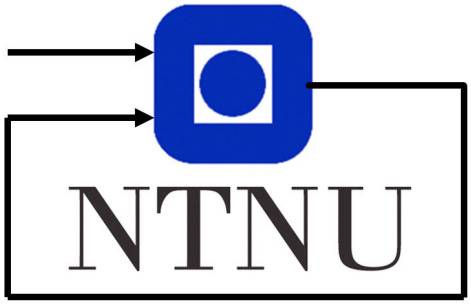
Model Predictive Control for the Attitude Control of autonomous fixed-wing Unmanned Aerial Vehicles

TTK4551 Specialization project

Written by Tobias Vaage

December 21, 2020



Department of Engineering Cybernetics

Abstract

Attitude control of autonomous fixed-wing unmanned aircrafts can unfold in a number of ways, one of which is Model Predictive Control. This control method optimizes a process given a set of constraints, and depending on this process and the constraints, the controller can be configured in different ways yielding different performances. In this project, the trade-off between computation time and flight performance is explored. Both a fully converged nonlinear MPC and a linear MPC is used, and also the effect of Real-Time Iteration is studied. By exposing the aircraft to different turbulence levels, the performance of the different controller configurations is explored. The simulation study showed that the linear controller is about twice as fast as the nonlinear, but the effect on the aircraft dynamics is minimal. As the wind conditions worsened, the linear MPC resulted in least error and faster computation time.

Contents

1 List of abbreviations

MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
LMPC	Linear Model Predictive Control
RTI	Real Time Iteration
QP	Quadratic programming
SQP	Sequential Quadratic Programming
NLP	Nonlinear programming
OCP	Optimal Control Problem
AOA	Angle of attack

2 Introduction

Attitude control of aerial vehicles can be done in several ways, such as optimization-based controllers as Linear-Quadratic Regulator (LQR) and Model Predictive Control (MPC), or PID controllers which correct error with proportional, integral and/or derivative terms. In this report the MPC is the control method that is to be studied. This type of controller solves a pre-defined optimization problem based on a cost function and a set of constraints on the states and inputs. A linear MPC, which minimizes the cost function based on a linear model, can be deployed relatively fast thanks to algorithms solving the underlying quadratic programs (QPs) [5]. A Nonlinear MPC (NMPC), which minimizes the cost function based on a nonlinear model, requires more elaborate algorithms. These algorithms require longer computational time, and the robustness of the controller becomes a topic as the algorithm may not be able to find a solution at a given timestep. The NMPC solves the nonlinear program (NLP) with a SQP solver which solves a sequence of QPs using QP solvers.

The unmanned aerial vehicle (UAV) may experience a turbulent environment due to strong wind gusts, and therefore the controller needs to be updated at a high rate. If the controller is not updated fast enough, the aircraft may be in a configuration not reversible, ultimately resulting in structural damage or loss of the aircraft. An irreversible configuration may be an orientation where the angle of attack is too large, resulting in loss of lift. In this project, we analyze the potential loss of performance when using a fully or partially converged solution by a NMPC, compared to the linear MPC. There exists a trade-off between flexibility, memory usage and speed when it comes to developing software for embedded optimal control [7]. So reducing the computation time should yield in a faster control algorithm, but what consequences does this have to the actual dynamics of the system?

The report is organized in the following manner. Section 3 introduces the framework around the model and controller. This section is necessary for understanding the results and to get a clear picture on what terms the simulation study is based on. Section 4 is about the linear and nonlinear MPC, where the differences and similarities are discussed in relation to the UAV. Section 5 explains in detail how these MPCs are to be tested in order to compare them. The content of the section creates the framework for the simulations and the data base for comparing them. Section 6 is about processing the data from the simulations. Here the results of the various simulations are presented and discussed. Lastly, section 7 concludes the report in a way that sheds light on the trade-off between computation time and aircraft dynamics.

3 Background

There has been extensive work done regarding the modeling and control of the aircraft. The aircraft in question is the Skywalker X8. The model of the aircraft can be represented in several ways, resulting in several state-spaces, parametrizations and frames. In this project, the full state-space has been used to represent the aircraft. This state-space depends on the chosen parametrization and frame, the various representations are presented below.

The framework that the model is built in allows for three parameterizations; quaternions, euler angles, and the rotation matrix. Quaternions is a four-dimensional representation where the issue of a singularity is nonexistent. However, this representation is not intuitive to humans, which makes the simulation and results hard to interpret. The euler angles are very intuitive, but there is a singularity at $\theta = \pm 90\,\text{deg}$. Hopefully the aircraft will never find itself in such an orientation, but singularities should be avoided either way. The rotation matrix is based on the rotations following the zyx-convention. This is the parametrization that has been used throughout the project.

There are also different coordinate frames one can develop the model and its dynamics within. The relevant frames in this case are NED $[n]$, BODY $[b]$, STABILITY $[s]$ and WIND $[w]$. References [1] and [4] has been used in the discussion of these coordinate frames. The n frame (North-East-Down) is an inertial frame in which the x-axis is directed north, y-axis is directed east, and z-axis is directed toward the center of the earth, or down. This frame is useful to get an understanding of where an aircraft is in the world, but not ideal to model the dynamics.

The b frame is fixed to the aircraft body, and this is where the aeordynamic forces (and others) act, see figure 1 for illustration.

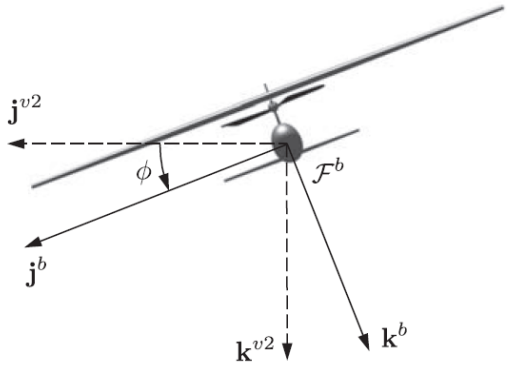


Figure 1: b frame [1].

The s frame is a rotation from b where the angle of attack plays a key role. This frame is useful for analyzing the stability of an aircraft, and in this case also place constraints on the AOA (α). For an illustration of this frame, see figure 2.

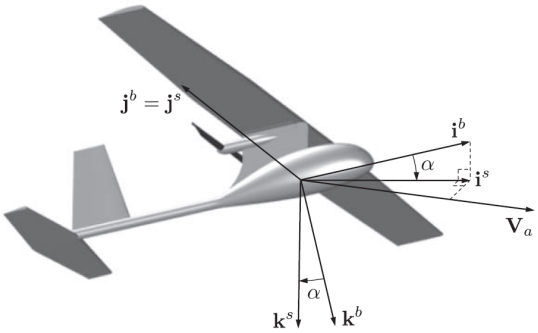


Figure 2: s frame [1].

In most cases it is useful to include forces and moments due to wind. For that the w frame is used. This frame is a rotation of the s, which is rotated about the side-slip angle (β). See figure 3 for illustration.

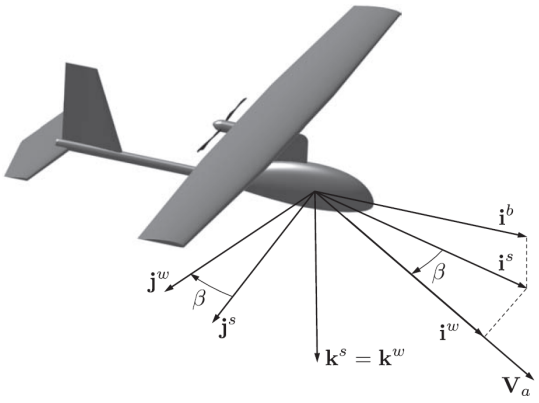


Figure 3: w frame [1].

All of these frames are related to each other, and it is necessary to rotate between them to include all relevant parameters in the model dynamics and simulations. Following the zyx-convention mentioned earlier, used to rotate from b to n, the following rotation matrix is defined:

$$\mathbf{R}_b^n = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{1}$$

where $c = cos()$, $s = sin()$ and $[\phi, \theta, \psi]$ are the euler angles. The rotation from b to s is useful when analyzing AOA is of value. The transformation between these two frames are given by the following rotation matrix, which is a dependent on AOA (α):

$$\mathbf{R}_b^s = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{2}$$

Lastly, the wind frame is obtained by rotating the s frame with the side-slip angle β. The allows for including the wind forces to the analysis through the side-slip variable. This rotation is given by:

$$\mathbf{R}_s^w = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

As mentioned earlier, the wind forces enter the equations of motion in the b frame, so using the defined relationships it is possible to derive a rotation from w to b. This relationship is of interest when the environmental effects is introduced later in the report.

$$\mathbf{R}_w^b = (\mathbf{R}_b^w)^T = \mathbf{R}_s^w \mathbf{R}_b^s = \begin{bmatrix} \cos\beta\cos\alpha & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \cos\beta\sin\alpha & -\sin\beta\sin\alpha & \cos\alpha \end{bmatrix} \tag{4}$$

Through Newtons laws of motion it is possible to derive a model for the aircraft dynamics. By relating the external forces $\mathbf{F}$ and moments $\mathbf{M}$ to a rigid-body, being the aircraft, the Newtons law of motion results in the following dynamics for the aircraft [6]:

$$m\dot{\mathbf{V}} + \omega \times m\mathbf{V} = \mathbf{F} \tag{5a}$$

$$\mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\mathbf{V} = \mathbf{M} \tag{5b}$$

Where $\mathbf{I}$ is the inertia matrix, m is mass, $\mathbf{V}$ are the linear velocities $[u, v, w]$ and ω are the angular velocities $[p, q, r]$. The forces and moments acting on the aircraft consists of several subcomponents. The external force vector includes aerodynamic forces, gravitational forces, and propulsion forces [1]:

$$\mathbf{F} = \mathbf{F}_a + \mathbf{F}_g + \mathbf{F}_p \tag{6}$$

The external moments are a combination of aerodynamic and propulsion moments [1]:

$$\mathbf{M} = \mathbf{M}_a + \mathbf{M}_p \tag{7}$$

In conlusion, there are several parametrizations and frames available. They all serve their purpose, and it is necessary to switch between them in order for computational efficiency and including all relevant parameters. They result in a number of states and parameters, in which a few has been chosen to study further. The airspeed is useful to observe and so are the rotations of the aircraft. That is why the following states has been plotted in the simulation study:

$$\mathbf{x}_{plot} = \begin{bmatrix} V_a & R_{13} & R_{23} & R_{33} \end{bmatrix} \tag{8}$$

where R_{13}, R_{23} and R_{33} represent the z-axis of the n frame in the b frame:

$$\begin{bmatrix} R_{13} \\ R_{23} \\ R_{33} \end{bmatrix} = \mathbf{R}_b^n \cdot \mathbf{b}, \quad \mathbf{b} = \begin{bmatrix} 0 & 0 & n \end{bmatrix} \tag{9}$$

4 MPC and NMPC

A simplified overview of the different controllers is illustrated in figure 4. It is a general description showing the main modules in the optimization problem. As illustrated, the right path is the nonlinear MPC, which is solved by a SQP solver. On the left is the linear MPC which is solved with a QP solver. It is expected that the NMPC path would be more time-consuming compared to the other path.

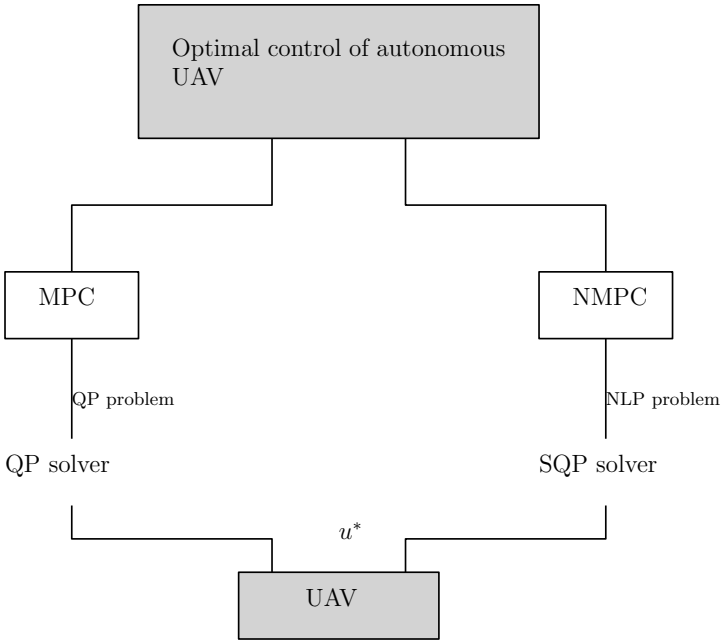


Figure 4: Optimal control using MPC.

In general, the difference between a linear and nonlinear MPC is that the constraints are linear or nonlinear. Expanding on this, we note that what makes this specific OCP nonlinear, is to use a nonlinear model for the UAV. A linear model results in a convex QP problem, and a nonlinear model turns the convex QP problem into a nonlinear and nonconvex problem [3]. These kinds of optimization problems require a NLP solver. However, the functionality of the MPC is the same, where you have an objective function that is to be minimized given a set of constraints. According to [3] the MPC principle is described as:

Model predictive control is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant.

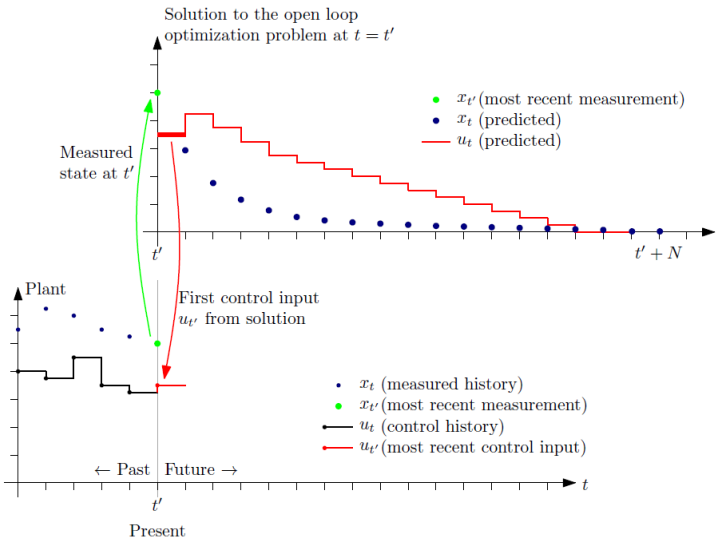


Figure 5: Illustration of MPC principle [3].

Figure 5 demonstrates this principle in a clear way. In regards to the NMPC, it is the SQP algorithms that could result in a "slower" controller. The QP subproblem solution is typically one of the most extensive steps in the SQP schemes, the other being the simulation and sensitivity computations of dynamical systems. The MPC controller is implemented through acados, which is a tool for optimal control of nonlinear problems [7]. See problem formulation below:

$$
\begin{aligned}
&\min_{x(\cdot),u(\cdot),z(\cdot),s(\cdot),s^{\mathrm{e}}} && \texttt{/* Cost Function */}\\
& && \int_0^T l(x(\tau),u(\tau),z(\tau),p) + \frac{1}{2}\begin{bmatrix} s_{\mathrm{l}}(\tau) \\ s_{\mathrm{u}}(\tau) \\ 1 \end{bmatrix}^{\top} \begin{bmatrix} Z_{\mathrm{l}} & 0 & z_{\mathrm{l}} \\ 0 & Z_{\mathrm{u}} & z_{\mathrm{u}} \\ z_{\mathrm{l}}{}^{\top} & z_{\mathrm{u}}{}^{\top} & 0 \end{bmatrix} \begin{bmatrix} s_{\mathrm{l}}(\tau) \\ s_{\mathrm{u}}(\tau) \\ 1 \end{bmatrix} \mathrm{d}\tau + \\
& && \quad m(x(T),z(T),p) + \frac{1}{2}\begin{bmatrix} s_{\mathrm{l}}^{\mathrm{e}} \\ s_{\mathrm{u}}^{\mathrm{e}} \\ 1 \end{bmatrix}^{\top} \begin{bmatrix} Z_{\mathrm{l}}^{\mathrm{e}} & 0 & z_{\mathrm{l}}^{\mathrm{e}} \\ 0 & Z_{\mathrm{u}}^{\mathrm{e}} & z_{\mathrm{u}}^{\mathrm{e}} \\ z_{\mathrm{l}}^{\mathrm{e}\top} & z_{\mathrm{u}}^{\mathrm{e}\top} & 0 \end{bmatrix} \begin{bmatrix} s_{\mathrm{l}}^{\mathrm{e}} \\ s_{\mathrm{u}}^{\mathrm{e}} \\ 1 \end{bmatrix}\\
&\text{s.t.} && \texttt{/* Initial value */}\\
& && \underline{x}_0 \le J_{\mathrm{bx},0}x(0) \le \bar{x}_0,\\
& && \texttt{/* Dynamics */}\\
& && f_{\mathrm{impl}}(x(t),\dot{x}(t),u(t),z(t),p) = 0,\\
& && \texttt{/* Path Constraints with lower bound */}\\
& && \underline{h} \le h(x(t),u(t),p) + J_{\mathrm{sh}}s_{\mathrm{l,h}}(t),\\
& && \underline{x} \le J_{\mathrm{bx}}x(t) + J_{\mathrm{sbx}}s_{\mathrm{l,bx}}(t),\\
& && \underline{u} \le J_{\mathrm{bu}}u(t) + J_{\mathrm{sbu}}s_{\mathrm{l,bu}}(t),\\
& && \underline{g} \le Cx(t) + Du(t) + J_{\mathrm{sg}}s_{\mathrm{l,g}}(t),\\
& && s_{\mathrm{l,h}}(t), s_{\mathrm{l,bx}}(t), s_{\mathrm{l,bu}}(t), s_{\mathrm{l,g}}(t) \ge 0,\\
& && \texttt{/* Path Constraints with upper bound */}\\
& && h(x(t),u(t),p) - J_{\mathrm{sh}}s_{\mathrm{u,h}}(t) \le \bar{h},\\
& && J_{\mathrm{bx}}x(t) - J_{\mathrm{sbx}}s_{\mathrm{u,bx}}(t) \le \bar{x},\\
& && J_{\mathrm{bu}}u(t) - J_{\mathrm{sbu}}s_{\mathrm{u,bu}}(t) \le \bar{u},\\
& && Cx(t) + Du(t) - J_{\mathrm{sg}}s_{\mathrm{u,g}} \le \bar{g},\\
& && s_{\mathrm{u,h}}(t), s_{\mathrm{u,bx}}(t), s_{\mathrm{u,bu}}(t), s_{\mathrm{u,g}}(t) \ge 0,\\
& && \texttt{/* Terminal Constraints with lower bound */}\\
& && \underline{h}^{\mathrm{e}} \le h^{\mathrm{e}}(x(T),p) + J_{\mathrm{sh}}^{\mathrm{e}}s_{\mathrm{l,h}}^{\mathrm{e}},\\
& && \underline{x}^{\mathrm{e}} \le J_{\mathrm{bx}}^{\mathrm{e}}x(T) + J_{\mathrm{sbx}}s_{\mathrm{l,bx}}^{\mathrm{e}},\\
& && \underline{g}^{\mathrm{e}} \le C^{\mathrm{e}}x(T) + J_{\mathrm{sg}}^{\mathrm{e}}s_{\mathrm{l,g}}^{\mathrm{e}} \le \bar{g}^{\mathrm{e}},\\
& && s_{\mathrm{l,h}}^{\mathrm{e}}, s_{\mathrm{l,bx}}^{\mathrm{e}}, s_{\mathrm{l,bu}}^{\mathrm{e}}, s_{\mathrm{l,g}}^{\mathrm{e}} \ge 0,\\
& && \texttt{/* Terminal Constraints with upper bound */}\\
& && h^{\mathrm{e}}(x(T),p) - J_{\mathrm{sh}}^{\mathrm{e}}s_{\mathrm{h}}^{\mathrm{e}} \le \bar{h}^{\mathrm{e}},\\
& && J_{\mathrm{bx}}^{\mathrm{e}}x(T) - J_{\mathrm{sbx}}s_{\mathrm{bx}}^{\mathrm{e}} \le \bar{x}^{e},\\
& && C^{\mathrm{e}}x(T) - J_{\mathrm{sg}}^{\mathrm{e}}s_{\mathrm{g}}^{\mathrm{e}} \le \bar{g}^{\mathrm{e}}\\
& && s_{\mathrm{u,h}}^{\mathrm{e}}, s_{\mathrm{u,bx}}^{\mathrm{e}}, s_{\mathrm{u,bu}}^{\mathrm{e}}, s_{\mathrm{u,g}}^{\mathrm{e}} \ge 0
\end{aligned}
$$

4.1 RTI - Connecting the NMPC to LMPC

To be able to discuss the differences between a linear and nonlinear MPC, one has to understand the underlying mechanisms from the problem formulation to the solvers. As stated, a NMPC uses a SQP algorithm to iteratively find a solution. When SQP is used in NMPC, and the reference is used as an initial guess, the first step of a full step Gauss-Newton SQP delivers the same control solution as a linear MPC [5]. This property is used to transfer the already-existing NMPC to a LMPC. There is another method in the NLP solver that draws parallels to the LMPC, and that is Real-Time Iteration (RTI). The main idea behind this is to at all times use the latest information on the system dynamics in the algorithms computing the NMPC solution [5]. This allows for approximating a solution based on the most recent information, instead of an accurate solution based on outdated information. The implementation of this on our system is a single line of code in which one chooses the SQP solver to use or not use RTI within the acados tool [7]. An illustration of this method is visualized in figure 6.

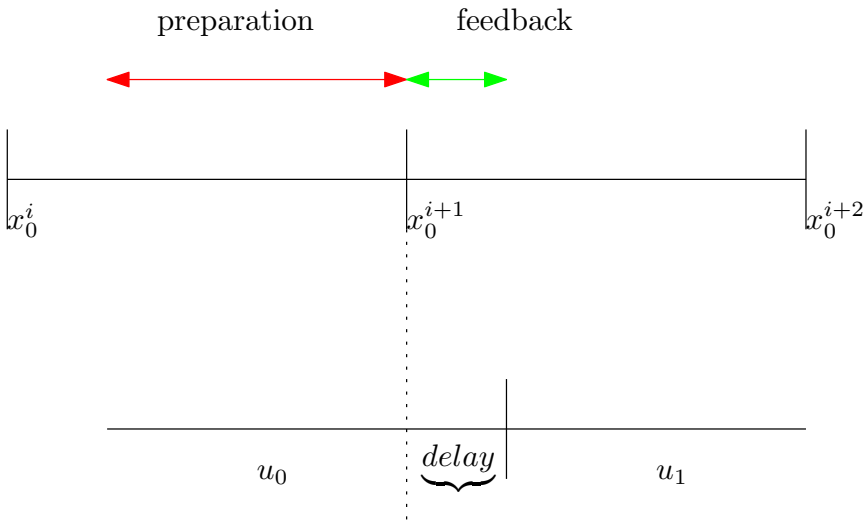


Figure 6: Illustration of preparation and feedback phase.

4.2 Real-time optimization strategies

While the real system evolves, the optimization problem is solved, and this optimization problem itself takes time. This is sometimes referred to as "the real-time dilemma" [5]. There exist different strategies that can be implemented to decrease the time of the optimization solver such that the real system is updated fast enough. The RTI can be viewed as one of these strategies, but Moritz Diehl [2] proposes a few easily implementable strategies.

Offline precomputations

The idea behind offline precomputations is to ensure that what can be solved offline, is solved offline. This is to minimize the number of computations done online, ensuring that the online solver computes as fast as possible. There are some simple actions one can take to implement this strategy:

- Factorize matrices
- Code generation (C-code)
- Model simplifications

In regard to the X8 aircraft, these actions have to a certain degree been implemented. There is still a lot potential in simplifying the model however. It could be interesting to explore how streamlining the algorithms potentially has an impact on the computation time. Ultimately, this simple although time-consuming process could yield in a faster controller without altering the controller itself, resulting in better flight dynamics. When comparing the different controllers, it is not relevant how offline precomputations improves online computation time. However, it might improve the benchmark computation times, rendering online controllers overall faster.

Delay compensation by prediction

Instead of acting on faulty dynamics after the fact, one can use the strategy of compensating by prediction. What this entails is that while the optimization computation takes place, one can use the previous input sequence as inputs to the plant until a new solution has been computed. We have to implement some control input to the plant anyways, so why not use the prediction that is already computed. A simple example is that if the optimization takes 5 timesteps to compute, first 5 elements of the input sequence could be fed to the plant, while the solver computes the next sequence. This method is sensitive to large pertubations and disturbances, and the predicted input sequence may be outdated to the point that controlling the system becomes impossible. This is especially something to consider for small UAVs in harsh environments. In those cases, the perturbations can be severe and rapidly changing, and the predicted inputs could make the situation worse.

5 Simulations

In this section, the framework for the various simulations are described. Unfortunately, there has not been any actual flight tests, so all the experiments and data is a result of simulating the aircraft model. A total of 9 simulations have been done, all with different environmental conditions and controller settings. In addition, at each of these simulations, the CPU time at each iteration in the MPC loop has been sampled.

5.1 Controller configuration

The controller is designed as a NMPC, where it has to iteratively converge to an optimal solution. However, by only allowing one iteration of the SQP solver, the controller becomes linear at the current state. By manipulating these controller settings, it is possible to simulate the system with both a nonlinear and linear MPC. In the cases where NMPC is experimented with, the maximum number of SQP iterations has been set to 1000 in order to ensure full convergence. Full convergence can be hindered if the maximum iterations constraint is reached. This is avoided when maximum iterations is set to 1000. On the other end, for the LMPC, the maximum SQP iterations has been set to 1, such that the system becomes linear. As discussed earlier, the RTI functionality is useful for reducing computation time. In the optimization algorithm it is possible to choose between SQP with and without RTI, so a third controller configuration is NMPC with RTI. Including this feature in the nonlinear case could be interesting in order to see how much it reduces the computation time compared to the linear case.

5.2 Wind conditions

Atmospheric disturbances have a great impact on the aircraft dynamics, and in the case of slow flying UAVs, the wind velocity is large relative to the airspeed compared to larger aircraft [6]. This translates to the wind having more impact on a small aircraft, which has to be dealt with in the design of the controller. As mentioned before we want to analyze how the different controller configurations handle various levels of turbulence. When simulating wind, it is useful to divide into the two subcategories steady ambient wind and wind gusts. The former is easily dealt with by the controller, as it acts as a constant disturbance. The latter however, depending on the level of turbulence, could render the aircraft hard to control because of the abrupt changes in wind velocity. The steady wind velocity is modeled as a constant speed vector in the n frame. For all the simulations that were done, these constants were set to a headwind of 5 m/s and a crosswind of 4 m/s. And so the static part of the wind takes the shape of:

$$\mathbf{V}_{static}^{n} = \begin{bmatrix} -5 & 4 & 0 \end{bmatrix} \tag{10}$$

The gust component of the wind is modeled as a stochastic process, where the Dryden transfer functions are used to transfer white noise into wind velocities [1]. Without going into detail in how these transfer functions are developed, it is worth to mention that they are dependent on airspeed, altitude, wingspan, and turbulence intensity. To see how the aircraft responds in different environments, those are the factors that are manipulated. The wind gusts are calculated in w and has to be rotated to b.

$$\mathbf{V}_{gust}^{b} = \mathbf{R}_{w}^{b}\mathbf{V}_{gust}^{w} \tag{11}$$

5.3 Test scenarios

The controller has been configured and a model describing the wind impact in different environments has been established. To test what happens to the aircraft dynamics in different environments with different controller configurations, it is important to run the simulations and extract the relevant data in a systematic way. Across all the simulations, the airspeed reference is set to 19 m/s, a normal cruising speed for the X8 aircraft. In addition, the simulation time is 20 seconds. The control objective is to reach and hold the desired airspeed while turning in a radius of 50 meters. That also means setting the flight path angle reference to zero. The flight path angle is defined as the angle between the speed over ground vector and the horizontal plane. Lastly, the altitude reference is set to 300 meters. Below is a description of the different simulations that is to be done.

Table 1: Description of test scenarios.

Simulation no.	**Controller**	**Wind conditions**
1	LMPC	Static wind only
2	NMPC w/RTI	Static wind only
3	NMPC	Static wind only
4	LMPC	Light turbulence
5	NMPC w/RTI	Light turbulence
6	NMPC	Light turbulence
7	LMPC	Moderate turbulence
8	NMPC w/RTI	Moderate turbulence
9	NMPC	Moderate turbulence

6 Results and discussion

6.1 Computation time

For determining the computation time of all the different scenarios, the total CPU time of every iteration has been sampled. In appendix B are plots showing these CPU times throughout every simulation. In one simulation there are 2000 timesteps ($\frac{\text{simulation time}}{\text{timestep}} = \frac{20s}{0.01s} = 2000$). This amount of data is too vast to extract information from, which is why the mean computation time during the simulation has been used. In table 2 is a complete overview of these mean computation times. The simulations have been done on the same computer, meaning the same CPU has been used every time. However, the CPU performance may be varying, which could be a source of error in these experiments.

Table 2: Computation times in different simulations.

Simulation no.	**CPU time**
1	13.4 µs
2	12.1 µs
3	23.1 µs
4	14.5 µs
5	12.2 µs
6	31.2 µs
7	14.8 µs
8	11.9 µs
9	32.2 µs

These results are better interpreted in a diagram categorizing the testing conditions. The computation times are categorized in a diagram shown in figure 7.

The NMPC computation is more time-consuming than LMPC. On the other hand, the impact that RTI has on the nonlinear controller is quite significant. Through all simulations, the NMPC with RTI computes faster than the LMPC. The objective however, was to compare the linear case to the nonlinear case, as as shown in figure 7 the linear case computes faster in all simulations by a good margin. It is only logical, as the linear controller does one SQP iteration, while the nonlinear controller does as many as necessary in order to converge to an optimal solution.

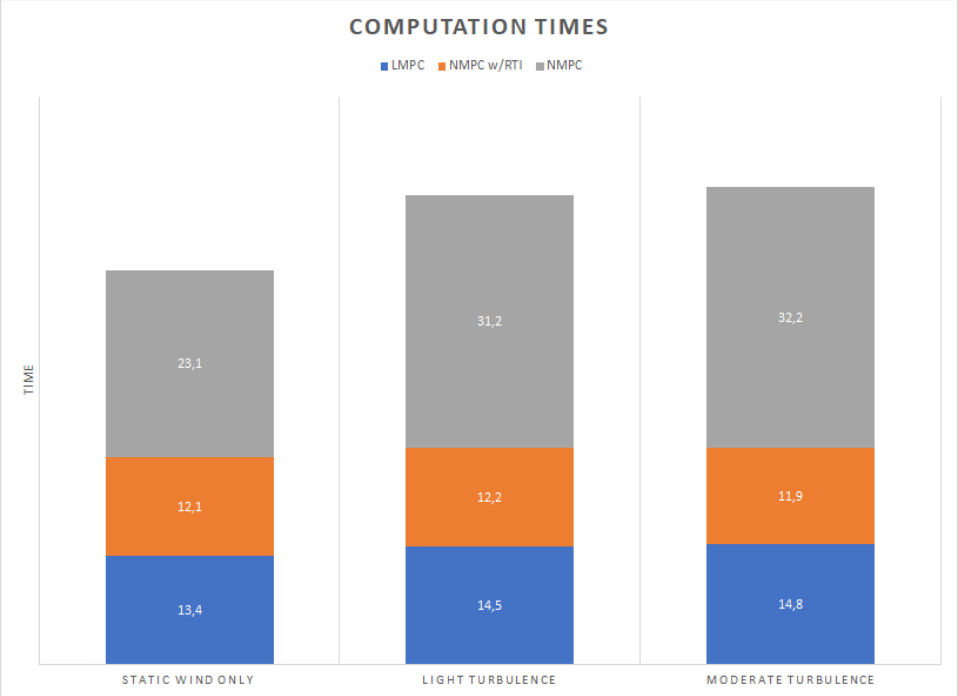


Figure 7: Visualization of computation times.

In the static wind case, the LMPC is 1.7 times faster, in the light turbulence case 2.1 times faster, and in the moderate turbulence case 2.2 times faster. This indicates that as the environmental conditions gets worse, the NMPC controller computes slower relative to the LMPC. A worsening turbulence level leads to more significant pertubations of the aircraft, which the controller needs to compensate for. As the LMPC only allows one SQP iteration no matter the situation, the NMPC needs and does more calculations in order to compensate.

The variations in computation times were quite significant. In some instances the computation time spiked, which for example can be seen in figure 21 in appendix B. It should be noted that the simulations were done on a CPU of type "AMD Ryzen 5 2500U" which can be subject to delays and irregular performance. These spikes does not affect the mean computation time across the simulation to any important degree, and occur in every simulation, so they are regarded negligible. An interesting observation between the LMPC and NMPC computation times is the way these times develop through the simulation. In figure 24 the LMPC controller only needs milliseconds to work its way down in computation time, while the NMPC in figure 26 needs a couple of seconds to do so. This occurs in the beginning of the simulation when the aircraft approaches the references, where the most volatile dynamics are

expected. This phenomenon indicates that the NMPC controls the aircraft more actively than the LMPC especially in this time period, which might be necessary in order for the aircraft dynamics to be acceptable.

6.2 Aircraft performance

In appendix A one can find plots describing the key performance parameters in each scenario. For each scenario, there are two plots; The first one consisting of the four states in $\mathbf{x}_{plot}$ (8), and the second one being the error of the same states in addition to the cost function. The first plot serves as a visualization of how the aircraft dynamics unfold throughout the simulation, and the second figure can be used as a gauge to observe how much the states deviate from the references. The second figure also includes the cost function, which is a good way of illustrating how well the controller manages to minimize the cost function. If the cost function has a small value, it indicates that the solution is more optimized than a cost function of larger value.

In general, it is the case that the aircraft dynamics become less stable as more and more turbulence is introduced to the system. This is due to the wind forcing the aircraft in an orientation not desirable, which the controller has to compensate for. This issue can also be observed in the plots illustrating the cost function. For instance, the cost function in the LMPC controller is very different from the static wind case to the moderate turbulence case. Below is a figure showing the differences in the cost function in the two different environmental conditions.

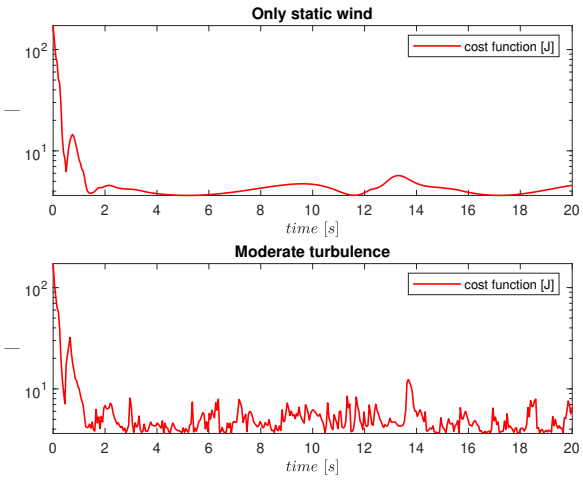


Figure 8: Cost function for static wind and moderate turbulence.

Clearly the OCP is better optimized when the aircraft is subject to less disturbances. In some parts of the simulation, the difference is as large as an order of magnitude.

What is more interesting however, is how the various controller configurations deal with the same environmental conditions. Overall, the environmental conditions does not impact the error dynamics to a large degree, which indicates that all controller configurations manages to control the aircraft in a satisfying manner. The state R_{13} will be used to represent the aircraft dynamics. As the cost function described, there are larger errors as the disturbances intensifies. In the case of no wind gusts, i.e only static wind, the aircraft performance is overall very good. Figure 9 describes this scenario. The maximum error ($e = y - y_{ref}$) is 0.2, and quickly approaches and oscillates around zero. In addition, there is little difference between the different controllers. The beginning of the simulation is the most eventful, and that is where the biggest differences are. Here the NMPC with RTI sticks out as worse controller in regards to error, and it seems that it overcompensates in some instances. Comparing the LMPC with the NMPC, there are little no to differences in error in the static wind case.

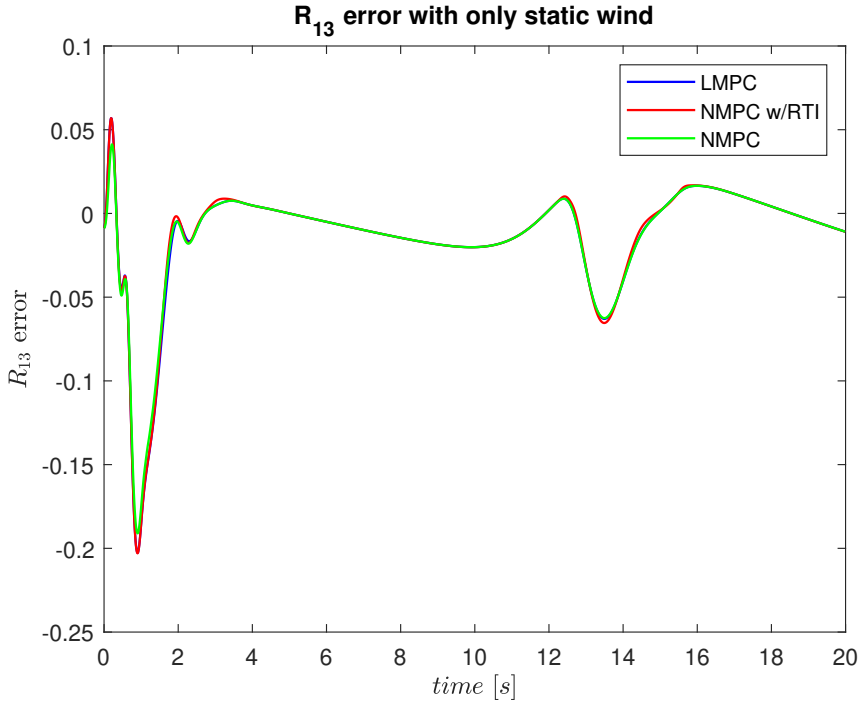


Figure 9: Error with static wind.

Introducing light turbulence to the system also introduces more error. This is illustrated in figure 10. In this case the maximum error is 0.25, which is a relatively small change compared to the static wind case. There are also differences between the controllers, and it can be said that the LMPC error dynamics seems a bit delayed compared to the nonlinear controller. Overall, the LMPC displays least error throughout the simulation, and the NMPC with RTI results in the most error. It should be noted that both nonlinear controllers have similar error dynamics, despite the large difference in computation time.

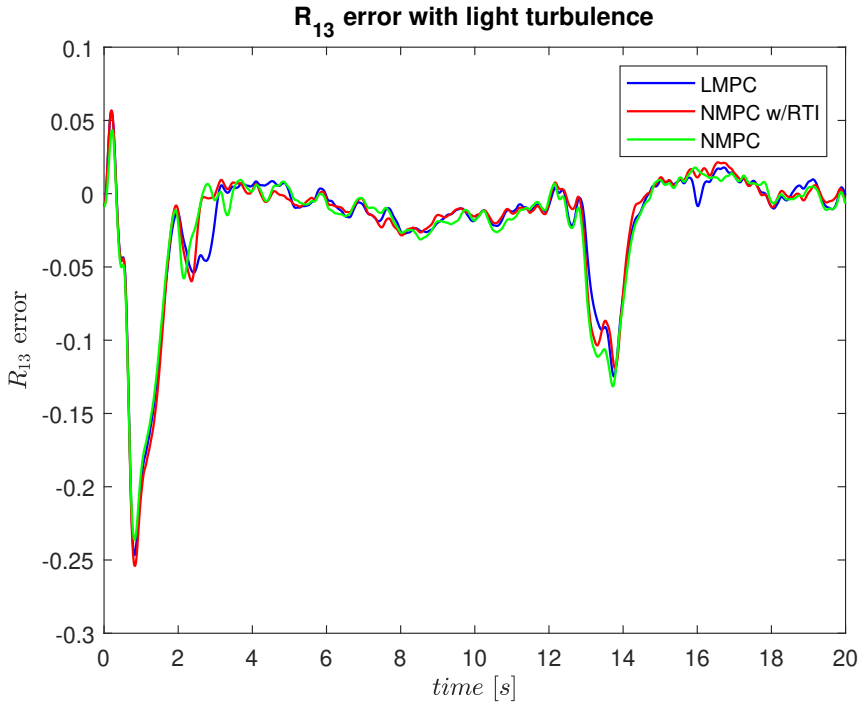


Figure 10: Error with light turbulence.

When the aircraft is exposed to moderate turbulence, the already discussed effects are amplified. The maximum error is now increased to 0.3, and the LMPC controller continues to be the better controller in regards to error. The error dynamics does not change much besides the amplitude, which is logical as the environmental conditions are worsened. See figure 11 for error development in the moderate turbulence case.

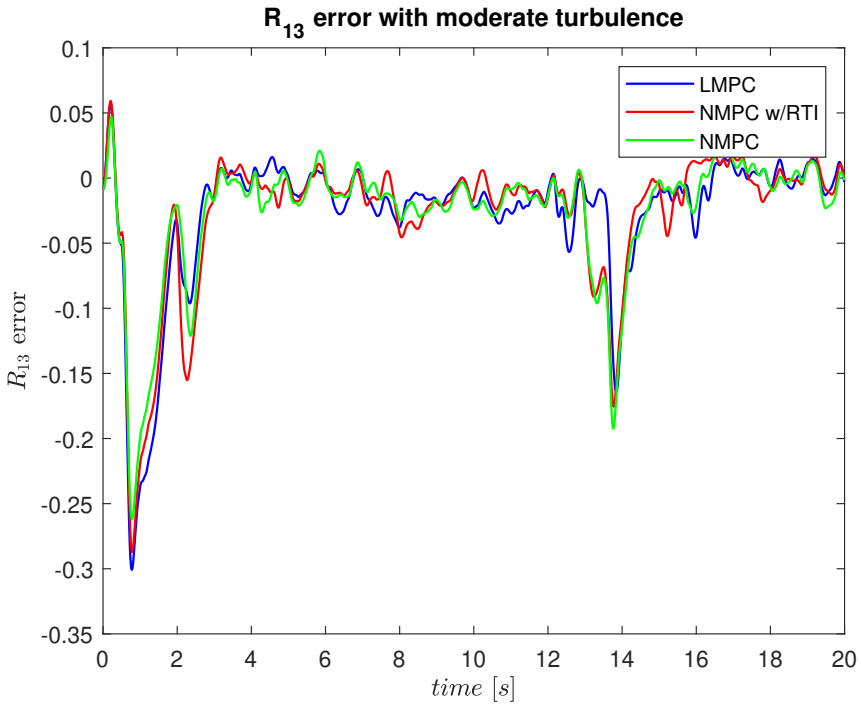


Figure 11: Error with moderate turbulence.

7 Conclusion

A total of 9 different simulations have been performed in order to determine the trade-off between computation time and aircraft dynamics. The LMPC has been tested against the NMPC, and also compared to NMPC with RTI. These have been compared in a range of environmental conditions, whose purpose is to single out the better performing controller.

In regards to computation time, it is clear that the nonlinear MPC with RTI is the fastest. This is a smart algorithm which takes advantage of recent dynamics to more effectively compute the nonlinear solution. When comparing the LMPC to NMPC, there is a large gap in computation time. The linear controller is approximately two times faster than the nonlinear controller, and this gap increases as the disturbances intensify. The nonlinear controller also needs some time to work its way down in computation time due to demanding dynamics in the beginning of the simulation. This is much better handled in the linear case.

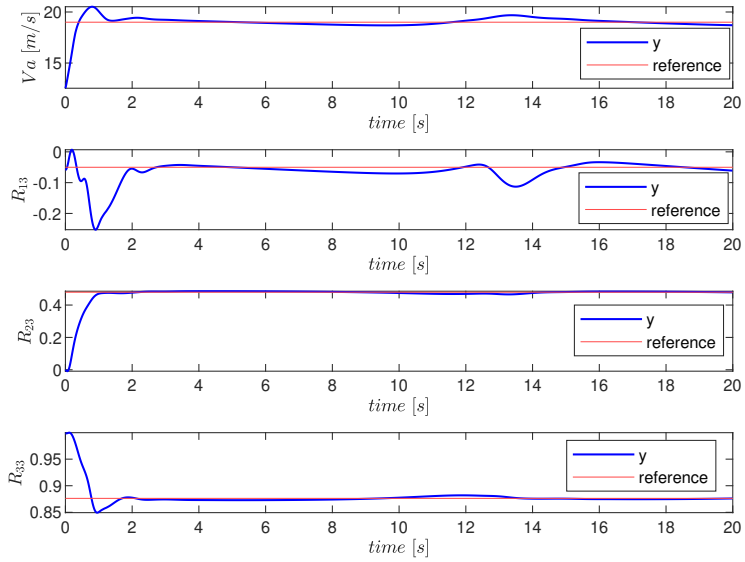
The aircraft dynamics are overall satisfying given the environmental conditions. There were not any large errors in either of the controllers, which indicates that computation time does not affect the system in a significant way. As the wind conditions gets worse, the LMPC is the better performing controller. The error dynamics expressed more overcompensation in the nonlinear controller, and the linear controller has slower and calmer error dynamics.

Further work

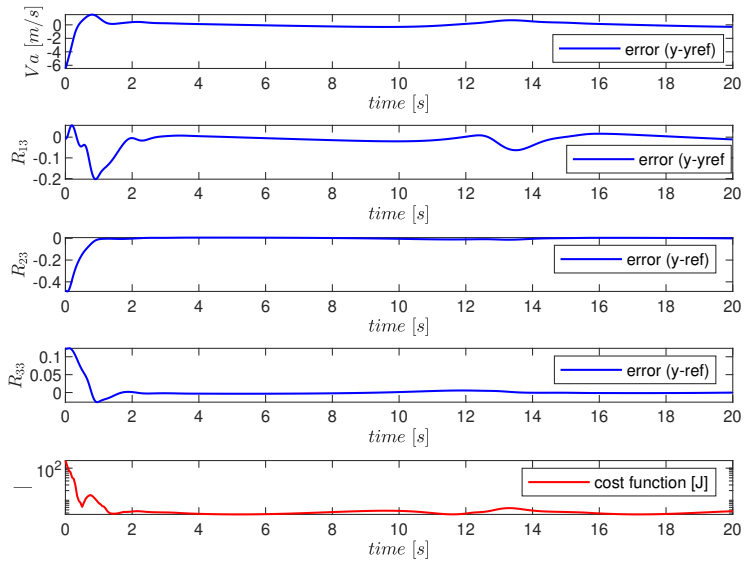
A natural continuation for this simulation study would be to test the results in real-world studies with the physical aircraft. It would also be interesting to identify the potential of how model simplifications and streamlining the algorithms improve computation time and controller performance. These tests resulted in a positive result for MPC in general, and comparing this control method with other methods could substantiate the performance of this controller even further or render it less effective.

A Simulation plots

A.1 Simulation 1



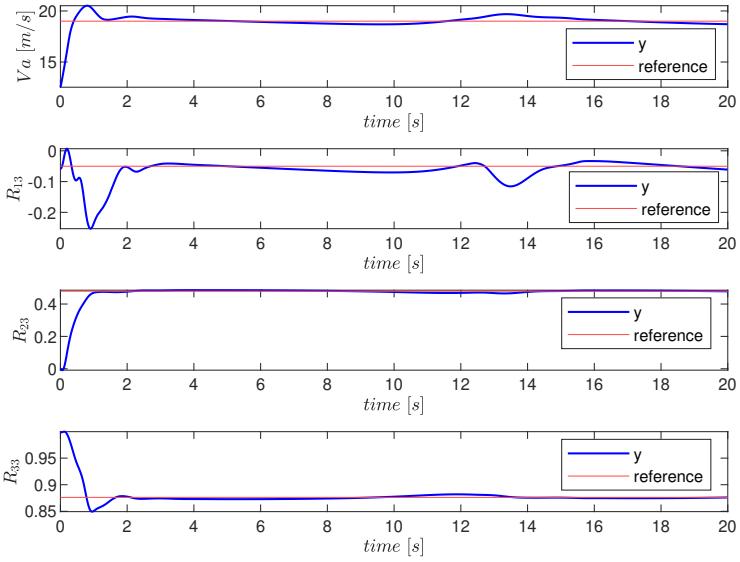
(a)



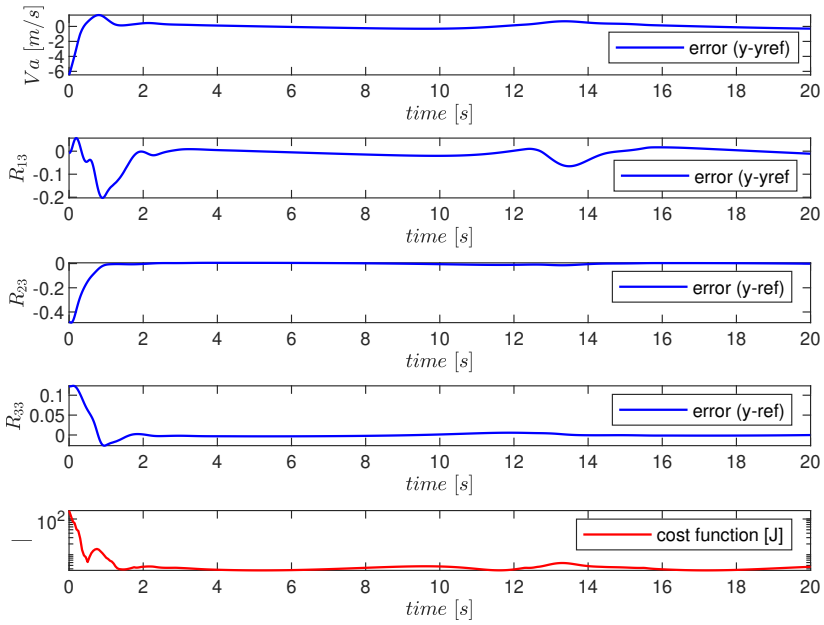
(b)

Figure 12

A.2 Simulation 2



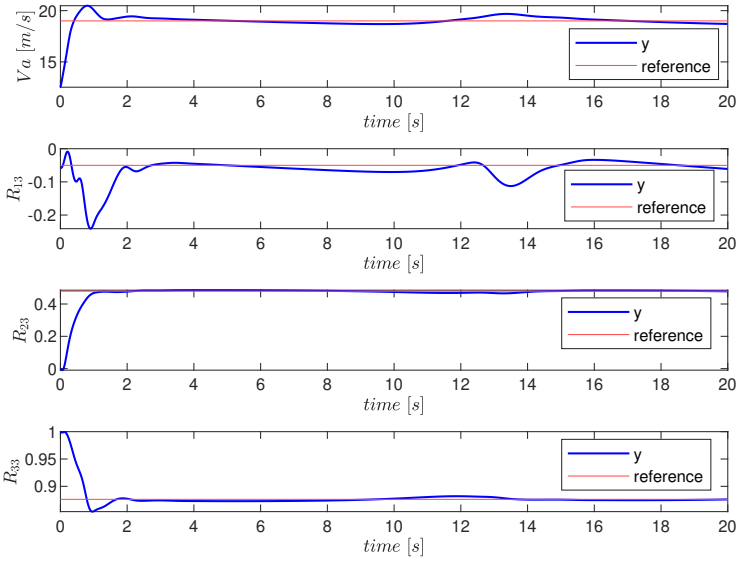
(a)



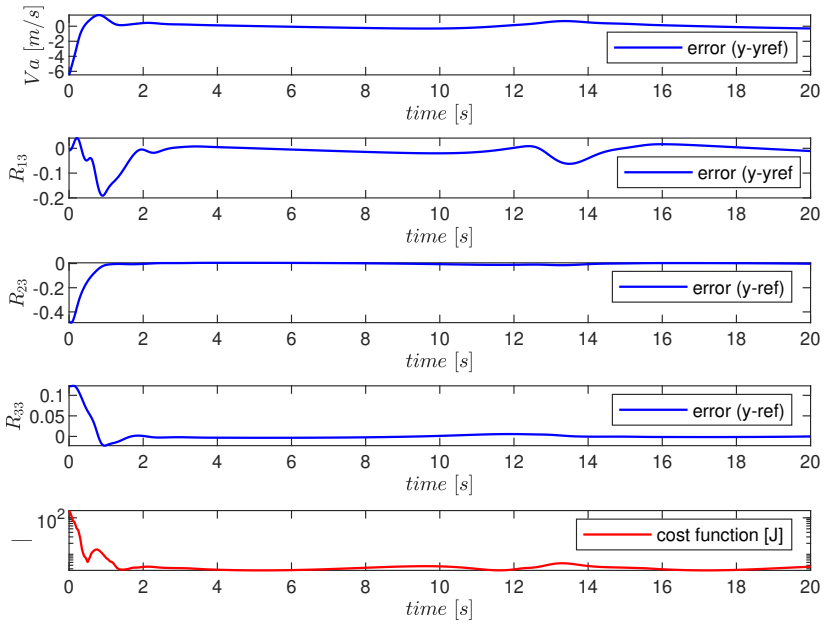
(b)

Figure 13

A.3 Simulation 3



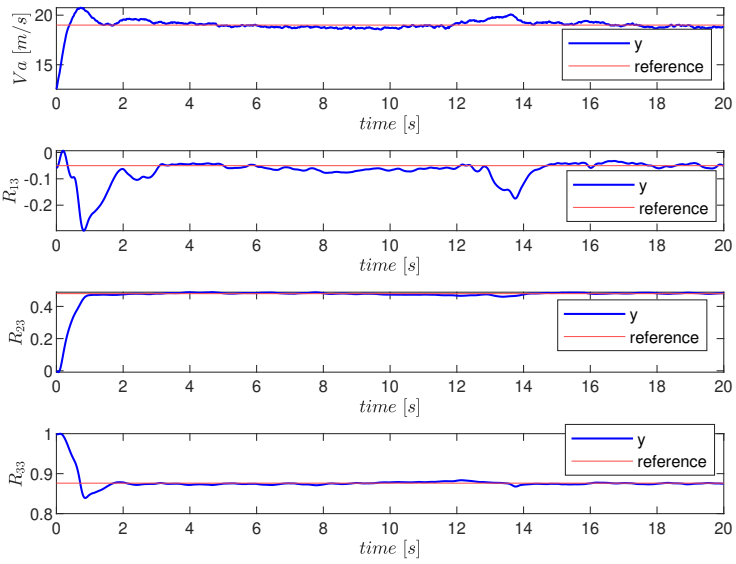
(a)



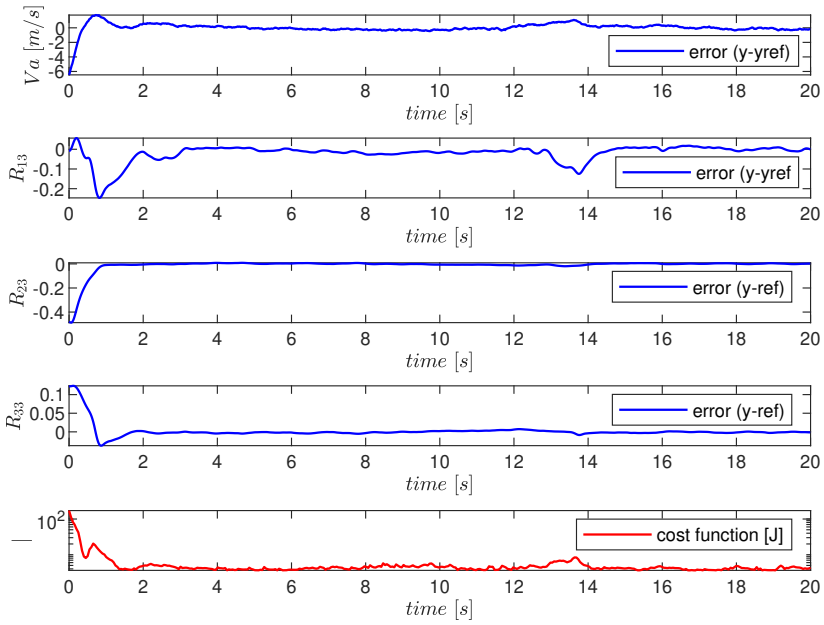
(b)

Figure 14

A.4 Simulation 4



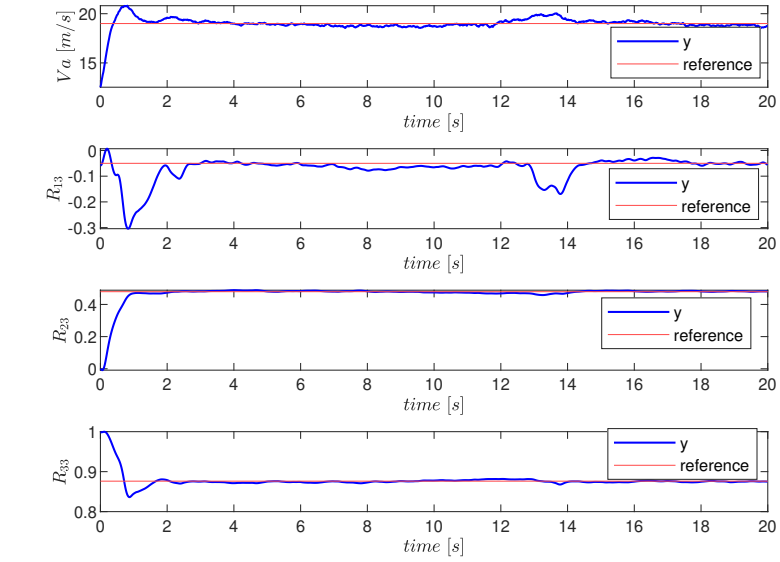
(a)



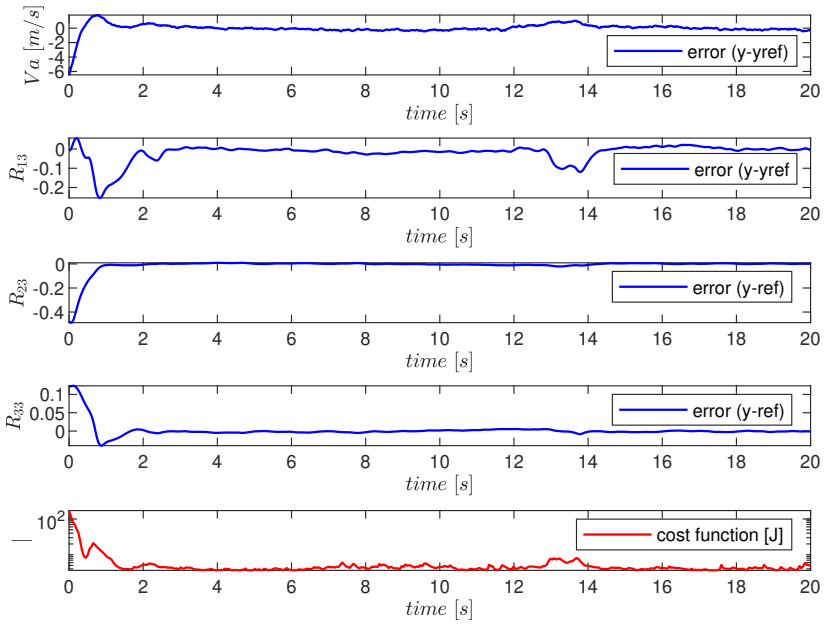
(b)

Figure 15

A.5 Simulation 5



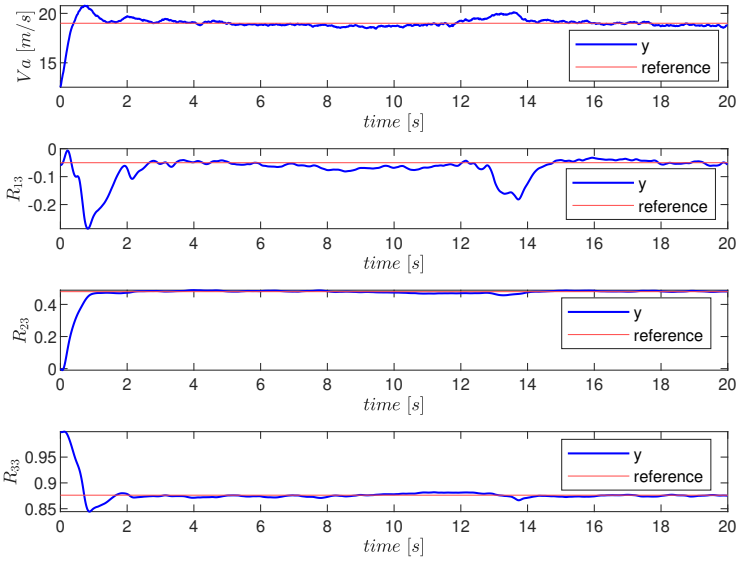
(a)



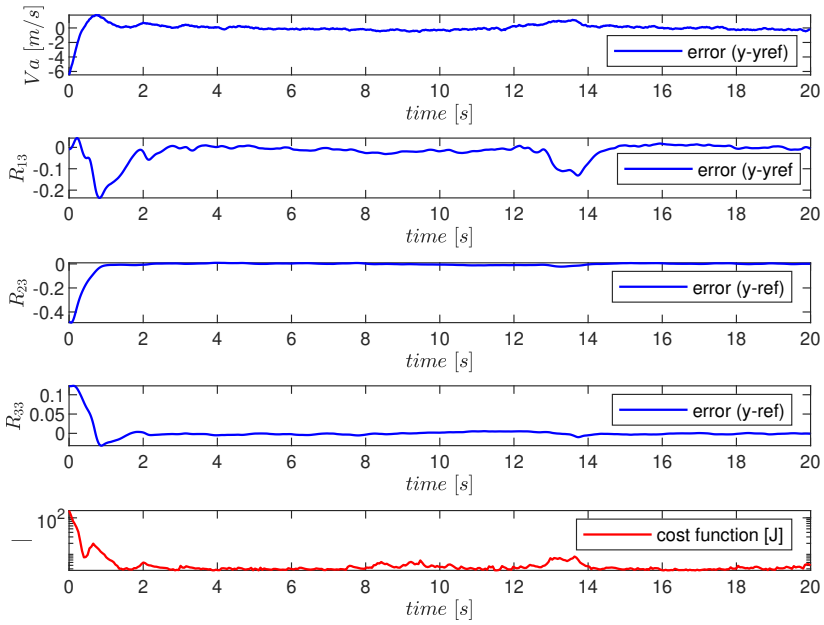
(b)

Figure 16

A.6 Simulation 6



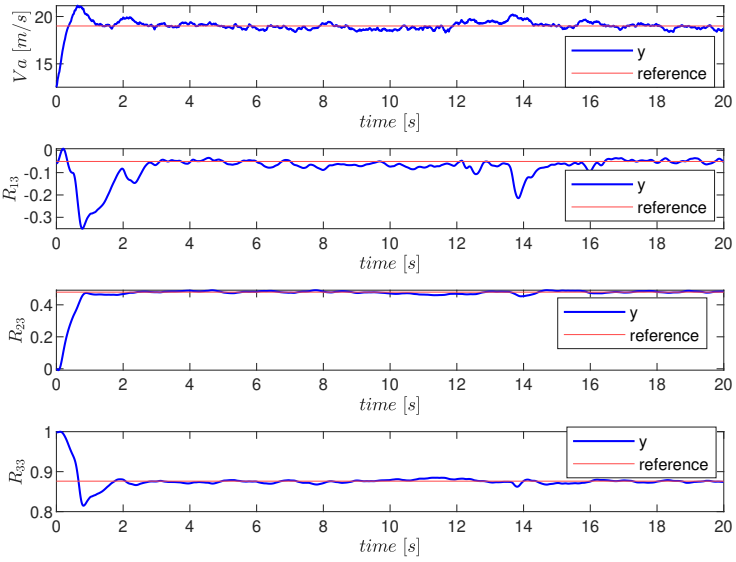
(a)



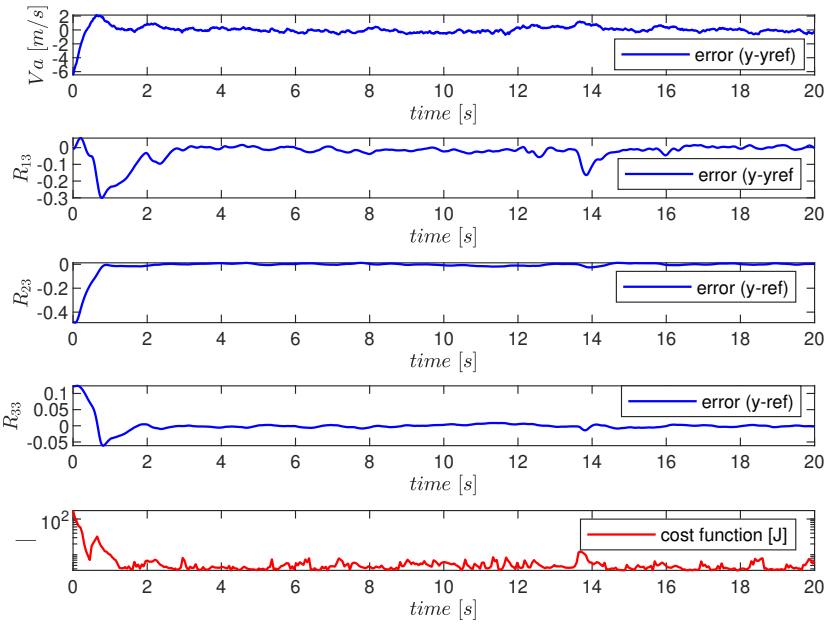
(b)

Figure 17

A.7 Simulation 7



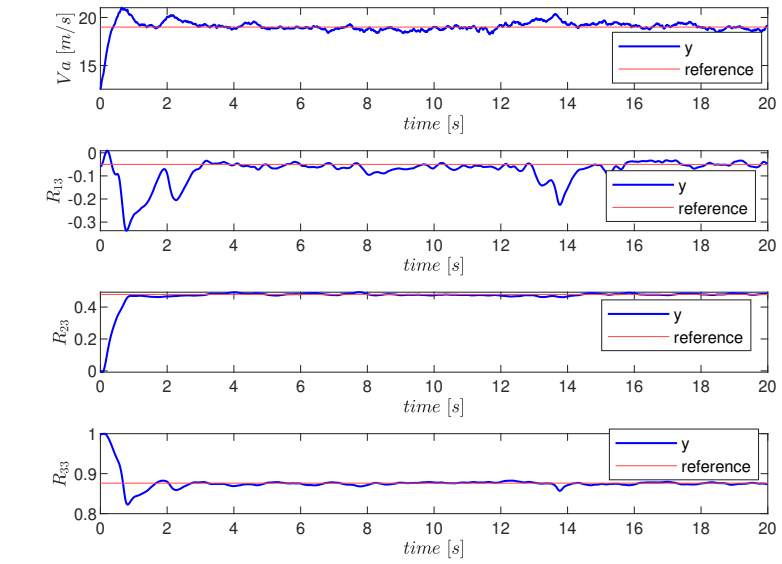
(a)



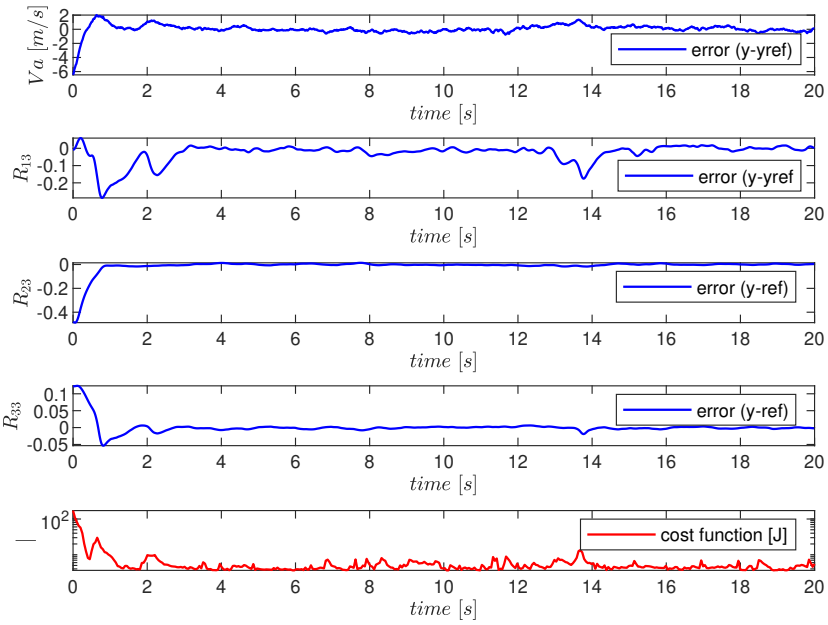
(b)

Figure 18

A.8 Simulation 8



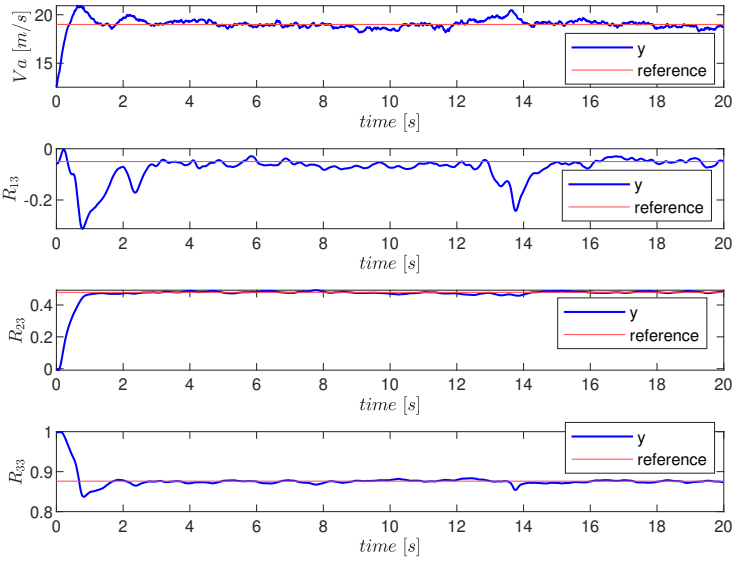
(a)



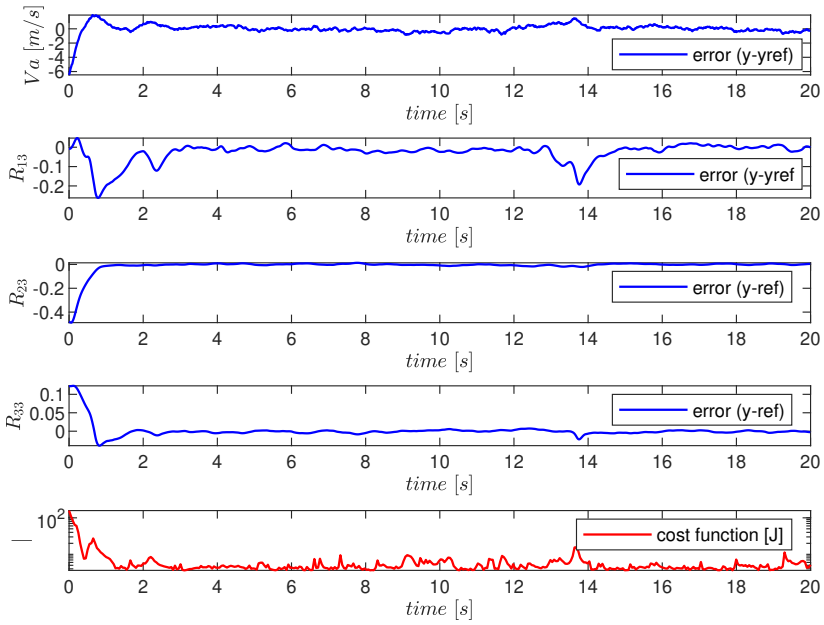
(b)

Figure 19

A.9 Simulation 9



(a)



(b)

Figure 20

B Computation times

LMPC - static wind (simulation 1)

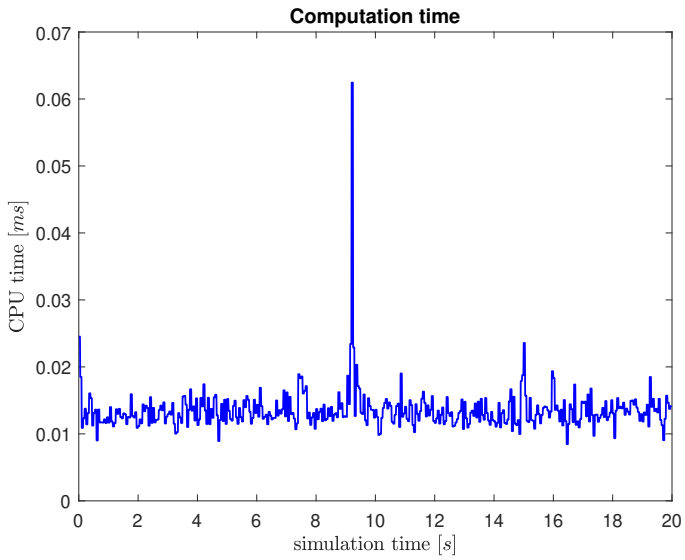


Figure 21

NMPC w/RTI - static wind (simulation 2)

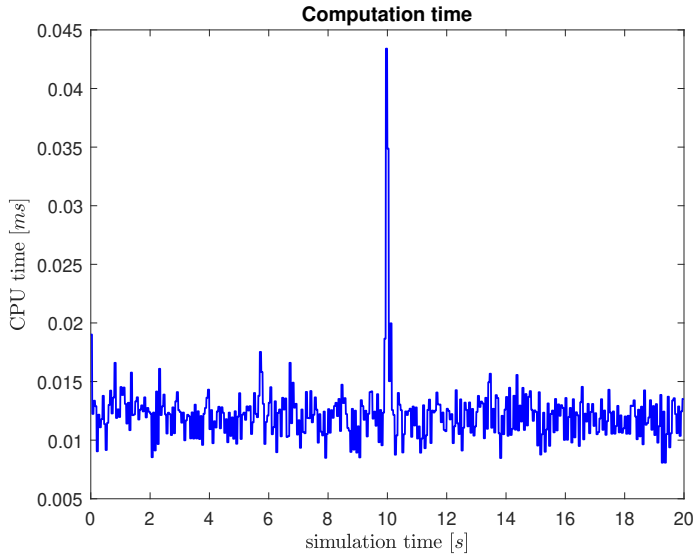


Figure 22

NMPC - static wind (simulation 3)

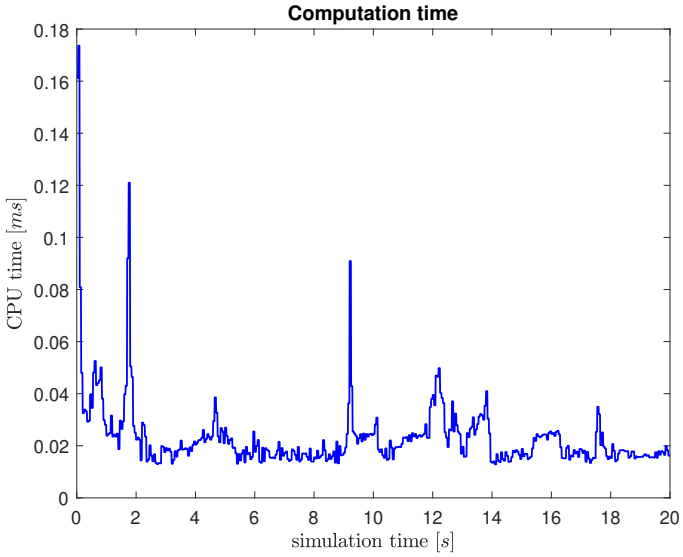


Figure 23

LMPC - light turbulence (simulation 4)

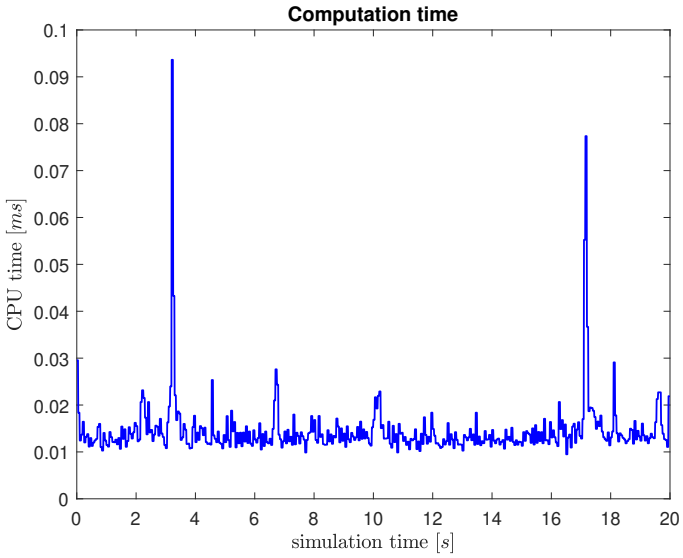


Figure 24

NMPC w/RTI - light turbulence (simulation 5)

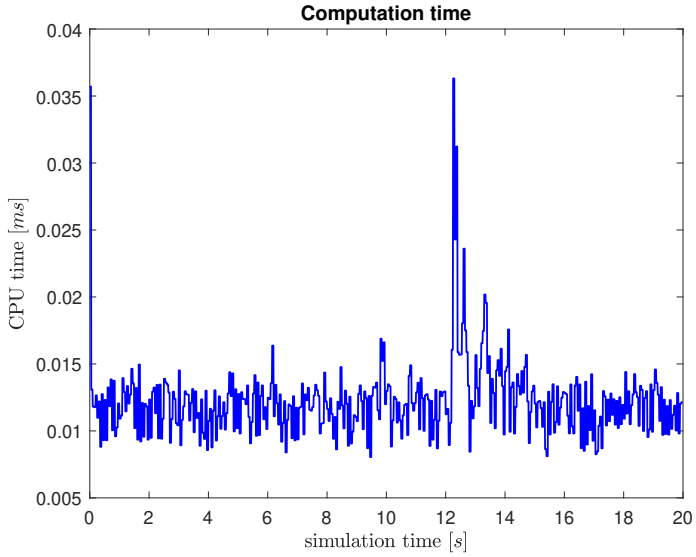


Figure 25

NMPC - light turbulence (simulation 6)

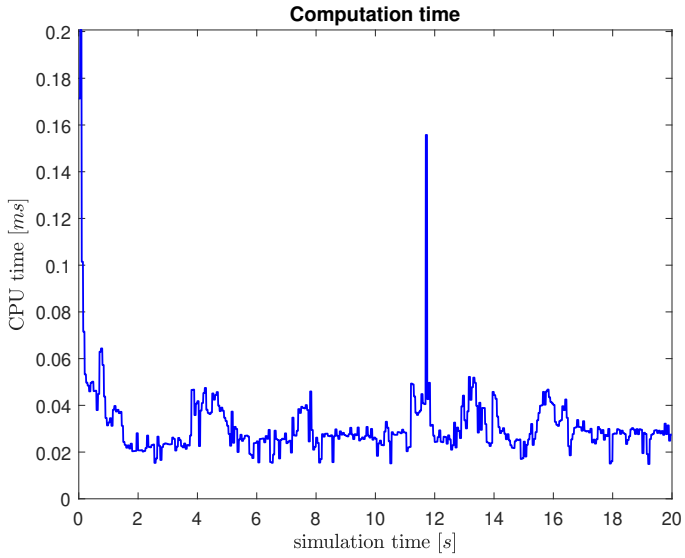


Figure 26

LMPC - moderate turbulence (simulation 7)

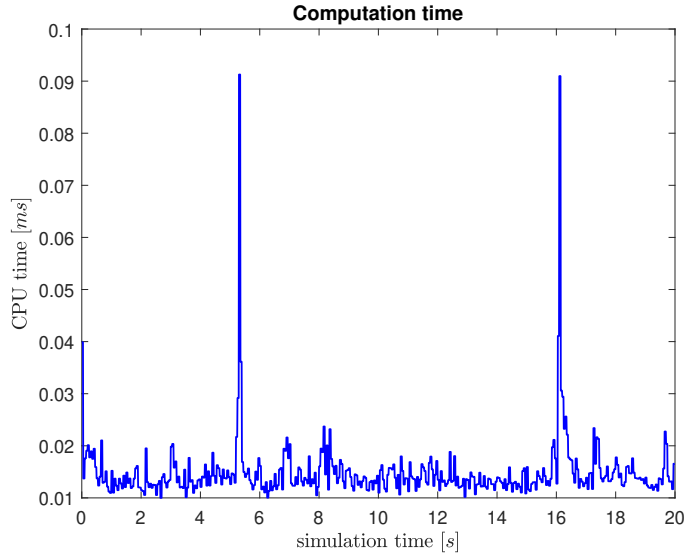


Figure 27

NMPC w/RTI - moderate turbulence (simulation 8)

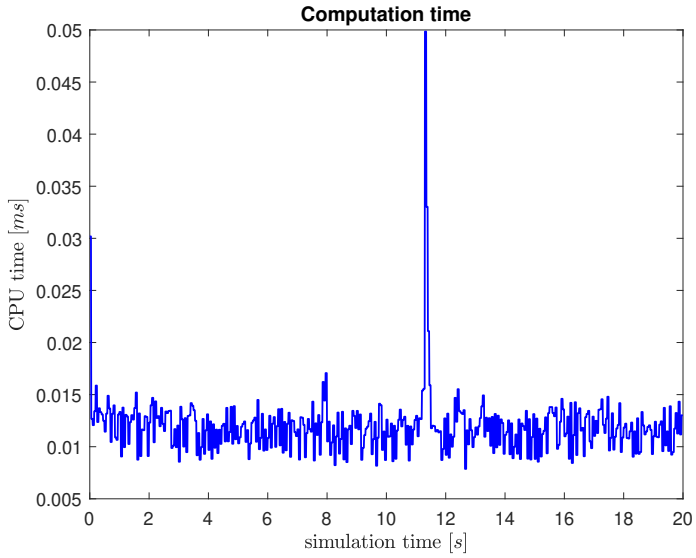


Figure 28

NMPC - moderate turbulence (simulation 9)

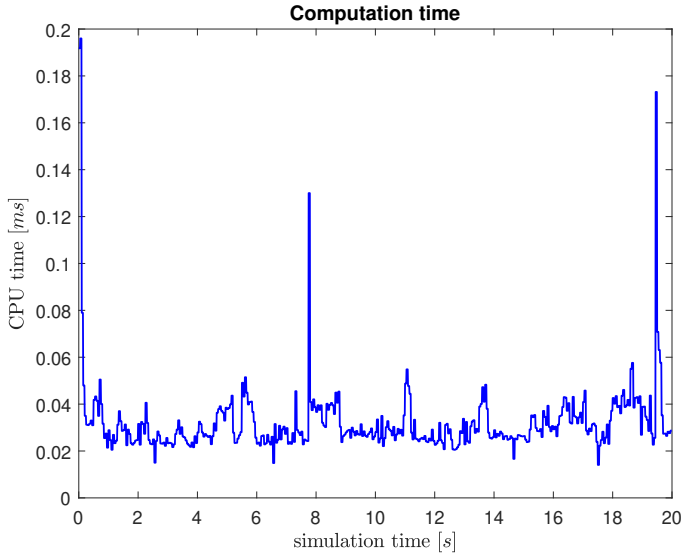


Figure 29

References

[1] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[2] Moritz Diehl. *Numerical Optimal Control (online)*. URL: `https://www.syscop.de/teaching/ss2020/numerical-optimal-control-online`. (accessed: 26.10.2020).

[3] Bjarne Foss and Tor Aksel N. Heirung. *Merging Optimization and Control*. 2016.

[4] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, 2021.

[5] Sebastien Gros et al. *From linear to nonlinear MPC: bridging the gap via the real-time iteration*. Sept. 2016.

[6] Kristoffer Gryte et al. "Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle." In: June 2018, pp. 826–835.

[7] Robin Verschueren et al. *acados: a modular open-source framework for fast embedded optimal control*. Oct. 2019.