Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Engineering - Robotics
30.0 credits

AN ADAPTIVE CONTROL SYSTEM BASED ON PID, I^{2}PD AND RLS, A SIMULATED DESIGN FOR UAVS

Johan Gärtner
jgr12001@student.mdh.se

Philip Johansson
pjn11001@student.mdh.se

Examiner: Mikael Ekström
Mälardalen University, Västerås, Sweden

Supervisor: Alessandro Papadopoulos
Mälardalen University, Västerås, Sweden

June 8, 2017

Abstract

During this master thesis an adaptive control system for angular velocity was developed. The approach focuses on radio controlled quadrotors and thereby targets a computational efficient solution. The adaptation is divided into two steps. First, an Integral Squared Plus Time Delay (I^2PD) model estimation is done through the Recursive Least Squares (RLS) algorithm. Second, control parameters for the quadrotors Proportional Integral Derivative (PID) controller are found by using empirically modified Tuning Rules (TRs) which are based on the I^2PD model. Simulation results show the benefits of an adaptive tuning algorithm in regards of the time consuming process of manually tuning a PID controller as well as robustness against both predictable and unpredictable system changes. This includes actuator degradation, battery consumption, changes in weight or inertia etc. However, it is apparent that the tuning algorithm demands an arbitrary initial approximation to be stable at initialization.

Keywords *- RLS, PID Controller, UAV, quadrotor, adaptive control, I^2PD, simulation*

Table of Contents

1 Abbreviations **4**

2 Introduction **5**
2.1 Background . . . 5
2.2 Hypothesis . . . 6
2.3 Problem formulation . . . 6
2.3.1 Research questions . . . 6
2.4 Expected outcome . . . 7
2.5 Limitations . . . 7

3 Related works **8**
3.1 The PID controller . . . 8
3.2 Adaptation and automatic tuning techniques . . . 8
3.2.1 Manual and automatic tuning . . . 9
3.2.2 Gain scheduling . . . 9
3.2.3 Adaptive control . . . 9
3.3 Control algorithms and system knowledge . . . 10
3.3.1 Process identification . . . 10
3.3.2 Extremum seeking . . . 10
3.3.3 Neural network . . . 10
3.3.4 Fuzzy logic . . . 11
3.3.5 Evolutionary algorithm . . . 11
3.3.6 Tuning rules . . . 12
3.3.7 Model predictive control . . . 12
3.4 Motivations . . . 13

4 Method **14**
4.1 Requirements . . . 14
4.1.1 Software simulation . . . 14
4.1.2 System performance . . . 14
4.1.3 Hardware . . . 14
4.2 Simulation . . . 14
4.3 Evaluation . . . 15

5 Technical description **16**
5.1 Closed loop system . . . 16
5.2 Model identification . . . 16
5.2.1 Least squares . . . 17
5.2.2 Recursive least squares . . . 18
5.2.3 Computational simplification . . . 19
5.3 FOPDT model . . . 19
5.3.1 FOPDT to RLS . . . 20
5.3.2 Tuning rules . . . 21
5.4 I^2PD model . . . 21
5.4.1 I^2PD to RLS . . . 22
5.4.2 Tuning rules . . . 22
5.4.3 Empirically modified tuning rules . . . 23

6 Simulation environment **24**
6.1 Mathematical model . . . 24
6.2 Control setup . . . 25
6.3 Manual tuning . . . 26

7 Simulation results **28**
7.1 RLS initialization and convergence 29
7.2 Test 1 - normal operation 30
7.2.1 Z-axis 30
7.2.2 X-axis 32
7.3 Test 2 - modified inertia 33
7.3.1 Z-axis 33
7.3.2 X-axis 35
7.4 Test 3 - battery degradation 37
7.4.1 Z-axis 37
7.4.2 X-axis 39
7.5 Test 4 - smooth flying 41
7.6 Summary 42

8 Discussion **44**
8.1 RLS convergence 44
8.2 Model accuracy 45
8.3 Real-time capabilities 45

9 Conclusion **46**

10 Future work **47**

11 Acknowledgment **48**

References **52**

12 Appendix 1 **53**
12.1 Test 1 - normal operation 53
12.1.1 Z-axis FOPDT 53
12.1.2 Z-axis I^2PD 55
12.1.3 X-axis I^2PD 57
12.2 Test 2 - modified inertia 59
12.2.1 Z-axis FOPDT 59
12.2.2 Z-axis I^2PD 61
12.2.3 X-axis I^2PD 63
12.3 Test 3 - battery degradation 65
12.3.1 Z-axis FOPDT 65
12.3.2 Z-axis I^2PD 67
12.3.3 X-axis I^2PD 69
12.4 Test 4 - smooth flying 71

1 Abbreviations

UAV Unmanned Aerial Vehicle

ESC Electronic Speed Controller

DoF Degree of Freedom

MPC Model Predictive Control

MI Model Identification

IAE Integrated Absolute Error

RLS Recursive Least Squares

MISE Mean Integrated Squared Error

MAE Mean integrated Absolute Error

PID Proportional Integral Derivative

VTOL Vertical Take Off and Landing

LiPo Lithium-Polymer

GA Genetic Algorithm

EA Evolutionary Algorithm

ZN Ziegler-Nichols

IFT Iterative Feedback Tuning

FOPDT First Order Plus Dead Time

I^{2}PD Integral Squared Plus Time Delay

MT Manual Tuning

NN Neural Network

FL Fuzzy Logic

TR Tuning Rule

V-REP Virtual Robot Experimentation Platform

2 Introduction

The following work has been done as a master thesis at Mälardalen University Sweden. The work is connected to the science of unmanned multi-rotors and deals with the field of on-line adaptation and optimization of the parameters for a PID controller. To understand the entire content, it is beneficial to have some basic knowledge in the science of control theory. The work has focused on creating an adaptive control system and evaluate the performance in a simulation environment. To perform the evaluation, the algorithm has been implemented on a quadrotor which was provided by the simulation environment.

2.1 Background

The science of quadrotors is a popular, growing topic with several unexplored possibilities. The field has a continuously growing number of features and applications, the popularity is increasing as the research is expanding. The initial quadrotor platforms were built in the beginning of the 1900s and were the first invented Vertical Take Off and Landing (VTOL) vehicles [1]. Unfortunately, the technology was abandoned during its birth due to the systems inability to maintain attitude and entering into unbridled oscillations. The instability required technical solutions not available at the time. The limitations demanded a new generation of quadrotors which used more advanced electronics, sensors and some kind of embedded solution for control. The science of control systems was an answer. Control theory is an comprehensive subject with many challenges. In general, a system needs to read a reference signal such as a desired set point and act according to the desired control objective. It includes tracking the reference signal, rejecting disturbances, minimizing energy consumption etc. Also, undesired conditions have to be prevented in time. Consequently, it is favorable to be aware of a mathematical model which represents the actual dynamic of a quadrotor. System knowledge is an essential key to designing and calibrating a robust control system with the desired behavior. As the topic was growing, and with the knowledge of predicting a quadrotors behavior, it was possible to simulate system responses of different situations. Several kinds of controllers could be tested, and provide a more comprehensive understanding of their parameters and their outcome in behavior [2]. However, as time passed, new questions arose and new technologies emerged. As systems become more complex together with continuously rising ambitions, the challenges also grow bigger.

Over the past few years, Unmanned Aerial Vehicles (UAVs) have been widely used for a variety of civilian applications. Some of these applications require accurate guidance and control to confront the markets high demands. It is a field which is growing within the areas of emergency services, hobbies, construction overviews and many more, the opportunities are infinite. A commonly used approach to control the attitude of a UAV and obtain stability is the widely established PID closed loop controller [3]. It is well known for increasing control ability (with respect to no control system) as well as reducing or eliminating the steady state error. It has the advantage of a simple structure, good stability and high reliability. The key to make use of its abilities is the approach of tuning and optimizing the PID controlling parameters. The tuning of the process may be achieved in different ways, several approaches are based upon being aware of a mathematical model which corresponds to the system dynamics [4]. However, in practice these systems have nonlinear dynamics such as time delays and time-invariances. They are affected by different external forces such as bad weather, strong winds, unpredicted mass distribution etc. Additionally they may become affected by mechanical damage, reduced battery charge or actuator degradation. As a result from this nonlinearities, the optimal tuning parameters may vary during run-time and thereby contribute to degraded performance and possibly leading to instability. Accordingly, system changes may demand a retuning process to maintain performance. Thus, the general tuning procedure is a time consuming process which requires some insight in basic control theory. The anomalies have encouraged to much research over the years which has lead to several approaches within the field of auto tuning and adaptive control [5]. However, increased embedded intelligence usually requires more comprehensive calculations. As a result, extended CPU power is vital but not always available, and due to this reason, it is favorable to provide adaptive solutions which are computational efficient.

2.2 Hypothesis

Radio controlled multi-rotors are normally stabilized by a PID controller. The extent of stability is dependent on the tuning done on the control parameters. However, avoiding unbridled oscillations and minimizing the time of obtaining a minimum value of the steady state error is difficult with respect to different external factors [6]. The dynamics of a quadrotor may change during runtime and become affected by the environment which will require a retuning. Taking into account that several kinds of quadrotors share a similar software design containing a conventional PID controller, they won′t have the ability to adapt against such anomalies. However, adding an adaptive tuning algorithm to the PID controller will counteract these anomalies and extend robustness against unexpected events [7].

2.3 Problem formulation

A common way to ensure desirable behavior of a PID controller (tune the PID parameters) is to know the mathematical model of the system. However, tuning a PID controller for a multi-rotor is often performed manually by trial and error. The Manual Tuning (MT) process includes classical methods such as Ziegler-Nichols (ZN), Iterative Feedback Tuning (IFT) and others. Nevertheless, it is difficult to obtain optimal performance of a system meaning high robustness as well as minimizing the Mean Integrated Squared Error (MISE) and Mean integrated Absolute Error (MAE) [8][9][10]. The work of re-calibrate the tuning parameters is a time consuming process, it requires much patience and the knowledge of basic PID behavior. In addition, optimal parameters (parameters obtained by methods such as ZN, IFT, MT and others) may change due to various factors such as hardware alterations after system maintenance, decreased battery level or the use of the same control system on different physical non-identical platforms. As a result of these changes, system performance may deteriorate and demand a re-tuning to preserve optimal performance. With respect to these reasons, an adaptive controller is required. This work focuses on the dynamics of a quadrotor for which a general model is illustrated in Figure 1.

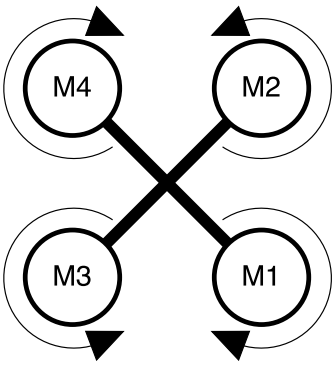

Figure 1: Quadrotor

2.3.1 Research questions

In the view of the hypothesis and problem formulation, this thesis will address the following topics:

1. *Is it possible to make a quadrotor robust against unpredictable events?*
   This includes changes in dynamics such as increased or decreased weight and inertia, battery degradation, strong winds etc.

2. *Which algorithms are suitable of performing on-line tuning of a PID controller?*
   It regards both Model Identification (MI) and tuning the controller in real-time. Accordingly, it is necessary to consider the computational complexity of different algorithms.

3. *To which extent should a plant model of a system be known to initiate an adaptive on-line tuning without producing an unstable state?*
   The approximated system model will be used to adjust the parameters of the PID controller. If the model is too far from the truth the control system will enter an unstable state. This regards both the mathematical model (the plant) as well as the approximation of its variables.

4. *How fast may an on-line adaptive tuning process minimize the MISE and MAE with respect to changed system variables?*
   If changes are made during runtime which affect the system dynamics such as changes in weight or payload. How long will it take for the algorithm to find a new good approximation.

2.4 Expected outcome

The desired outcome is to find an adaptive tuning algorithm with the ability to optimize the parameters of a PID controller. It is expected that the resulting approach will be possible to implement into a number of different quadrotor systems as a diversity of these share similar control designs (see the open-source softwares [11][12][13][14][15]). Accordingly, the resulting algorithm should be robust and at the same time able to minimize the MISE and MAE (which are described in Equations (1) and (2)) in real-time while the controller is running. Lastly, it is essential that the algorithm is able to perceive platform alterations, adapt and converge against a better condition (a decreased MISE and MAE compared to no adaptive tuning algorithm). As a consequence, it would perform better than a standard PID controller.

$$\frac{1}{n}\sum_{1}^{n}(r-y)^2 \tag{1}$$

where r = set-point and y = sensor value

$$\frac{1}{n}\sum_{1}^{n}|r-y| \tag{2}$$

where r = set-point and y = sensor value

2.5 Limitations

Scenarios may arise in real life that where not accounted for in simulation. For example the convergence could take a turn towards an erroneous model approximation. Another example might be if adaptation is activated before the quadrotor is lifted from the ground which will give the wrong system response, something that will not be accounted for. Also a fair first estimation of the system model needs to be in place for the quadrotor to be able to fly and progress its adaptation.

3 Related works

The science of control theory contains several challenges. A major task is to design an appropriate controller and utilize its full potential (obtaining control parameters which provide desirable performance). A common course is to estimate a system model and use this information to adapt the behavior of the controller. Thus, the current section covers known techniques from different fields considering identification of a system model, controllers, tuning techniques and how optimal control parameters can be adapted. The collection includes approaches from the field of artificial intelligence and adaptive techniques for continuous optimization of system performance.

3.1 The PID controller

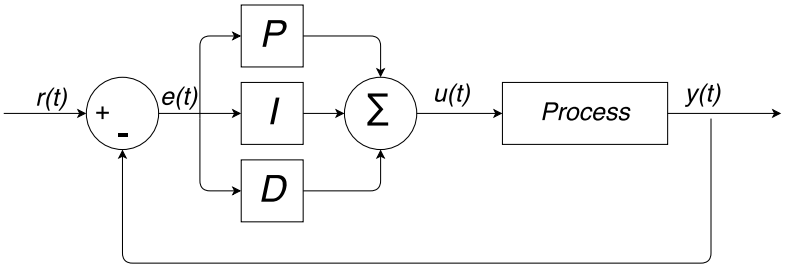

Figure 2: PID Controller

The PID controller, almost a hundred years old is derived from the most basic of controls, the linear Proportional Controller (P-Controller) where a correction is made with a set magnitude in linear relation to the error [6]. The added Integral and Derivative terms of the PID gives it drift correction and prediction. The related formula can be seen in Equation (3). An illustration of the PID controller can be seen in Figure 2 where the P, I and D components from Equation (3) are represented in three respective blocks. The I term compensates for the error history to avoid drift and works like a buffer adding the current error continuously to the same. The D term is the derivative of the error and is used as a predictor of the future to change the control behavior depending on the speed of correction, for example to dampen the correction when the system is quickly reaching its desired set-point and prevent overshoot. In practice the PI controller is the most common combination of the PID controller meaning there is no derivative part. Because of its robustness, simple structure and ease of implementation the PID controller is today the most used controller for industrial applications. The PID controller is widely used in the field of UAVs and an example of an implementation can be seen in the article [3]. A formula of a control system in time discrete form can be seen in Equation (4).

$$\begin{gathered} u(t) = K_p e(t) + K_i \int_0^\tau e(\tau)d\tau + K_d \frac{d}{dt} e(t), \\ \text{where } K_i = \frac{K_p}{T_i} \text{ and } K_d = K_p T_d \end{gathered} \tag{3}$$

$$\begin{gathered} u(t_k) = K_p e(t_k) + K_i \sum_0^{t_k} e(t_k) T_h + K_d (e(t_k) - e(t_{k-1})), \\ \text{where } T_h = \text{time between two samples} \end{gathered} \tag{4}$$

3.2 Adaptation and automatic tuning techniques

It is common that the characteristics of a process change with time or operating conditions. Adaption and automatic tuning were originally developed to deal with such anomalies. The field includes different techniques to approach these kind of difficulties, it concerns techniques such as adaptive control, automatic tuning and gain scheduling [6]. An overview is given in the following subsection.

3.2.1 Manual and automatic tuning

Although the PID controller has been around for almost a century and even with its simple structure and ease of implementation, there are still problems of finding the most suitable control parameters [9]. It is a complex task taking into account that the behaviors of different systems are non-linear. Also, note that over the years these control systems have been implemented either mechanically or electrically. Thus, tuning could require changes to the mechanical structure or the exchange of electrical components. There are several techniques and heuristics rules on how to find these parameters [16]. In later years and with the use of computers, methods for automatic tuning have been widely explored. Many of the MT techniques have been implemented to perform these calibration tasks automatically without the need of an operator. The most known tuning method is called the Ziegler-Nichols (ZN) method which comes in two versions, one which analyses the step response and another which analyses frequency response. The latter is the most used in the field of UAV's and in practice consists of zeroing the K_d and K_i terms and increasing solely the proportional gain K_p on a system until it reaches an oscillating state. The resulting gain is referred to as the unstable gain or K_u on which a scaling is applied to the rest of the system. In [16] the tuning of a speed controller for a DC motor using the ZN method is described and in [17] the authors explore an extension of the same approach. Many similar methods to the ZN rules exist like the Kappa-Tau, some of which are mentioned in [18]. In [10], an extension of the Kappa-Tau is presented where tuning of a 2-Degree of Freedom (DoF) PI controller is aiming towards low Integrated Absolute Error (IAE). The presented rules are dependent on the characterization of three parameters, i.e. the static gain K_p, the gain at a 180 degree face shift K_{180} and the frequency at this gain ω_{180}. The article [18] also describes other well known analytical methods for tuning such as Pole-placement where a systems is analyzed in the frequency domain to find a suitable damping ratio and natural frequency. In addition, the article also gives an introduction to the learning algorithms for automatic tuning such as Genetic Algorithm (GA). If a mathematical model of the system is available then TRs can be applied. These are functions that map values of a models parameters to the PID parameters. These are rarely optimal but can often give satisfactory performance or be seen as a good starting point for further tuning [9].

3.2.2 Gain scheduling

Even though it is possible to find a good linear approximation for a control design, a system might have operating points at which it changes its behavior in a way that would make a new tuning beneficial. These stages can be more or less distinct and are divided into different parts called "schedules". A schedule means a scenario where new tuning parameters are used [6]. An example can be a hydro-power plant when the water reaches different levels. At one level a new section of sluices might be present and at another a natural outflow could be connected to the pond. In this example, at each event, the system changes drastically but keeps close to linear behavior within these regions. Here gain scheduling is at its best. The variables used to identify set parameters as when the water is between these different levels are called the scheduling variables. In the introduction of [19], a system is described where gain scheduling is commonly used. Here the water level of a nuclear steam generator is controlled at different power levels.

3.2.3 Adaptive control

Gain scheduling, described in Section 3.2.2, works well for predictable situations where system parameters within certain regions are known. For more complex systems, where these are changing continuously and often exerted in an unpredictable way, another approach might be needed. Adaptive control techniques solve this problem by continuously making new system parameter estimations. In [20], RLS is used to find an estimate of the plant model. In [21], the addition of a pole placement algorithm has been developed to identify the tuning parameters of the plant. Many advantages exist over conventional methods, some of which are a better optimization as approximations in the system model are compensated for and the lack of need for MT. The approach introduced robustness as it compensates for actuator degradation, disturbances etc. Any adaptive control technique can be ran as an automatic tuning (auto-tuning) if it is ran for a constrained time

period [22], for example if it runs for at set amount of time or until desired system performance is achieved. An optimization criteria is often needed to define the goal of the convergence.

3.3 Control algorithms and system knowledge

There are many suggestions on how algorithms may be used as add-ons to a conventional control system. The purpose may be anything from obtaining a process model to automatically tune a controller. Thus, the following subsection gives an overview of algorithms and important aspects of adaptation.

3.3.1 Process identification

The tuning of a PID controller is dependent on the process. The process is a set of activities that interacts with the actuators to achieve an response of the actual entity [6]. In addition, it is possible to describe the process mathematically, the concept may be known as a plant. Many approaches on finding optimal tuning parameters are based on the plant, and due to this reason, it is mandatory to known the plants properties. A plant may be described and identified in many ways. One way is through an experiment in an open-loop, where the set-point signal is set to a constant value and the process output is measured until it has reached a steady state error. The experiment is repeated until a series of different set-points are covered. The same approach may also be applied as a closed loop experiment. RLS is one method where the parameters of a plant may be identified [23]. It is an algorithm which recursively finds the coefficient that minimizes a weighted linear least squares cost function with respect to input-output signal relations. Another identification algorithm is the Smooth Variable Structure Filter (SVSF), it is a prediction-correction method which is based on a sliding mode observer [24]. However, the RLS shows faster convergence time compared to SVSF according to the [24]. Similar approaches of model and parameter identifications are found in [21][25][26].

3.3.2 Extremum seeking

Extremum Seeking is an optimization approach for dynamical problems where limited knowledge of a system is available. It is suitable when a system has a nonlinear equilibrium map which contains a local minimum or maximum. In [27] a method is presented which optimizes a step response of a closed-loop system consisting of a PID controller and an unknown plant with a discrete version of extreme seeking. A major advantage of the algorithm is that it does not require a system plant model to improve its performance.

3.3.3 Neural network

Neural Networks (NNs) combined with back propagation is one of the research fields which have been conducted to overcome system problems such as nonlinearities [28]. It is originated in attempts to make simple models of neural activity in the brain. The algorithm endeavors to recognize patterns and carry out simple learned tasks. The computational approach is based on a large collection of neural units where each neural unit is connected with many others. A neuron contains a base function which combines the values of all its inputs together. A general model is shown in Figure 3. Neural networks typically consist of multiple layers and the signal traverses from the input, thought the layers to finally deliver the final, desired outputs. A final interesting property of the neural network is the learning procedure. It is an algorithm which makes it possible to find its own parameters in the form of weights (each line between neurons represents a weight) such that the network matches given input-output relations. The weights are typically obtained recursively, a more comprehensive exposition is found in [29][6]. The application of neural networks is widely used, it has been proposed for identification and control of nonlinear dynamics plants [30][31]. It is favorable at coping with difficulties such as system modeling and unknown parameter identification. However, common critic is the demand of a large diversity of training data, slow convergence rate, arriving at local minimum and over-fitting problems.

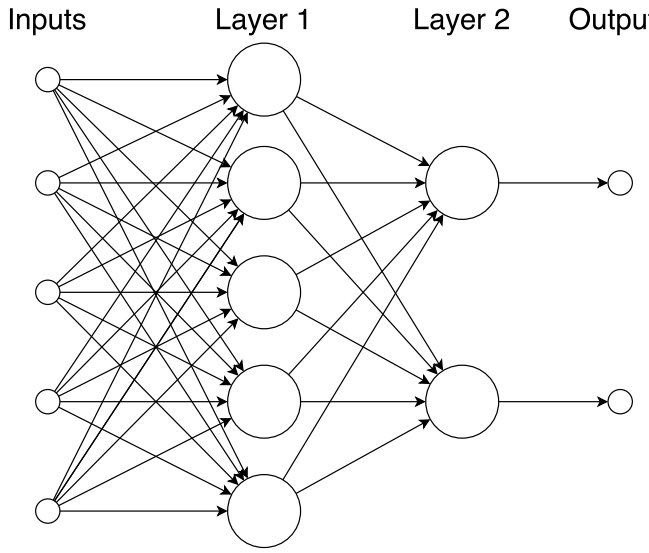

Figure 3: Neural Network

3.3.4 Fuzzy logic

Fuzzy Logic (FL) or fuzzy control is a paradigm which has received much attention during the development of artificial intelligence. The applications diversity reaches from consumer products such as washing machines, microwaves and industrial process control to cameras and medical instrumentation. The fuzzy control is based on fuzzy logic and deals with ordinary boolean quantities which are either true or false [32]. It is an attempt to develop a method for logic reasoning with less accuracy. The method if performed by establish linguistic variables and associating them with membership functions. Fuzzy control consists of an input stage, a processing stage and an output stage. The input stage maps input data to a collection of linguistic variables. The processing stage includes a set of rules expressed in fuzzy logic. Each linguistic variable is invoked by the appropriate rule and further processed by combining each result of each rule. The final output converts the combined result back into a specific control output value. The benefit of fuzzy control is its aptitude of dealing with nonlinearities and uncertainties. Different kinds of approaches are suggested in [33][34]. However, fuzzy logic deals with approximations and is not a good candidate for managing systems which require extreme precision. The algorithm does not have the ability to learn and adapt. Expert knowledge is required to create the optimal rule set (the rules are decided by experts) and achieve good system performance.

3.3.5 Evolutionary algorithm

Evolutionary Algorithm (EA) is inspired by biological evolution and utilizes the concept "survival of the fittest", where the fittest are found in the process of evolution [35]. An overview of the general steps is described in Figure 4. The algorithm is initiated by creating randomized values as different solutions to a problem (a population of several individuals). Each individual is evaluated according to their performance compared to every other individual within the population, the measure is called fitness. The fitness value is used by the selection algorithm to assign each individual a rank, the rank is defined as the probability of becoming selected to the process of recreation. All selected individuals are parents for the next generation. The new population is created by combining different parents in various ways, mutate individuals and retain certain parents. The last step may vary depending on the approach of the EA. Finally, as the new population is generated, the new generation is evaluated and the process restarts. There are several kinds of suggestions on how to tune a PID controller with the approach of EA as in [4]. The article presents a simulation where EA obtains better performance compared with the ordinary ZN method. Similar approaches are also suggested in [36][4]. The main disadvantages of EA are the uncertain convergence time, no guarantee of finding a global optimum solution and choosing the correct approach to the problem.

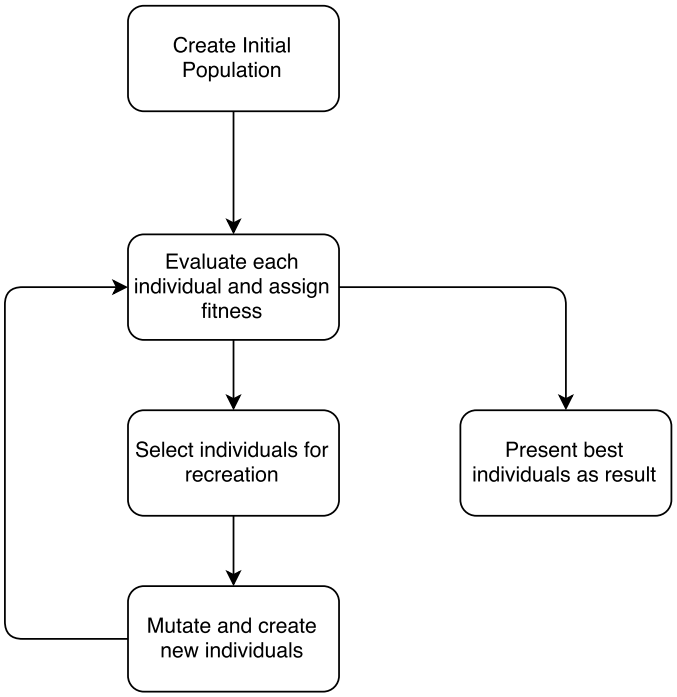

Figure 4: The process of EA

However, the concept is very easy to understand, the algorithm is modular and provides an answer which will always improve over time.

3.3.6 Tuning rules

In light of the amount of previous mentioned algorithms, it appears that there are several methods to approach and find arbitrary PID control parameters. However, since the initial work of ZN, much research has been done in developing rules for tuning. TRs refers to a set of rules which generate certain parameters to a controller. The rules are based upon the characteristics of a certain plant. Accordingly, there may be different rules which strive against providing different kinds of controller behavior (robustness, minimizing the time of obtaining the steady state error etc.). O′Dwyer, in the book [9], presents a large collection of approaches. It is mentioned that several proposed TRs wont have any impact on industrial practice. One reason is that such TRs are not accessible, taking into account that they are all scattered throughout the control literature. Nevertheless, TRs are mostly easy and accurate methods of finding optimal values to different kinds of processes. One method is described in [37]. The article implies a simple recipe on how TRs may be applied. The method is favorable with respects to its high performance and easy, straight forward calculation. The drawback is the difficulty of finding an accurate process model and apply the correct TRs to the system. These are static and will have to be reevaluated if any changes are applied to the system. Also, after system changes a new model will have to be found before new rules can be applied.

3.3.7 Model predictive control

Model Predictive Control (MPC) is a control technique that falls in the class of optimal control born in the early 1970s which with the help of a prediction horizon explores a variety of possible outcomes [38]. The idea of MPC is to set an optimization problem that is solved iteratively, at every control instant. The solution of the optimization problem provides a plan of future control actions that minimizes a given cost function. The first action of the plan is applied, and a new solution is computed based on the actual behavior of the system, while the previous plan is discarded. The MPC relies on the existence of a mathematical model of the system dynamics

which is often found by some kind of iterative model identification technique. Besides configuration parameters for the prediction horizon, other tuning parameters are introduced. Some of these are weights (mainly because these parameters are expressed in different units) that define prioritization between affecting system variables, also called independent variables and saturation values for these parameters. The independent variables are those the controller can affect in a direct way such as set-points for further control loops or control elements such as valves, dampeners etc [39]. Other variables such as disturbance that the MPC cannot control are called dependent variables. MPC works well with open-loop stable systems and increases robustness in noisy systems or where the system model has accuracy problems [40]. The article also presents the development of a MPC controller which uses the prediction of the set-point. This can be done as there is a known schedule for the operation. Results show that the prediction horizon can be reduced significantly with this knowledge.

3.4 Motivations

To the best of our knowledge it was found that the PID controller is the most common approach to control a quadrotors attitude and obtain stability [3][41]. To further increase the stability, some approaches suggests a controller with a combined Kalman filter where results show more accurate control and less overshoot [42][41][43]. However, different papers suggest a variety of other alternative methods to increase the stability, robustness and performance of the controller. Although most approaches show good results, it is worth mentioning that many of these approaches have only been evaluated in simulations. Their may be suggestions like an on-line tuning with pole placement such as [44], adaptive controller with a state observer [45], sliding mode controller [46], nonlinear fuzzy logic external disturbance [34], backstepping [47] for nonlinear fast maneuver of UAV[48] and many more. Further approaches explore the bacterial foraging algorithm for PID tuning [49], an adaptive control of cable suspended load on UAV [50] and an L1 adaptive fault tolerant backstepping [51]. It may be beneficial to further explore methods and combinations of approaches which have been evaluated on real platforms. These may be approaches which include AI algorithms NN [28][30][31], EA [4][36] or FL [32][33]. Unfortunately it appears, to the best of our effort, that those algorithms may have high computational complexity with respect to the given task. Another interesting approach is the established method of MPC [39][40][38]. In addition, MPC is very complex and demands an iterative procedure which might not be feasible together with real-time requirements (despite the fact that there are implementations targeting real-time [52]). It is stated in [39], that the computational requirements and memory constrains will limit the depth of the prediction horizon, which if minimized too much can lead to erroneous results. Thus, if optimization is pushed, non-optimal outputs may arise. Hence, the method of MPC requires an increased amount of memory compared to the PID controller and might not suit excising drone hardware. Accordingly, TRs seem to encourage to minor computational complexity and are well established [9][37][22]. The foundation of any TRs are based on the existance of some kind of model plant. Algorithms of model approximation and estimation are found in [22]. One approach to identify such a model and encourages to less computational complexity is the RLS algorithm. It is a straight forward algorithm which recursively obtains a mathematical pattern by previous weighted input-output values of an optional entity.

4 Method

This section presents the research and development methods used for the design of an adaptive tuning algorithm. The goal was to adapt the model to a quadrotor. Furthermore an iterative work methodology based on the top to bottom approach was chosen. The initial state concerns current knowledge combined with research, followed by implementation and evaluation. The results where then used to refine the method, continued research and finally restart the process into the next iteration. A general approach to solve the thesis problem formulation is described in Figure 5. The proposed solution is divided into four major blocks which describe a general closed loop control system with the extension of adaptation. The Controller provides a control signal which is managed by the Process (the Process represents the actual quadrotor). The MI block identifies and approximates the current Process for each DoF. The next step concerns the block TR, as the MI converges against an accurate Process estimation. The MI provides further improved tuning values based on established TRs.

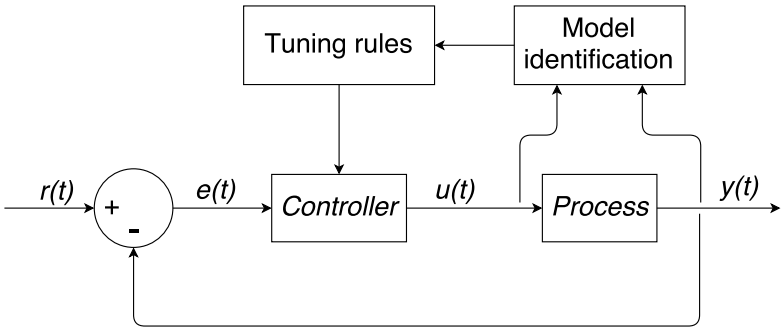

Figure 5: System design solution

4.1 Requirements

The problem formulation was divided into a set of requirements. Although, this thesis focuses on simulations, there are constrains regarding future implementation on hardware.

4.1.1 Software simulation

S_1: The "Process" must be represented by an arbitrary model of the actual quadrotor

4.1.2 System performance

C_1: The resulting work should not reduce flight performance compared to MT

C_2: The MISE and MAE should be minimized within a reasonable time limit

4.1.3 Hardware

E_1: The algorithm should be able to run on-line, in real-time in synchronization with the control loop

E_2: The system design solution should be able to run on an embedded system without interfering with existing process scheduling

4.2 Simulation

The tools used for simulation are MATLAB and Virtual Robot Experimentation Platform (V-REP) [53]. The simulation includes the inverse kinematics model of a quadrotor and is built to mimic its behavior in three DoF. Also the Controller, MI and TR blocks are implemented for each DoF following the design in Figure 5.

4.3 Evaluation

As the goal of this work is to increase performance and robustness through adaptation, the chosen method of evaluation is based on the MISE and MAE. The following two scenarios has been evaluated in simulations:

1. A performance comparison between the proposed method and MT using conventional tuning techniques. This leaves out external disturbance and changed dynamics such as actuator degradation or weight changes.

2. An evaluation of changed inertia and robustness of the proposed approach. Thus, the dynamics are changed drastically in different ways to determine if and at the rate of which the system will converge into an arbitrary solution. Accordingly, the thesis method is compared to MT using conventional tuning techniques.

5 Technical description

The following section provides the information and the technical descriptions and solutions to the thesis problem formulation. It describes how the closed loop system works and how the process identification is done. It covers the technique of implementing TRs and every evaluated suggested approach of this thesis.

5.1 Closed loop system

The architectural system design is presented in Figure 6. It is apparent that the design is based on the closed-loop control system combined with process identification and TRs [17][21][6]. The solution is used on each DoF in the same way independently to provide stability for the system as a whole. It is also known as a feedback control system which uses the concept of an open loop system as a forward path but has one or more feedback loops. Feedback is defined as some portion of an output action that is returned to modify the next action. Closed-loop systems are designed to automatically achieve and maintain the desired output condition. It is done by comparing the desired state r with the actual state y. Consequently, the error signal e is generated, which is the difference between r and y, a "closed-loop system". In addition, signal e is sent forward to the controller which in this thesis was chosen to be a PID controller [6]. Then, the controller will provide a control signal u which is the command sent to the Process. The Process is represented by an arbitrary estimated quadrotor model in simulation. On a real quadrotor, the process output consists of the sensor values measured as a result from actions taken by the UAV′s actuators. The actuators are controlled by the different control signals u which is mixed to its corresponding DoF in the motormixer.

5.2 Model identification

Moving forward to the MI block which is the step of identifying a mathematical model of the process. Accordingly, the model (plant) shall be able to represent the actual behavior of the process. The extent of a models complexity determines the accuracy of an estimated process. However, an advanced plant model with very high complexity may affect the computational rate and decrease the performance with heavy calculations (CPU expensive). Thus, the model identification involves two parts, the first challenge is to choose a modeling strategy. There are a variety of options and the used strategy will influence the values of the plants parameters. The second step concerns the problem of selecting a proper model with the appropriate complexity [37].

For this thesis and further explained in [23], RLS is a suitable tool as a modeling strategy. By harnessing the measurements of the input and output signals u and y, it is possible to estimate a mathematical model of the process. The result of the RLS is a differential equation which describes the output as sum of previous weighted input-output signals. A more detailed explanation is given in the Sections 5.2.1 and 5.2.2. Concerning the selection of the plant model, there are a variety of different options, a common plant is the First Order Plus Dead Time (FOPDT) model. In process control, it is a regular tool because its simple and very efficient at approximating the dynamics of several industrial processes. Accordingly, as the FOPDT is used as an observer to represent the process, it provides precomputed TRs which have been found by experts. Thereby, the rules bring the opportunity to find suitable tuning parameters for a PID controller. The derivation of the FOPDT TRs is described in the Section 5.3. Another common plant is the I^2PD model. It is similar to the FOPDT model, but differs in the manner of containing an additional integrator and excluding the dead time variable. Thus, in conformity with the FOPDT model, the I^2PD model also provides subtitle tuning rules which are adapted to a PID controller. The derivation of the TRs may be found in Section 5.4. A more detailed explanation about several TRs can be found in the book [9]. A detailed overview of the system solution is shown in Figure 6.

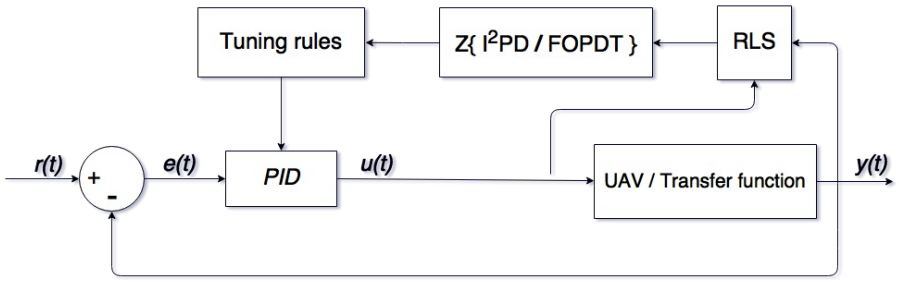

Figure 6: Demonstrates a detailed description of the implemented scheme in Figure 5. It contains the additional block Z{...} that maps the discrete-time model identified by the RLS to the continuous-time model needed by the TRs

5.2.1 Least squares

Applications of numerical techniques in the field of science and engineering involve curve fitting of experimental data. The method of least squares is a standard approach in regression analysis to determine a system model that relates to experimental data. For this work, the identification process concerns an AutoRegression with an eXogenous input (ARX) model [54]. It basically implies that the current output y(k) is predicted as a weighted sum of past output values and past input values. The ARX model is described in Equation (5) and is in discrete form:

$$y(k) = \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{j=1}^{n_b} b_j u(k-j) + \xi(k) \tag{5}$$

The variable y is the value of the process output, u describes the input and ξ represents noise, unmodeled dynamics and measurement errors. With respect to this work, it is assumed that ξ has a fixed probability distribution and a constant expected value and variance which gives an unpredictable contribution to the system dynamics. Regarding the described ARX equation, it is possible to reshape it into a more suitable form which concerns this work. Accordingly, denoting the parameters ϑ and φ, they may be expressed as:

$$\vartheta = [a_1 \ a_2 \ ... \ a_{n_a} \ b_1 \ b_2 \ ... \ b_{n_b}]^T \tag{6}$$

$$\varphi = [y(k-1) \ y(k-2) \ ... \ y(k-n_a) \ u(k-1) \ u(k-1) \ ... \ u(k-n_b)]^T \tag{7}$$

Consequently Equation (5) can be rewritten as:

$$y(k) = \varphi(k)^T \vartheta + \xi(k) \tag{8}$$

The Equation (8) leads to a simple predictive model which represent the actual function of the approximated model:

$$\hat{y}(k) = \varphi(k)^T \vartheta \tag{9}$$

The final aim is to solve Equation (9) by finding its weights and thereby obtaining the approximated model. Thus, the prediction error of the real and approximated model can be calculated as:

$$\epsilon(k) = y(k) - \hat{y}(k) \quad or \quad \epsilon(k) = y(k) - \varphi(k)^T \vartheta \tag{10}$$

A cost function is used to estimate the parameters of vector ϑ and minimize the error, N describes the cardinality which is expressed in Equation (11):

$$J = \frac{1}{N} \sum_{k=1}^{N} \epsilon(k)^2 \tag{11}$$

By substituting $\epsilon(k)$ in Equation (11) with (10) and deriving the expression as $\frac{dJ}{dk} = 0$, it is possible to obtain all the minimum point of the cost function. The minimum point can be obtained as the solution of the following formula:

$$\left[\sum_{k=1}^{N} \varphi(k)\varphi(k)^T \right] \vartheta = \sum_{k=1}^{N} \varphi(k) y(k) \tag{12}$$

To facilitate the equation overview, the $S(N)$ matrix is introduced:

$$S(N) = \sum_{k=1}^{N} \varphi(k)\varphi(k)^T \tag{13}$$

If the $S(N)$ matrix is invertible, it is possible to find the weights $\vartheta(k)$ and thereby solve the identification problem as:

$$\hat{\vartheta}_N = S(N)^{-1} \sum_{k=1}^{N} \varphi(k) y(k) \tag{14}$$

5.2.2 Recursive least squares

The on-line identification procedure for the ARX model is called the RLS, the method recursively finds the coefficients that minimize a weighted linear least square cost function (observe Equation (11)) with respect to the input-output signals. Accordingly, the derivation is described below. Consider Equation (14) where it is possible to expand the last term as:

$$\sum_{i=1}^{k} \varphi(i) y(i) = \sum_{i=1}^{k-1} \varphi(i) y(i) + \varphi(k) y(k) \tag{15}$$

By writing Equation (14) at time k - 1, directly yields:

$$\sum_{i=1}^{k-1} \varphi(i) y(i) = S(k-1)\hat{\vartheta}_{k-1} \tag{16}$$

Applying the definition of Equation (16) in Equation (15) gives:

$$\sum_{i=1}^{k} \varphi(i) y(i) = S(k-1)\hat{\vartheta}_{k-1} + \varphi(k) y(k) \tag{17}$$

Moreover, it appears from Equation (13) that:

$$S(k) = S(k-1) + \varphi(k)\varphi(k)^T \tag{18}$$

As a consequence, Equation (17) may be described as:

$$\sum_{i=1}^{k} \varphi(i)y(i) = (S(k) - \varphi(k)\varphi(k)^T)\hat{\vartheta}_{k-1} + \varphi(k)y(k) \tag{19}$$

As a final step, by substituting (19) in (14), the first form of the RLS is discovered as:

$$\begin{aligned} \hat{\vartheta}_k &= \hat{\vartheta}_{k-1} + K(k)\epsilon(k) \\ K(k) &= S(k)^{-1}\varphi(k) \\ \epsilon(k) &= y(k) - \varphi(k)^T\hat{\vartheta}_{k-1} \\ S(k) &= S(k-1) + \varphi(k)\varphi(k)^T \end{aligned} \tag{20}$$

5.2.3 Computational simplification

With the appropriate algebraic manipulations, it is also possible to rewrite the RLS algorithm in other forms. The book [54] provides the following extended derivations:

$$\begin{aligned} \hat{\vartheta}_k &= \hat{\vartheta}_{k-1} + K(k)\epsilon(k) \\ K(k) &= V(k)\varphi(k) \\ \epsilon(k) &= y(k) - \varphi(k)^T\hat{\vartheta}_{k-1} \\ V(k) &= \frac{1}{\mu}V(k-1) - \beta_{k-1}^{-1}V(k-1)\varphi(k)\varphi(k)^T V(k-1) \\ \beta_{k-1} &= \mu + \varphi(k)^T V(k-1)\varphi(k) \end{aligned} \tag{21}$$

The alternative RLS is a more appropriate option for this thesis. From a computational point of view, it is less complex and requires less CPU power.

5.3 FOPDT model

The first order linear system FOPDT with a time delay is a common empirical description of many stable dynamic processes. Is has en easy mathematical structure with only three unknown parameters, the equation is described as:

$$P(s) = \frac{K}{1+Ts}e^{-Ls} \tag{22}$$

Accordingly, the unknown parameters represent:

- K = Process gain
- T = Process time constant
- L = Process Dead time

The process gain is the variable which affects the magnitude of the output response, meaning the relation between the input and output. The time constant describes the amount of time needed for the output too reach 63.3 % of the steady state condition. It is a variable which affects the speed of the response of the system. The last parameter L is a shift delay. Assume that the input signal is a step function that normally changes from 0 to 1 at time t = 0. However, due to physical and computational constrains, the output signal is delayed by L_k. As a result, the final system model has a delay and the control signal is affected as $u(k - L_k)$.

5.3.1 FOPDT to RLS

Taking into account that the RLS provides a discrete-time process function where the output is equal to weighted previous input and output values. It is of major importance to map the discrete-time model to the continuous-time model needed by the TRs. Currently, the FOPDT is described in continues-time through its Laplace transform and requires a conversion to the discrete-time counterpart, expressed in the Z-transform domain. Here the Forward Euler approximation is adopted for performing this mapping:

$$s = \frac{z-1}{h}, \quad \text{where} \quad h = \textit{sampling time} \tag{23}$$

Thus, the Equation 22 can be converted to the discrete time domain as:

$$P(z) = \frac{Y(z)}{U(z)} = \frac{K}{1 + T(\frac{z-1}{h})} e^{-L(\frac{z-1}{h})} = \frac{hK}{h + Tz - T} \left[e^{-\frac{L}{h}(z-1)} \right] \tag{24}$$

Moving forward to the next step, the $e^{-\frac{L}{h}(Z-1)}$ term may be ignored since it is assumed that the dead time is less than the sampling time:

$$Y(z) = \frac{hK}{h + Tz - T} U(z) \;\Leftrightarrow\; (h + Tz - T)Y(z) = hKU(z) \tag{25}$$

$$TzY(z) + (h - T)Y(z) = hKU(z) \tag{26}$$

The inverse Z-transform is described in the book [55] and written as:

$$\mathcal{Z}^{-1} \left\{ z^{\pm k} Y(z) \right\} = y(n \pm k) \tag{27}$$

The Z-transform converts Equation (26) to:

$$Ty(k+1) + (h - T)y(k) = hKu(k) \;\Leftrightarrow\; y(k+1) = \frac{(T-h)}{T} y(k) + \frac{hK}{T} u(k) \tag{28}$$

Equation (28) may be simplified to the following form:

$$y(k) = \left[\frac{T-h}{T} \right] y(k-1) + \left[\frac{hK}{T} \right] u(k-1) \;\Leftrightarrow\; y(k) = \vartheta_1 y(k-1) + \vartheta_2 u(k-1) \tag{29}$$

Notice the similarity between Equations 5 and 29. The functions are equivalent and make it possible to find the parameter of the FOPDT function, the following relation is discovered:

$$\vartheta_1 = \frac{T-h}{T} \Leftrightarrow T = \frac{h}{1-\vartheta_1} \tag{30}$$

$$\vartheta_2 = \frac{hK}{T} \Leftrightarrow K = \frac{\vartheta_2}{1-\vartheta_1} \tag{31}$$

5.3.2 Tuning rules

The next step concerns the parameters of the FOPDT model. The TRs belonging to the FOPDT aim against minimizing the MAE and MISE, these are described in the article [37]. The equations are presented as:

$$K_p = \frac{a_1}{K}\tau^{b_1}, \qquad T_i = \frac{T}{a_2 + b_2\tau}, \qquad T_d = a_3 T \tau^{b_3}, \qquad where \quad \tau = L/T \tag{32}$$

Constants	τ = 0.1-1.0	τ = 1.1-2.0
a_1	1.048	1.154
b_1	-0.897	-0.567
a_2	1.195	1.047
b_2	-0.368	-0.220
a_3	0.489	0.490
b_3	0.888	0.708

Table 1: TRs parameters

In this context, the TRs are adapted to a specific parametrization of the controller. For this case, it is adapted to a controller with a low pass filter of the derivative term. The transfer function of the controller $K(s)$ is:

$$K(s) = K_p\left(1 + \frac{1}{T_i s} + \frac{T_d}{1 + (T_d/N)s}\right) \tag{33}$$

The K_p value is the proportional gain, T_i is the integral time constant and T_d is the derivative time constant. The derivative time noise filter constant N usually takes values within the range of 5-33 without loosing its functionality [9]. For this paper, it was found that the value of 18 provided satisfactory performance and noise filtering.

5.4 I^{2}PD model

The Integral Squared Plus Time Delay (I^{2}PD) represents a non-self-regulating process which only contains two unknown parameters. The transfer function is described as:

$$P(s) = \frac{K}{s^2}e^{-s\tau} \tag{34}$$

The unknown parameters represent:

- K = Process gain
- τ = Process Dead time

The I^{2}PD function is a second order system which contains the same parameters as the FOPDT model apart from the time constant T. A major difference is the output behavior. The goal is to find the most accurate model approximation to represent the plant.

5.4.1 I^{2}PD to RLS

The I^{2}PD model may be converted into the time discrete domain in the same way as the FOPDT model. Observe Equation (23), it provides the following extension of the I^{2}PD:

$$P(z) = \frac{Y(z)}{U(z)} = \frac{K}{(\frac{z-1}{h})^2} e^{-(\frac{z-1}{h})\tau} = \frac{Kh^2}{z^2 - 2z + 1} \left[e^{-(\frac{z-1}{h})\tau} \right] \tag{35}$$

According to previous assumptions, the $e^{-(\frac{z-1}{h})\tau}$ term may be ignored since it is assumed that the dead time is less than the sampling time. This provides:

$$Y(z)(z^2 - 2z + 1) = U(z)(Kh^2) \tag{36}$$

$$Y(z)z^2 - 2Y(z)z + Y(z) = Kh^2 U(z) \tag{37}$$

The inverse Z-transform, described in (27) gives:

$$y(k+2) - 2y(k+1) + y(k) = Kh^2 u(k) \tag{38}$$

$$y(k+2) = 2y(k+1) - y(k) + Kh^2 u(k) \tag{39}$$

$$y(k) = 2y(k-1) - y(k-2) + Kh^2 u(k-2) \;\Leftrightarrow\; y(k) = 2y(k-1) - y(k-2) + \vartheta_3 u(k-2) \tag{40}$$

Accordingly, it appears that Equation (5) is equivalent to Equation (41). The gain K may be discovered:

$$\vartheta_3 = Kh^2 \;\Leftrightarrow\; K = \frac{\vartheta_3}{h^2} \tag{41}$$

5.4.2 Tuning rules

The TRs for the I^{2}PD model are based on a method provided by Åström and Hägglund, it can be found in the book [9] and is written as:

$$K_p = \frac{0.02140}{K\tau^2}, \qquad T_i = 17.570\tau, \qquad T_d = 14.019\tau \tag{42}$$

The TRs for this model are adapted to the specific controller:

$$K(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \tag{43}$$

Furthermore, the current transfer function corresponds to the controller described in Equation 33. It contains the same parameters with the major difference of the absence of an integrated filter for the derivative term.

5.4.3 Empirically modified tuning rules

The following TRs are a modification of the I^2PD TRs described in Section 5.4.2. They have been found by empirical evaluations and the continued process will show how the modified rules in this approach make it favorable with respect to the other evaluated approaches (note Section 7 which contains an ensemble of every method evaluated in within this thesis). The modified rules are written as:

$$K_p = \frac{0.02140}{K\tau^2}, \qquad T_i = 2155.2\tau, \qquad T_d = 1.7586\tau \tag{44}$$

Observe that the corresponding PID controller is equivalent with the controller described in Equation (43). Also, in order to obtain the proper rules, focus has been towards the goal of minimizing the MISE and MAE for different kinds of tests on each axis.

6 Simulation environment

For this thesis the simulation environment V-REP has been used to simulate the behavior of the quadrotor [53], while MATLAB has been used for the implementation of the controller. An existing quadrotor model has been used which includes its inverse kinematics as well as the simulation of particles passing through the thrusters. Sensor data as angles, angular velocity and position are gathered and the actuation is done controlling the four thrusters. The controlled parameter for these in the simulation is a desired particle velocity which is adjusted by its own controller within the simulation environment. This can be compared to the desired revolution of a motor on a real quadrotor which has its own Electronic Speed Controller (ESC) for each motor.

6.1 Mathematical model

A quadrotor has fixed pitch propellers, hence thrust T on each motor is regulated by changing their rotational speed (Equation (45)). Adjusting the thrust of the motors makes it possible to control the lift, roll and pitch. The yaw is controlled by utilizing the combined torque Q from each motor and propeller pair and is described in Equation (46).

$$T = c_T\omega^2, \text{ where } c_T = \textit{propeller thrust coefficient} \tag{45}$$

$$Q = c_Q\omega^2, \text{ where } c_Q = \textit{motor and propeller torque coefficient} \tag{46}$$

It is important that each degree of freedom is independent, meaning that the actuation on one axis should maintain the combined torque and force on all other axis unchanged. Roll, pitch and yaw represent the rotations around the x, y and z-axis, x facing forward towards the flight trajectory. To achieve this, a motor configuration is set that for all three axis of rotation means that actuation will increase and decrease two pairs of motors proportionally where each pair consists of two motors rotating in the opposite direction to each other. An illustration can be seen in Figure 7 where the front of the quadrotor is facing upwards. Equation 47 describes the thrusts and torques for each axis where $\sum T$ is the combined thrust of all motors.

$$\begin{bmatrix} \sum T \\ \tau_{roll} \\ \tau_{pitch} \\ \tau_{yaw} \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ -dc_T & -dc_T & dc_T & dc_T \\ dc_T & -dc_T & dc_T & -dc_T \\ -c_Q & c_Q & c_Q & -c_Q \end{bmatrix} \begin{bmatrix} \omega_{M1}^2 \\ \omega_{M2}^2 \\ \omega_{M3}^2 \\ \omega_{M4}^2 \end{bmatrix}, \tag{47}$$

where $d = \textit{distance from each rotation axis}$

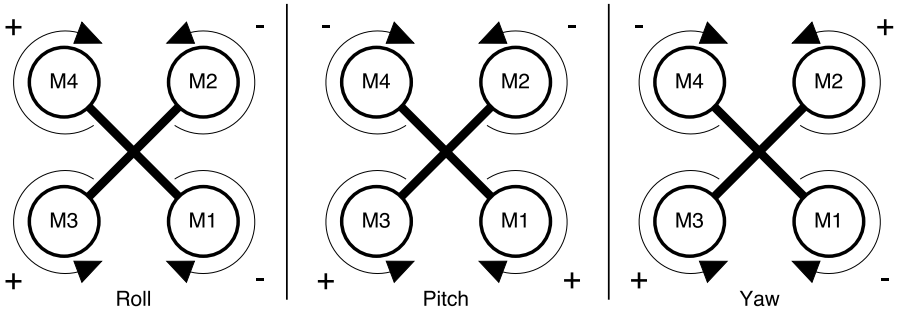

Figure 7: Quadrotor Control

6.2 Control setup

There are in total 11 control loops controlling the quadrotors position in space. These control loops are connected in a cascade structure for each axis as seen in Figure 8 and then in turn the structures for each axis are connected in parallel into the motormixer as is illustrated in Figure 9. Position control (the position in x and y) is ultimately controlled by the control loops for roll and pitch, thus the cascade for these both share the same structure. The cascade is divided into a number of blocks. Each block comprises a certain controller which is in charge of a specific control abstraction. The cascade is divided into the blocks position, velocity, angle and angular velocity. An illustration can be seen in Figure 8.

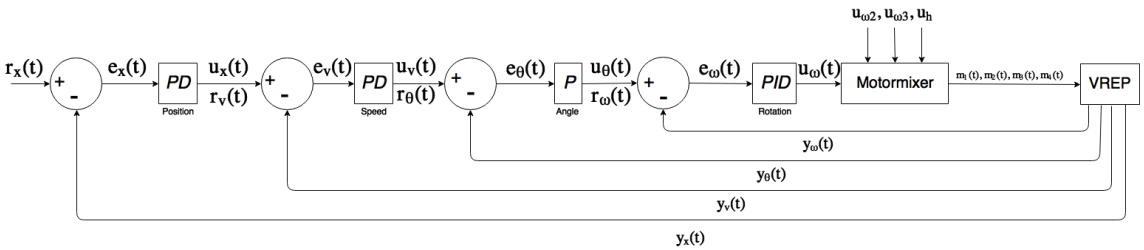

Figure 8: Position Control

For the z-axis (yaw) the structure is much simpler. Here a P-controller sets the desired heading in relation to the global reference frame and sends its output to the PID controller for angular velocity on the axis. Hence Figure 8 does not apply for the z-axis. Lastly comes the altitude control which consists of one single PID controller. All the resulting outputs are then combined in the motormixer which is illustrated in Figure 9.

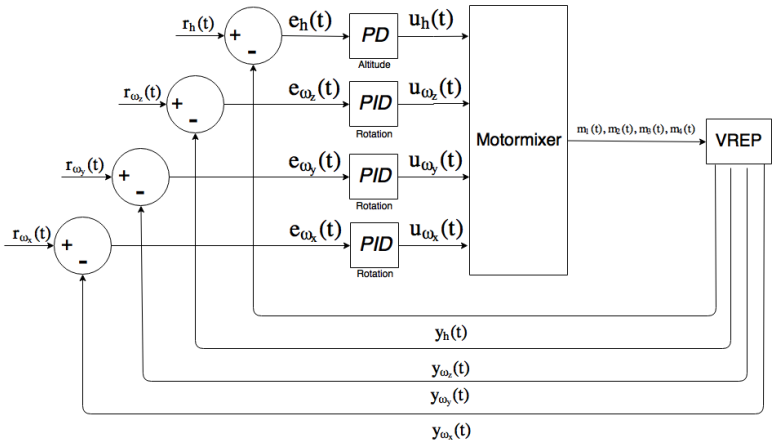

Figure 9: Control inputs-outputs to the motormixer

The motormixer itself can be explained in a simplified way as a block which sums all outputs and rescales them if necessary. If any of the motors reach their saturation level, then a rescaling of all motor outputs is done and the control signals are updated for the adaptive algorithms to get an accurate identification of the process. This means the response corresponds to an accurate estimation of the control signal for each axis being identified. A simplified illustration is shown in Figure 10.

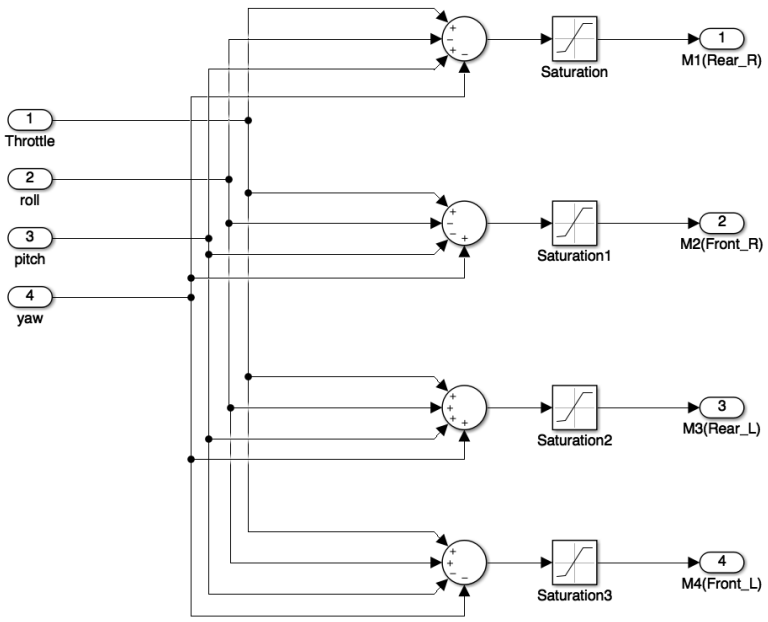

Figure 10: Motormixer

6.3 Manual tuning

A rule-based empiric tuning was done on all 11 control loops by trial and error [7]. The Ziegler-Nichols method was used to get a first estimation which was then optimized manually. The tuning process was done in a bottom up manner meaning tuning of the control loops closest to the actuation was done first and then tuning was continued working backwards up until the highest level was reached. The Ziegler-Nichols method can be simplified into two steps which include finding the critical gain K_u at which the critical oscillation frequency F_u and the oscillation period T_u are present. From there the K_p, T_i and T_d values are set according to a table depending on the desired behavior and performance (for instance Table 3). K_u is identified by zeroing K_d and K_i and increasing the gain K_p of the system until it finds constant oscillations after a change of input r. The gain is increased in small steps to find the value at which this happens. The frequency F_u is then observed and its inverse leads to the oscillation period T_u. Another approach to find F_u is to stimulate the control signal u with a sinusoidal function disconnecting the controller and then increase this frequency until a phase difference between the control signal u and the output y is equal to -pi (-180 degrees). The two have been compared and give similar results. Table 2 shows the TRs for systems with and without integral and derivative part.

P Control	$K_p = 1/2K_u$		
PI Control	$K_p = 5/11K_u$	$T_i = 5/6T_u$	
PID Control	$K_p = 2/3K_u$	$T_i = 1/2T_u$	$T_d = 1/8T_u$

Table 2: Ziegler-Nichols TRs

Using the rules from Table 2 as a base, there are many adaptations of the classical Ziegler-Nichols rules developed with different goals and measures of performance, some of which can be seen in Table 3. The continued optimization of the system is done by trial and error analyzing the results of changed parameters. It should be noted that performance of the control systems higher up in the hierarchy will be dependent on the performance of the loops further down, closer to the actuation. Also the definition of performance changes higher up the hierarchy as a smooth and calm movement is more desirable when navigating to a new position in space than quickly reaching the set-point.

Rule name	**Parameters**		
Classic Ziegler-Nichols	$K_p = 2/3K_u$	$T_i = 1/2T_u$	$T_d = 1/8T_u$
Pessen Integral Rule	$K_p = 7/10K_u$	$T_i = 2/5T_u$	$T_d = 15/100T_u$
Some Overshoot	$K_p = 1/3K_u$	$T_i = 1/2T_u$	$T_d = 1/3T_u$
No Overshoot	$K_p = 1/5K_u$	$T_i = 1/2T_u$	$T_d = 1/3T_u$

Table 3: Ziegler-Nichols adaptations

This means in practice that the gains for these loops are relatively lower than for example the ones for the angular velocity where performance is the measure of how fast the target value is reached. As the focus of this work is on the angular velocity, here is where most optimization has been done to reach the best possible results for later comparison with the proposed approach of the thesis. The summary in Table 4 describes some simple rules when finding the best solution for the system. The K_p, T_i and T_d values can be represented as K_p, K_i and K_d as per Equation 48.

$$K_p = K_p,\ K_i = \frac{K_p}{T_i},\ K_d = K_p T_d \tag{48}$$

Parameter	+	+ +	-	- -
K_p	Increased responsiveness	Quick oscillations Instability	Increased robustness Decreased performance	Unresponsive system
K_i	More drift correction	Slow oscillations	Faster error decay	High drift
K_d	Less overshoot Better wind tolerance Higher K_p are possible Higher robustness	Noisy output Hot motors Decreased responsiveness	More overshoot after changed set-point	K_p cannot be as high

Table 4: Effect of changed parameters

By applying these methods of MT, suitable parameters where found for the control loops used in the simulations. The parameters for the control loops of angular velocity are presented in Table 5. It should be noted that the accuracy of these parameters will change depending on the experience of the operator performing the tuning. Roll is rotation around the x-axis, pitch around the y-axis and yaw around the z-axis.

	K_p	K_i	K_d
Roll	0.0013	0.00012	0.00011
Pitch	0.0025	0.00014	0.00010
Yaw	0.0520	0.00410	0.00051

Table 5: Resulting values from Manual Tuning

7 Simulation results

The evaluation of this work consists of a comparison in performance between three adaptive approaches and MT. The approaches have been executed over four major test scenarios. All three adaptive methods consist of identifying a model and tune the controlling parameters. To refer to these in a more simple manner they have been named accordingly:

A1 This method uses the FOPDT model from Section 5.3 as a plant and becomes approximated with the RLS (presented in Section 5.3.1). Corresponding TRs may be found in [37] and get explained in Section 5.3.2.

A2 This method refers to the I^2PD model which is described in Section 5.4. The model is identified with RLS algorithm (observe Section 5.4.1) and uses the TRs described in Section 5.4.2.

A3 The proposed approach by this thesis. Identical to **A2** with the difference of empirically optimized TRs (note Section 5.4.3).

In total there are four kinds of tests (**T1**, **T2**, **T3**, **T4**), each one with its own specific setup. Every test is performed on all three adaptive approaches plus the MT, also for both the x and z-axis. The reason no evaluation is done for the y-axis is due to the x and y-axis sharing very similar dynamics, see Equation (47). During all tests, control on all other axis is active. For example, when evaluating the z-axis, then control on both the x and y-axis is active. For the tests **T1**, **T2** and **T3**, the method of MT is used on the non evaluated axis. Before running any tests, the conventional PID controller is tuned according to 6.3 (MT). Also, all the models for **A1**, **A2** and **A3** are identified by training every RLS data-set set to a converged state. For the battery degradation, a model has been used from [56]. The simulation model can be seen in Figure 11. The four different thesis tests are described as:

T1 The first test consists of measuring the performance of an unchanged system. The method for this is inducing a square wave signal to the controller with the frequency of 1Hz and amplitude of 5 deg/s. The same input signal r is used for tests on both axis.

T2 The second test is an extension of **T1**, the major difference is that the inertia will changed during run-time. For the z-axis, the first third of the test has a slightly raised inertia ($1.4\times$ Original Inertia (I_O)), the second part of the test returns to the original inertia and in the last third of test, the inertia is raised substantially ($2.0 \times I_O$). For the x-axis, inertia starts unchanged and is substantially increased at one third of the simulation ($4.0 \times I_O$).

T3 Still building on top of **T1**, battery degradation is emulated by decreasing the power of the actuators over time. Actuator degradation reaches around 40% at the end of the simulation and is based on the model for Lithium-Polymer (LiPo) batteries presented in [56]. The data-points have been fitted to a function with the help of the MATLAB function "polifit" and the resulting curve is presented in Figure 11.

T4 This test aims at emulating flight under normal operation. The quadrotor has to follow an object moving gently around in space while battery degradation is active. Evaluation is done on the position control loops.

To facilitate the understanding of comparisons between different methods within a specific test, the following convention was introduced:

C1 Refers to a comparison between **A3** and MT

C2 Refers to a comparison between **A3** and **A2**

C3 Refers to a comparison between **A3** and **A1**

The presented concepts will be used in the continued exposition. Each abbreviation is used in a different manner concerning the results and the final conclusion. Be aware of its importance since its also used in the appendix 12 to clarify certain aspect of the final evaluation.

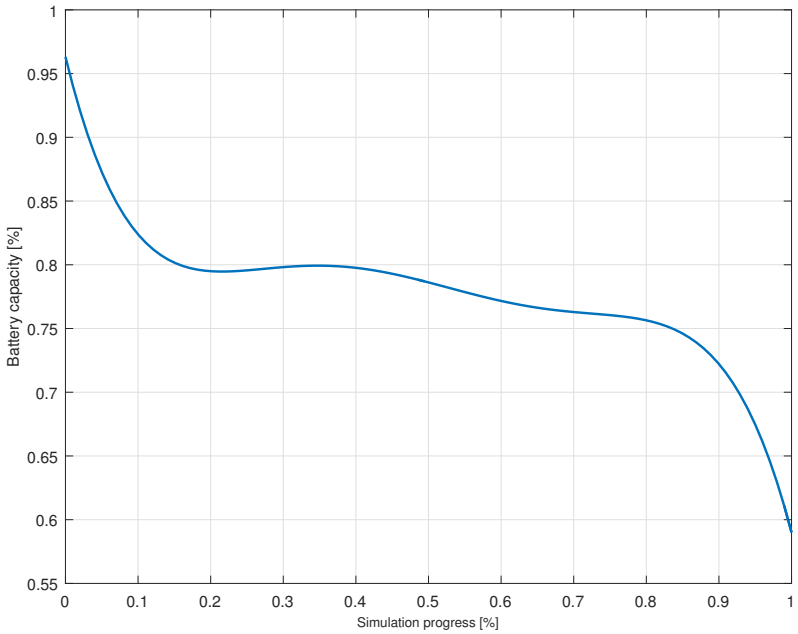

Figure 11: Battery degradation model

The current section focuses on the comparison **C1** between the results from MT against the proposed approach of this thesis **A3**. Results are shown for both the x and z-axis. See appendix (Section 12) for more detailed results on the comparisons **C2** and **C3** for the approaches **A2** and **A1**. In the simulation environment the quadrotor has due to the setup of a realistic scenario some constrains where the simulation can be deemed unsuccessful. The most common outcomes which lead to the abortion of the simulation are if the quadrotor finds itself in a state upside-down or that it was not able to hold its position in space. The later can occur due to instabilities in the angular velocity controller which are propagated into the position control. There is a set distance the quadrotor can drift away from its target position before the simulation is aborted. In these cases results are marked with "N/A".

7.1 RLS initialization and convergence

Figure 12 shows an example of the RLS weights being initialized from zero. At 0.375 seconds the adaptation is activated which can be seen in the change of the K_p value. At one second the set-point is changed which speeds up the identification process. The K_p value evolves in relation to the third weight of the RLS (θ from Section 5), which will be refereed to as W3 for here on. W3 identifies the gain of the system on which the TRs are applied to find the new K_p value. The K_d and K_i values are scaled accordingly through the change of K_p also with use of the TRs. These are also dependent on the systems dead-time and delta-time which can be read about more in detail in Section 5.4.3.

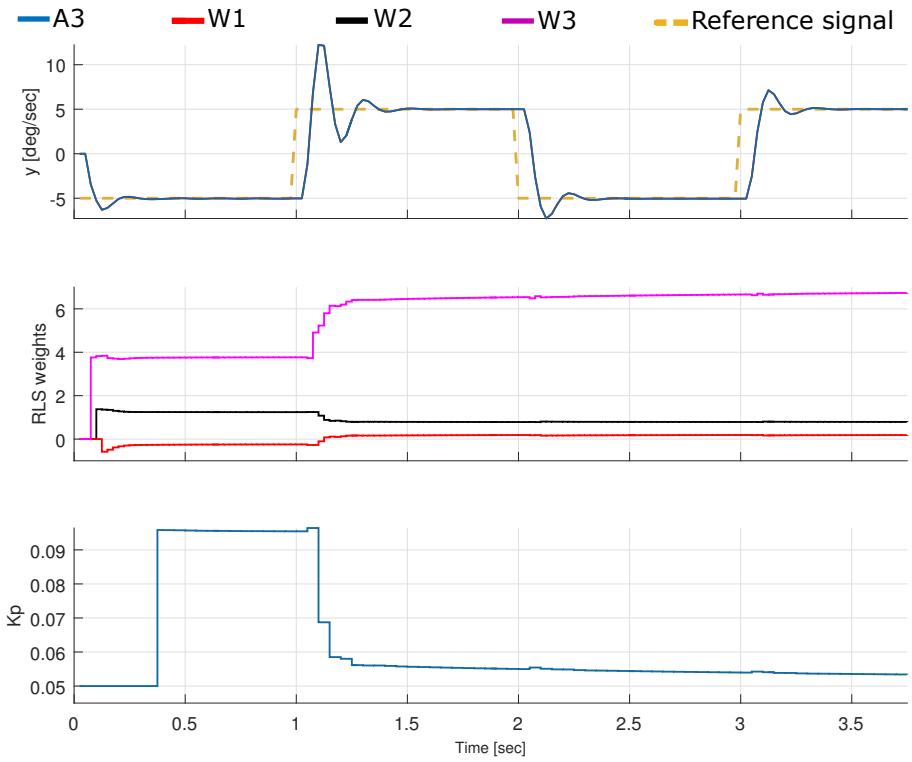

Figure 12: **A3** (z-axis), how the RLS evolves over time

7.2 Test 1 - normal operation

The current subsection presents a comparison in performance between different approaches through **T1**. Two Figures are demonstrated for each axis. Figures 13 and 15 demonstrate three different graphs which show different signals of the system. The top graph displays the systems output y, the second plot is the control signal u and the last graph shows how K_p evolves. Figures 14 and 16 present two different error signals per new set-point (MISE/sp and MAE/sp). The errors are the actual measurements of performance, each error is individually calculated for every new set-point. A last demonstration of the results are presented in tables 6 and 7. They represents a general overview of the total error for the entire **T1**.

7.2.1 Z-axis

Observe Figures 13 and 14. The simulation operates for an amount of 12.5 seconds and concerns a comparison between **A3** and MT (**C1**). It is apparent that both approaches maintain a stable state. The saturation of the control signal u is in the range of -1.5 to 1.5, accordingly it appears that neither approach reaches the saturation limit. However, as **A3** becomes active at the time of 1 second, it is clear that the current K_p value is then increased for 1 second before the RLS responds to the system change and decreases the K_p value to a more appropriate state. Notice how the control signal is magnified for a brief moment at the time 2 seconds. Accordingly, it can be seen that **A3** works towards an extension of providing a stable and desirable state. The errors maintain a stable condition without much deviation. As a further note, observe Table 6 which shows the total MISE and MAE of the entire simulation. The two approaches **A3** and MT perform equally, while the other approaches have a significantly bigger error.

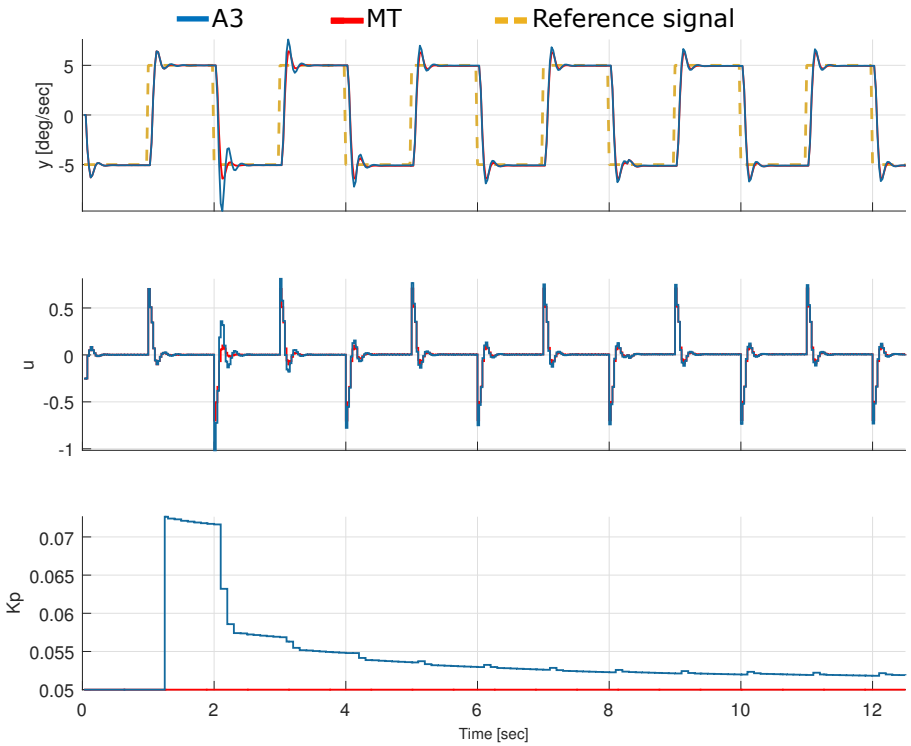

Figure 13: T1(C1), z-axis

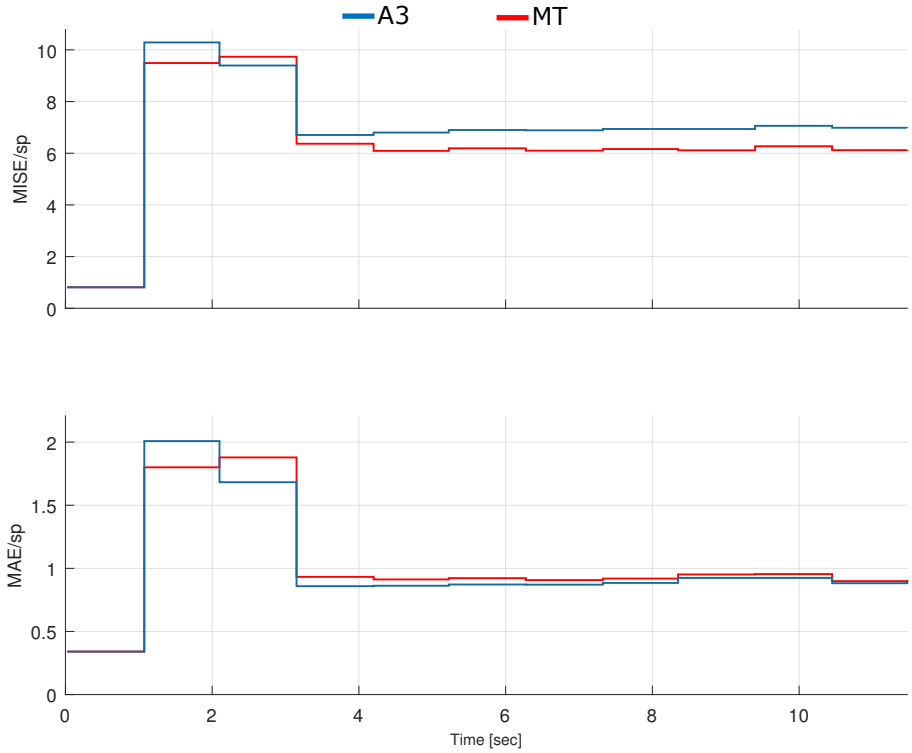

Figure 14: T1(C1), z-axis, errors

A further comparison of the approaches **A2** and **A1** may be found in Figures 30, 32 and 34 in the appendix Section 12.1. Additionally, **T1** demonstrates a comparison where it can be seen that **A3** has the best performance.

T1	**A1**	**A2**	**A3**	**MT**
MISE	48.953	11.586	6.691	6.705
MAE	5.857	2.812	0.947	0.923

Table 6: T1 results, z-axis

7.2.2 X-axis

See Figures 15 and 16, the current test simulation on the x-axis is performed in the same way as the previous test on the z-axis. The test has operated for an amount of 12.5 seconds and concerns an comparison between **A3** and MT (**C1**). As a similarity to the previous result, both methods remain within the saturation range of -1.5 to 1.5 (referring to the control signal u). The method **A3** becomes active at 1 second and begins to converge towards an even more stable state. To be noted is the decrease of K_p during convergence. Observe the output of MT in Figure 15. MT demonstrates small oscillations for each given change in set-point and thereby a longer settling time. However, both of the methods provide similar results and bring the UAV to a stable state (observe the errors of the MISE/sp and MAE/sp). The total MISE and MAE are presented in Table 7, thus it is apparent that both **A3** and MT perform better the remaining approaches.

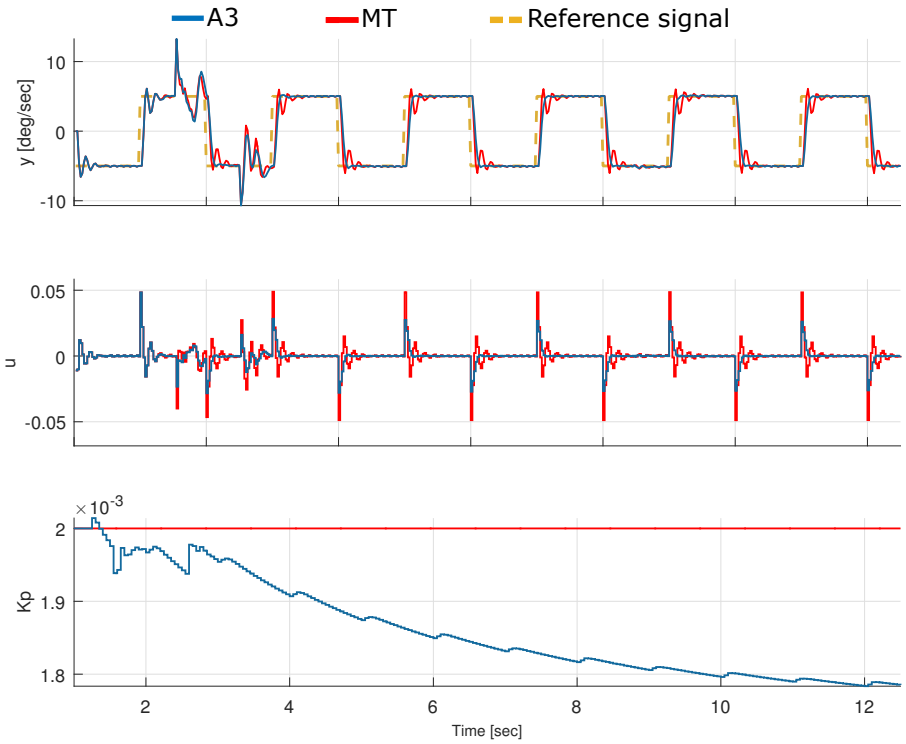

Figure 15: T1(C1), x-axis

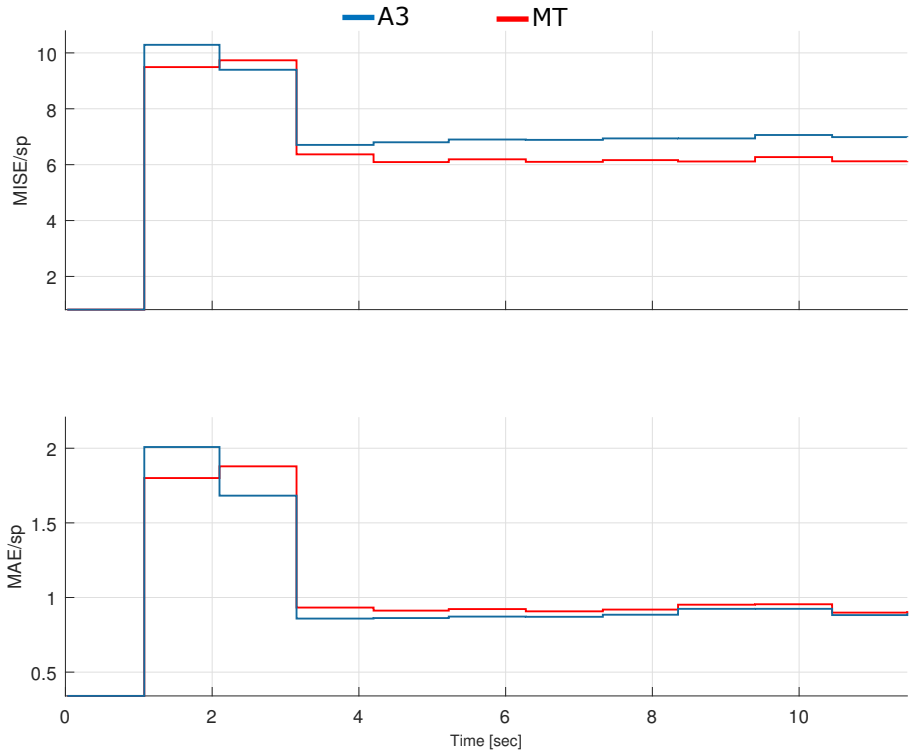

Figure 16: T1(C1), x-axis, errors

T1	A1	A2	A3	MT
MISE	N/A	23.495	7.185	6.547
MAE	N/A	2.835	1.033	1.054

Table 7: T1 results, x-axis

7.3 Test 2 - modified inertia

The following subsection presents a comparison between different approaches with the use of **T2**. Three figures are presented for each axis. Figures 17, 18, 20 and 21 correspond to the same graph model as pictured in Section 7.2. The new graph type in 19 and 22 is a detailed extension of how the system reacts to changed inertia. The images (19, 22) are divided into three different time ranges. Each range presents the output for a specific inertia.

7.3.1 Z-axis

It can be seen in Figure 17 that performance of **A3** adapts and converges within each block. The overshoot is higher at first when the inertia is decreased but stabilizes to the same performance as in the end of the previous block. When the inertia is raised substantially in the third and last block the performance seems slightly degraded. Analyzing the control signal u the reason for this is found to be saturation of the actuators which is reached at a value of 1.5. The evolution of the K_p value follows the logic of the inertia in an intuitive way and decreases with a lower inertia and increases when the inertia is raised. In Figure 18 it can be seen that the error increases at every change of inertia for **A3** but is reduced closely to the same value at the end of each block. In comparison to MT, **A3** reaches equal or superior performance for each block. A representation of the error at the end of each block is shown in Figure 19 where it is clear that the proposed approach **A3** reaches similar performance at the end of each block as per the MT looses performance when inertia is changed.

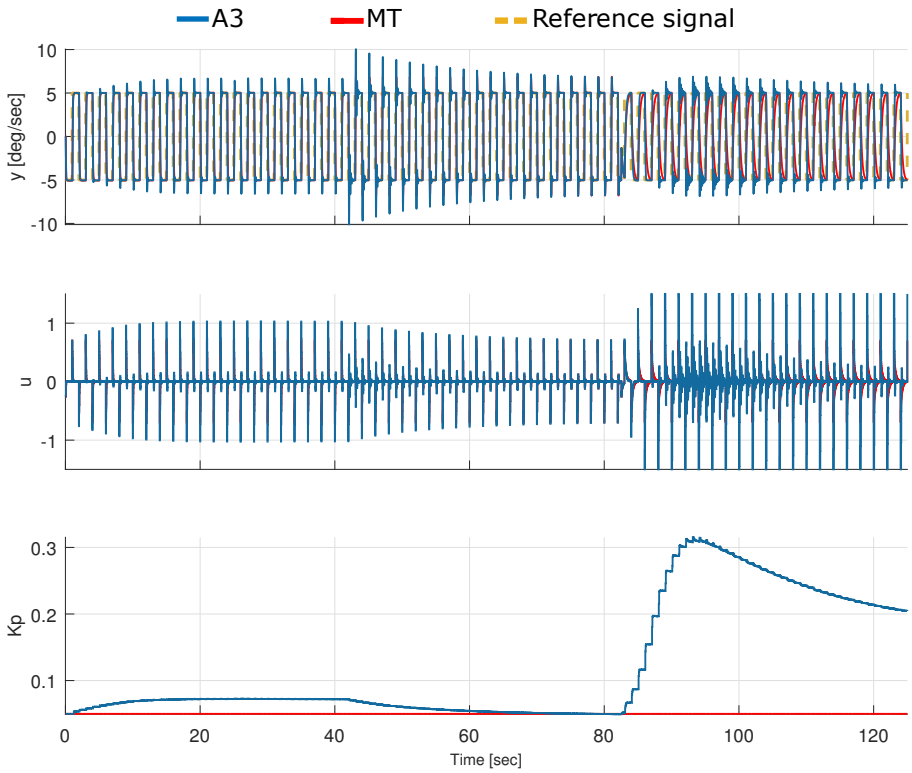

Figure 17: T2(C1), z-axis

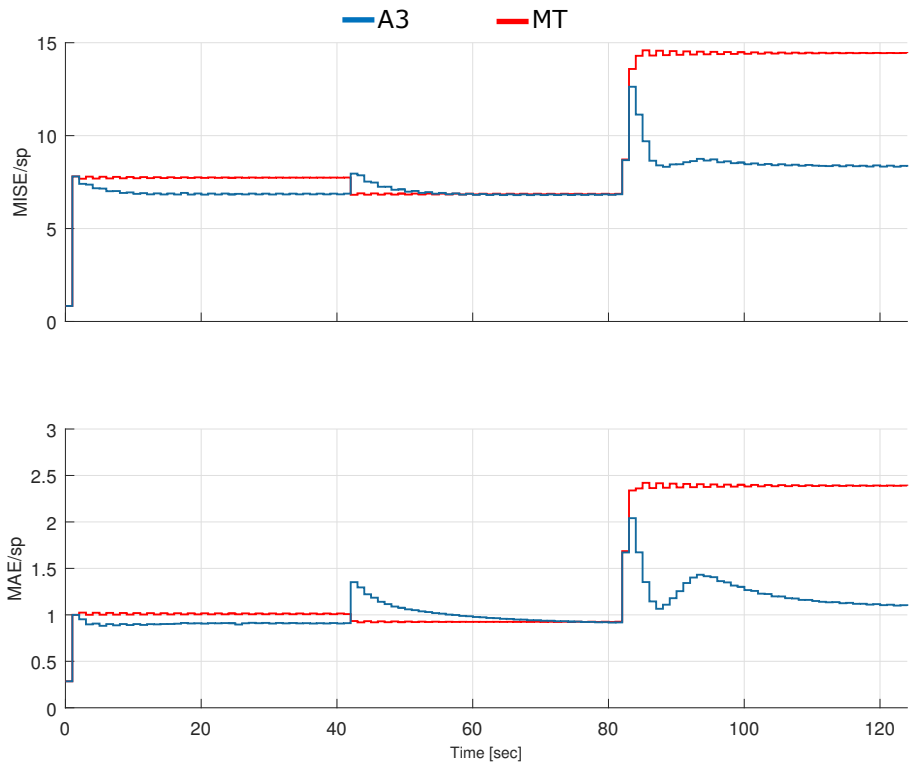

Figure 18: T2(C1), z-axis, errors

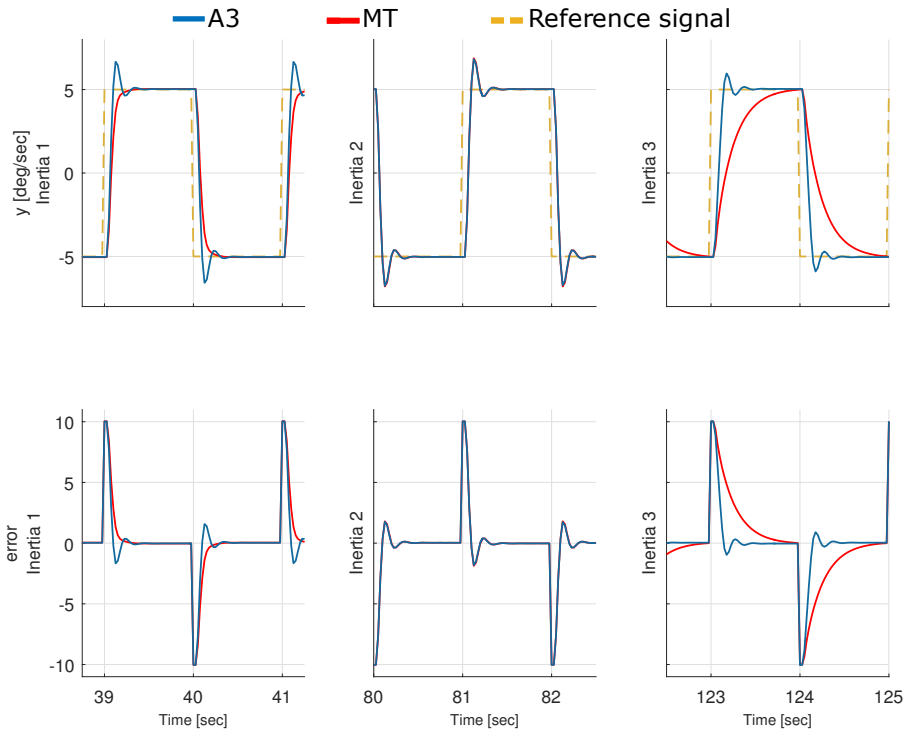

Figure 19: T2(C1), z-axis, different inertias

As presented in Table 8, **A3** achieves the best performance in this test. Section 12.2.1 shows in Figure 36 that **A1** adapts to the changed inertia but performance is very poor. Section 12.2.2 shows slightly better performance for **A2** but both approaches have huge amounts of overshoot and control signals are often saturated.

T2	**A1**	**A2**	**A3**	**MT**
MISE	33.255	17.002	7.508	9.677
MAE	4.665	3.249	1.055	1.448

Table 8: T2 results, z-axis

7.3.2 X-axis

Once more, the proposed approach shows convergence and improvement in performance over time. Some oscillations can be seen just before the adaptation is activated at around 3 seconds in Figure 20. After the inertia is changed at 1/3 of the simulation, degradation in performance is seen for both **A3** and MT. For **A3** the K_p value is raised during the identification until good performance is reached. As can be seen in Figure 21 the error decreases and the system reaches close to the same performance as before the change. Here, in difference from the z-axis, the control signal is far from saturated allowing better performance. The error of MT is now more than twice of the one of **A3**. Looking at Table 9 together with Figure 44 it can be seen that **A2** has similar adaptation but shows unstable behavior with much oscillations. As for **A1**, performance was not good enough to complete the simulation.

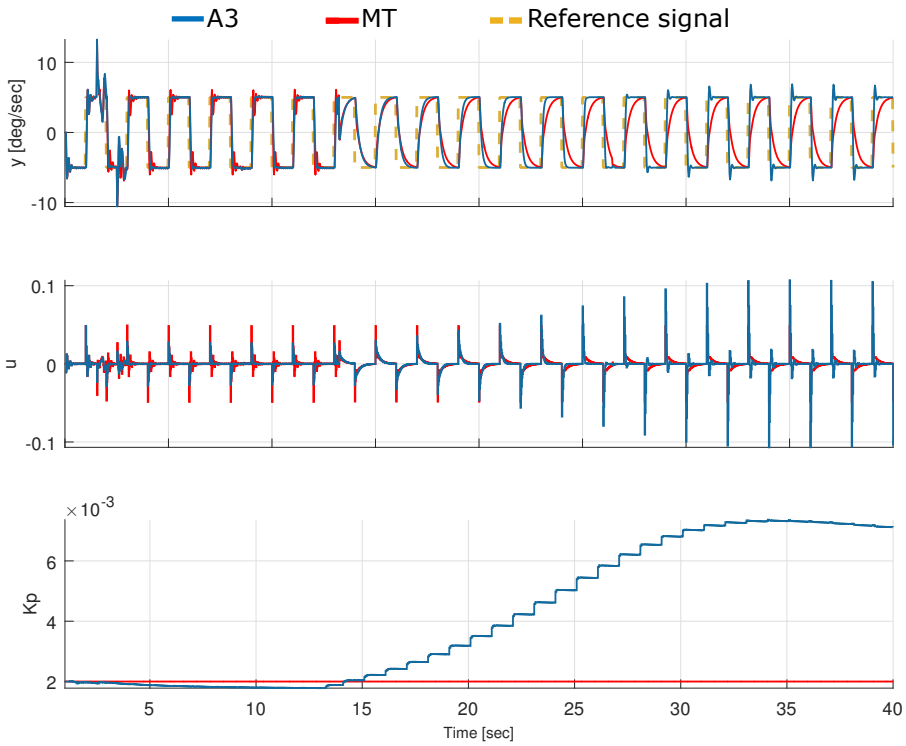

Figure 20: T2(C1), x-axis

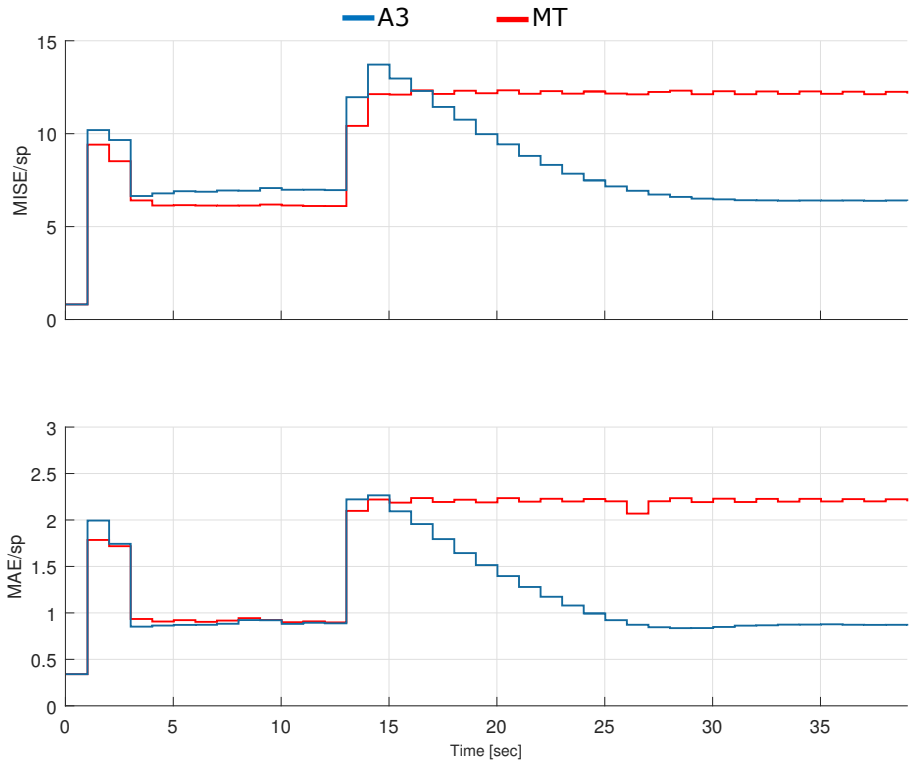

Figure 21: T2(C1), x-axis, errors

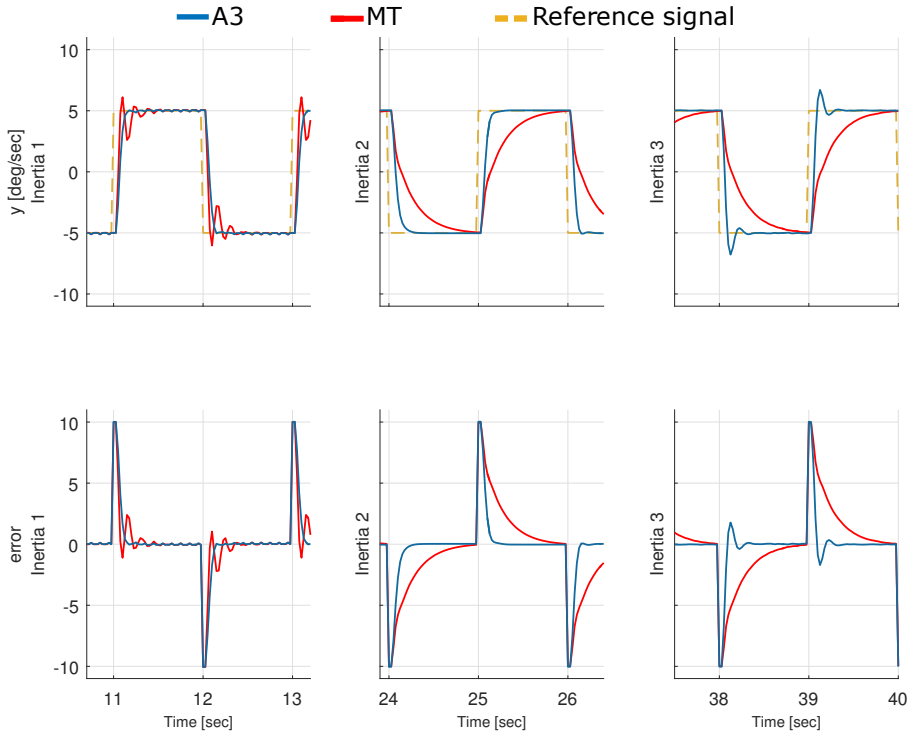

Figure 22: T2(C1), x-axis, different inertias

T2	A1	A2	A3	MT
MISE	N/A	16.6628	7.8866	10.2727
MAE	N/A	2.5869	1.1403	1.8186

Table 9: T2 results, x-axis

7.4 Test 3 - battery degradation

The current section compares different methods through **T3**. In accordance with the previous test **T2**, three figures are presented for each axis. The Figures 23, 26, 24, 27 provide the same information as in **T2**. The third figure for each axis, 25 and 28 shows different moments in time during battery degradation.

7.4.1 Z-axis

Figure 23 clearly shows the adaptation and robustness of **A3**. As the battery is depleted and the actuators loose performance, the overall performance of the controller is kept in a steady state. Looking at the control signal u, it can be seen that the actuation is magnified over time in proportion to the loss of power. A clear picture of how the conventional controller with MT looses performance is shown in Figure 24, in difference to the proposed approach **A3** which keeps a steady error. Analyzing **A1** in Figures 45 and 46, there seems to be no clear adaptation. Instead performance seems to increase with battery degradation. For **A2** based on Figures 48 and 49, adaptation is similar to **A3**, though performance is lower and there is more oscillation. Looking at the error, it can also be seen that it is not steady as for **A3**. Although fairly unchanged, it varies in an unpredictable manner.

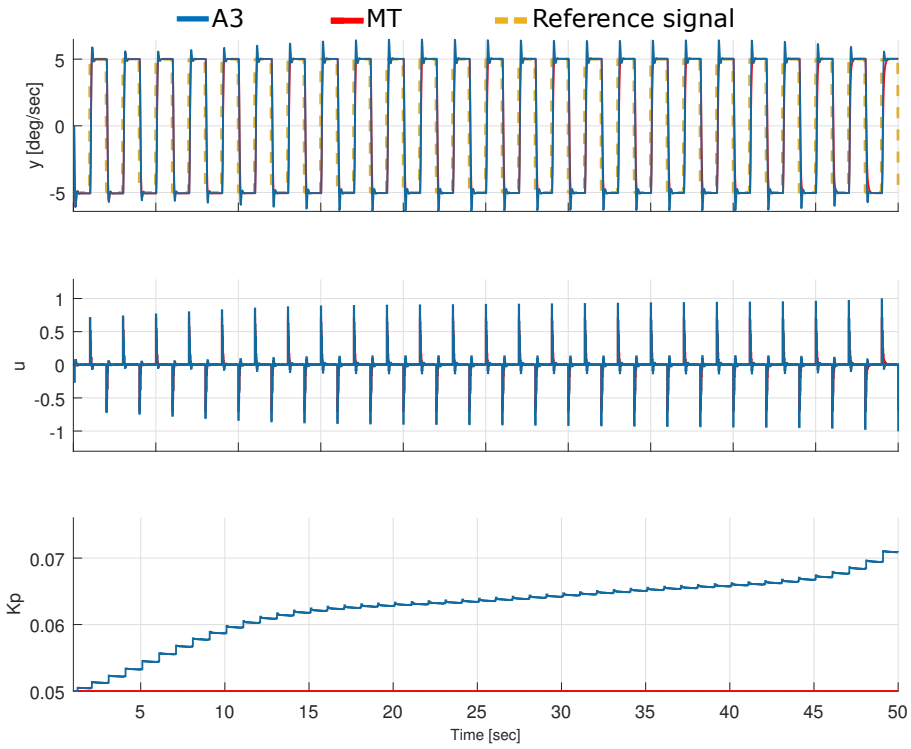

Figure 23: T3(C1), z-axis

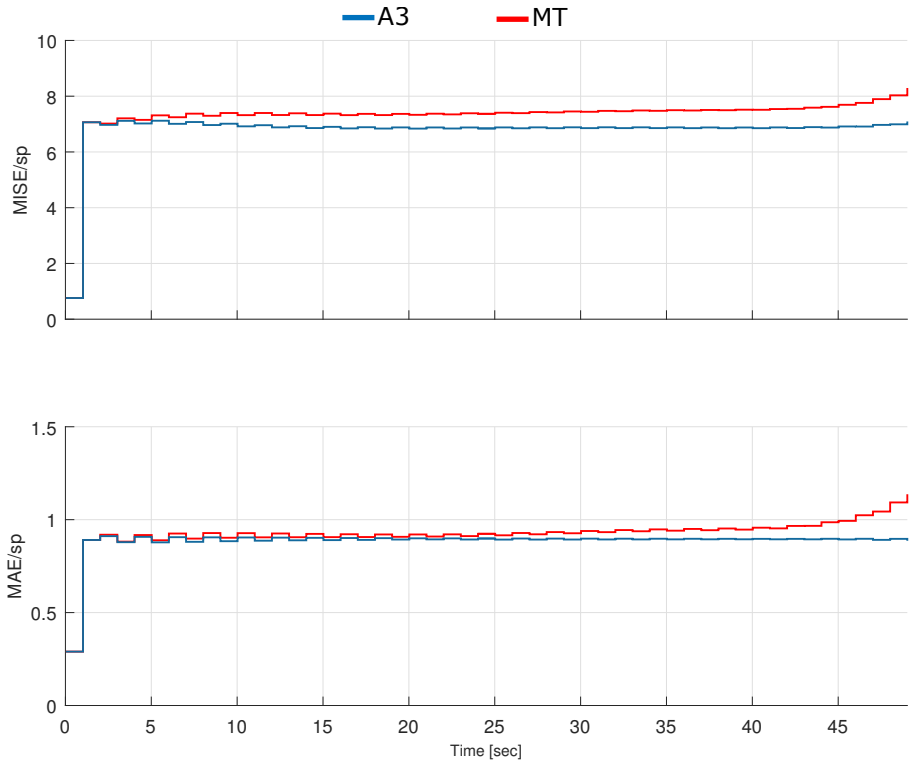

Figure 24: T3(C1), z-axis, errors

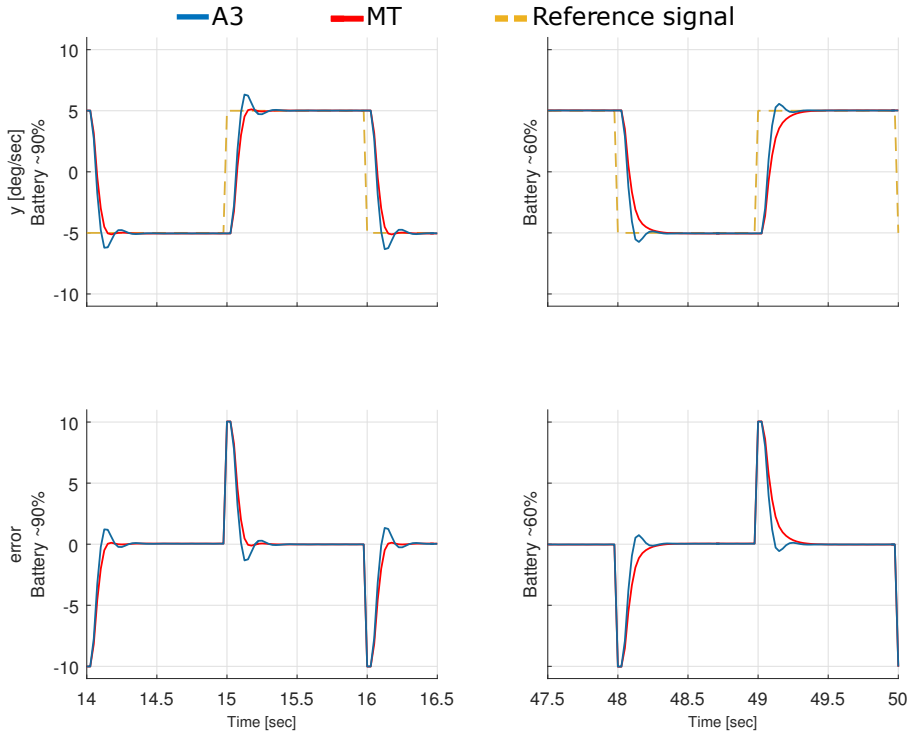

Figure 25: T3(C1), z-axis, beginning and end

T3	A1	A2	A3	MT
MISE	43.1395	11.270	6.844	7.365
MAE	5.6957	2.736	0.888	0.932

Table 10: T3 results, z-axis

7.4.2 X-axis

The same test for the x-axis shows the very same trend. In Figure 26 the augmented control signal can be seen over time as well as the increased K_p. Looking at the errors in Figure 27 they don't show the same steady line as for the z-axis for **A3** but the similarity is big as the error is kept low over time as per the MT error grows. Looking at Figure 28 it can be seen that the performance is similar at both the start and end of the simulation for **A3** as per the performance of the MT gets degraded. For this axis **A1** could not survive the simulation due to instability. **A2** shows good performance and adaptation as can be seen in Figures 51 and 52. Although having a marginally lower overall performance than **A3** it shows a stable and predictable behavior.

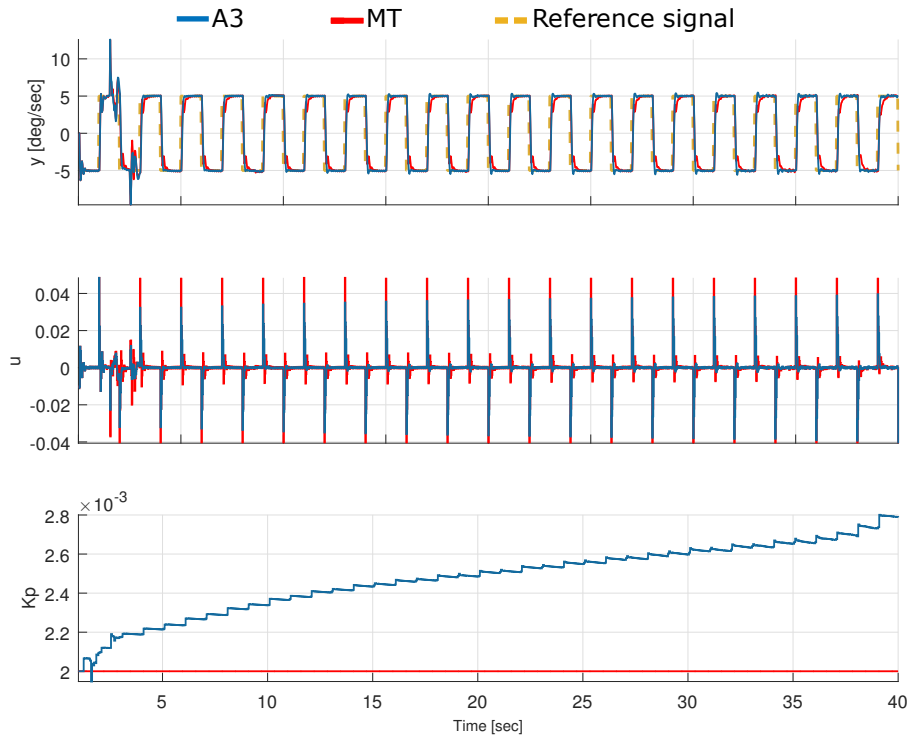

Figure 26: T3(C1), x-axis

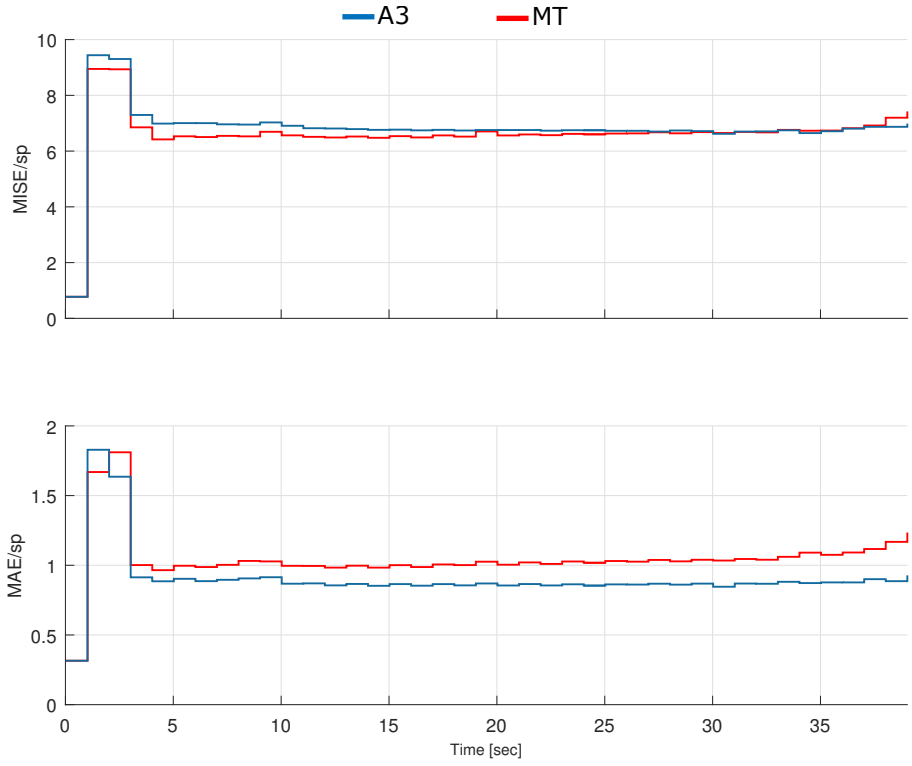

Figure 27: T3(C1), x-axis, errors

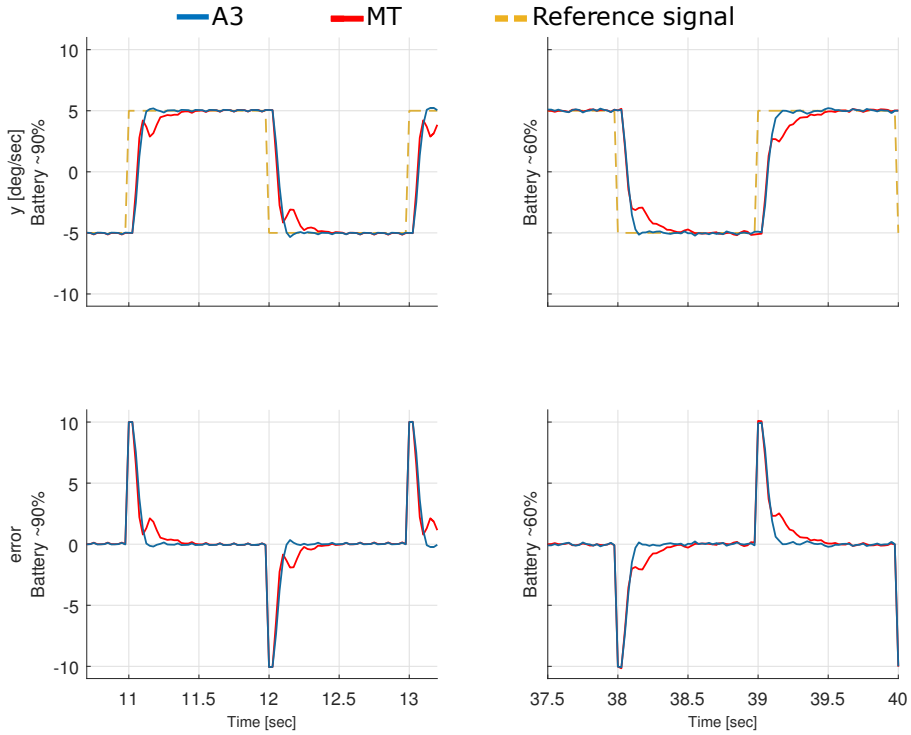

Figure 28: T3(C1), x-axis, beginning and end

T3	A1	A2	A3	MT
MISE	N/A	8.9254	6.857	6.690
MAE	N/A	1.8176	0.909	1.056

Table 11: T3 results, x-axis

7.5 Test 4 - smooth flying

As this test scenario simulates normal flight, it is less predictable compared to previous tests deeming graphical analysis difficult and is hence excluded. Also, as this test consists of position control, the evaluation is done on the utmost control loop for position, including the whole cascade; velocity control, angle control and ultimately angular velocity control. Observe Figure 29, which demonstrates the set-point or target value for the smooth path of **T4**. Results are presented in Table 13. Unfortunately neither **A1** or **A2** made it through the simulation. In difference with previous tests where each adaptive approach is tested on an isolated axis, this test applies adaptation on all three axis at the same time. The evaluation is done under the assumption that z-pos (height control which in it self has no adaptation) as well as x-pos and y-pos may be affected by the performance of the control loops for angular velocity around the quadrotors own axis x, y and z.

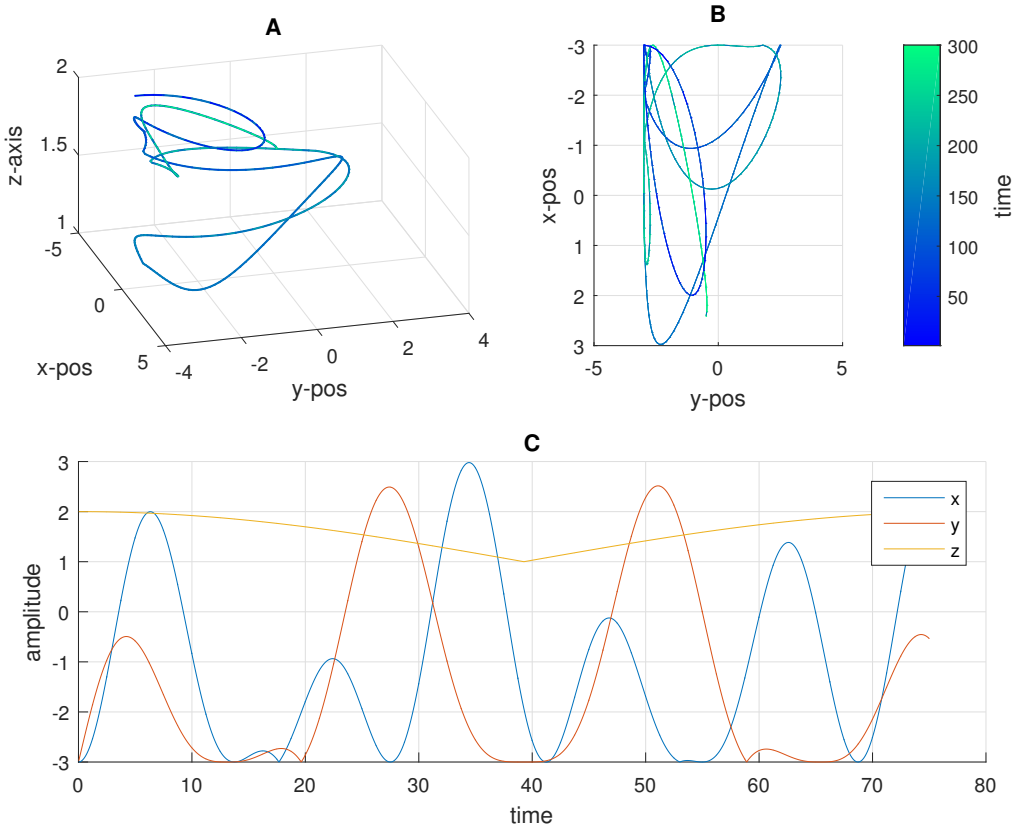

Figure 29: Presents T4 smooth flying, A: smooth path in 3D space, B: smooth path in 2D space, C: smooth path over time in 2D space for each individual axis

7.6 Summary

Observing the result in Sections 7.2, 7.3 and 7.4, it is apparent that **A3** minimizes the MAE and MISE compared to MT, **A1** and **A2**. A summary of the results is given in Table 12. Moreover, notice Table 13, which presents the results of a smooth quadrotor flight, it can be seen from the numerical results of the MISE and MAE that the method **A3** performs better than other approaches. **A3** is also superior with respect to settling times, predictability and stability.

Test			A1	A2	A3	MT
z-axis	T1	MISE	48.953	11.586	6.691	6.705
		MAE	5.857	2.812	0.947	0.923
	T2	MISE	33.255	17.002	7.508	9.677
		MAE	4.665	3.249	1.055	1.448
	T3	MISE	43.139	11.270	6.844	7.365
		MAE	5.6957	2.736	0.888	0.932
x-axis	T1	MISE	N/A	23.495	7.185	6.547
		MAE	N/A	2.835	1.033	1.058
	T2	MISE	N/A	16.663	7.887	10.273
		MAE	N/A	2.587	1.140	1.819
	T3	MISE	N/A	8.925	6.857	6.690
		MAE	N/A	1.818	0.909	1.056

Table 12: Result summary of the tests; **T1**,**T2** and **T3**

T4			A1	A2	A3	MT
Voltage drop	x-pos	MISE	N/A	N/A	11.069	12.303
		MAE	N/A	N/A	0.959	1.003
	y-pos	MISE	N/A	N/A	180.763	192.672
		MAE	N/A	N/A	4.119	4.305
	z-pos	MISE	N/A	N/A	9.780	11.416
		MAE	N/A	N/A	0.825	0.932
Fixed voltage	x-pos	MISE	N/A	N/A	11.223	11.598
		MAE	N/A	N/A	0.953	0.927
	y-pos	MISE	N/A	N/A	155.298	157.372
		MAE	N/A	N/A	3.589	3.679
	z-pos	MISE	N/A	N/A	10.117	11.992
		MAE	N/A	N/A	0.846	0.914

Table 13: Results of the test **T4**

8 Discussion

It has been shown that the proposed approach by this thesis adapts in a good manner to different situations. Under unchanged conditions, **A3** demonstrates similar performance as MT (**T1**), it has shown its capability of increasing performance over time after changed inertia (**T2**) and that errors are kept at a minimum during battery drain (**T3**). By analyzing the graphs it can be stated that errors decrease along with the convergence of the adaptation after a change in dynamics and reach a minimum in both the MISE and MAE close to the one present before any system changes where applied. This as long as the system response is not limited by actuator saturation as can be seen in the last third of Figures 17 and 18. **T4** shows that adding adaptiveness to the system increases performance also in control loops higher up in the chain as can be seen in Table 13 where even tests without battery degradation show better results.

In **T2**, numerical results are conclusive and show the superiority of the proposed approach **A3**. Also looking at Figure 21, it is clear that performance is restored to its optimum after a big dynamical change.

For **T3**, the same cannot be said numerically as the x-axis performs slightly worse than MT. Although looking at the graphs for the error on the x-axis in Figure 27, it is clear that as MT is constantly loosing performance along with battery drain, the proposed approach is reducing the error over time proving good adaptation and performance. If the simulation would run for an increased amount of time the error would increase for MT while it would stay approximately constant for **A3**. The reason simulation is not ran longer for the x-axis, is because of the constrains in position that exist in the simulation environment. As position control is deactivated for the x-axis during this test, the quadrotor drifts away leading to the termination of the simulation.

It should be noted that in the comparison **T1**(**C1**), results will depend much on the precision of the empirical tuning process. For each case, as both the MT and adjustment of the TRs for **A3** have been done manually (see Sections 5.4.3 and 6.3), results are hard to evaluate. It should be stated that performance according to the results both numerical and graphical show similar performance for both **A3** and MT on both the z and x-axis in **T1**. On another note, MT stays robust while loosing performance when inertia is increased or battery degradation is present. In opposite scenarios as when inertia is decreased or if the power to weight ratio is changed (case of less weight or more powerful motors), MT decreases in stability while **A3** is consistent in its performance.

8.1 RLS convergence

For the RLS algorithm, there are two scenarios to evaluate. First, the convergence of "zeroed" or otherwise called untrained data sets meaning the weights are all initialized to zeros. This is not a realistic scenario as it is best to have a moderate model approximation when introducing and activating the proposed approach on a system. Although, it is of interest to see how fast a good model approximation can be found. In Section 7.1 such a scenario is ran. Adaptation is activated at 0.375 seconds and the RLS converges towards a solution reaching good performance quickly. The simulation is ran for 150 iterations or 3.75 seconds in which it can be assessed that an arbitrary solution is found.

The second scenario is when a system model is known and then system changes are applied. Looking at the graphs for the test of changed inertia in Figures 17 and 18 for the z-axis and Figures 20 and 21 for the x-axis, it can be seen that upon a change, it takes between 10 to 20 seconds for the proposed approach to fully converge.

Lastly, the behavior of **T3** can be mentioned when changes are made in a smooth matter. Here adaptation keeps up well with changes. Battery degradation of approximately 40% is emulated over a time period of 50 seconds for the z-axis and looking at Figure 24 it is hard to see any decrease in performance. For the x-axis, the simulation is done over 40 seconds and looking at the

errors in Figure 27, it can be noted that adaptation is slightly slower but that overall performance is kept high throughout the entire simulation. As mentioned previously, a trend can be seen that the performance of the proposed approach over a longer simulation, with battery degradation, **A3** will grow in superiority against MT.

8.2 Model accuracy

According to the results, it is clear that the I^2PD model corresponds to a better approximation of the system compared to the FOPDT model. The dynamics of a process may be determined from the response of a pulse, step, ramp or other deterministic signals. However, this requires that the system is at rest before the input is applied. Thus, in practice it may be hard to estimate a system response and determine it′s dynamics. This may be due to external factors such as disturbance or the difficulty of isolating such a test to one specific axis. Be aware that stabilization on the remaining axis will affect this evaluation. Bare in mind also that altering variables such as particle dynamics, or sensor noise on a real implementation may disturb the system. As a result, the initial assumption of the FOPDT being a good model approximation provided poor performance (observe appendix Section 12, results from **A1**). Additionally, a first order system does not overshoot or oscillate, it is limited to the extent of not containing the property of inertia. Thus, the dynamic of a quadrotor system contains the characteristics of inertia which provided the inability to obtain desirable behavior. However, as the simulation evaluation continued, it was found that the I^2PD model corresponded to a more accurate behavior of a quadrotor system. The I^2PD is a simple approximation of a model and holds the property of being a non self regulating process. It basically implies that the processes has integrating characteristics compared to the FOPDT which by itself is a self regulating process (no integrating characteristic).

8.3 Real-time capabilities

In the simulations performed in this thesis, control has been done in MATLAB and the actuation sent to V-REP as can be read about in Section 6. The process runs in a synchronous way, waiting for the next control signal to be sent from MATLAB for V-REP to continue the simulation which runs for a period of 25 ms each iteration. This configuration is in this regard not running in real-time. Although, the algorithm itself is found to be efficient and CPU cheap. It has in total 35 multiplication operators, 3 division operators and 3 plus and minus operators on individual floats. There are some more demanding operators as square products that can be ran just once in the beginning as they won't change during runtime. These initial calculations present values used in the process of finding the correct TRs for the I^2PD model. They are depend on the delta-time which is normally known in the system and the dead-time for each axis which will be a fixed value once identified. The delta-time, being the time between each control loops might vary from time to time depending on the implementation off the systems scheduler. If it is not constant it will induce another square operator that will have to run in real-time. Some changes may be needed to the PID controller depending on the platforms implementation and may add some extra operators. The added load of these operations is assessed to be relatively low in comparison to other algorithms running on these kind of systems. The assessment therefore concludes that the proposed approach could be implemented on a real platform an be ran on-line in real-time together with the control loops which in practice normally means a frequency in the magnitude of kHz.

9 Conclusion

This master thesis has presented a design for an adaptive control system. The work has focused on a solution for radio controlled quadrotors where the algorithm is computationally efficient. The method aims at automatically tuning a PID controller in real-time by combining the I^2PD model and RLS. With the use of an adaptive tuning algorithm, it has been shown that a quadrotor may be robust against unpredictable events. Additionally, it is shown that an adaptive solution may improve performance by minimizing the MISE and MAE with respect to conventional tuning methods.

The research questions described in Section 2.3.1 have been answered by executing a variety of tests explained in Section 7. As a result, it is apparent that the exposure to unpredicted anomalies decreases performance of a conventional, manually tuned control system. However, the suggested adaptive approach of this thesis will counteract such anomalies. The algorithm has the ability to detect system changes and adapt accordingly. A disadvantage of the proposed approach emerges as an initially untrained RLS demands a certain amount of data before it can provide an arbitrary system model. A faster training will be obtained if the input is stimulated (a diversity of input signals) as the identification will have more data to use when finding the relations between the input-output signals. During that time, the quadrotor may decrease its performance and enter unbridled states. Due to this reason, it is of most importance to provide the RLS algorithm with adequate data before the entire algorithm (both MI and adaptation) is activated. This requires a control system to handle the flight at an initialization stage of the RLS.

The main advantage of the proposed approach is that the solution is computationally efficient. The exhibited calculations may be implemented to several existing platforms without the demand of extensive CPU power. The proposed approach is adjusted to suit a system as an extension. Thus, it is essential to consider the structure of the present PID controller and make proper adjustments if needed.

10 Future work

In order to continue the work and confirm the simulation results, extended comparisons against other methods could be done. An interesting evaluation would be to compare it against the performance of a MPC. Thus, it is of great importance to consider the added computational and memory requirements. The evaluation would involve minimizing the prediction horizon while trying to maintain desirable performance of the MPC. Another step would be to implement the thesis approach onto a real UAV quadrotor platform. However, there are several aspects which needs to be taken into account when implementing **A3** to a real-time system. Firstly, it is of great importance to identify which kind of PID-controller is used. It has to conform with the the control structure of **A3**. Secondly, data structures and code optimization have to be revised before implementation is done on a real-time system with respect to the system hardware.

A restriction of a real-time implementation and evaluation is the difficulty of executing similar tests. It is in practice impossible to execute the exact same tests due to factors like wind disturbances, initial altitude positions or to keep a constant battery charge.

Lastly, to further improve the adaptive tuning algorithm, an extended identification approach of the dead-time for each axis could be developed. It is also desirable to continue the work by developing a method which defines the accuracy of the RLS model. This may be used as a tool to define when the adaptiveness should be activated on a new platform.

11 Acknowledgment

We would like to express our gratitude towards Alessandro Papadopoulos who supervised us through the project by good engagement and valuable remarks on the work. We would also like thank Mikael Ekström and Giacomo Spampinato who provided valuable input to the thesis structure and general advice in the initial stages of this work. Finally we would like to thank our loved ones which have supported us throughout the entire process.

References

[1] D. Sonntag, "A study of quadrotor modelling," Master's thesis, Linkping University, Department of Computer and Information Science, 2011.

[2] S. Bagheri, "Modeling, simulation and control system design for civil unmanned aerial vehicle (UAV)," Master's thesis, Umeå University, Department of Applied Physics and Electronics, 2014.

[3] Y. Mou, Q. Zhang, S. Liu, and K. Liang, "The flight control of micro quad-rotor UAV based on PID," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Nov 2016, pp. 353–356.

[4] D. H. W. S. Zhang Jinhua, Zhuang Jian, "Self-organizing genetic algorithm based tuning of PID controllers," Nov 2008.

[5] G. Lin and G. Liu, "Tuning PID controller using adaptive genetic algorithms," in *2010 5th International Conference on Computer Science Education*, Aug 2010, pp. 519–523.

[6] K. J. Åström and T. Hägglund, "PID controllers: theory, design, and tuning," 1995.

[7] K. J. Åström and T. Hägglund, *Advanced PID Control*. ISA - The Instrumentation, Systems and Automation Society, 2006.

[8] M. Zhuang and D. P. Atherton, "Automatic tuning of optimum PID controllers," *IEE Proceedings D - Control Theory and Applications*, vol. 140, no. 3, pp. 216–224, May 1993.

[9] A. O'Dwyer, "Handbook of PI and PID controller tuning rules," 2009.

[10] J. J. Gude and E. Kahoraho, "Kappa-tau type pi tuning rules for specified robust levels: The frequency response method," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, Sept 2012, pp. 1–8.

[11] "Cleanflight," http://cleanflight.com/, accessed: 2017-05-08.

[12] "Betaflight," https://github.com/betaflight/betaflight/wiki, accessed: 2017-05-08.

[13] "Ardupilot," http://ardupilot.org/, accessed: 2017-05-08.

[14] "Librepilot," https://www.librepilot.org, accessed: 2017-05-08.

[15] "Multiwii," http://www.multiwii.com/, accessed: 2017-05-08.

[16] P. M. Meshram and R. G. Kanojiya, "Tuning of PID controller using Ziegler-Nichols method for speed control of dc motor," in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, March 2012, pp. 117–122.

[17] Z. Juric and B. Perunicic, "An extension of the ziegler-nichol's method for parametric identification of standard plants," in *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No.04CH37521)*, vol. 1, May 2004, pp. 363–366 Vol.1.

[18] P. Cominos and N. Munro, "PID controllers: recent tuning methods and design to specification," *IEE Proceedings - Control Theory and Applications*, vol. 149, no. 1, pp. 46–53, Jan 2002.

[19] M. G. Na, "Auto-tuned PID controller using a model predictive control method for the steam generator water level," *IEEE Transactions on Nuclear Science*, vol. 48, no. 5, pp. 1664–1671, Oct 2001.

[20] M. S. Holzel and E. A. Morelli, "Real-time frequency response estimation from flight data," 2012.

[21] E. Yu and Y. Hu, "A novel modified PID controller applied to temperature control with self-tuning ability," in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 7025–7029.

[22] K. J. Åström and T. Hägglund, "Automatic tuning and adaptations," *PID controllers: theory, design, and tuning*, pp. 230 – 243, 1995.

[23] L. G. Junior, J. O. P. Pinto, J. A. B. Filho, and G. Lambert-Torres, "Recursive least square and genetic algorithm based tool for PID controllers tuning," in *2007 International Conference on Intelligent Systems Applications to Power Systems*, Nov 2007, pp. 1–6.

[24] M. A. Al-Shabi, K. S. Hatamleh, and A. A. Asad, "UAV dynamics model parameters estimation techniques: A comparison study," in *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Dec 2013, pp. 1–6.

[25] O. M. K.S. Hatamleh and R. Paz, "A uav model parameter identification method," 2009.

[26] H. Tnunay, M. Q. Abdurrohman, Y. Nugroho, R. Inovan, A. Cahyadi, and Y. Yamamoto, "Auto-tuning quadcopter using loop shaping," in *2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, Nov 2013, pp. 111–115.

[27] N. J. Killingsworth and M. Krstic, "PID tuning using extremum seeking: on-line, model-free performance optimization," *IEEE Control Systems*, vol. 26, no. 1, pp. 70–79, Feb 2006.

[28] J. O. Pedro and A. J. Crouse, "Direct adaptive neural control of a quadrotor unmanned aerial vehicle," in *2015 10th Asian Control Conference (ASCC)*, May 2015, pp. 1–6.

[29] W.-D. Chang, R.-C. Hwang, and J.-G. Hsieh, "A multivariable on-line adaptive PID controller using auto-tuning neurons," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 1, pp. 57 – 63, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095219760300023X

[30] Y. Wang, H. Zhang, and D. Han, "Neural network adaptive inverse model control method for quadrotor UAV," in *2016 35th Chinese Control Conference (CCC)*, July 2016, pp. 3653–3658.

[31] Q. Lin, Z. Cai, Y. Wang, J. Yang, and L. Chen, "Adaptive flight control design for quadrotor UAV based on dynamic inversion and neural networks," in *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, Sept 2013, pp. 1461–1466.

[32] A. Visioli, "Tuning of PID controllers with fuzzy logic," *IEE Proceedings - Control Theory and Applications*, vol. 148, no. 1, pp. 1–8, Jan 2001.

[33] K. S. Tang, K. F. Man, G. Chen, and S. Kwong, "An optimal fuzzy PID controller," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 4, pp. 757–765, Aug 2001.

[34] J. O. Pedro and C. Mathe, "Nonlinear direct adaptive control of quadrotor UAV using fuzzy logic technique," in *2015 10th Asian Control Conference (ASCC)*, May 2015, pp. 1–6.

[35] P. Nyxn and G. Borgersen, "Using an evolutionary algorithm to search for control algorithm parameters with a quadrocopter as hardware in the loop," May 2015.

[36] P. Zhang, M. Yuan, and H. Wang, "Self-tuning PID based on adaptive genetic algorithms with the application of activated sludge aeration process," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 2, 2006, pp. 9327–9330.

[37] A. V. O. Arrieta and R. Vilanova, "PID autotuning for weighted servo/regulation control operation," *Journal of Process Control*, vol. 20, pp. 472 – 480, 2010.

[38] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems*, vol. 20, no. 3, pp. 38–52, Jun 2000.

[39] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodriguez, "Model predictive control x2014;a simple and powerful method to control power converters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1826–1838, June 2009.

[40] S. Drogies and D. D. Geest, "Predictive control: propositions for the design methodology," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 1, 1999, pp. 647–651 vol.1.

[41] Z. Peng and L. Jikai, "On new UAV flight control system based on kalman amp; PID," in *2011 2nd International Conference on Intelligent Control and Information Processing*, vol. 2, July 2011, pp. 819–823.

[42] T. Magnusson, "State estimation of UAV using extended kalman filter," Master's thesis, Linkping UniversityLinkping University, Automatic Control, The Institute of Technology, 2013.

[43] M. Pettersson, "Extended kalman filter for robust UAV attitude estimation," Master's thesis, Linköping UniversityLinköping University, Automatic Control, Faculty of Science and Engineering, 2015.

[44] J. Yang, Z. Cai, Q. Lin, and Y. Wang, "Self-tuning PID control design for quadrotor UAV based on adaptive pole placement control," in *2013 Chinese Automation Congress*, Nov 2013, pp. 233–237.

[45] O. K. Ablesimov and A. Y. Gonchar, "Adaptive control system for UAV with state observer," in *2013 IEEE 2nd International Conference Actual Problems of Unmanned Air Vehicles Developments Proceedings (APUAVD)*, Oct 2013, pp. 203–205.

[46] H. Bouadi, A. Aoudjif, and M. Guenifi, "Adaptive flight control for quadrotor UAV in the presence of external disturbances," in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, May 2015, pp. 1–6.

[47] C. Lijia, H. Xiaoxiang, and G. Yang, "Robust adaptive backstepping control of UAV with lumped uncertainties," in *2014 International Conference on Mechatronics and Control (ICMC)*, July 2014, pp. 961–965.

[48] Y. Yang, X. Chen, and C. Li, "Rapid turning maneuver flight control for a high agility UAV using robust adaptive augmented backstepping," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, Aug 2014, pp. 1397–1402.

[49] J. Oyekan and H. Hu, "A novel bacterial foraging algorithm for automated tuning of PID controllers of UAVs," in *The 2010 IEEE International Conference on Information and Automation*, June 2010, pp. 693–698.

[50] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass," in *53rd IEEE Conference on Decision and Control*, Dec 2014, pp. 6149–6154.

[51] M. E. N. Srensen and M. Breivik, "UAV fault-tolerant control by combined l1 adaptive backstepping and fault-dependent control allocation," in *2015 IEEE Conference on Control Applications (CCA)*, Sept 2015, pp. 1880–1886.

[52] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, no. 3, pp. 683 – 694, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109813005360

[53] C. R. GmbH. (2017) Virtual robot experimentation platform (v-rep). [Online]. Available: http://coppeliarobotics.com/

[54] A. Leva, M. Maggio, A. V. Papadopoulos, and F. Terraneo, *Control-Based Operating System Design*. Institution of Engineering and Technology, 2013.

[55] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing: Pearson New International Edition*. Pearson Higher Ed, 2013.

[56] W.Pasman, "Lipo batteries," Apr. 2017. [Online]. Available: http://mmi.tudelft.nl/~wouter/helipage/docs/lipo/lipo.html

12 Appendix 1

The appendix is divided into a number of subsections and includes extended illustrations of the thesis results. It contains the test results for the approaches **A1** and **A2** which are compared with **A3** in the comparisons **C2** and **C3**. The tests are demonstrated in the same manner as in the result Section 7.

12.1 Test 1 - normal operation

In this test a square wave is induced on the input as presented in **T1**. The set-point, sensor values, control signal and the evolution of the K_p value are presented along with the errors. Additionally, notice that there is no log for **A1** on the x-axis as the platform reaches an unstable state and is unable to complete the experiment.

12.1.1 Z-axis FOPDT

The results from **T1** on **A1** show that the system is following its target value but is oscillating at a great magnitude (see Figure 30). The errors presented in Figure 31 show that performance is at least 6 times lower than the one of the proposed approach by the thesis (**A3**). To notice is that the control signal is maxing out at each oscillation.

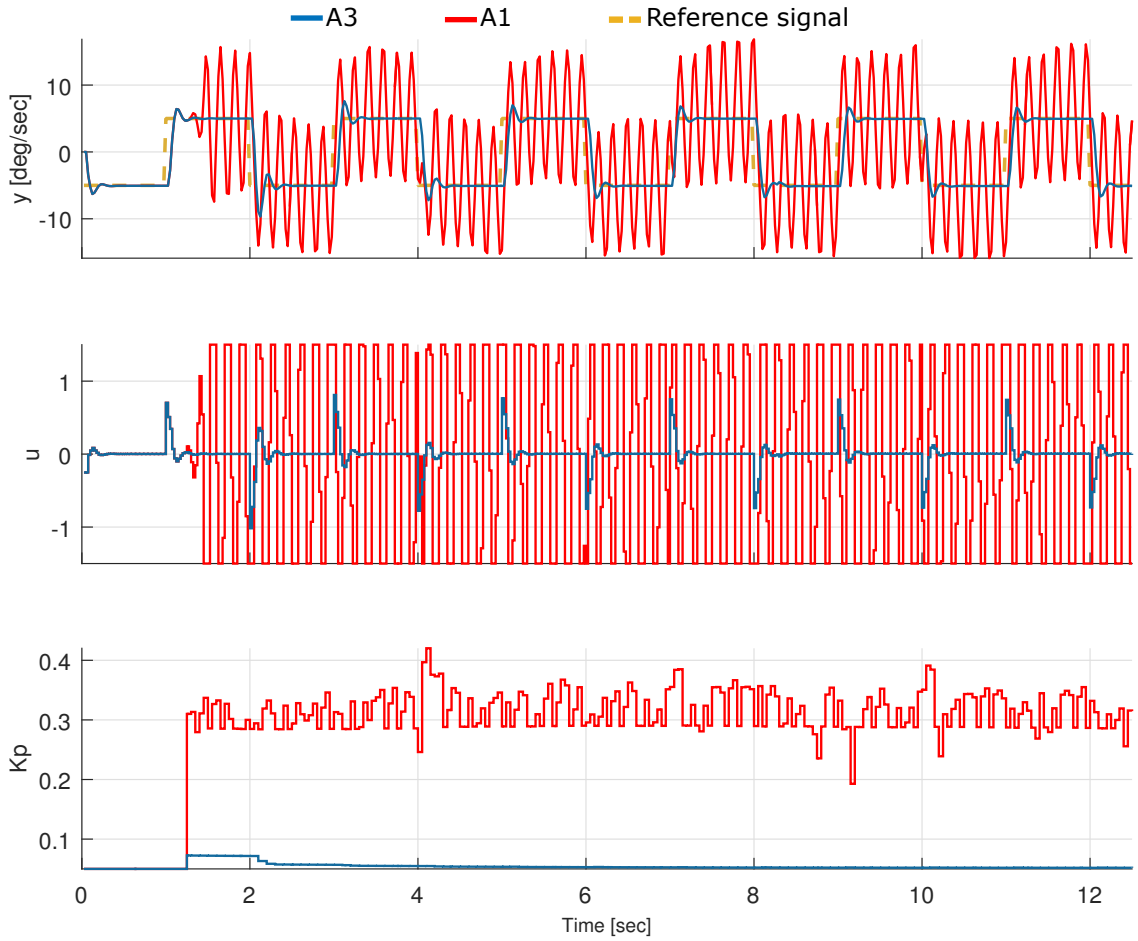

Figure 30: T1(C3), z-axis

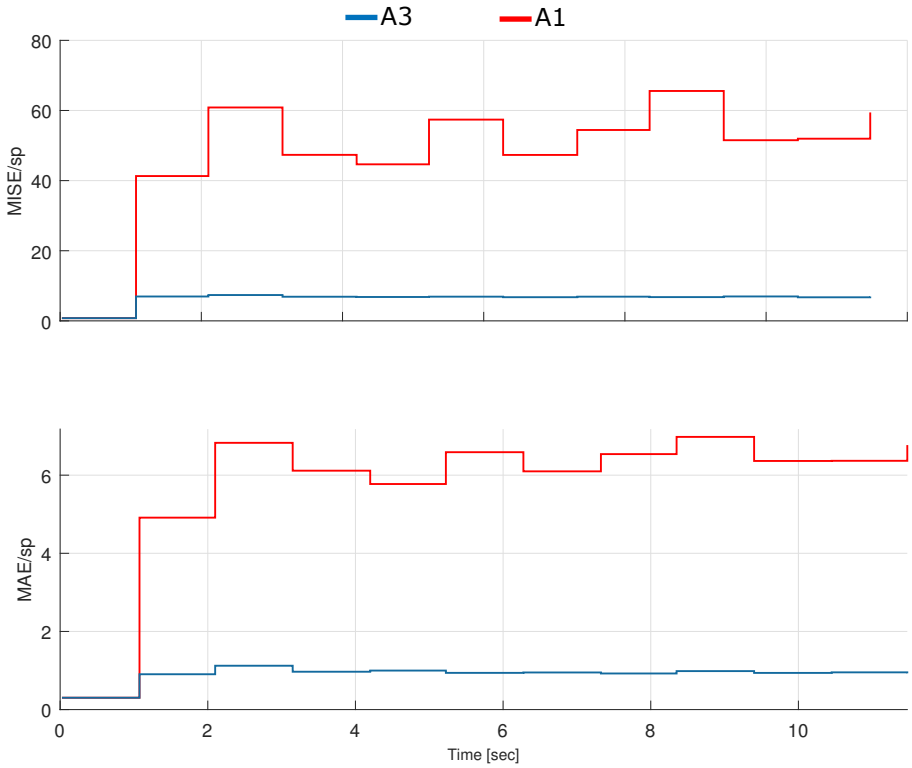

Figure 31: T1(C3), z-axis, errors

12.1.2 Z-axis I^{2}PD

Observe Figure 32, the performance is clearly better than the one of the approach **A2**. Though, oscillations are still big and performance is clearly worse than with the proposed approach **A3**. The system does oscillate around the set-point and stays in a marginally stable. In comparison to **A1**, the control signal never reaches saturation. Errors in Figure 33 show the lower performance compared to **A3**.

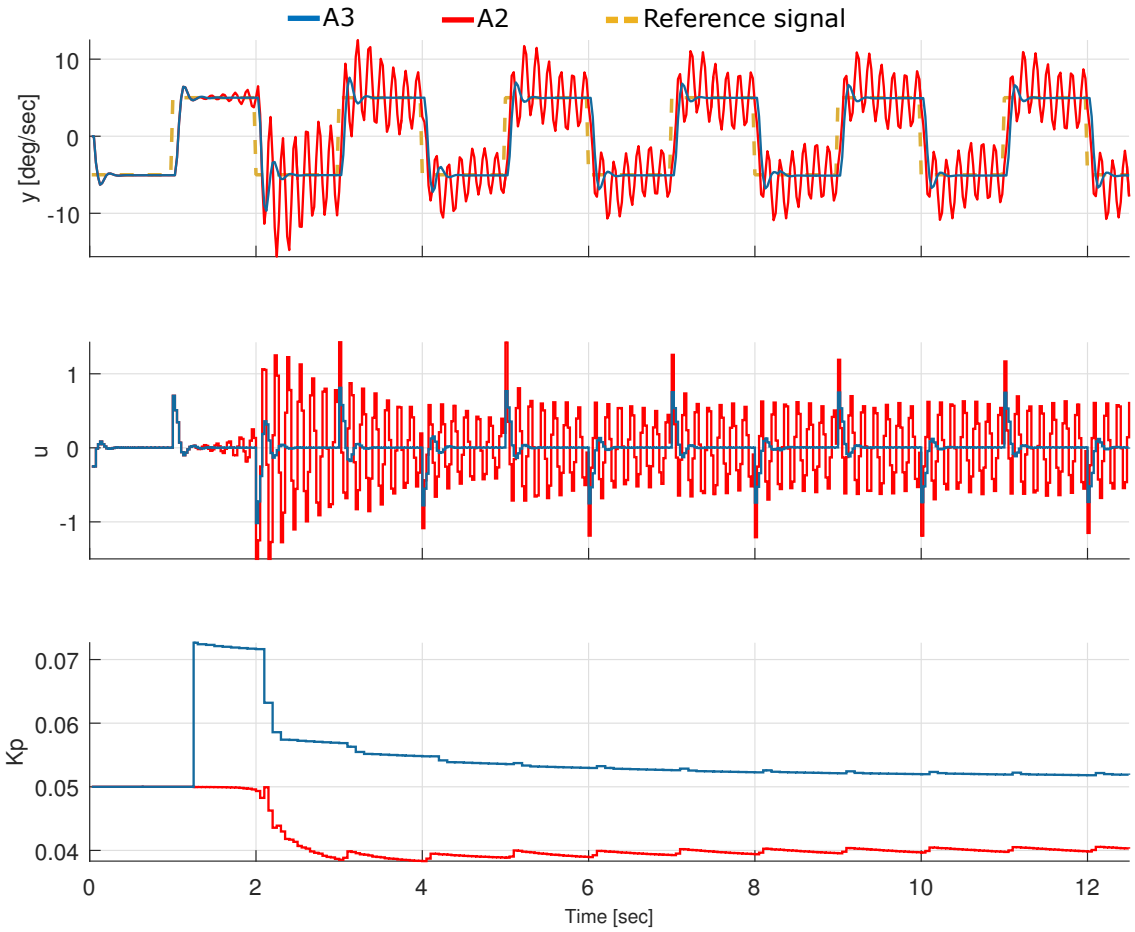

Figure 32: T1(C2), z-axis

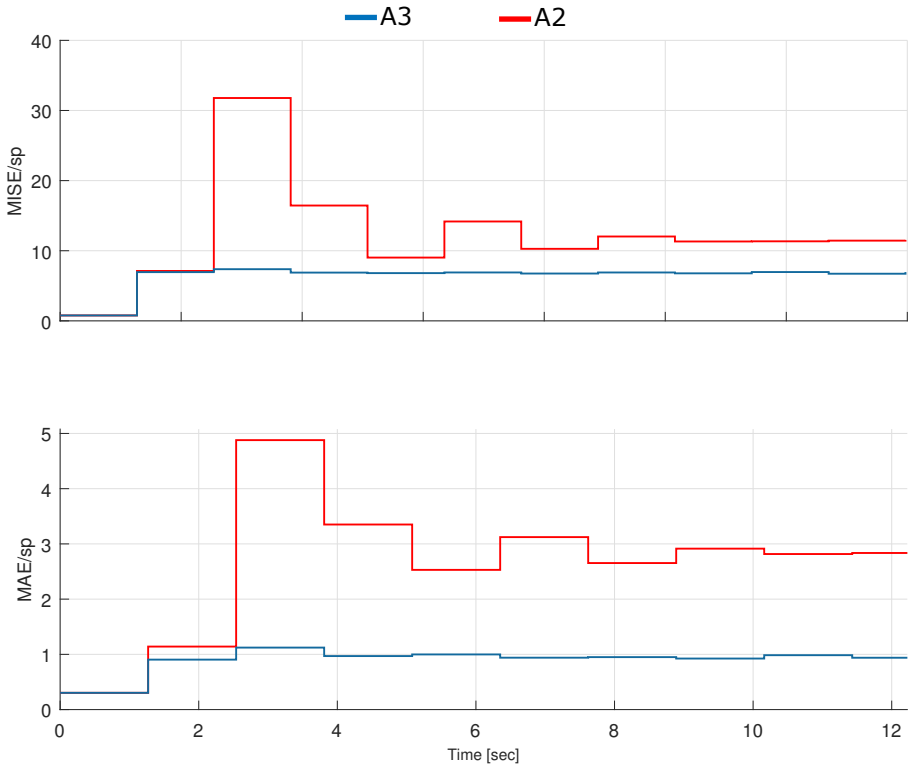

Figure 33: T1(C2), z-axis, errors

12.1.3 X-axis I²PD

For the x-axis, performance is better for **A2** than on the z-axis. The system is stable with constantly decreasing overshoot as seen in Figure 34, the system identification is also adapting, decreasing the K_p value and lowering the error over time as seen in Figure 35. The errors remain bigger on the MAE but keep closing in to the same performance as **A3** on the MISE, although with bigger oscillations. It can be seen that convergence is still ongoing at the end of the simulation.

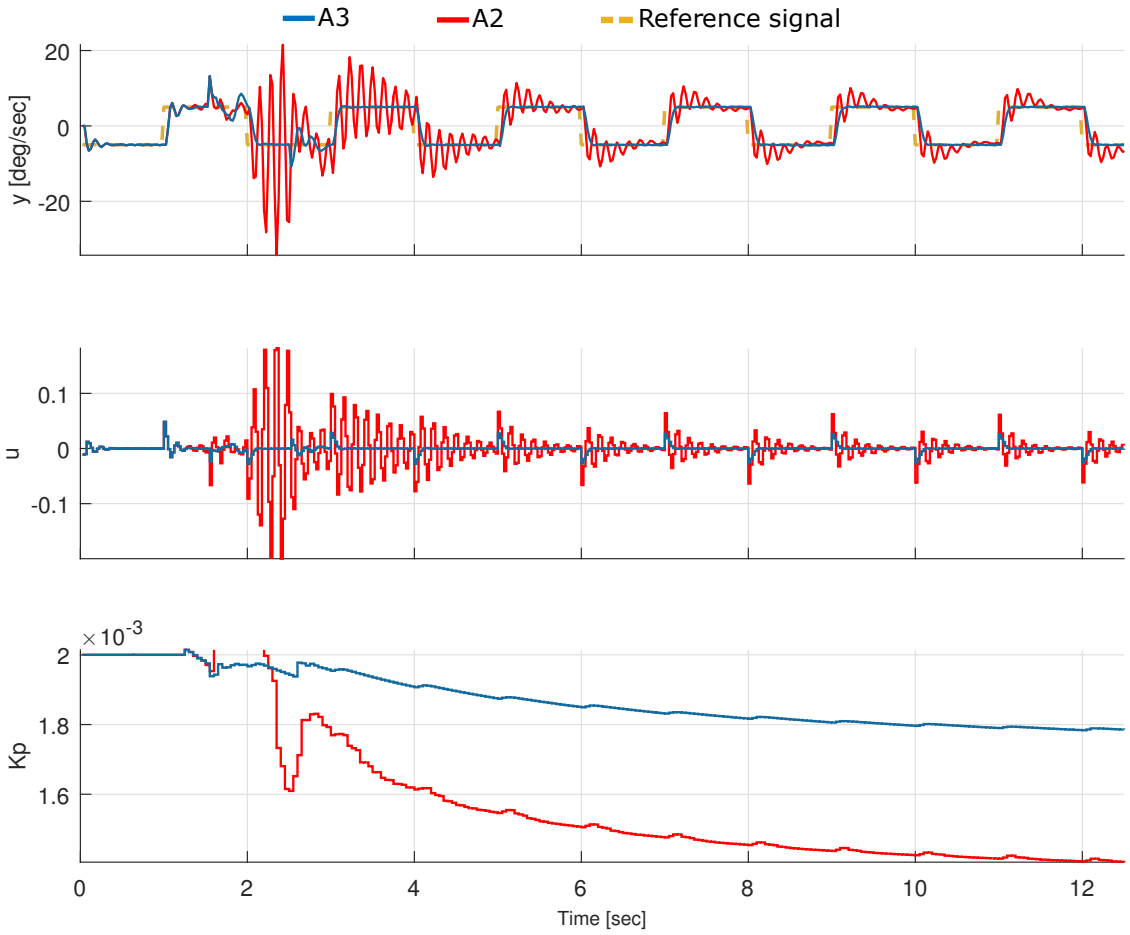

Figure 34: T1(C2), x-axis

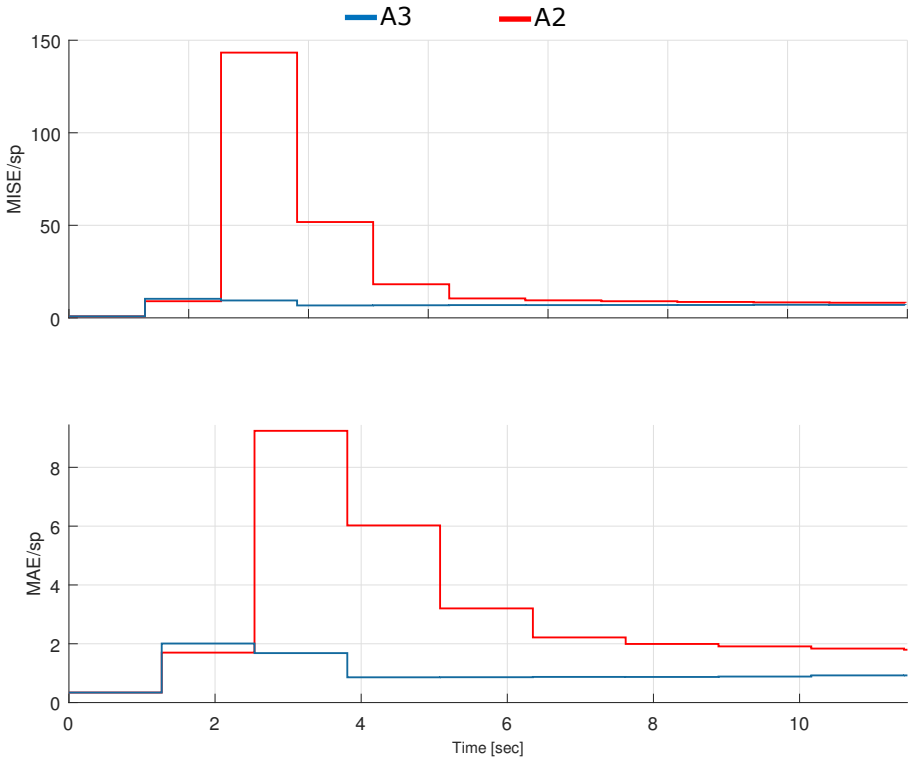

Figure 35: T1(C2), x-axis, errors

12.2 Test 2 - modified inertia

In this test the same square wave is induced as in **T1**. The addition is changes in inertia to evaluate the convergence within different approaches. The test is slightly different for the x and z-axis, see **T2** for a detailed description.

12.2.1 Z-axis FOPDT

On the z-axis, two changes are made dividing the test in three blocks. The starting inertia is higher than what the RLS data is trained on. In Figure 36, the K_p value is increased on the higher inertia, decreased when it is lowered and again increased when the inertia is at last set substantially higher. However, even though the RLS identifies the system model appropriately, performance on this approach is poor. As can be seen from large overshoots and a maxed out control signal. Looking at the errors in Figure 37, the trend shows a decrease over time along with convergence but performance is very poor. Figure 38 shows the performance and error from the last part of each block.

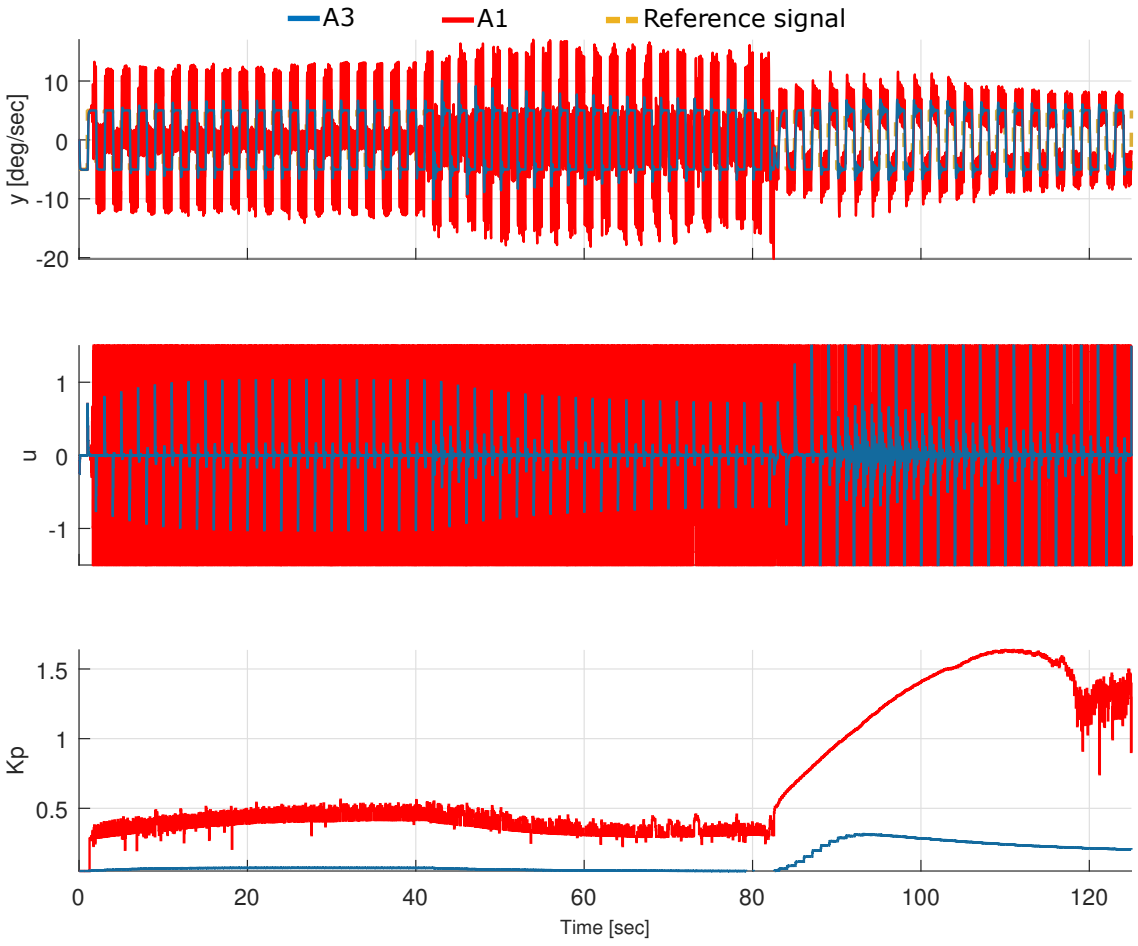

Figure 36: T2(C3), z-axis

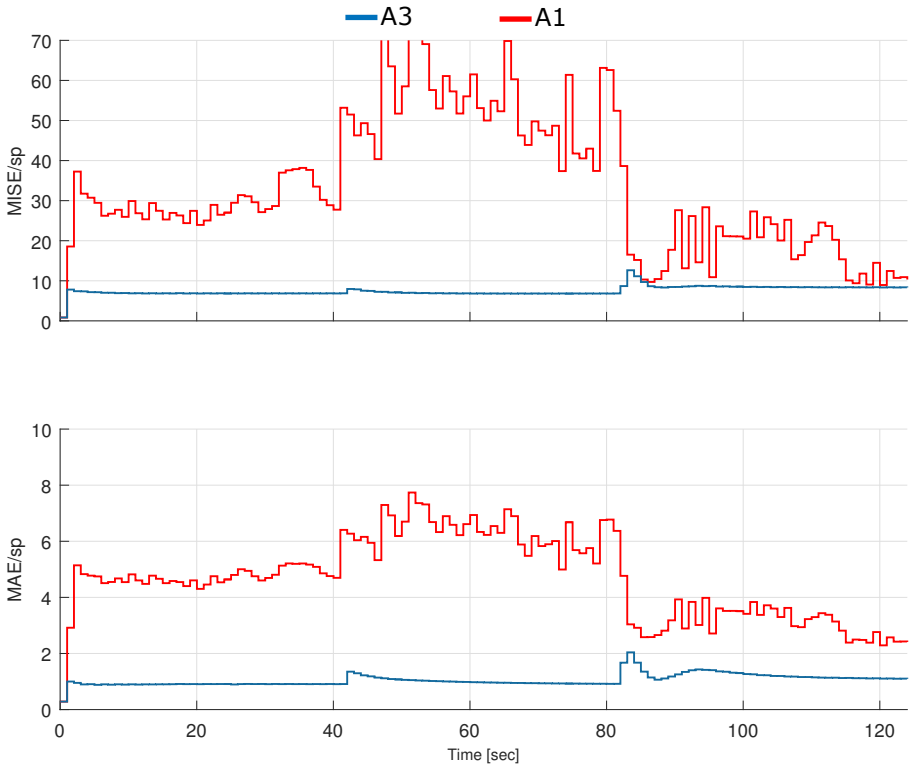

Figure 37: T2(C3), z-axis, errors

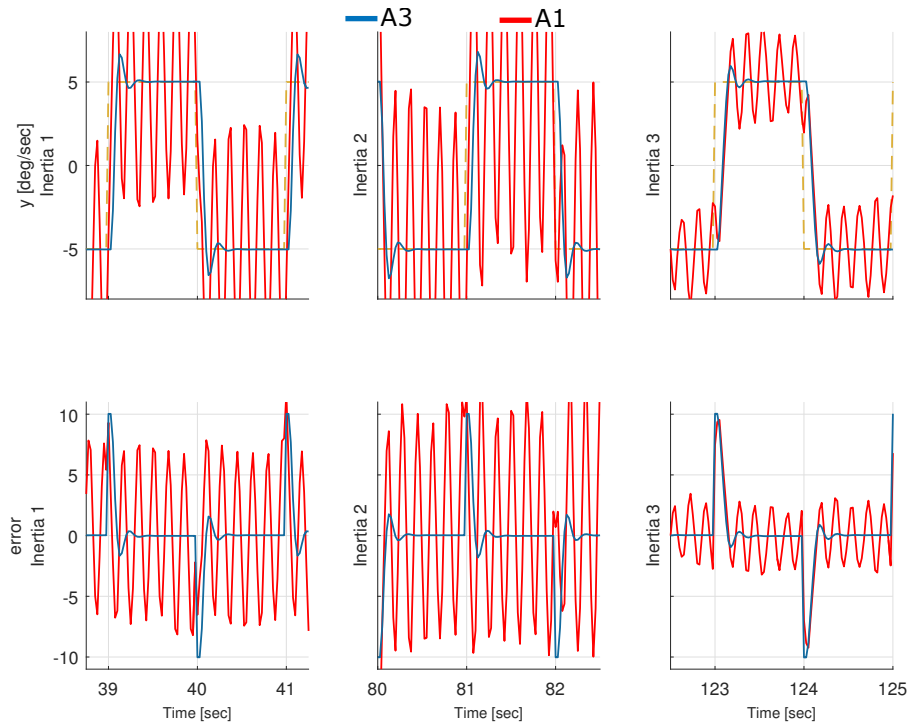

Figure 38: T2(C3), z-axis, changing inertia

12.2.2 Z-axis I²PD

A2 also shows convergence as seen in Figure 39, although performance is poor and the control signal is often maxed out. Errors in Figure 40 show a decrease over time but an inferior performance in comparison to the proposed approach **A3**. Lastly looking at the end of each section in Figure 41, its clear that the the set-point is followed with oscillations for when inertia is increased but the same cannot be said for when it is restored back to its original state in the middle block.

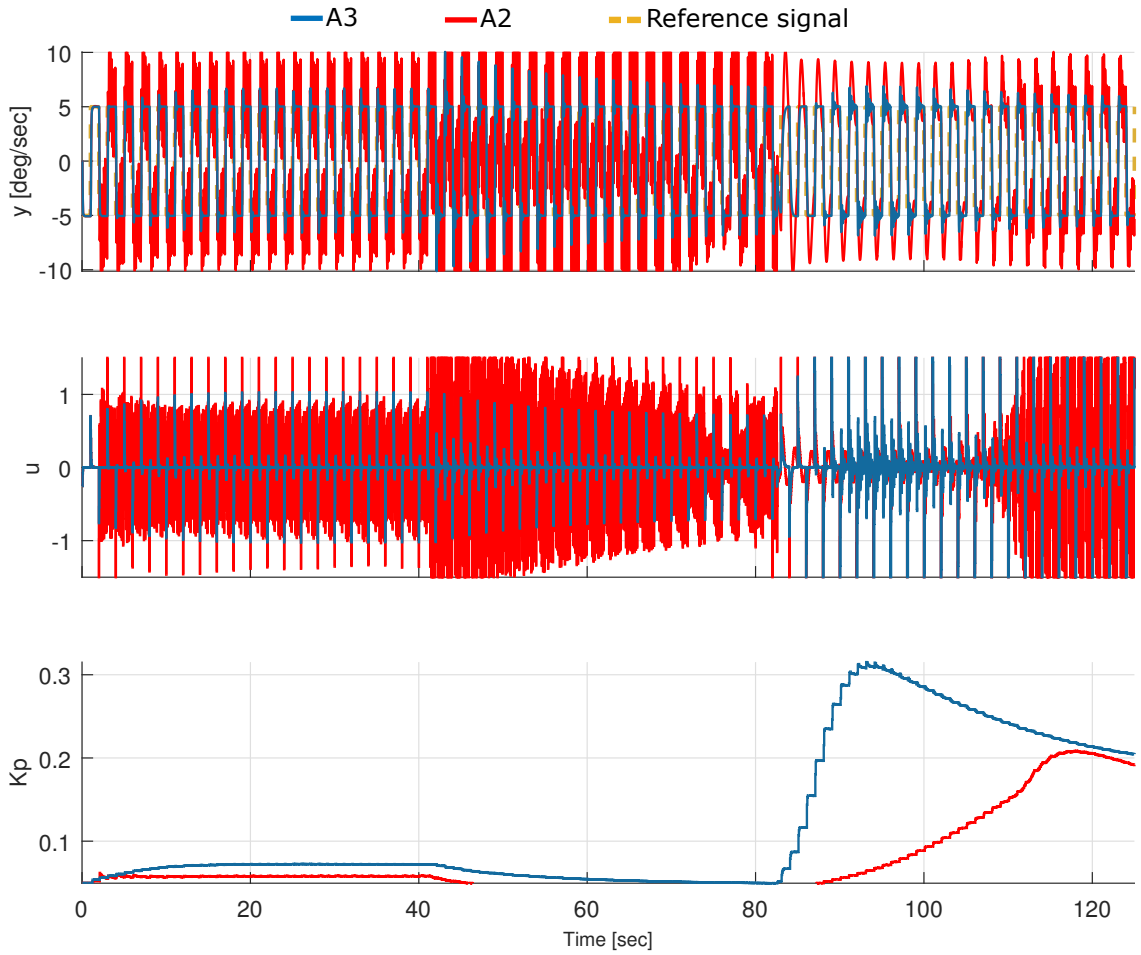

Figure 39: T2(C2), z-axis

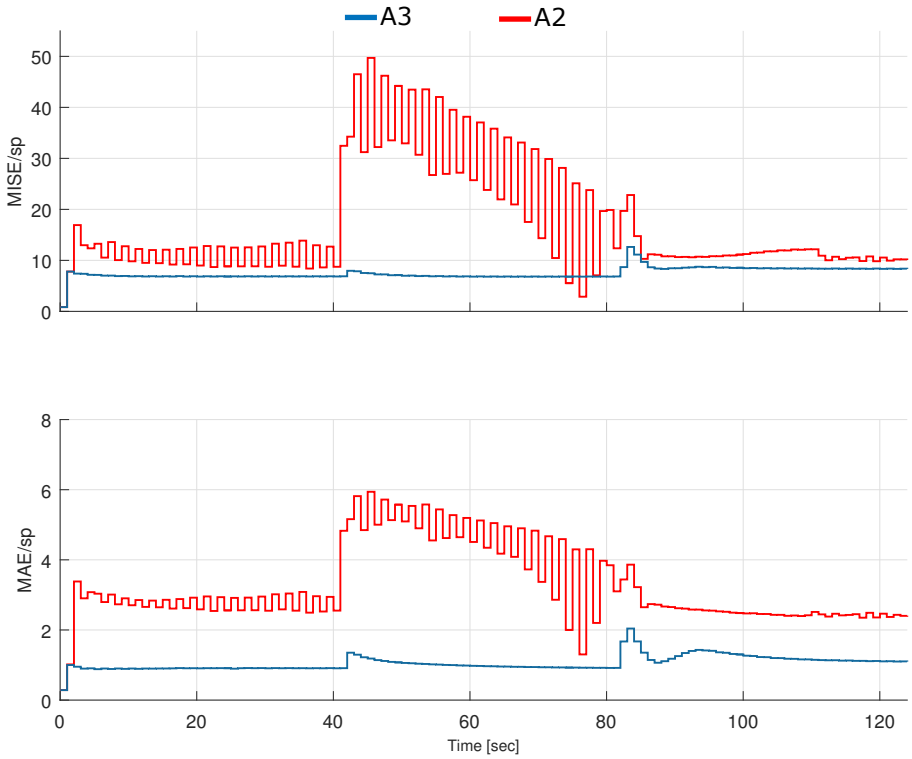

Figure 40: T2(C2), z-axis, errors

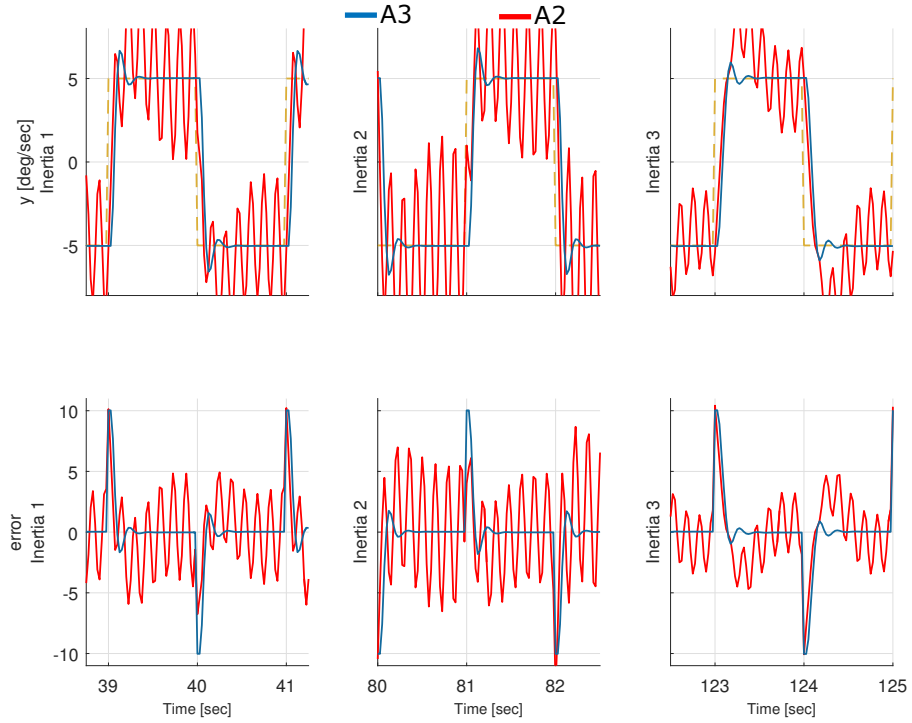

Figure 41: T2(C2), z-axis, different inertias

12.2.3 X-axis I²PD

Here inertia is only changed once at 33% of the simulation where it is increased substantially. As can be seen in Figure 42, the K_p evolves and converges for **A2**. After the change in inertia, response changes and oscillations disappear for some time. Although, oscillations reappear in the last third of the simulation. There is always big overshoot but errors come close to **A3** as seen in Figure 43. Although the resulting behavior is not desired with big amounts of overshoot and oscillations. Note the spike at around 27 seconds in Figure 43. Figure 44 shows the set-point being followed in an acceptable manner but behavior seems unpredictable.

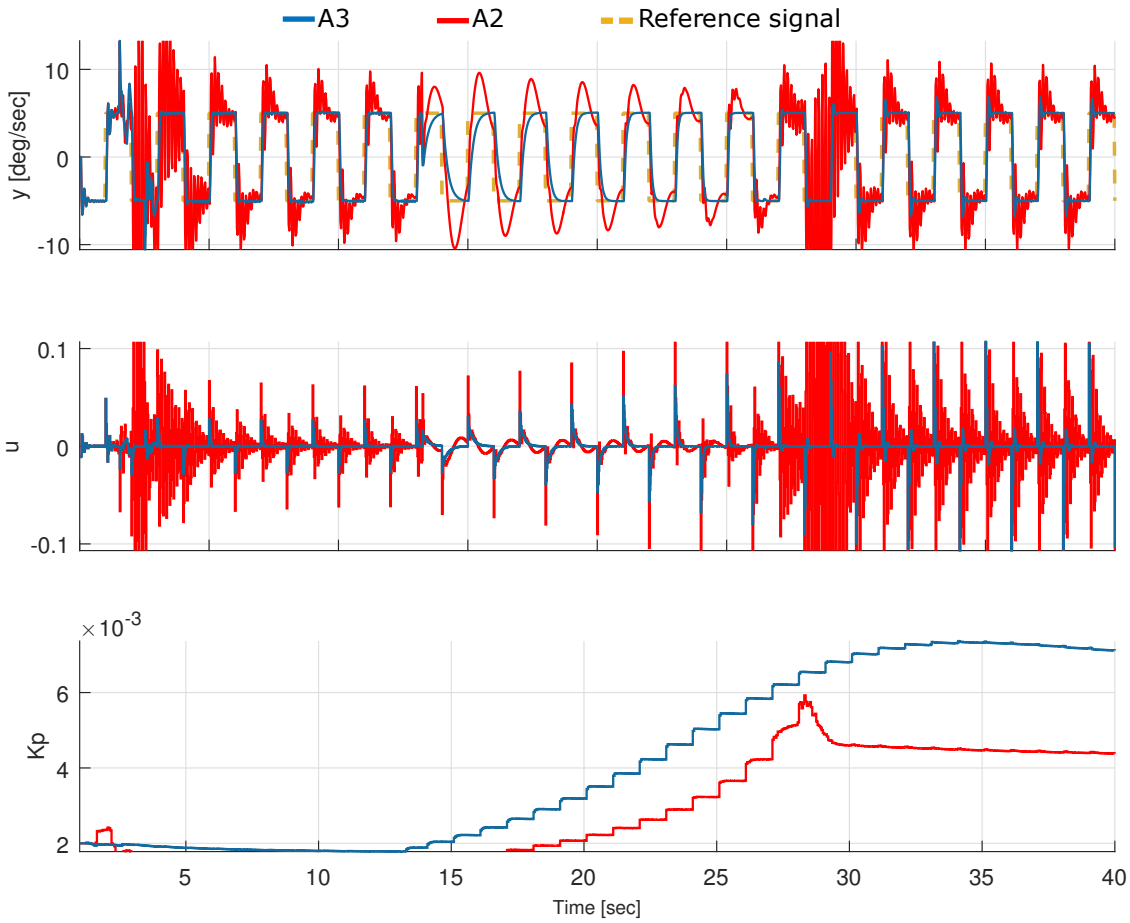

Figure 42: T2(C2), x-axis

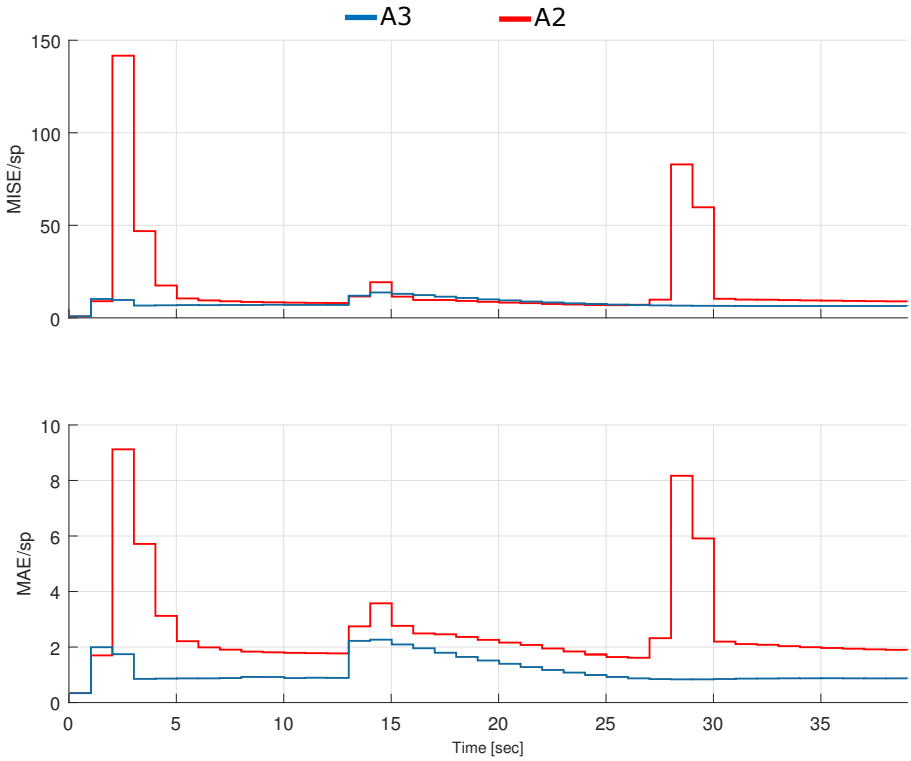

Figure 43: T2(C2), x-axis, errors

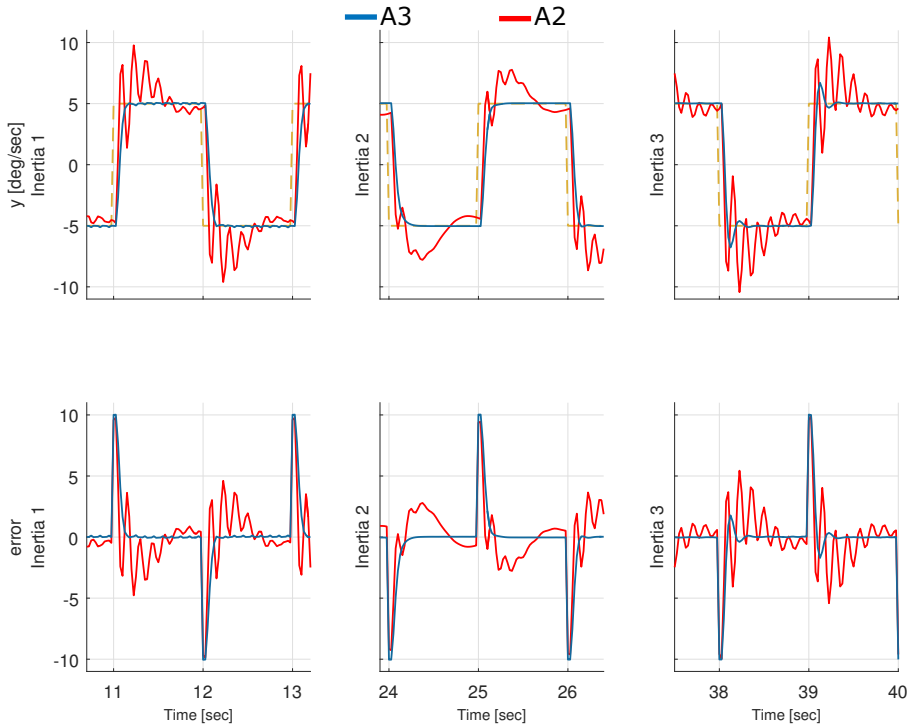

Figure 44: T2(C2), x-axis, different inertias

12.3 Test 3 - battery degradation

In this test the same square wave is induced as in **T1**, the difference being battery degradation is simulated as well, following the model in Figure 11.

12.3.1 Z-axis FOPDT

Convergence follows the battery model in a good manner. From Figure 45 both overshoot and oscillations are big. Each time a new set-point is set, the control signal maxes out. Errors stay stable at the same rate as seen in Figure 46 where the MAE is decreased at the end. It can also be seen that at the end of the simulation performance is better (see Figure 47) which is likely due to the systems loss in power and not the approach **A1** itself.

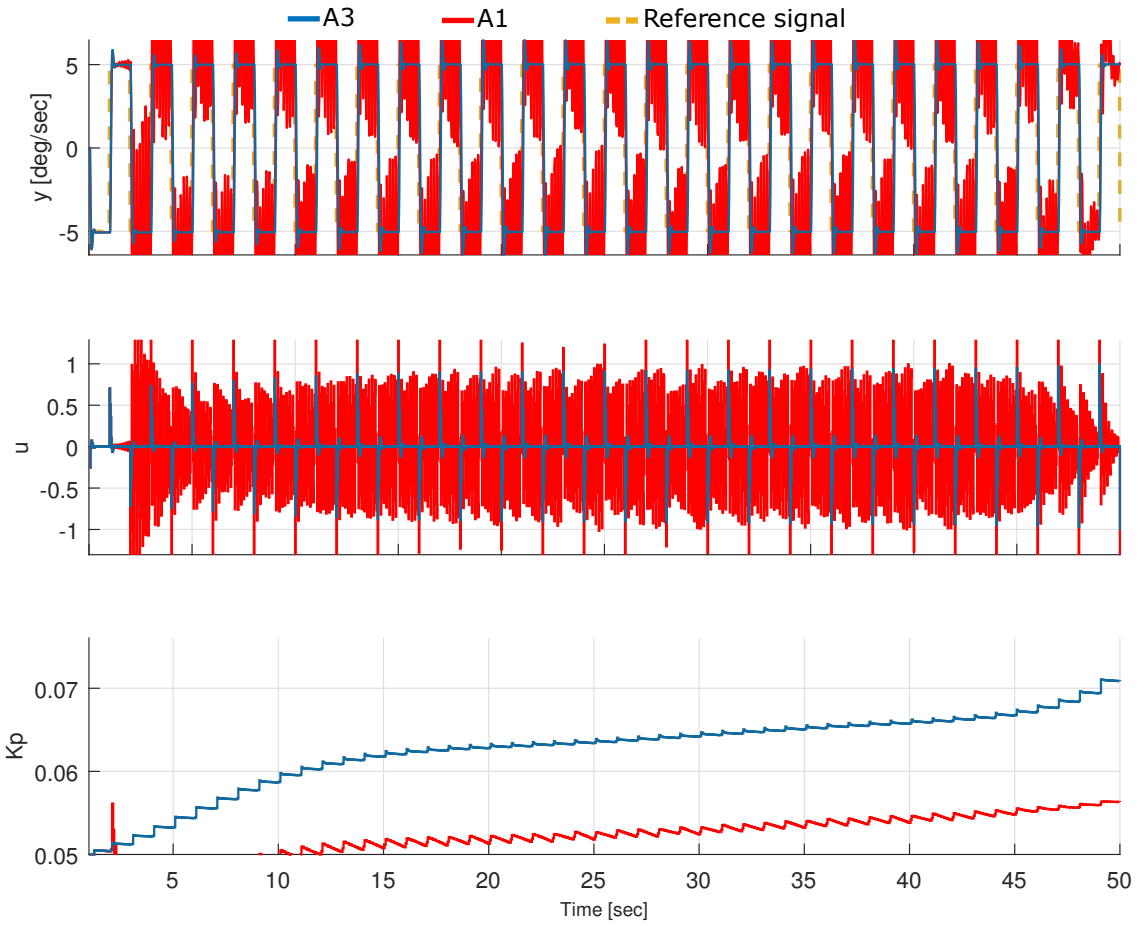

Figure 45: T3(C3), z-axis

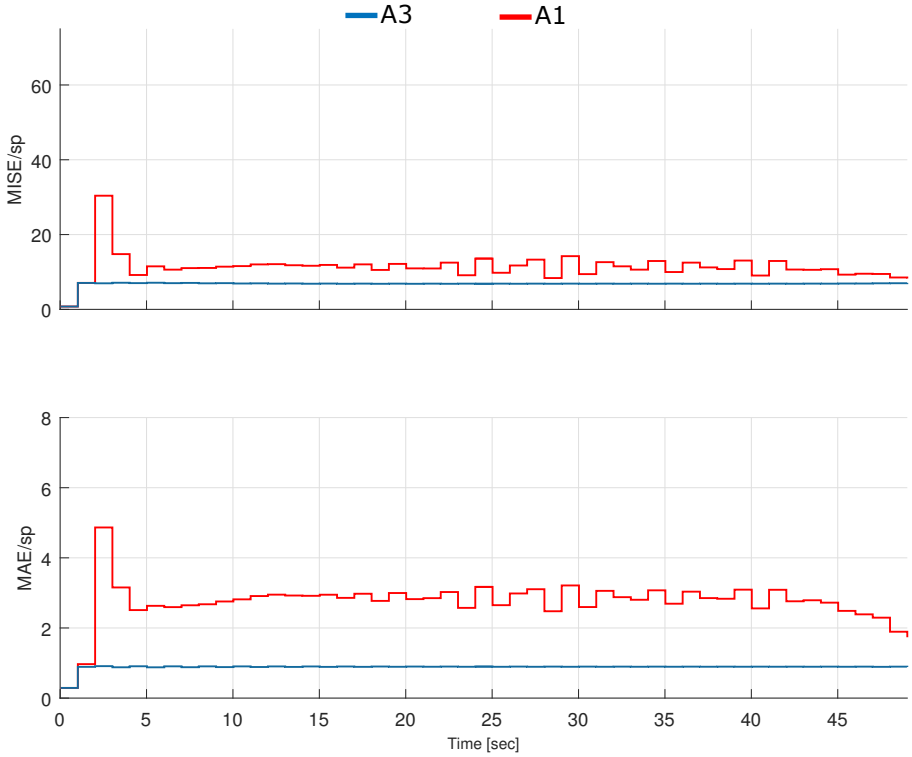

Figure 46: T3(C3), z-axis, errors

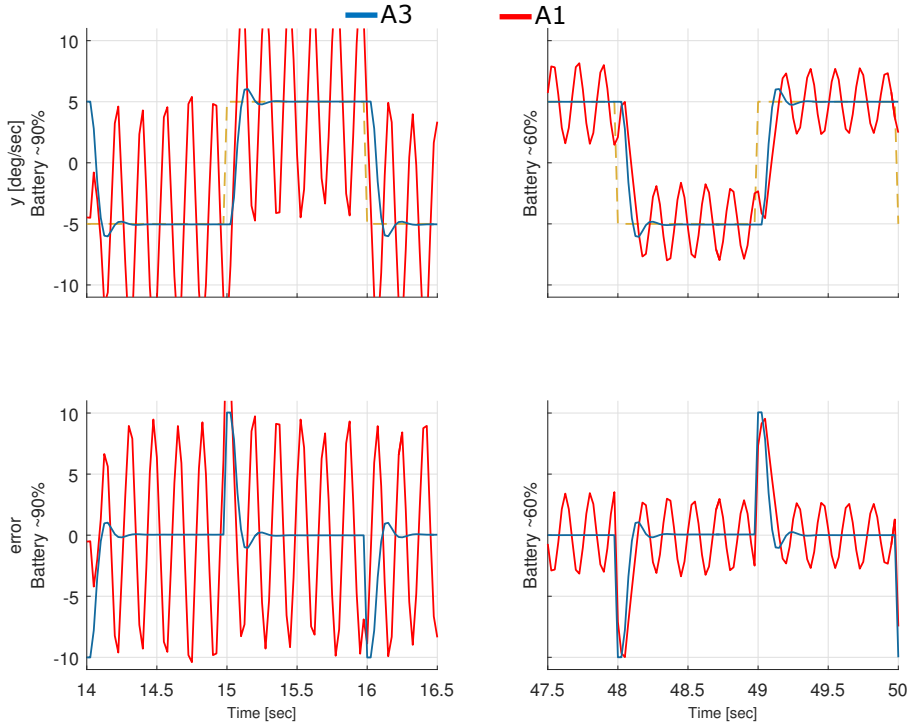

Figure 47: T3(C3), z-axis, time blocks from the beginning and end

12.3.2 Z-axis I²PD

The method **A2**, in the same way as **A1**, maxes out the control signal at each change in set-point as seen in Figure 48. The performance stays quite constant and convergence follows the battery curve well. From Figure 49 the same behavior as **A1** can be seen at the end, likely caused by less overshoot due to lower actuation power. The same analysis can be done in Figure 50 as the system turns from marginally stable to stable when the battery reaches its lowest charge.

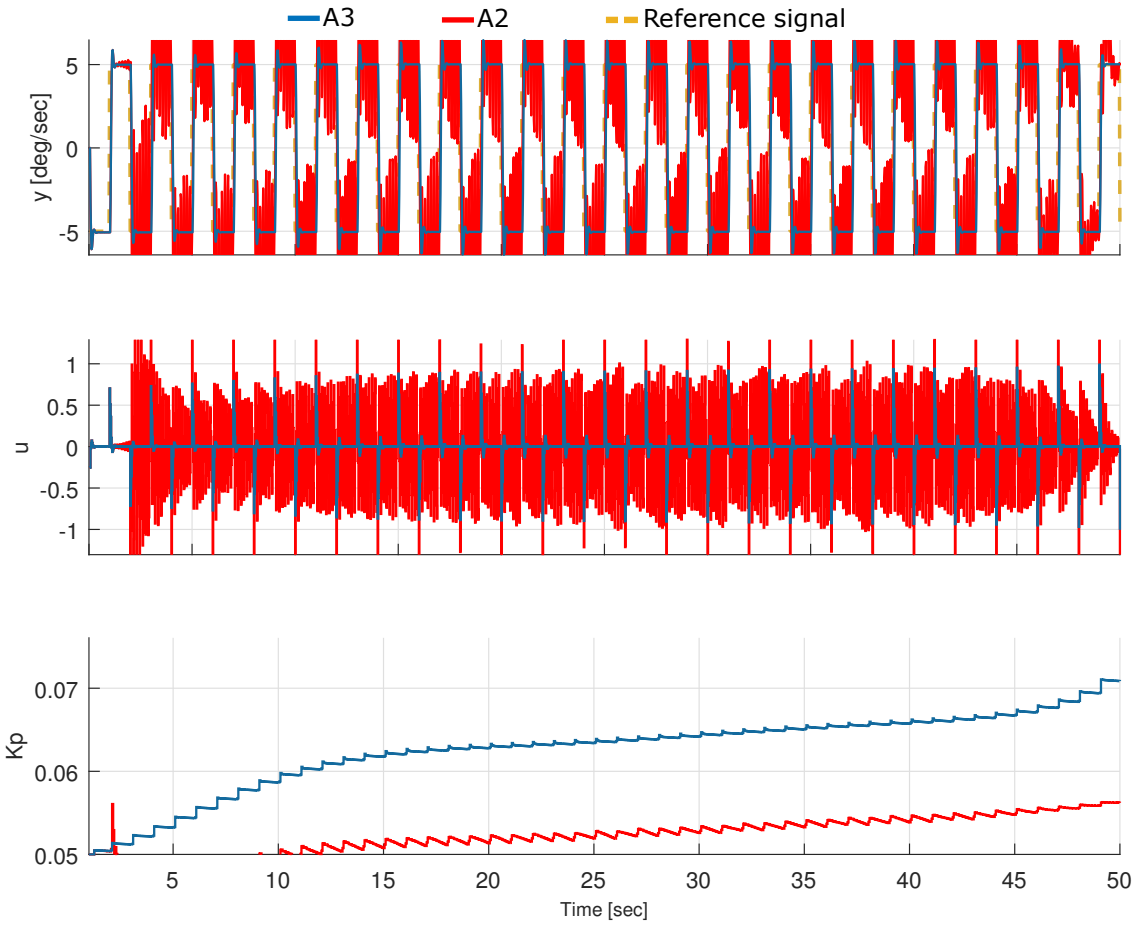

Figure 48: T3(C2), z-axis

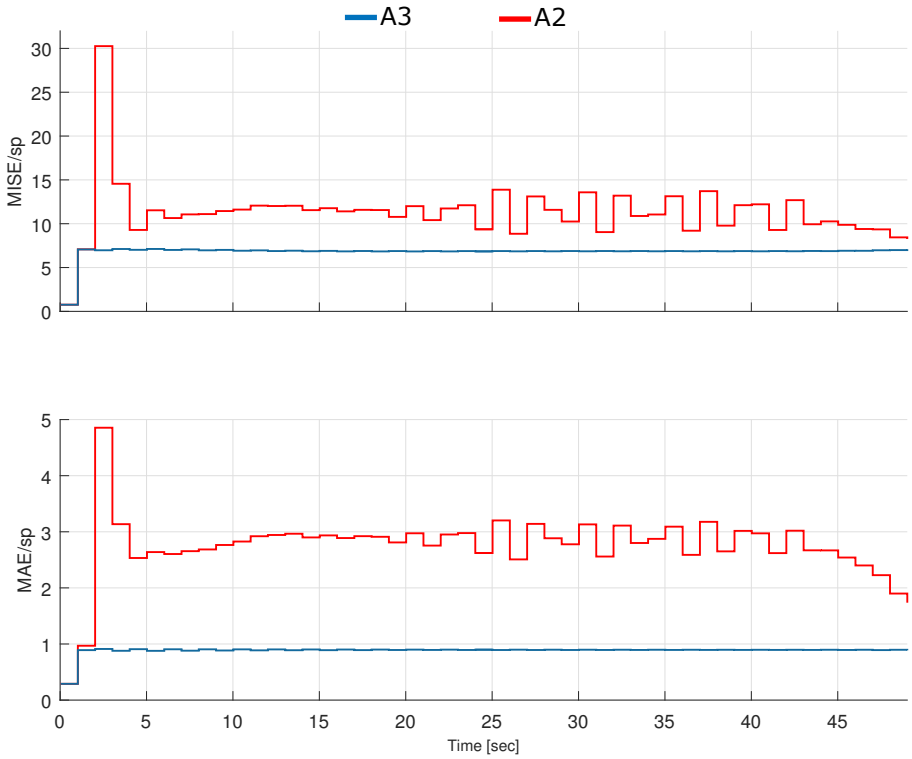

Figure 49: T3(C2), z-axis, errors

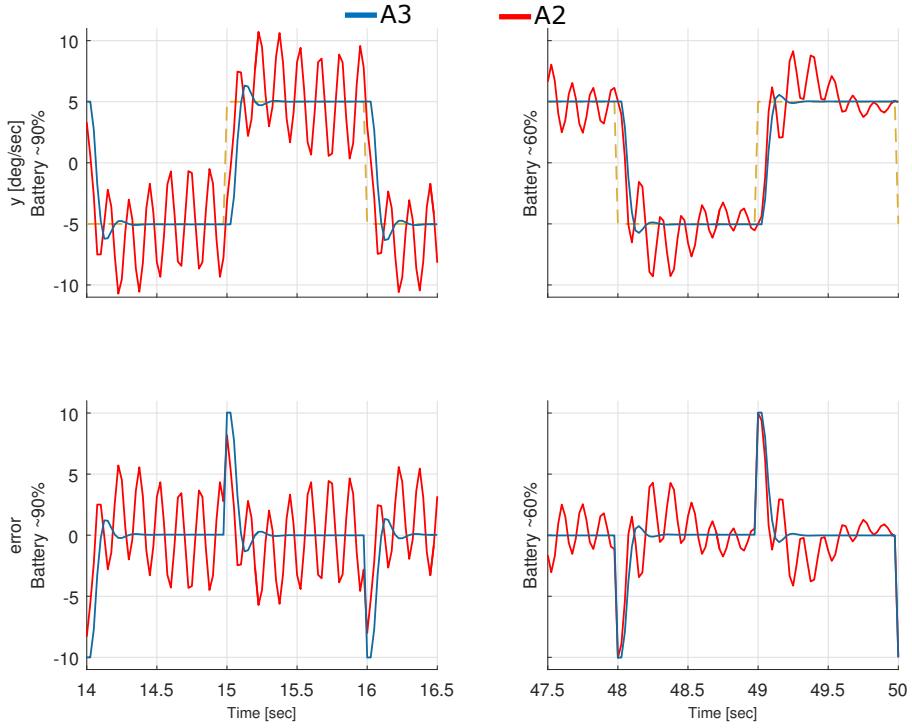

Figure 50: T3(C2), z-axis, time blocks from the beginning and end

12.3.3 X-axis I^{2}PD

The test on the x-axis is performed in the same way as for the z-axis. Figure 51 shows very consistent performance on **A2**. Errors are also very consistent as can be seen in Figure 52. A zoomed in graph of start and end confirms the consistency in Figure 53. Although the proposed approach by the thesis **A3** still shows superior performance.

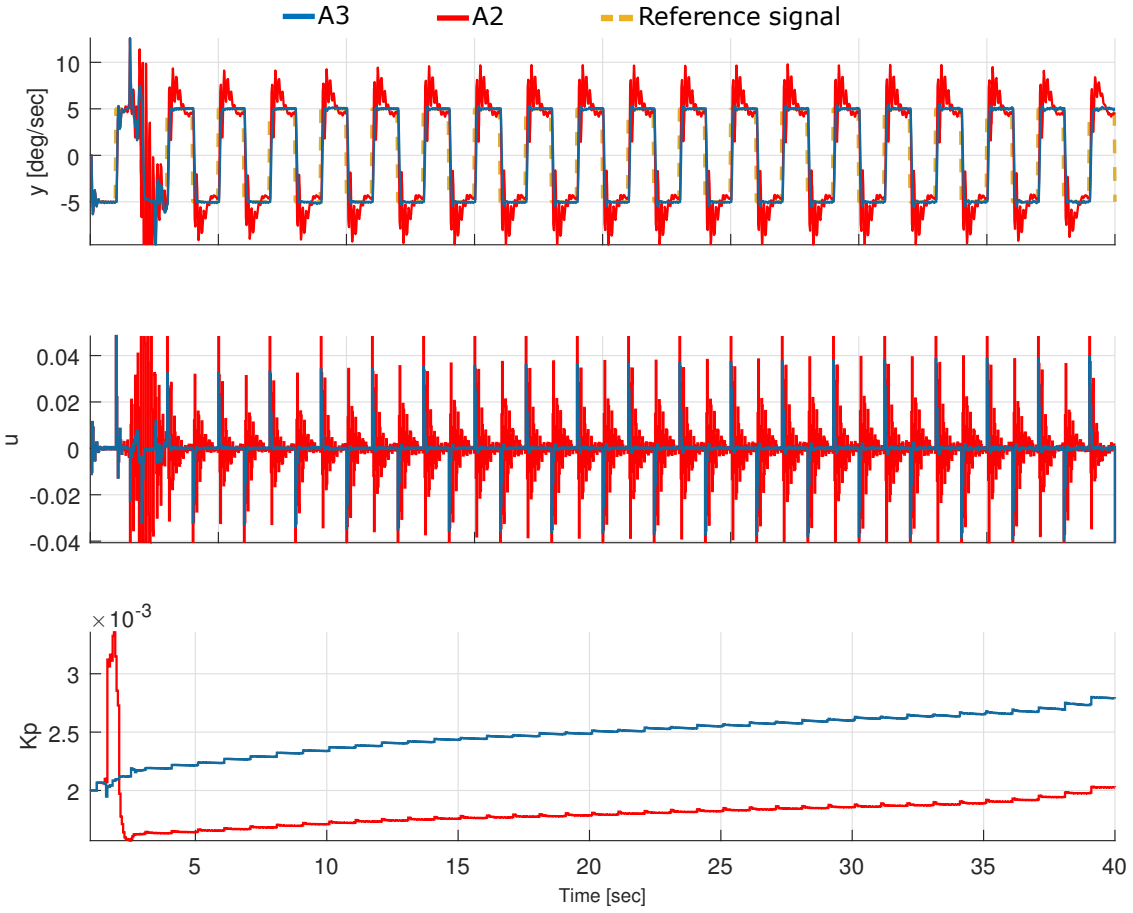

Figure 51: T3(C2), x-axis

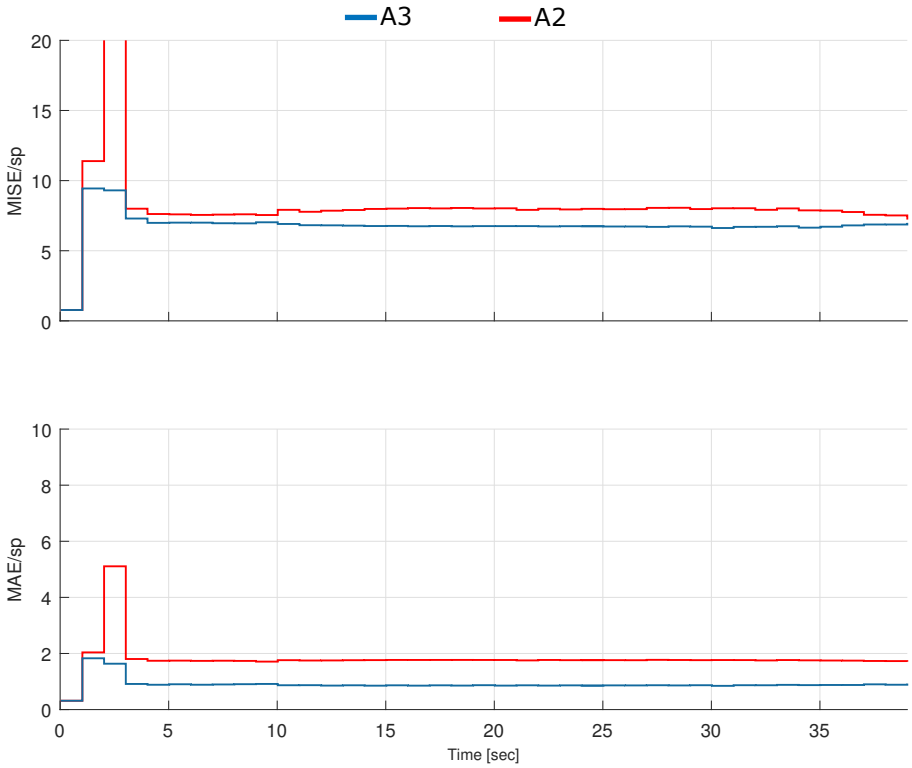

Figure 52: T3(C2), x-axis, errors

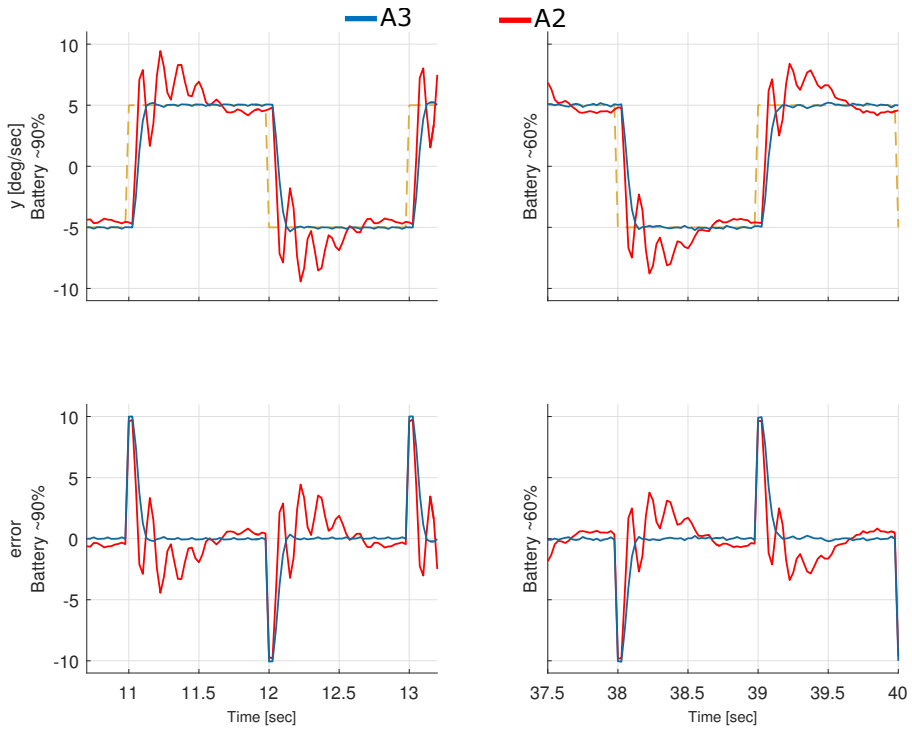

Figure 53: T3(C2), x-axis, beginning and end

12.4 Test 4 - smooth flying

Results have been presented numerically in Section 7. The following picture 54 shows an illustration of the quadrotor in the simulation environment V-REP. The quadrotor follows the target (presented as green globe) as the test is executed, also observe Figure 29 which demonstrates the smooth path.

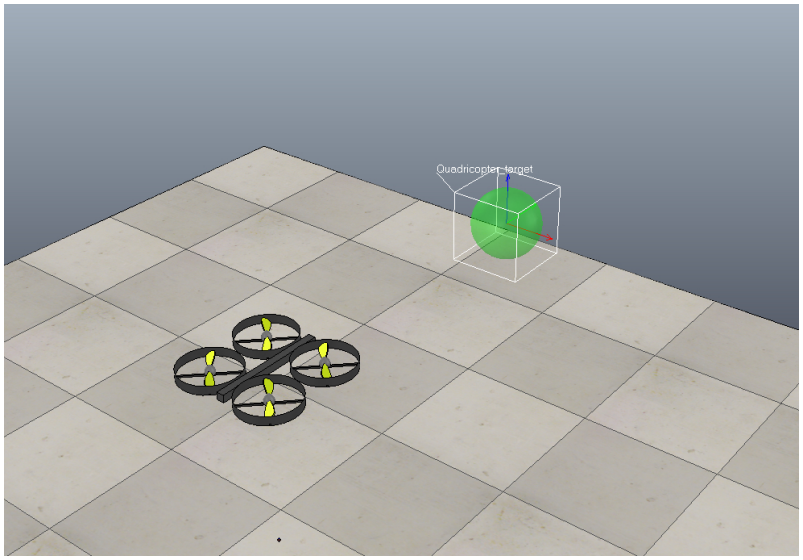

Figure 54: T4, smooth test, screenshot of V-REP simulation environment