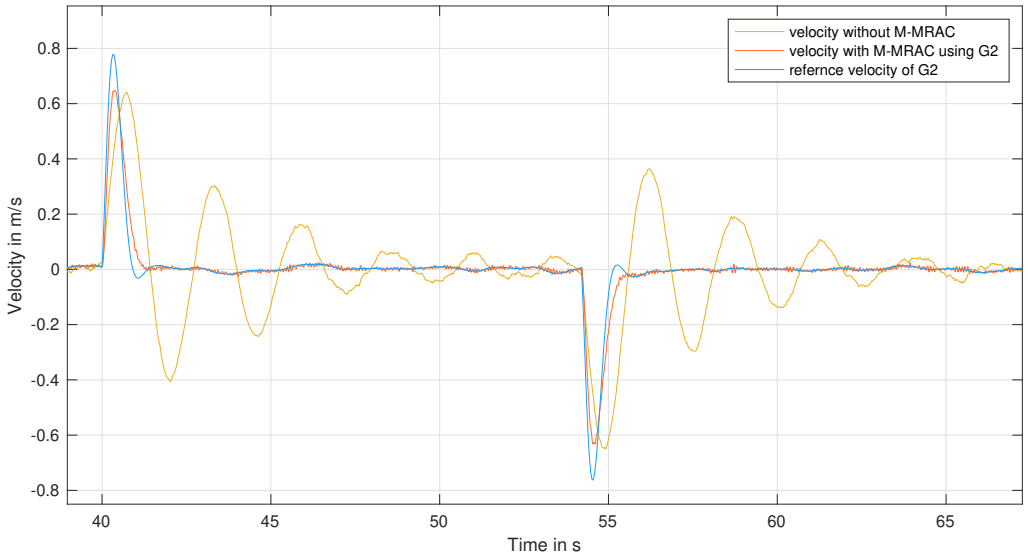

Figure 5.4: Results of the simulation showing the vertical velocity of rising to 1.90 m from 1.50 m with and without M-MRAC using $G_2(s)$ at higher PID values of the first controller

In Figure 5.5 the same maneuver is flown with and without M-MRAC using $G_1(s)$ and with a normal and double mass. The corresponding velocities can be seen in Figure 5.6. The second PID-controller was now tuned, becoming a well parameterized controller without overshoot.

This was done to prove three points.

First, the adaptation is taking away the oscillation with the double mass system, and allows for a faster rise and settling time. The double mass system with using M-MRAC reaches midway a certain altitude 0.09 seconds before the double mass system without M-MRAC. This cannot be transferred to the rising time (reaching 0.95% of the change), due to the first controller slowing down the desired velocity when approaching the desired altitude to overcome overshoot, while the double mass system just rushes through and has a non-negligible overshoot. Therefore, with all three others, the settling time is the rise time, while the system with double mass and without M-MRAC needs 3.00 more seconds before reaching settle time.

Second, at second 41 there is a slight oscillation in the system with normal mass and M-MRAC which is not noticeable in the system without M-MRAC. The exact oscillation is in the double mass system with M-MRAC. This is seemingly a characteristic of the reference model. This shows that the M-MRAC does not necessarily make a system optimal, but only makes it behave like the reference model. In this simulation, the requirements of the use would decide, if the faster-rising M-MRAC system or the non-oscillating normal system would be desired. The optimal way would be to design a new reference system that is fast and does not oscillate. This was the intention of $G_2(s)$, but for the demonstration of M-MRAC keeping the characteristics of the reference model, $G_1(s)$ was chosen for this simulation.

Third, without mathematical proof, it can be seen that the difference between the two M-MRAC systems is smaller when the altitude is lowered again, although the double mass system without M-MRAC seems not to have improved its overshoot etc. This is because θ has to be integrated, which needs time and situations where the reference and real system differ in a noticeable way, which is not given with the drone staying at 1.50 m.

5.1.2 Answer to System Change

The second complex of evaluation with the simulation is the response to sudden system changes. In Figure 5.7 a sudden mass increase to the factor 1.5 is shown because the real system is also tested with approximately 1.5. With M-MRAC using $G_1(s)$ the lowest point was 1.472 m and with using $G_2(s)$ 1.469 m with a desired altitude of 1.5 m. Without MRAC it was 1.34 m. There is no major difference between $G_1(s)$ and $G_2(s)$. The associated velocity can be seen in Figure 5.8. It will not be possible to prohibit any kind of fall of velocity because there has to be a difference

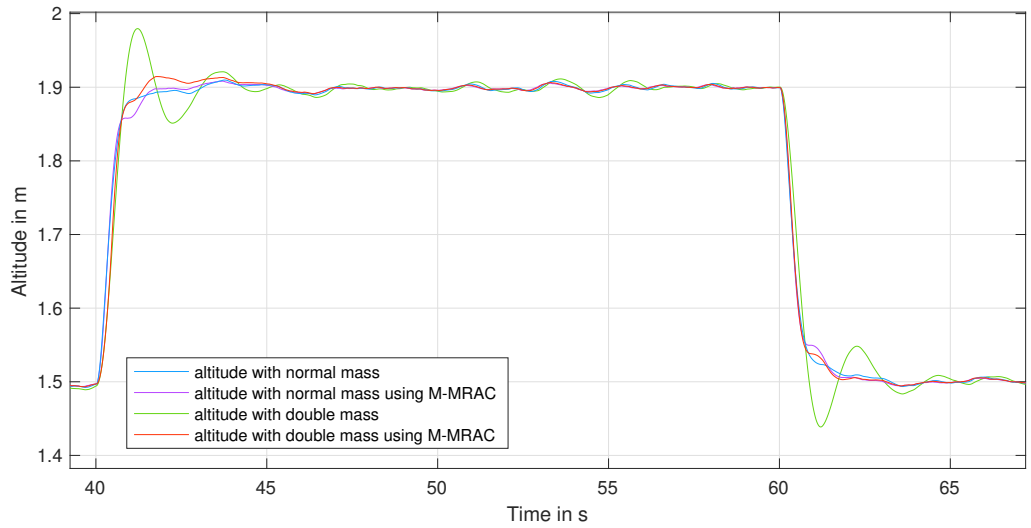

Figure 5.5: Results of altitude of the simulation of rising to 1.90 m from 1.50 m with and without M-MRAC using $G1(s)$ with a mass of 63 grams and 126 grams

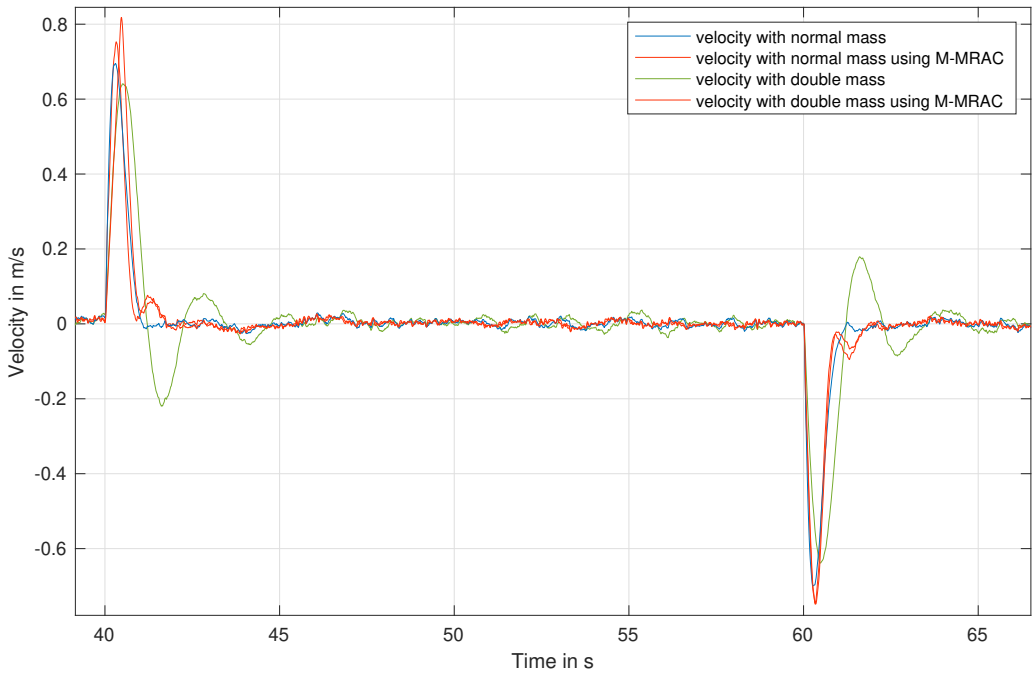

Figure 5.6: Results of velocity of the simulation of rising to 1.90 m from 1.50 m with and without M-MRAC using $G1(s)$ with a mass of 63 grams and 126 grams

for an adaptive controller to adapt. With variation of the parameters of the adaptive controller, only the peak and return time of the system can be changed.

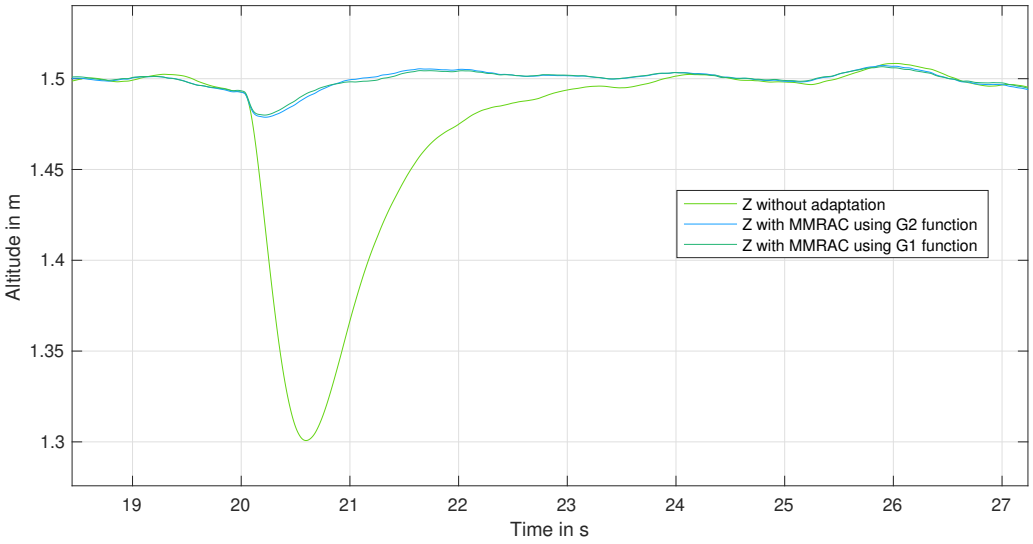

Figure 5.7: Results of simulation of holding altitude and suddenly increasing mass to factor 1.5

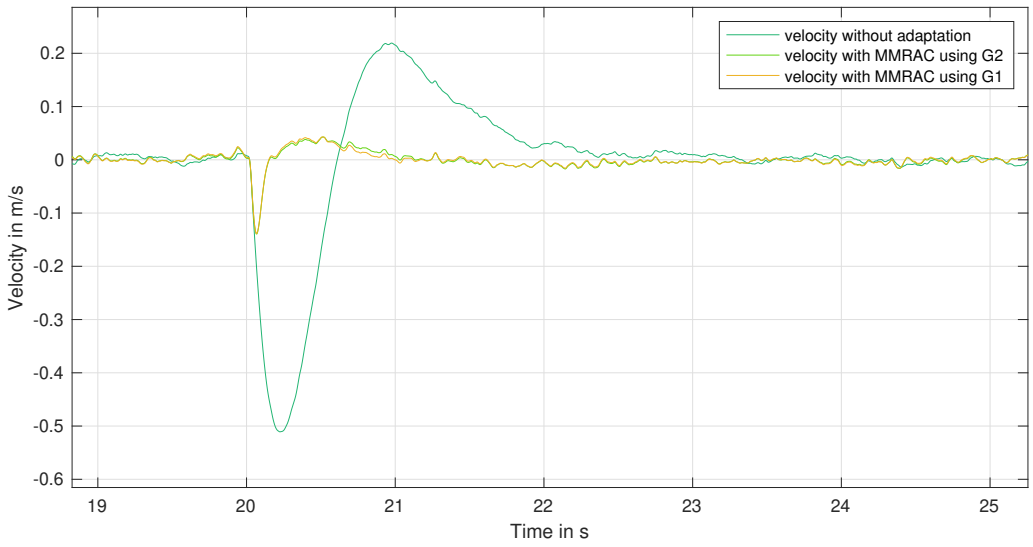

Figure 5.8: Results of simulation of the vertical velocity while holding altitude and suddenly increasing mass to factor 1.5

The next evaluation of the simulation is the sudden mass increment seen in Figure 5.9. It has to be said that with the actuators reaching their limit, 2.4 is the maximum factor of its own weight the virtual drone is still capable of flying with. But even starting at the maximum simulation height of 5 meters, the system without M-MRAC still crashed with mass increased to factor 2.4.

Therefore it cannot be said whether it would ever get back to the altitude from where it started or not. Therefore the second extraction with a factor of only 2.3 is more meaningful. Only $G_1(s)$ is shown, due to $G_2(s)$ having no visible difference in that plot. The same behavior of the drone with to high mass will be seen with the real system in the next section.

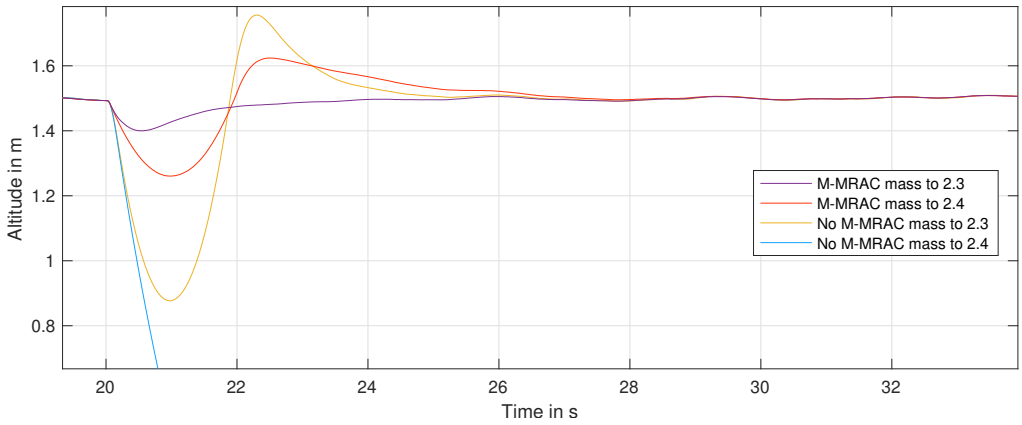

Figure 5.9: Results of the simulation of holding altitude and suddenly increasing mass to factor 2.3 and 2.4 with and without M-MRAC

The last evaluation is a short comparison of MRAC and M-MRAC. In chapter 3.4.3 the need for a modified version of MRAC was pointed out, because of the late response of MRAC due to the integration of the error. Figure 5.10 shows the same values as in Figure 3.8, but additionally the MRAC with the gain 4 was modified to an M-MRAC. The modified version like as expected a quicker response and an overall less error.

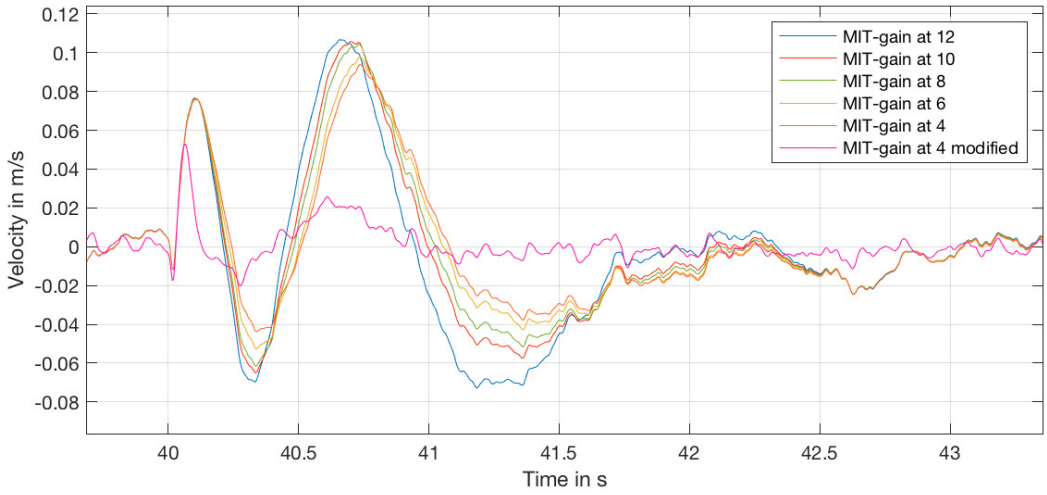

Figure 5.10: Error between reference velocity and real with MRAC with MIT rue adapted with gains from 4 to 12. The height was increased at 40 s which leads to acceleration and deceleration.

5.2 Real System

Both evaluations, the flight dynamics and the answer to system changes, are now tested on the real system. $G_2(s)$ is used as the transfer function of the reference system for the real system, because the simulations showed slightly better results with it than with $G_1(s)$. The system itself is a small drone developed by the University of Wuerzburg with the mass of 194 g, four RS1106 motors from *EMAX* and a Cicada-30A-4in1 ESC from *Sunrise*. The micro-processor is a STM32F407VG from *STMicroelectronics* and the barometer a MS5611 from *Measurement Specialties*. The altitude is calculated by a Kalman-filtered combination of the internal barometer and the external, optical based, OptiTrack™-system, which tracks with eight infrared cameras four infrared-reflecting balls mounted on the drone. With the visual information the pose is calculated. For the mass-increment two clamps with a mass of 52 g and 120 g are used. They are clipped to the bottom of the quadrocopter by hand.

Figure 5.11: Real system drone with two clamps for mass increment

5.2.1 Flight Dynamics

The behavior of the velocity when increasing in altitude with and without M-MRAC with a default mass without change is shown in Figures 5.12 and 5.13. With M-MRAC, the desired velocity is reached faster but with an overshoot of about 8.3%. A comparison of settle and rise time with the definition introduced in Chapter 4 is not useful, because the system without M-MRAC never reaches the 0.95% threshold, but has a maximum of 0.94 m/s with a desired velocity of 1.00 m/s. Therefore, the threshold for the definition of the rise and settle time is set to 90%. As the desired velocity has at every point a change of $\neq \infty$, the starting point for the measurement of rise and settle time is set to the point where the desired velocity starts to rise.

With this definition, the system without M-MRAC has a rise and settle time of 0.28 s and with M-MRAC 0.17 s. This speaks for a much higher acceleration. The acceleration can be seen in Figure 5.14. It was lowpass-filtered with a passband frequency of 20 Hz to filter micro vibrations. The acceleration was not measured but mathematically created a posteriori out of the velocity data. There is no limit implemented for the acceleration, which would be passed over by the M-MRAC implementation: the system without M-MRAC would have the possibility to reach the same accelerations. When rising, the acceleration of the system with M-MRAC is 1.9 times bigger compared to the system without. When sinking it is only factor 1.2 lower. Taking the two sinking velocities of -11.4 $\frac{m}{s^2}$ and -13.3 $\frac{m}{s^2}$ into account, it should be clear that the mathematical calculation of acceleration out of measured velocity has some bias and/or uncertainty. Otherwise the drone would sink faster than it would only by gravity, which is not possible, because the drone does not have any reverse thrust.

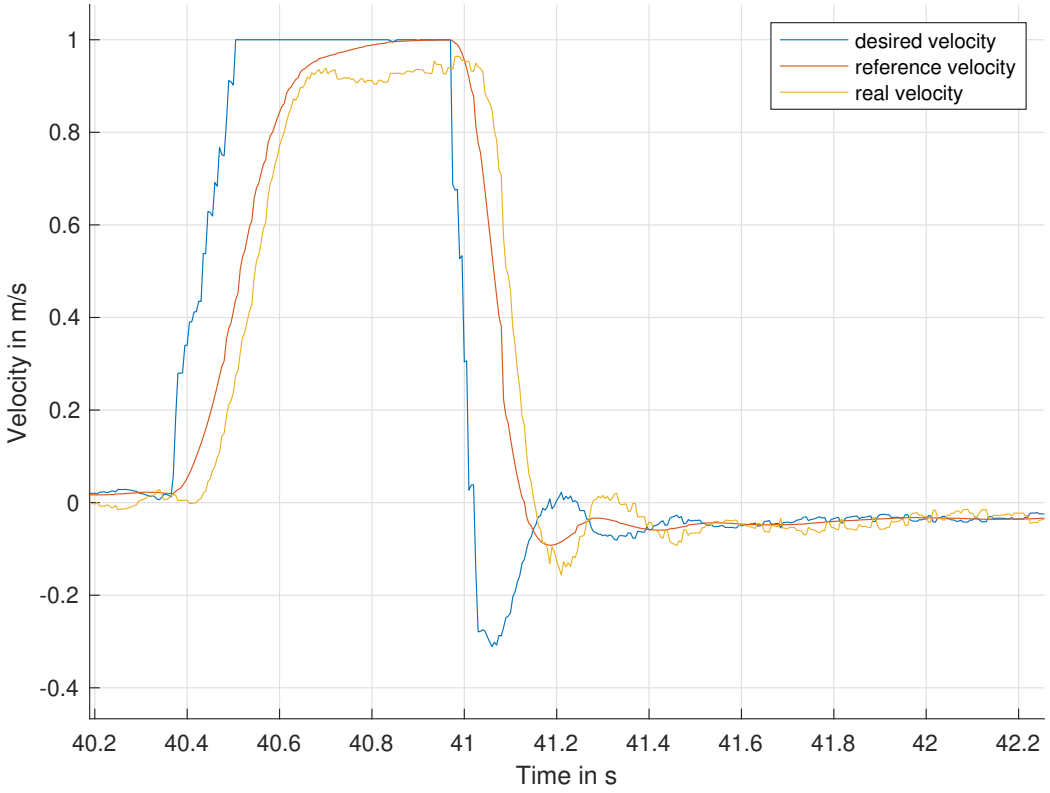

Figure 5.12: Velocity of the real quadrocopter without using M-MRAC when rising altitude

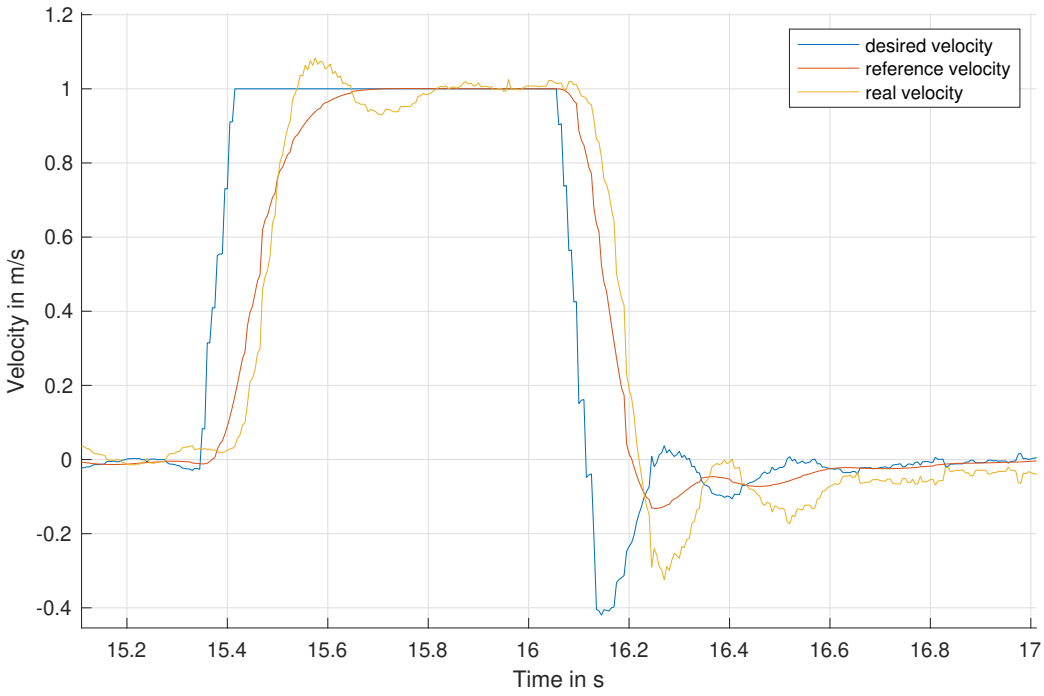

Figure 5.13: Velocity of the real quadrocopter with using M-MRAC when rising altitude

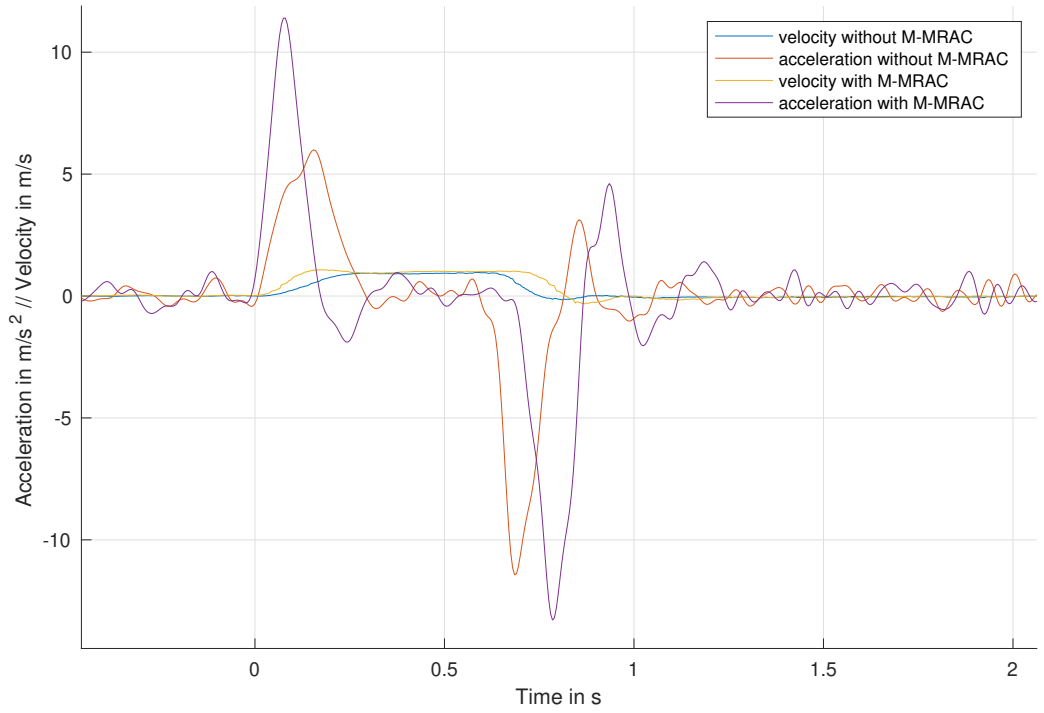

Figure 5.14: Velocity and acceleration of the real quadrocopter with and without using M-MRAC when rising altitude

5.2.2 Answer to System Change

Figure 5.15 shows the behavior of the quadrocopter with and without using M-MRAC when 27% of the weight of the drone is attached with a clip. The short rise at the beginning is due to the method of attaching manually a clip, which leads to a short push. From the altitude 0 just after attaching the clip the system without adaptation has a rise time of 3.71 s, and a settle time of 11.8 s, with the lowest point at 0.283 m below the starting point and a overshoot of 16.7%. The system with M-MRAC has a rise time of 1.12 s, a settle time of 2.92 s, the lowest point at 0.197 m below the starting point and an overshoot of 13.3%.

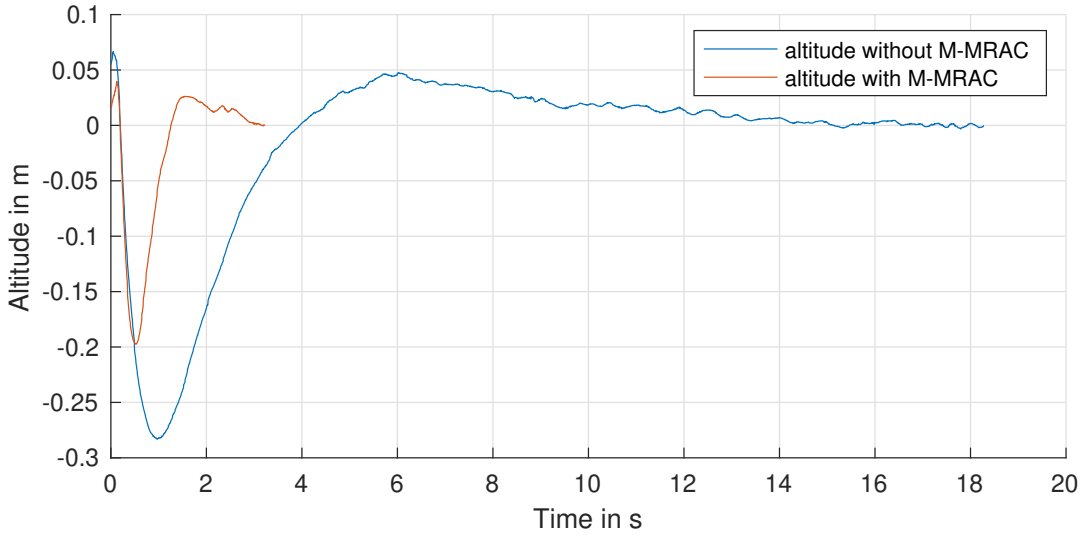

Figure 5.15: Results of the real quadrocopter of holding altitude and suddenly increasing mass to factor 1.27 with and without M-MRAC

Figure 5.16 shows the behavior of the drone when increasing the mass to factor 1.62. The system without M-MRAC touches the ground, starting from the maximum possible height of the experimental setup of 2 meters. Even when it touches the ground, it stays there, not able to return back to 2 m. The system with M-MRAC has a rise time of 0.74 s, a settle time of 2.36 s, the lowest point at 0.41 m below the starting point and an overshoot of 27%. The main difference to the simulation is, that when increasing to factor 1.62 and to 1.27 respectively, the system with M-MRAC has no real difference until at about 0.4 seconds. That is an indicator of the real system being more lethargic than the simulation. Figure 5.17 shows the vertical velocity when increasing mass with factor 1.27 without M-MRAC and Figure 5.18 shows it with M-MRAC. It can be clearly seen that with M-MRAC the real velocity reaches the reference behavior after

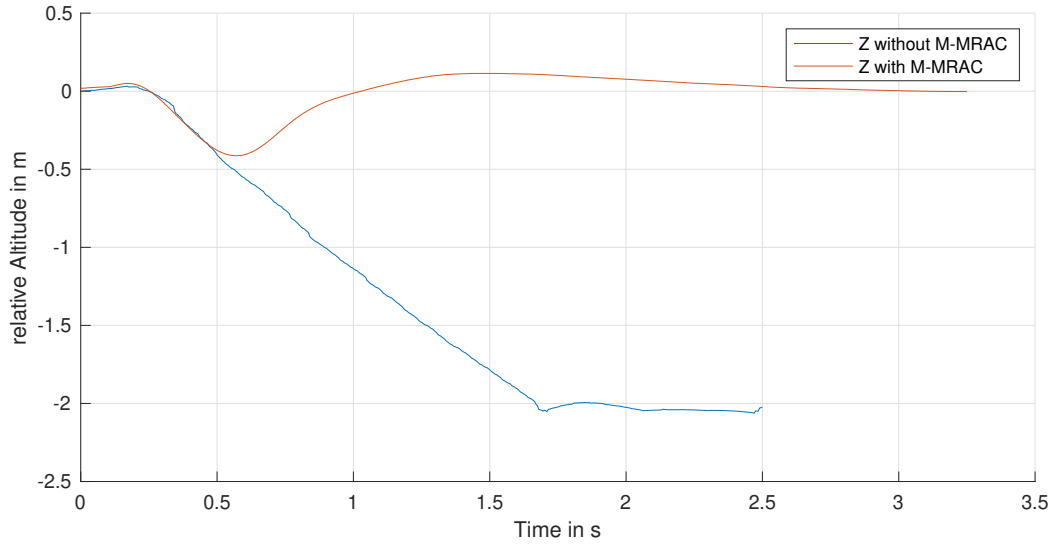

Figure 5.16: Results of the real quadrocopter of holding altitude and suddenly increasing mass to factor 1.62 with and without M-MRAC

about 0.6 s and then converts together to 0; without M-MRAC the real velocity only reaches the reference and desired velocity when it is about 0, meaning the drone reached the desired altitude.

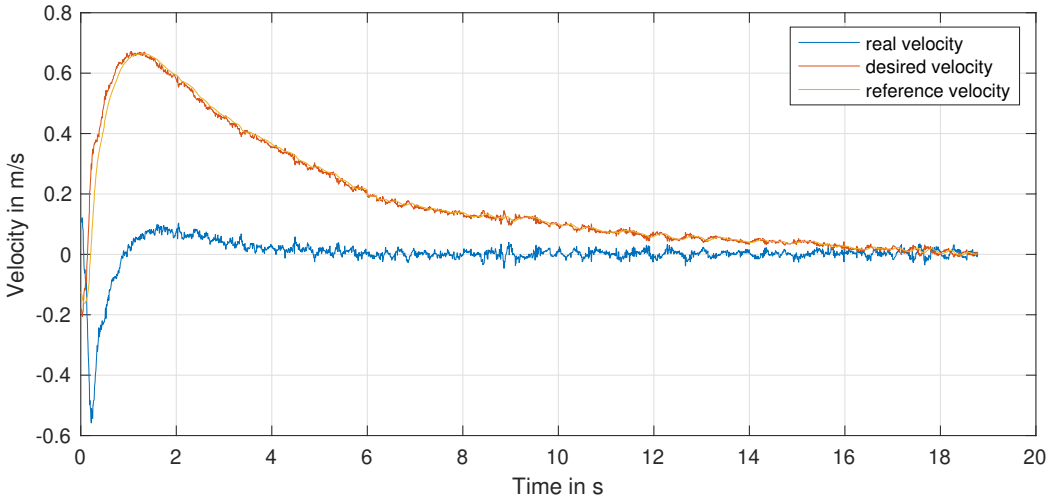

Figure 5.17: Vertical velocity of the real quadrocopter of holding altitude and suddenly increasing mass to factor 1.27 without M-MRAC

The last point is the flight dynamics of the changed system. Figure 5.19 shows the system behavior with the 52 g clamp attached and Figure 5.20 the behavior with the use of M-MRAC. It can be seen that the system without M-MRAC does not reach the desired maximum velocity in the maximum possible time without flying out of the experimental setup. It is inert in compar-

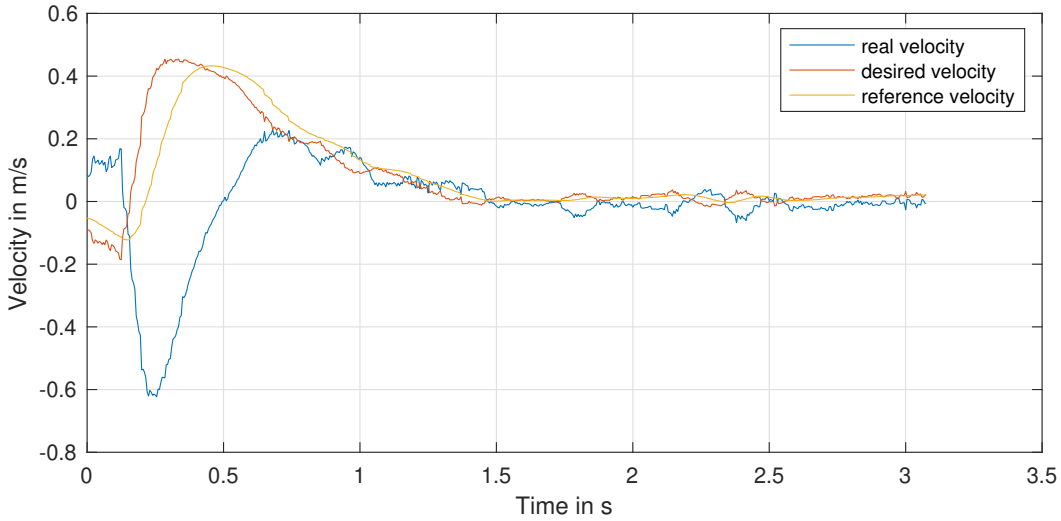

Figure 5.18: Vertical velocity of the real quadrocopter of holding altitude and suddenly increasing mass to factor 1.27 with M-MRAC

ison to the behavior without an attached weight from Figure 5.12. Without M-MRAC the rise time increases from 0.28 s to 0.39 s. With M-MRAC the rise time remains 0.17 s . So the system without M-MRAC had a 39% increases in rise time, while the system with M-MRAC, considering the accuracy, had no increase. However, the overshoot of the system with M-MRAC rises from 8.3% to 11.8% , which means that rise and settle time differ here. The settle time of the system with M-MRAC and attached weight is 0.25 s, which is still faster than both rise times of the system without M-MRAC.

During the flights there was a noticeable behavior of the attitude control, that should be mentioned, but cannot be shown due to the lack of data. Because of the faster rise and sink rates when using M-MRAC, often the throttle of the single motors came to the limitations of the allowed maximum power output of the ESCs. This leads to problems with the attitude control, because it had no more possibilities to reduce the throttle when at minimum or increase it when at maximum power. Although this has no impact on the altitude control, there has to be an adjustment to avoid this behavior. This could be an easy limitation of the maximum and minimum throttle by the altitude control, to give the attitude control room to adjust, even at the maximum sink rate.

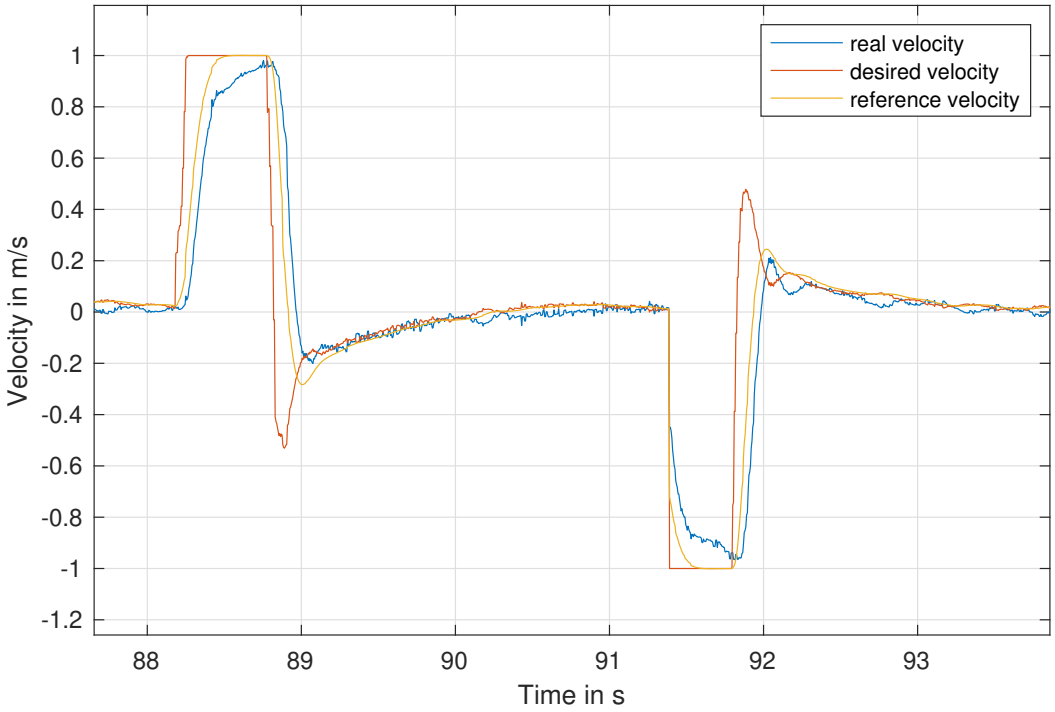

Figure 5.19: Vertical velocity of the real quadrocopter when changing altitude with a mass factor of 1.27 and without using M-MRAC

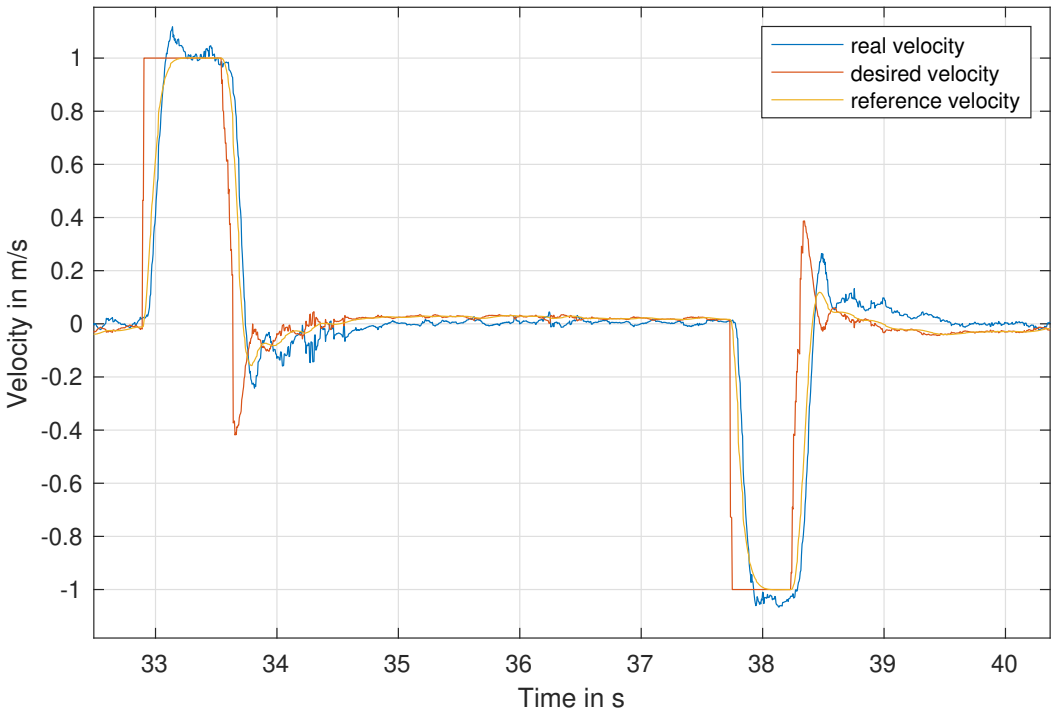

Figure 5.20: Vertical velocity of the real quadrocopter when changing altitude with a mass factor of 1.27 and with using M-MRAC

6 Conclusion

In this thesis an M-MRAC for altitude control of a quadrocopter with a cascaded controller structure was implemented and tested. The adaptive controller was implemented, so it did not rely on the limitations of the inner controller. This makes it more applicable also to other than the to the specific quadrocopter. The results of the evaluation of the simulations as well as the real system show an improvement of behavior when changing the mass of the system up to the point where the normal system was not able to fly properly anymore. This shows that an M-MRAC is suitable as an extension for the MIDRAS-copter's altitude control.

Without any change of system settings, the normal behavior was affected in a way that can be traced back to the M-MRAC, leading to faster rise times but higher overshoots, even with the reference system having no overshoot. This needs further investigation of the parameters and the gains of M-MRAC and for the reference model itself. The simulation showed that M-MRAC can lead to desired flight behavior, but it also showed that it is not able to rebuild the exact reference model without deviation. It has to be clear that for M-MRAC to have an impact, there first needs to be a difference to the reference model.

Therefore M-MRAC is not an alternative to tuning the default controller properly to the desired behavior.

The implementation now has to be tested on several different systems, for confirmation of its flexible use. First tests were positive, but are outside the scope of this thesis. The focus is on the large MIDRAS-copters that will carry the net to catch drones, and therefore have the impact of a rapidly changing mass. Furthermore, as explained in the last paragraph of Chapter 5.2, there is the need for some modifications regarding the interplay of the altitude control-complex with other controllers.

A further step for the adaptive altitude control is the implementation of a simple identification of the maximum possible rise time of a system, and adapting the reference model to this rise time. This would be the first step of a simple MIAC.

For adaptive control in general, the other 5 degrees of freedom of the pose of the quadrocopter could be controlled with adaptive controllers, to a have the complete drone following an optimal virtual reference drone.

7 Abbreviations

CMRAC	Combined Model Reference Adaptive Control
ESC	Electronic Speed Control
LQR	Linear-Quadratic Regulator
MIAC	Model Identification Adaptive Control
MIT	Massachusetts Institute of Technology
M-MRAC	Modified Model Reference Adaptive Control
MRAC	Model Reference Adaptive Control
PID	Proportional Integral Derivative
SOAS	Self-Oscillating Adaptive Systems
SPR	Stability Proof Rule
STR	Self-Tuning Regulator
UAV	Unmanned Aerial Vehicle

8 Bibliography

[1] Khan Afaque and Shanmukha Swamy. *Modified MRAC based on Lyapunov theory for improved controller efficiency*. 2016. doi: 10.1109/ICACDOT.2016.7877735.

[2] Vladimir Bobal, Petr Chalupa, and Petr Dostal. Adaptive dual control of nonlinear liquid system. *IFAC Proceedings Volumes*, 40(13):483–488, 2007. doi: 10.3182/20070829-3-ru-4911.00075.

[3] Zachary T. Dydek. Adaptive control of unmanned aerial systems, 2010.

[4] Rudy Gianto. Review of self–tuning controller and its application in electrical power system. *international journal of scientific & technology research*, 4:137–142, 2015. ISSN 2277–8616.

[5] Dr. Irene M. Gregory. Fundamentals of adaptive control, 2012.

[6] Jingqing Han. From pid to active disturbance rejection control. *IEEE Transactions on Industrial Electronics*, 56(3):900–906, 2009. ISSN 0278–0046. doi: 10.1109/TIE.2008.2011621.

[7] Priyank Jain and Madhav J. Dr. Nigam. Design of a model reference adaptive controller using modified mit rule for a second order system. *Advance in Electronic and Electric Engineering*, 3:477–484, 2013. ISSN 2231–1297.

[8] Suresh K. Kannan, Girish V. Chowdhary, and Eric N. Johnson. Adaptive control of unmanned aerial vehicles: Theory and flight tests. pages 613–673, 2016. doi: 10.1007/978-90-481-9707-1_61.

[9] Ibrahim Kaya and Mustafa Nalbantoglu. Cascade controller design using controller synthesis. 2015. doi: 10.1109/ICSTCC.2015.7321265.

[10] Thomas R. Kurfess and Mark L. Nagurka. *High Gain Control System Design with Gain Plots*. Pittsburgh, 1991. doi: 10.1184/R1/6489980.v1.

[11] Hausi Müller and Norha Villegas. *Runtime Evolution of Highly Dynamic Software*. Springer, 2014. doi: 10.1007/978-3-642-45398-4_8.

[12] Richard M. Murray, Zexiang Li, and Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. ISBN 978–0–849–37981–9. doi: 10.1201/9781315136370.

[13] Rupali J. Pawar and Bhagsen J. Parvat. Design and implementation of mrac and modified mrac technique for inverted pendulum. pages 1–6, 2015. doi: 10.1109/PERVASIVE.2015.7087168.

[14] Rupali J. Pawar and Bhagsen J. Parvat. Mrac and modified mrac controller design for level process control. pages 217–222, 2018. doi: 10.1109/INDIANCC.2018.8307981.

[15] Prof. Dr. Karl Johan Åström. Adaptive control – a perspective. *IFAC Proceedings Volumes – Adaptive Systems in Control and Signal Processing*, 23, 1990. doi: 10.1016/S1474-6670(17)52687-7.

[16] Prof. Dr. Karl Johan Åström and Björn Wittenmark. *Adaptive Control*. Addison–Wesley, Amsterdam, 1989. ISBN 978–0–201–09720–7.

[17] Murillo F. Santos, Leonardo M. Honorio, Exuperry B. Costa, Edimar J. Oliveira, and Joao P. P. G. Visconti. Active fault–tolerant control applied to a hexacopter under propulsion system failures. pages 447–453, 2015. doi: 10.1109/ICSTCC.2015.7321334.

[18] Shankar Sastry and Marc Bodson. *Adaptive Control – Stability, Convergence, and Robustness*. Prentice Hall, London, 1989. doi: 10.1093/imamci/1.1.27.

[19] Marc Steinberg. Historical overview of research in reconfigurable flight control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 219 (4):263–275, 2005. doi: 10.1243/095441005X30379.

[20] Yahui Xiao, Yufei Fu, Chengfu Wu, and Pengyuan Shao. Modified model reference adaptive control of uav with wing damage. pages 189–193, 2016. doi: 10.1109/iccar.2016.7486724.

9 Appendix

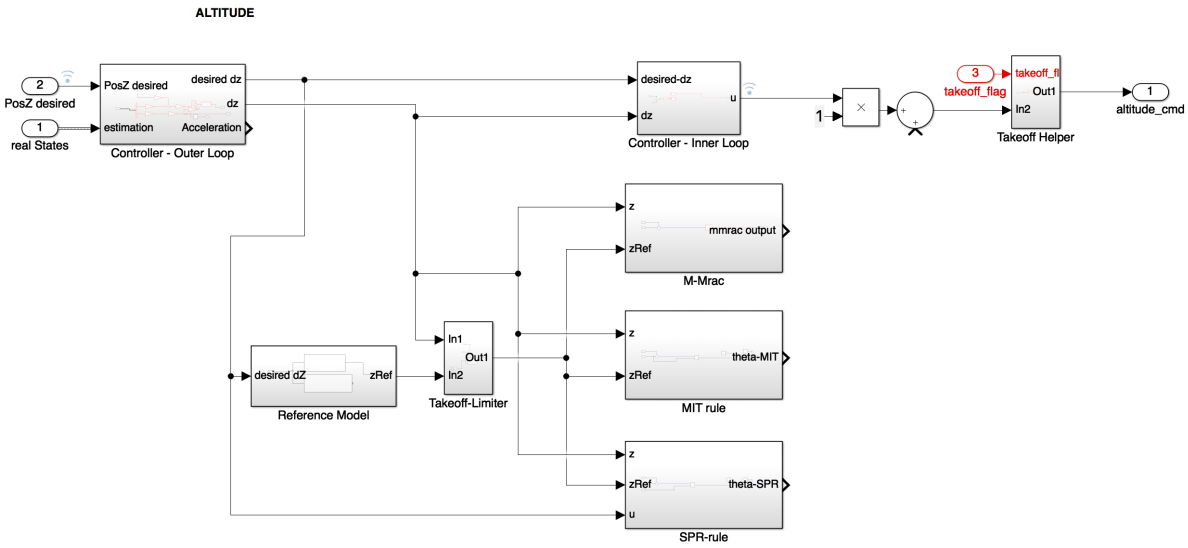

Figure 9.1: MATLAB/Simulink implementation for the altitude control using the two controllers. MRAC with MIT and SPR can be attached to the multiplication and when additionally the M-MRAC block is attached to the unconnected SUM-block the MRAC becomes M-MRAC

C++ code representing the reference system for 4.3

```
/* Storage of the last 10 derivatives. Because of a queue
system the position of the latest data has to be remembered
*/
int startWith = 9;
double dLast[10]={0,0,0,0,0,0,0,0,0,0};
/* auxiliary variables for the beta function */
```

```
double feedbackBeta=0;
double i1=0;
double i2=0;
/* transfer Function of form (a*s+b)/(s^2 + c1*s + c2)*/
double a=0;
double b=90;
double c1=17;
double c2=90;

double lastData=0;
/* calculating the next reference velocity with the input
data. Dt is the time since the last data */
double nextRef(double data, double dt) {
    //---------PD-Controller----------------
    if(dt<=0)dt=0.005;
    dLast[startWith]=(lastData-data)/dt;
    lastData=data;
    double d=0;
    for(int i=0;i<=8;i++)
    {
        d+=dLast[startWith--];
        if (startWith == -1)
            startWith= 9;
    }
    d+=dLast[startWith];
    d/=10;
    d*=a; //factor for D
    double p=b*data; //factor for P
    double outputAlpha=p+d;
   //--------- double integrateion-controller----------------
```

```
    double inputBeta=outputAlpha-feedbackBeta;
    i1+=inputBeta*dt;
    i2+=i1*dt;
    feedbackBeta=i1*c1+i2*c2;
    return i2;
    }
```

C++ code representing the M-MRAC adaptation for 4.2

```
double maxAdaptationIntegration = 15;

double refV = 0;

//For MMRAC
double factorLinear = 2.5;
double factorIntegral = 1.0;
double integralAdpatation = 0;

//Gain with MIT
double maxAdaptationIntegrationMIT =5;
double integralMIT=1;
double mitGain = 2;

double getNewPWM(double setVelZ, double VelZ, double dt, bool
inFlight, double pwm) {
    if (!inFlight)
        return pwm;

    //getting next reference velocity
```

```
   double refV = nextRef(setVelZ,dt);
    //error of reference velocity to real Velocity
    double e = (VelZ-refV);
    double e_MIT=e*refV;
    if(refV<0){
     e_MIT*=-1;
    }
//using MIT rule and M-MRAC-concept of calculating new pwm
double e_MIT =refV;
    double a = integralMIT - e_MIT * dt * mitGain;
    double b = integralMIT - e_MIT * dt * mitGain;
    if (!(a > maxAdaptationIntegrationMIT || b <=0.1))
        integralMIT += e_MIT * dt * -1 * mitGain;

//Integration of the error fpr M-MRAC
   if (!(integralAdpatation + e * dt > maxAdaptationIntegration
                    || integralAdpatation + e * dt < -
maxAdaptationIntegration))
        integralAdpatation+= e * dt;
//integral + linear part
    double addM_MRAC = integralAdpatation * factorIntegral +
e * factorLinear;


    return ((pwm+50)*integralMIT - addM_MRAC)-50;
}
```