Nonlinear Data-driven Model Predictive Control for 3-DoF Helicopters

Fabian Jakob

***Abstract*—In this work the Data-driven Model Predicitve Control approach is tested on a nonlinear 3-DoF helicopter application example. The proposed control algorithm uses the concept of artificial reachable setpoints using only measured input/output data for the linearized system parametrization. Therefore, three different concepts of data update schemes will be analyzed, providing simulation results for a corresponding Software-in-the-Loop model. Also, regarding real-time ability, different quadratic programming solvers and solution methods will be investigated.**

I. Introduction

Data-driven Control has received increasing attention in recent years due to the availability of large data amounts, growing computational power and simplicity comparing to difficult model descriptions. The idea of data-driven control combined with the accomplished framework of Model Predictive Control (MPC) leads to the promising data-driven MPC approach. Different theoretic guarantee providing schemes and algorithms are existing in literature, as well as many successful practical applications. In this work, the control method will be applied to a nonlinear unstable 3-DoF helicopter. In contrast to other works about avionic applications of data-driven MPC, the pre-stabilizing inner controller will be omitted, as the helicopter will be purely controlled by the proposed data-driven controller. This work mainly focuses on the simulative preliminary work. Further steps will be the direct implementation and case study on the real helicopter.

Mathematical preliminaries: This work makes use of the following notations:

(i) $u_{[a,b]}$ denotes the subvector of u, starting with index a and ending with index b.
(ii) I_n denotes the identity matrix of size $n \times n$. $\mathbf{1}_n$ indicates a column vector of size n, containing only ones as entries. In some places the index will be omitted, if the corresponding matrix multiplication inherits the dimension.
(iii) $\underline{\sigma}(A)$ denotes the minimum singular value of a A.
(iv) The weighted $2-$norm is defined by $\|x\|_Q^2 = x^T Q x$.
(v) The *Kronecker product* is defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}.$$

II. The 3-DoF Helicopter

The system to be controlled is a 3-DoF Helicopter by Quanser actuated by two DC motors driving the propellers and three encoders measuring the respective angles. The degree of freedoms are the travel angle α, elevation angle β and pitch angle γ as depicted in figure 1.

The plant inputs are given by the front- and backmotor force

$$u = \begin{pmatrix} F_F \\ F_B \end{pmatrix}, \tag{1}$$

measured in Newton and resulting from the applied input voltages U_F and U_B. The relations of input voltage and resulting motor forces are given by the characteristic force-voltage curve, which depends on the motor mechatronics and can vary for each helicopter. To reduce the linearization error, the motor forces will be used as plant inputs for analysis and design. A simple inversion of the force-voltage relation then gives the respective motor voltages to be applied. Since the voltages are bounded by $U_B, U_F \in [0\,\mathrm{V}, 4\,\mathrm{V}]$, the resulting forces are also bounded through the force-voltage curve. The determination of the curve can be done via regressing using measured data, see (1). In this work, a force bound of $F_{\max} = 0.98\,\mathrm{N}$ will be taken.

The controlled outputs are the measured travel, elevation and pitch angles

$$y = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}, \tag{2}$$

given in radians.

The relation between input u and output y is the underlying system dynamics

$$\begin{aligned} x_{k+1} &= f(x_k, u_k), \qquad x_0 \in \mathbb{R}^n \\ y_k &= h(x_k, u_k), \end{aligned} \tag{3}$$

which is specified by the unknown vector fields f, h and the order n of the system. The discrete-time series values

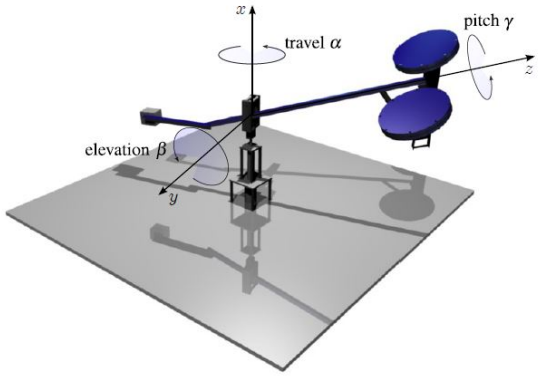


Figure 1. 3-DoF Helicopter [(1)].

$y_k = y(kT_s)$, $k \in \mathbb{Z}$ correspond to the time instances of the underlying continuous-time function $y(t)$, specified by a sampling time T_s (e.g. the cycle time of the microcontroller).

Intuitively, this system is unstable, which also can be shown by simple linearization based system analysis. An example of Newton-Euler based modelling of the 3-DoF Helicopter is given in (1), resulting in a nonlinear system of order $n = 6$. This order will also be used as an estimate for the real system dimension n in the following sections. The input and output dimensions are given by $m = 2$ and $p = 3$, respectively.

Stabilization of the 3-DoF helicopter based on LQG design is possible, as shown in (1). However, modelling and parameter identification becomes the main difficulty then, which can be skipped directly by simply obtaining I/O-data of the system within a data-driven control algorithm. The control method to be used is a nonlinear data-driven tracking MPC scheme as proposed in (2), (3). The goal will be an asymptotically stable tracking of a given set point y_{goal} while satisfying input constraints specified by the actuator limitations.

III. Data-based system parametrization

The key underlying concept of data-driven MPC is to parametrize all unseen trajectories of an LTI system by one single I/O-trajectory $\left\{u_k^d,\, y_k^d\right\}_{k=0}^{N-1}$, as captured in Willems fundamental Lemma (4).

Definition 1. *For a given sequence $\{u_k\}_{k=0}^{N-1}$, $u_k \in \mathbb{R}^m$, the respective Hankel matrix is defined as*

$$H_L(u) \coloneqq \begin{pmatrix} u_0 & u_1 & \dots & u_{N-L} \\ u_1 & u_2 & \dots & u_{N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{L-1} & u_L & \dots & u_{N-1} \end{pmatrix}.$$

Definition 2. *A sequence $\{u_k\}_{k=0}^{N-1}$, $u_k \in \mathbb{R}^m$, is said to be persistently exciting of order L, if*

$$\operatorname{rank} H_L(u) = mL. \tag{4}$$

Theorem 1 ((4), Willems Fundamental Lemma)**.** *Suppose $\left\{u_k^d,\, y_k^d\right\}_{k=0}^{N-1}$ is a trajectory of a controllable LTI system, where u^d is persistently exciting of order $L + n$. Then $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L}$ is a trajectory of the system if and only if there exists an $\alpha \in \mathbb{R}^{N-L+1}$, such that*

$$\begin{pmatrix} H_L(u^d) \\ H_L(y^d) \end{pmatrix} \alpha = \begin{pmatrix} \overline{u} \\ \overline{y} \end{pmatrix}. \tag{5}$$

Note that the input hankel matrix has the dimensions $H_{L+n}\left(u^d\right) \in \mathbb{R}^{m(L+n)\times(N-L-n+1)}$. Therefore, in order to assure $\operatorname{rank} H_{L+n}(u) = m(L+n)$, the number of columns has to be larger than the number of rows, i.e.

$$N \geq (m+1)(L+n) - 1. \tag{6}$$

In other words, for persistence of excitation it is necessary to have sufficiently long data.

Overall, condition (5) is just another way to describe the I/O-dynamics of an LTI system by simply obtaining data and without any need for modelling. The Fundamental Lemma will be used to describe the linearization of the nonlinear system (3). However, linearizations around arbitrary operating points are generally not LTI, as additionally affine terms are emerging once $(\bar{x}, \bar{u})$ is not an equilibrium of the underlying system dynamics. Thus, the system to be parametrized is described by

$$\begin{aligned} \Delta x_{k+1} &= f(\overline{x},\overline{u}) + \underbrace{\left.\frac{\partial f}{\partial x}\right|_{\overline{x},\overline{u}}}_{A} \Delta x_k + \underbrace{\left.\frac{\partial f}{\partial u}\right|_{\overline{x},\overline{u}}}_{B} \Delta u_k \\ \Delta y_{k+1} &= h(\overline{x},\overline{u}) + \underbrace{\left.\frac{\partial h}{\partial x}\right|_{\overline{x},\overline{u}}}_{C} \Delta x_k + \underbrace{\left.\frac{\partial h}{\partial u}\right|_{\overline{x},\overline{u}}}_{D} \Delta u_k, \end{aligned} \tag{7}$$

where $f(\overline{x},\overline{u}), h(\overline{x},\overline{u})$ are generally not equal to zero. Hence, Theorem 1 cannot be applied. However, the following Theorem, proposed in (3), is an extended version of Willem's fundamental Lemma for affine systems, which will be used for the application of the data-driven MPC algorithm to the helicopter.

Theorem 2. *Suppose system (7) is controllable and the data $\left\{u^d, y^d\right\}_{k=0}^{N-1}$ with corresponding state $\left\{x^d\right\}_{k=0}^{N-1}$ satisfy*

$$\operatorname{rank} \begin{pmatrix} H_L\left(u^d\right) \\ H_L\left(x^d{}_{[0,N-L]}\right) \\ \mathbf{1}^T \end{pmatrix} = mL + n + 1. \tag{8a}$$

Then $\{\overline{u}_k, \overline{y}_k\}_{k=0}^{L}$ is a trajectory of the system, if and only if there exists an $\alpha \in \mathbb{R}^{N-L+1}$, such that

$$\begin{pmatrix} H_L\left(u^d\right) \\ H_L\left(y^d\right) \\ \mathbf{1}^T \end{pmatrix} \alpha = \begin{pmatrix} \overline{u} \\ \overline{y} \\ 1 \end{pmatrix}. \tag{8b}$$

The modified system parametrization (8b) differs only in the additional condition that all elements of α must sum up to 1. The modified persistence of excitation condition (8a) is non-trivial to verify. Further details are discussed in (3).

IV. The Data-driven MPC Algorithm

The control scheme to be applied is a data-driven linear tracking MPC for nonlinear systems, as proposed in (2), (3). The algorithm is based on solving a simple MPC scheme with a prediction model based on Willem's fundamental Lemma. The data trajectories in the Hankel matrices describe the local behaviour of the system, which corresponds to the affine linearization (7) around the operating points. In the scope of this work, different data update schemes will be investigated, for both fixed system parametrizing trajectories and time-varying data.

A. The algorithm

Analogous to the model-based MPC case, the control input is obtained by solving a finite horizon optimal control problem,

having the measured state as initial condition. In the data-driven case, the dynamics is replaced by Willem's fundamental Lemma in order to predict the open-loop trajectories. For nonlinear systems, the extended Fundamental Lemma for affine system parametrization will be taken. The initial condition will be replaced by a measured initial I/O-trajectory, taking the past n measurements. For convenience, define

$$\tilde{L} = L + n, \tag{9}$$

as this will be the horizon of the optimal control problem, consisting of the prediction horizon L and n initialization points. Then the control algorithm is as follows:

1) Collect the past n measurements $\{u_k,\, y_k\}_{k=t-n}^{t-1}$ at time t.
2) Solve

$$\min_{\substack{\alpha,\overline{u},\overline{y},\\ u^s,y^s,\sigma}} \sum_{k=0}^{\tilde{L}-1} \|\overline{y}_k - y^s\|_Q^2 + \|\overline{u}_k - u^s\|_R^2 + \|y^s - y_{\text{goal}}\|_S^2 + \lambda_\alpha \|\alpha\|^2 + \lambda_\sigma \|\sigma\|^2 \tag{10a}$$

$$\text{s.t.} \quad \begin{pmatrix} H_{\tilde{L}}\left(u^d\right) \\ H_{\tilde{L}}\left(y^d\right) \\ \mathbf{1}^T \end{pmatrix} \alpha = \begin{pmatrix} \overline{u} \\ \overline{y} + \sigma \\ 1 \end{pmatrix} \tag{10b}$$

$$\begin{pmatrix} \overline{u}_{[0,n-1]} \\ \overline{y}_{[0,n-1]} \end{pmatrix} = \begin{pmatrix} u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \end{pmatrix} \tag{10c}$$

$$\begin{pmatrix} \overline{u}_{[\tilde{L}-n-1,\tilde{L}-1]} \\ \overline{y}_{[\tilde{L}-n-1,\tilde{L}-1]} \end{pmatrix} = \begin{pmatrix} u^s \otimes \mathbf{1}_{n+1} \\ y^s \otimes \mathbf{1}_{n+1} \end{pmatrix} \tag{10d}$$

$$\overline{u}_k \in \mathbb{U}, \;\; u^s \in \mathbb{U}^s \tag{10e}$$

3) Apply $u_t = \overline{u}_n$ as control input, set $t \leftarrow t+1$ and go back to 1.

The trajectory $\{u^d, y^d\}$ of course has to be persistenly exciting and is usually generated offline before the control task. There are existing schemes in literature where the data is updated online during closed loop. However, even then a trajectory has to be pre generated for initialization. This topic will be adressed in section V.

The parameter sigma σ accounts for the model error due to measurement noise and linearization. In order to prevent overfitting, both α and σ are additionally penalized in the cost function. The weighting matrices $Q \succ 0, R \succ 0$ and $S \succ 0$ as well as the regularization and cost penalty $\lambda_\alpha \geq 0$ and $\lambda_\sigma \geq 0$ have to be defined a-priori by the engineer. The influence of those parameters on stability and performance will be discussed in section VI-A.

The setpoint (u^s, y^s) to be reached is an optimization variable by itself. Hence, with the dynamics constraints (10b), the algorithm calculates the optimal *reachable* setpoint within the horizon $\tilde{L}$. The weighting term $\|y^s - y_{\text{goal}}\|_S^2$ thereby assures to stay close to the actual reference, independent whether this is a stationary point or not.

Of course, the data $\left\{u_k^d,\, y_k^d\right\}_{k=0}^N$ should still cover only small operational ranges since theorem 2 only holds for linear (linearized) system behaviour (5). This might raise some issues if the system has to be controlled at different setpoints. One simple modification might be updating the data during closed loop. However, persistence of excitation is then not guaranteed and counterintuitive to the asymptotic stabilization of a setpoint. Another modification is scheduling the data in the Hankel matrix with different trajetories collected around different setpoints. Those three concepts, fixed data, updating the data and scheduling the data, will all be investigated in section VI-C.

B. The quadratic Program

Note that the optimization problem (10) is a quadratic program (QP) and can therefore be rewritten in matrix form as

$$\min_z \;\; \frac{1}{2} z^T H z + f^T z \tag{11a}$$

$$\text{s.t.} \;\; A_{\text{eq}} z = b_{\text{eq}} \tag{11b}$$

$$A_{\text{ineq}} z \leq b_{\text{ineq}}. \tag{11c}$$

The input constraints (10e) arise as $u_{\min} \leq u \leq u_{\max}$ and are therefore the only inequality constraints. For convenience, (11c) will be replaced by

$$z_{\min} \leq z \leq z_{\max}, \tag{11d}$$

which, for example, can be passed to `quadprog` in `MATLAB`.

This reformulation is key, as those matrices are inputs for most quadratic program algorithms, as `quadprog` or `qpOASES` for example.

For convenience, define the abbreviations

$$\begin{aligned} H_u &:= H_{\tilde{L}}\left(u^d\right), \quad H_y := H_{\tilde{L}}\left(y^d\right) \\ \boldsymbol{R} &:= R \otimes I_{\tilde{L}}, \qquad \boldsymbol{Q} := Q \otimes I_{\tilde{L}} \\ \boldsymbol{R_1} &:= R \otimes \mathbf{1}_{\tilde{L}}, \qquad \boldsymbol{Q_1} := Q \otimes \mathbf{1}_{\tilde{L}}. \end{aligned} \tag{12}$$

By stacking up all decision variables into a column vector

$$z = \begin{pmatrix} \alpha \\ \overline{u} \\ u^s \\ \overline{y} \\ y^s \\ \sigma \end{pmatrix}, \tag{13a}$$

the cost function matrices read as

$$H = 2 \begin{pmatrix} \lambda_\alpha I & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{R} & -\boldsymbol{R_1} & 0 & 0 & 0 \\ 0 & -\boldsymbol{R_1}^T & \tilde{L}R & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{Q} & -\boldsymbol{Q_1} & 0 \\ 0 & 0 & 0 & -\boldsymbol{Q_1}^T & S + \tilde{L}Q & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_\sigma I \end{pmatrix} \tag{13b}$$

$$f^T = \begin{pmatrix} 0 & 0 & 0 & 0 & -2y_{\text{goal}}^T S & 0 \end{pmatrix}. \tag{13c}$$

The constraint matrices result in

$$A_{\text{eq}} = \begin{pmatrix} H_u & -I & 0 & 0 & 0 & 0 \\ H_y & 0 & 0 & -I & 0 & -I \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} 0 & I \end{pmatrix} & -I \otimes \mathbf{1}_{n+1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \begin{pmatrix} 0 & I \end{pmatrix} & -I \otimes \mathbf{1}_{n+1} & 0 \\ \mathbf{1}^T & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{13d}$$

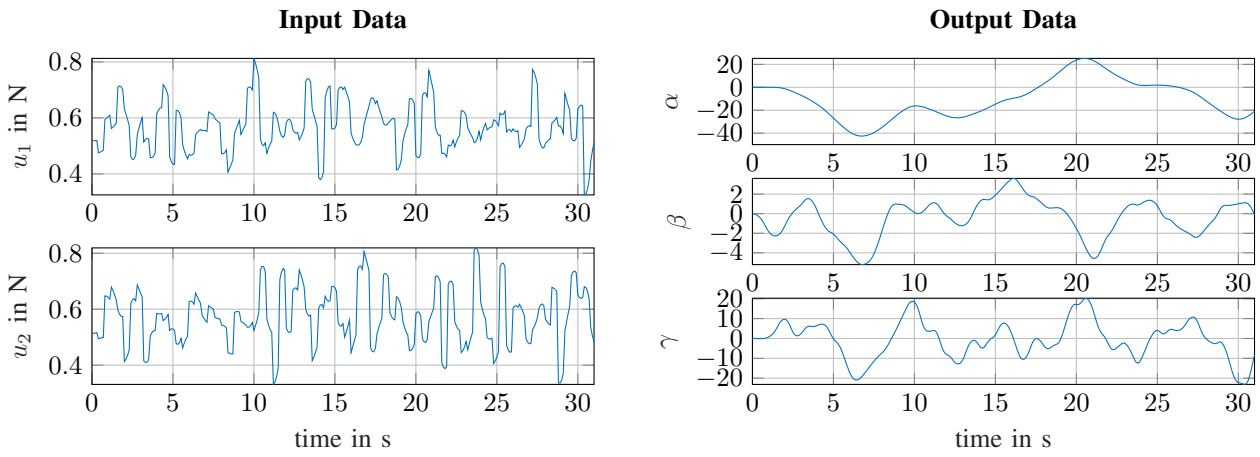


Figure 2. Data Trajectory for the travel task, resulting from simulation of the model.

$$b_{\text{eq}} = \begin{pmatrix} 0 \\ 0 \\ u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{13e}$$

Note the sparse structure of the Hessian matrix H, which can be exploited by numerical quadratic program solvers.

V. Data Trajectory Generation

As all system trajectories in the MPC algorithm are parametrized by one single I/O-trajectory, the generation of that data trajectory $\{u_k^d,\, y_k^d\}_{k=0}^{N}$ is naturally of significant performance, independent of the data update scheme. Even when updated in closed loop, an initialization trajectory is still needed.

However, data generation is not trivial for unstable plants, as instability does not automatically inherit persistence of excitation. Also, one might avoid damaging the unstable system during trajectory generation. Thus, in order to generate the initial I/O-trajectory, the plant first has to be pre-stabilized. Therefore an observer based LQG controller is designed for the linearized system around the zero setpoint, based on a simple model description with respective parameters obtained by Newton-Euler-methods as given in (1). Other methods might be manual joystick control by a human.

For the algorithm, two trajectories will be generated, purely simulative. One for the travel task and one for the elevation task. Of course, the solution of the MPC problem will depend on how suited the data trajectory is and therefore also dependends on the control parameters in the data generation. The following parameters have proven beneficial for the quality of the resulting trajectories and thus, for the closed loop performance.

The weight matrices for the LQG controller are chosen as

$$\begin{aligned} R_{\text{LQR}} &= I_m, \;\; Q_{\text{LQR}} = 5I_n \\ R_{\text{LQR}} &= I_m, \;\; Q_{\text{LQR}} = \text{diag}(500, 1, 5, 1, 1, 1) \end{aligned}$$

for the travel and elevation task, respectively, and for the observer

$$R_{\text{obs}} = I_p, \;\; Q_{\text{obs}} = 50I_n,$$

which then results in the respective LGQ controller and observer gain matrices K_{LQR} and L_{obs}.

In order to achieve persistence of excitation, a uniformly distributed random noise is added on the plant input. The noise is in the range of $u_{\text{disturb}} \in [-0.2\,\text{N}, 0.2\,\text{N}]$ with a sampling time of $t_{\text{disturb}} = 0.4s$ for the travel task and $t_{\text{disturb}} = 0.3s$ for the elevation task. The disturbance is large enough to ensure persistence of excitation and small enough to not endanger stability. Those parameters are also tuning parameters for the resulting control algorithm, since the system parametrization depends on the quality of the data trajectory used in the Hankel matrices.

For the travel task, the helicopter starts in the zero position $(\alpha_0, \beta_0, \gamma_0) = (0°, 0°, 0°)$ and is excited by the input disturbance. For the elevation task, the helicopter starts and flies around its home position $(\alpha_0, \beta_0, \gamma_0) = (0°, -30°, 0°)$.

The sampling rate is chosen to be $t_{\text{sample}} = 0.1\,\text{s}$ within a simulation time of $t_{\text{sim}} = 31\,\text{s}$. Figure 2 shows the resulting trajectory $\{u_k^d,\, y_k^d\}_{k=0}^{N-1}$ for the travel task.

To ensure that persistence of excitation is achieved, the corresponding input Hankel matrix $H_{L+n}(u^d)$ is generated. The prediction horizon is chosen as $L = 50$, which corresponds to $5\,\text{s}$ in the continuous system. This matrix has a minimum singular value of

$$\begin{aligned} \underline{\sigma}\left(H_{\tilde{L}}\left(u^d\right)\right) &= 0.0529, \qquad \text{(travel task)} \\ \underline{\sigma}\left(H_{\tilde{L}}\left(u^d\right)\right) &= 0.0488, \qquad \text{(elevation task)} \end{aligned}$$

which are unequal to zero. Thus, the input is indeed persistently exciting of order $L + n$.

VI. Simulation results with `quadprog`

To investigate the influence of each control parameter, a case study in `MATLAB/Simulink` with the simulation model will be performed. Problem (10) will be solved with the QP-solver `quadprog`.

A. Influence of Optimization Parameters

Particulary, the influence of Q, R, S, λ_α and λ_σ will be investigated. Therefore, the controlled helicopter has to perform a lift from home position $(\alpha_0, \beta_0, \gamma_0) = (0°, -30°, 0°)$ to its horizontal position $\beta = 0°$ and a turn to $\alpha = 90°$ on the same altitude and back. The tracking performance will be quantized by the tracking error of the angles α and β, i.e.

$$e = \frac{1}{T_{\text{sim}}} \sum_{k=0}^{T_{\text{sim}}} \|\alpha_k - \alpha_{\text{goal}}\|^2 + 3\,\|\beta_k - \beta_{\text{goal}}\|^2 ,$$

given in °, where β is weighted three times as much as α, since errors in the elevation might be in a smaller order of magnitude. Also, the norm of the steady-state error e_{ss} will be considered. The tracking error of γ is not considered since its only purpose is the stabilization of the helicopter.

The optimization will be performed by doing a line search, i.e. keeping all parameters but one fixed and optimize over that parameter. The system parametrization will stay fixed with the pre-generated travel data trajectory from section V as input to the Hankel matrices.

1) Input/output weightings: Taking pre-optimized values values for all parameters besides the cost weightings Q and R, their influence on the control performance will be investigated. The output weighting has been chosen to be diagonal, i.e. $Q = qI_p$, with $q > 0$ being the variable parameter. The error values are given in table I. The value $q = 15$ provides a good trade-off between small tracking error and steady-state error. Also as a remark, a smaller weighting of the third output γ, which intuitively should lead to a better tracking of the first two outputs, surprisingly lead to growing error. Also no clear effects on different weightings in y_1 and y_2 could be observed. Thus, the equally weighted diagonal structure seems reasonable. Fixing now the output weighting at $q = 15$, the resulting error values for an input weighting $R = qI_m$ of the same structure are computed, also given in table I. A value of $r = 10$ has been chosen, as this indeed not minimizes the error, but also leads to minimum steady state error.

Table I
INFLUENCE OF INPUT AND OUTPUT WEIGHTING. THE ERRORS ARE GIVEN IN DEGREE.

q	e	e_{ss}
0.1	21.65	4.24
1	21.31	3.21
15	20.09	1.67
20	19.97	2.11
30	19.93	2.64

r	e	e_{ss}
0.1	20.06	1.83
1	20.09	1.67
10	20.22	0.8
20	20.4	0.86
30	20.51	0.86

2) Artificial setpoint weighting: The parameter S ensures y^s to be close to y_{goal}. Thus, it interacts with the output weighting Q as this parameter ensures y to be close to y^s. As expected, a high choise of Q leads to a high deviation of y^s and y_{goal}, since the output cost term then dominates the cost function. This can be counteracted by a high choice of S. By observation, travel and elevation angle α and β typically deviate most, as a tracking of the respective reference angle

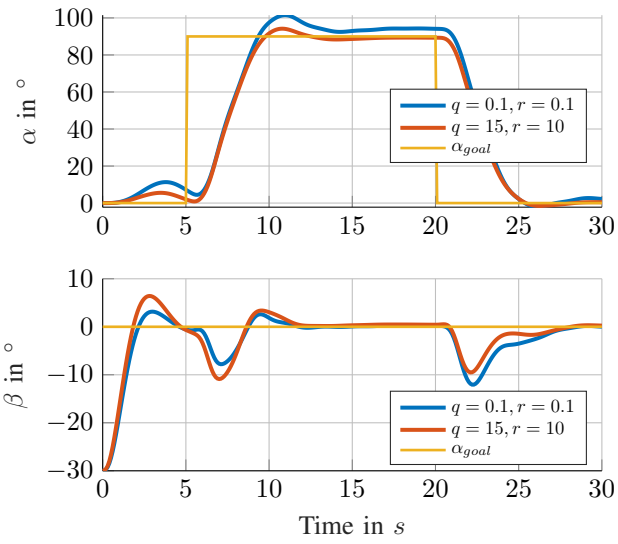


Figure 3. Comparison of different input/output weightings on the control performance.

required the most energy out of all angles. Therefore a rather high weighting for α is chosen. For comparison, a choice of

$$S_1 = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad S_2 = \begin{pmatrix} 5000 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

lead to tracking errors of $e_1 = 40.05°$ and $e_2 = 20.23°$, respectively. A comparison of the evolution of the artificial setpoint y_s for the travel angle is shown in figure 4.

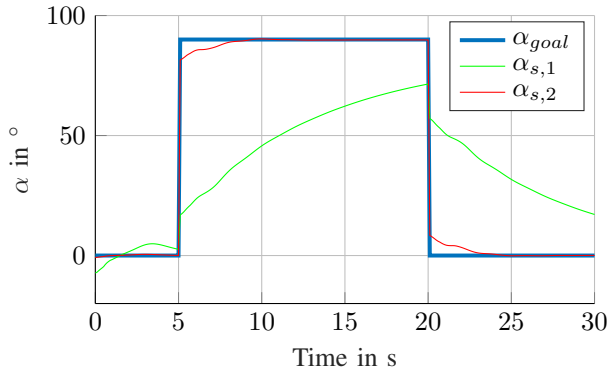


Figure 4. Evolution of the artificial setpoint for different weightings.

3) Regularization in the noise free case: Since the noise free case is considered, the regularizations should only compensate for nonlinearity errors. As no output measurement errors occur, it is expected that no big corrections by the slack variable σ are needed. Therefore, a rather high slack weighting λ_σ is expected for good tracking, as too moderate weighting of σ leads to large values and thus to inaccurate predictions. Otherwise, too aggressive weighting leads to no compensation at all. Indeed a parameter range of $10^5 - 10^6$ does achieve good behaviour, as table II shows. Outside of this range, deviations of y from y^s are observed. For too aggressive weightings, high oscillations in y_3 occured, see figure 5.

Table II
INFLUENCE OF REGULARIZATION PARAMETERS IN THE NOISE FREE CASE

λ_α	e	e_{ss}
0.1	unstable	unstable
1	26.24	6.02
10	20.17	3.09
100	20.91	0.51
150	21.37	0.42
200	21.74	0.69

λ_σ	e	e_{ss}
10^3	unstable	unstable
10^4	28.42	1.37
10^5	20.91	0.51
10^6	20.22	0.8
10^7	20.71	1.26
10^8	21.15	oscillating

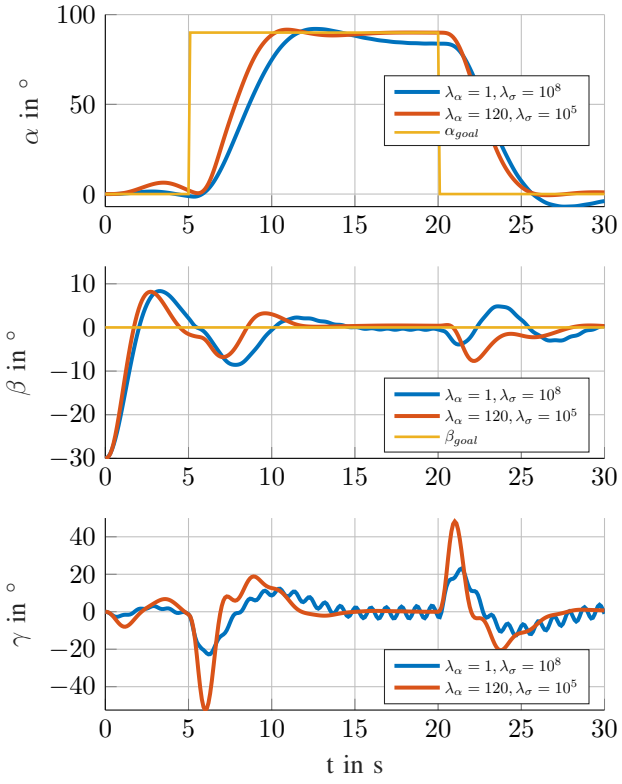


Figure 5. Comparison of different regularization parameters on the control performance.

4) Regularization in the noisy case: In order to study the effect of noise, a uniformly distributed random noise in the range of $y_{\text{noise}} \in [-1°, 1°]$ will be added to both data and closed loop output. The noise is chosen rather high to get conservative results, as angle sensors typically are much more precise. As expected, both e and e_s lie in significantly larger magnitudes than in the noise free case, see table III. Intuitively, larger regularization is needed in the noisy case, which corresponds to higher λ_α and smaller λ_σ values, which indeed can be observed in table III. However, the margin between the optimal slack weighting and the stability boundary gets smaller.

B. Influence of Data/Prediction Horizon

As it comes to stability and computational effort, the predicition and data horizon L and N are of significant importance, as stability guarantees grow with larger horizons at the cost of high order quadratic programms, which lead to higher online complexity. For complexity measurement, the mean CPU time t_{CPU} is yielded additionally to the performance

Table III
INFLUENCE OF REGULARIZATION PARAMETERS IN THE NOISY CASE

λ_α	e	e_{ss}
1	55.61	18.69
10	46.04	11.71
100	29.22	4.62
150	28.92	5.14
200	28.92	5.37

λ_σ	e	e_{ss}
10^3	unstable	unstable
10^4	34.89	1.72
10^5	28.34	5.16
10^6	27.72	5.38
10^7	27.66	5.40

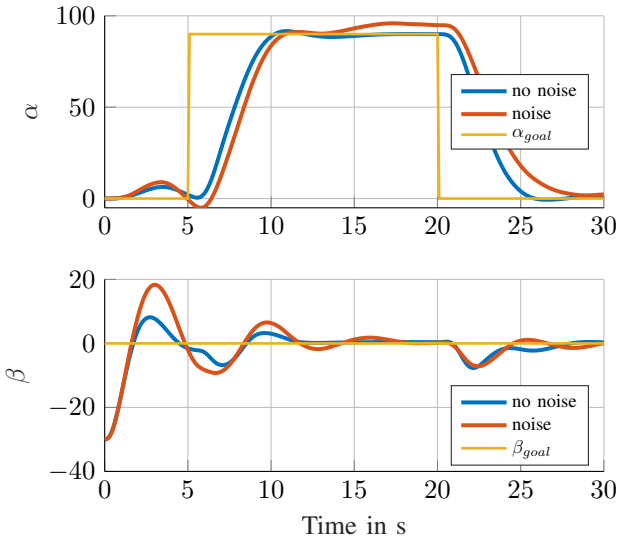


Figure 6. Comparison of simulation result with optimized parameters in noisy and noise free case.

measures. Note that this quantity depends on hardware and software and should not be taken as reference, but rather to get a quantitative feeling on how the computational effort evolves.

Table IV shows the simulation results for different horizons. The optimal behaviour is achieved for a data length of $N = 310$ and a prediction horizon in the range of $25 - 50$, where $L = 25$ leads to lower error but larger overshoots due to more aggressive behaviour.

Table IV
COMPARISON OF PERFORMANCE AND COMPLEXITY FOR DIFFERENT HORIZONS

L	$N = 200$			$N = 310$		
	e	e_{ss}	t_{CPU}	e	e_{ss}	t_{CPU}
10	unstab.	unstab.	0.008	unstab.	unstab.	0.026
25	unstab.	unstab.	0.012	18.95	0.43	0.033
50	25.16	15.85	0.021	21.36	0.41	0.059
75	-	-	-	20.99	11.25	0.112

C. Influence of the Data Trajectory

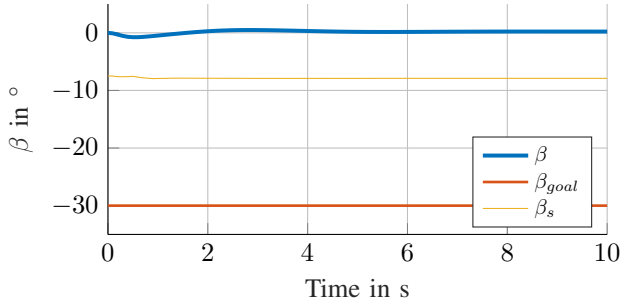
For a changing set of operating points, e.g. landing and flying, the data trajectory parametrizing the system might not cover a large enough range of state space in order to make proper predictions. On the other hand, a too large range might violate the affine dynamics assumptions. For this sake, different ways of scheduling the data in the hankel matrix will be investigated.

1) Fixed data: For an evaluation of the fixed data scheme for the travel task, refer to the last subsections, where the reference could be tracked very well once the parameters has been optimzed. However, the controller reaches its limit once landing is necessary, as the artificial setpoint does not follow the low altitude of the reference, yet the controlled helicopter, see figure 7. The home position of the helicopter does not seem to included in the set of reachable setpoints using this system parametrization. Hence, this data trajectory alone is not suited for an overall flight performance.

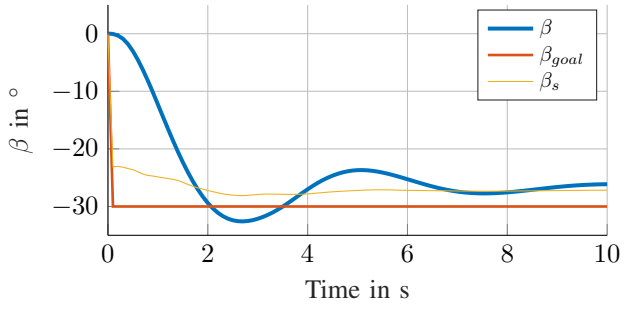
Using the second pre-generated trajectory, where the helicopter performed a low altitude flight, leads to a better behaviour once landing is required. Performing a travel task with the elevation data, however, is also not possible since the helicopter does not even reach the required high altitude to begin with.

A mix of travel and elevation in the data generation does also not lead to any good performance, as observed in simulation. An intuitive explanation, is that the assumption of affine system dynamics is then not valid anymore.

As a conclusion, using one single fixed trajectory is not suited for controlling the helicopter in all possible tasks.



(a) Travel trajectory



(b) Elevation trajectory

Figure 7. Performing a landing flight with different data trajectories.

2) Scheduling the data: In order to perform a landing flight, the elevation trajectory is much more suited for the system parametrization than the travel trajectory, which can be observed in figure 7. Thus, the data trajectory will be switched during closed loop, depending on the task to be performed. To do so, the helicopter will follow the same trajectory as in the previous subsections with additional landing to end the flight. The results can be seen in figure 8.

Observe that the helicopter is able to decrease height, however, the desired altitude is still not tracked perfectly.

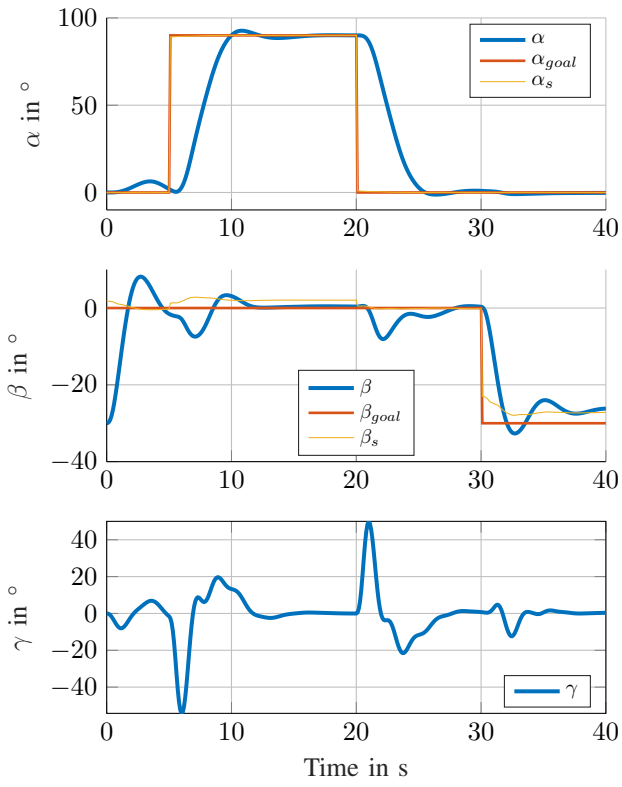


Figure 8. Simulation results for the data-driven MPC algorithm with time scheduled data.

Controlling the helicopter in low altitudes is remains a big difficulty, as it has also been with LQG control. Further steps of improvement are probably needed here. Another issue is also the spike in the computation time of the algorithm, see figure 9.

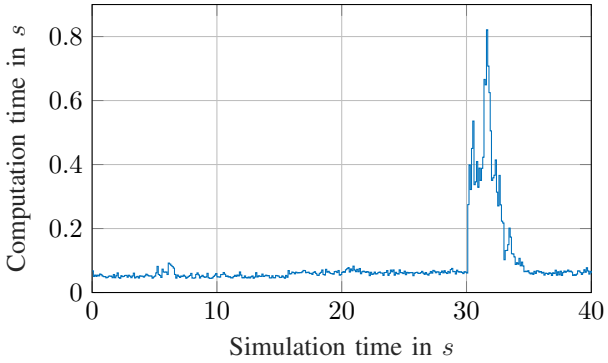


Figure 9. Comparison of different input/output weightings on the control performance.

3) Updating the data in closed loop: A method which could tackle the changing setpoint issue might be updating the data in closed loop. This assures the data trajectory always to be around the operating point. Of course, initialization data is needed, for which the travel trajectory will be used. The simulation results are shown in figure 10. The control scheme overall lead to stable behaviour and good performance for the travel task.

Nevertheless, observe that this scheme is still not able to cope with the landing setpoint. An explanation might be a too large data horizon, at which the part of the trajectory describing the zero equilibrium range might be too large. However, smaller horizons of data still lead to instability or bad performance, as discussed in the previous subsection.

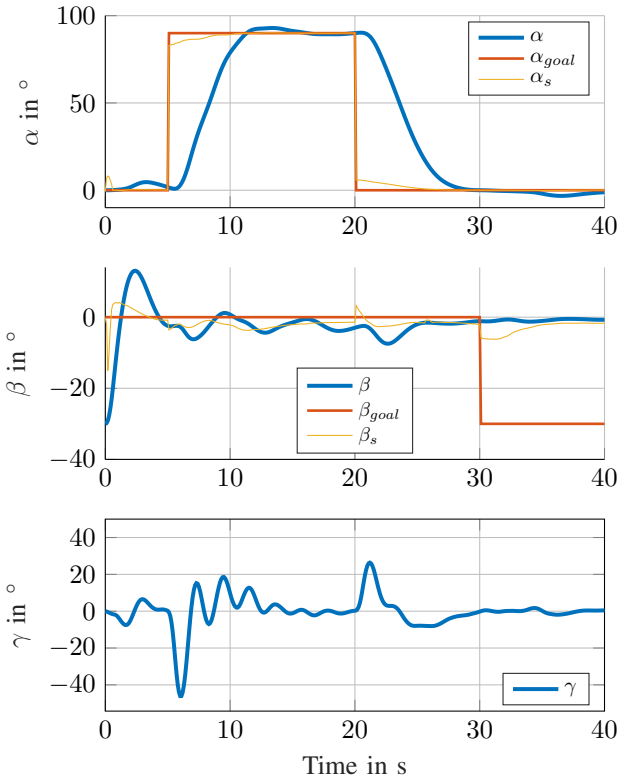


Figure 10. Simulation results for the data-driven MPC algorithm with closed loop data update.

Note, that the minimum singular value of the input Hankel matrix $\underline{\sigma}\left(H(u^d)\right)$, which is a measure for persistence of excitation, monotonically decreases when the helicopter reaches and stays in an equilibirum. In fact, after 100 s of remaining in the zero equilibrium, the controlled system became unstable, as the data loses the persistence of excitation property. This concludes that the scheme still needs adjustments, such as switching to another data indepenent controller, for example, once a region around the equilibrium is reached.

As a conclusion, the best suited data update scheme appears to be the scheduling fixed data trajectories in time. Thereby, the qualitiy of the elevation trajectory might still be improved.

VII. Setup of qpOASES

Since `quadprog` is not compatible for `C` Code generation in older `MATLAB` versions, the QP-Solver `qpOASES` will be used instead, anticipating the embedded real-time implementation on the microcontroller. The open source package admits a Simulink interface, allowing the solution of sequential quadratic programs which typically arise in MPC problems.

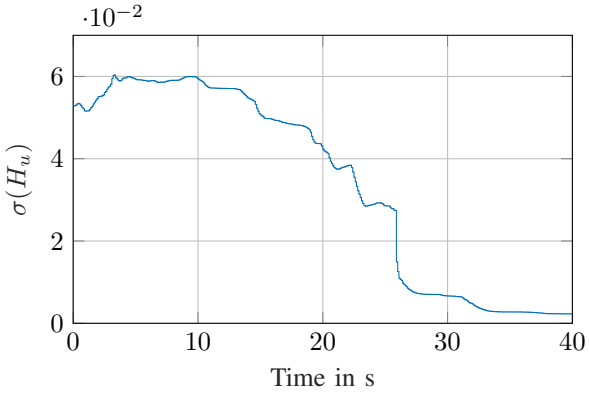


Figure 11. Evolution of the minimum singular value for closed loop data update.

A. Installation of qpOASES

The package can be downloaded from `https://github.com/coin-or/qpOASES`.

Execution of the `make` files is not needed, since only the `Simulink` interface will be used. An installation guide is given in the manual (6) section 6.2.

B. Usage of qpOASES

An example on how to use the `Simulink` interface is also given in the package directory `.../interfaces/simulink`. This directory is key for the use and should be included into the `MATLAB` path, since it contains the important `C++` files with the respective `make` file for compilation.

In order to use the `qpOASES` interface, the S-function has to be included into the `Simulink` block diagram. For the solution of sequential MPC QPs, the S-function corresponding to `qpOASES_SQProblem.cpp` will be used. The block contains the input ports for the matrices H, f, A, lb, ub, lbA, ubA in order to solve the quadratic program

$$\begin{aligned} \min_z \ & z^T H z + f^T z \\ \text{s.t. } & \text{lbA} \leq Az \leq \text{ubA} \\ & \text{lb} \leq z \leq \text{ub}. \end{aligned} \tag{14}$$

All matrices has to be passed as stacked row vectors. The outputs of this block are the solution z, the value of the cost fval, the exitflag and the number of iterations. For further explanations, see (6).

Note that problem (11) can be easily transformed into (14) with

$$\begin{aligned} \text{lbA} = \text{ubA} = b_{\text{eq}}, \quad & A = A_{\text{eq}} \\ \text{lb} = z_{\min}, \quad & \text{ub} = z_{\max} \end{aligned}$$

It should be emphasized, that the header of the `C++` file `qpOASES_SQProblem.cpp` has to be edited by the user for modifications. In line 57-60, the sampling time of the simulation, the number of yielded optimization variables, the maximum number of iterations and the type of the Hessian have to defined. For example, the code sequence

```
#define SAMPLINGTIME      -1
#define NCONTROLINPUTS    710
#define MAXITER           100
#define HESSIANTYPE       HST_SEMIDEF
```

specifies inherited simulation time, 710 decision variables and a positive semi-definite Hessian H.

After every modification (for example a change of optimization parameters due to changing horizon or condensing), the `make.m` has to be executed again for compilation.

C. Bounding the computation time

In order to assure the real-time ability of the control algorithm, an interface for bounding the computation time of the QP-solver will be implemented. As indicated by (6), another parameter `cputime` can be passed to the solver methods. Therefore, in the head of the file `qpOASES_SQProblem.cpp` another constant

```
#define MAXCPU     0.1
```

will be defined, where 0.1 indicates the chosen sampling time, which will be used on the microcontroller. In order to pass this constant to the algorithm, a `real_t *` pointer has to be created, as inferred by the documentation [6]. This is done in line 350

```
double maxAllowedCpuTime = MAXCPU;
cputime = (real_t*) &maxAllowedCpuTime;
```

and used in line 398, line 404 and in line 412 within the commands

```
init(H,g,A,lb,ub,lbA,ubA,nWSR,cputime);
...
hotstart(H,g,A,lb,ub,lbA,ubA,nWSR,cputime);
```

Violations of the maximum computation time are indicated by the error message
ERROR: Maximum number of working set recalculations performed
in the `MATLAB` command line.

VIII. Simulation results with qpOASES

A. Condensing the problem

`qpOASES` is not well suited for solving QPs with sparse matrices as in (13). In fact, trying to solve (14) analogous to section VI was only possible for a maximum CPU time above 1 s, which is much larger than the allowed sampling time. It can be assumed that neither a microcontroller will perform the calculations fast enough. In order to reduce the computation time, the optimization problem will be condensed by eliminating some variables. This effectively reduces the problems dimension at the cost of dense QP-matrices, which though can be tackled within the use of the Linear-Algebra-Package in `qpOASES`.

In order to reduce the problems dimension, the control input $\bar{u}$ and output $\bar{y}$ will be parametrized by α and σ, i.e.

$$\begin{aligned} \bar{u} &= H_{\tilde{L}}\left(u^d\right)\alpha \\ \bar{y} &= H_{\tilde{L}}\left(y^d\right)\alpha - \sigma. \end{aligned} \tag{15}$$

This in fact reduces the number of variables by $(m + p)\, L$, which corresponds to roughly 40% with the parameters values from section VI-B.

Take the abbreviations from (12) and define

$$H_\alpha := H_y^T \boldsymbol{Q} H_y + H_u^T \boldsymbol{R} H_u + \lambda_\alpha I. \tag{16}$$

Then, stacking up all remaining variables into

$$z = \begin{pmatrix} \alpha \\ u_s \\ y_s \\ \sigma \end{pmatrix}, \tag{17a}$$

leads to the cost function matrices

$$H = 2 \begin{pmatrix} H_\alpha & -H_u^T \boldsymbol{R_1} & -H_y^T \boldsymbol{Q_1} & -H_y^T \boldsymbol{Q} \\ -\boldsymbol{R_1}^T H_u & \tilde{L} R & 0 & 0 \\ -\boldsymbol{Q_1}^T H_y & 0 & \tilde{L} Q + S & \boldsymbol{Q_1}^T \\ -\boldsymbol{Q} H_y & 0 & \boldsymbol{Q_1} & \boldsymbol{Q} + \lambda_\sigma I \end{pmatrix} \tag{17b}$$

$$f^T = \begin{pmatrix} 0 & 0 & -2 y_{\text{goal}}^T S & 0 \end{pmatrix}, \tag{17c}$$

which are much more dense comparing to the block diagonal structure in (13).

The new contraint matrices result in

$$A = \begin{pmatrix} \mathbf{1}^T & 0 & 0 & 0 \\ H_{u,\text{init}} & 0 & 0 & 0 \\ H_{y,\text{init}} & 0 & 0 & \begin{pmatrix} -I & 0 \end{pmatrix} \\ H_{u,\text{TEC}} & -I \otimes \mathbf{1}_{n+1} & 0 & 0 \\ H_{y,\text{TEC}} & 0 & -I \otimes \mathbf{1}_{n+1} & \begin{pmatrix} 0 & -I \end{pmatrix} \\ H_u & 0 & 0 & 0 \end{pmatrix}, \tag{17d}$$

$$\text{lbA} = \begin{pmatrix} 1 \\ u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \\ 0 \\ 0 \\ u_{\min} \otimes \mathbf{1}_{\tilde{L}} \end{pmatrix}, \qquad \text{ubA} = \begin{pmatrix} 1 \\ u_{[t-n,t-1]} \\ y_{[t-n,t-1]} \\ 0 \\ 0 \\ u_{\max} \otimes \mathbf{1}_{\tilde{L}} \end{pmatrix}, \tag{17e}$$

where the subscript *init* in (17d) indicates the first n rows and *TEC* the last $n+1$ rows in that matrix.

B. Test within the simulation model

Solving the uncondensed problem with the new `qpOASES` solver lead to the exact same results, since the matrices of the QP don't change and every minimum of a QP is a global one due to convexity. However, solving the condensed problem (17) leads to different behaviour in the closed loop, see figure 12.

Changing the reference leads only to a marginally small change in the artificial setpoint y_s, even for higher weightings in S_y. Also, the landing behaviour is critical as the helicopter runs into the boundary (i.e. the ground) which is obviously undesired. However, note that no output constraints have been given to the algorithm due to runtime issues. Further investigations and modifications are obviously needed however, due to time issues those problems couldn't be tackled anymore.

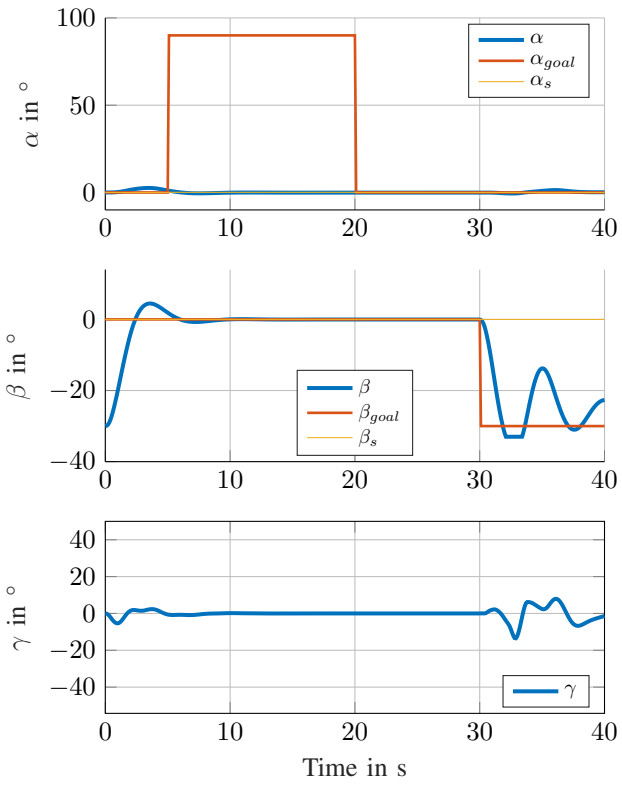


Figure 12. Simulation results for the data-driven MPC algorithm with closed loop data update.

IX. Conclusion

To summarize, it has been shown that stabilization and tracking of the nonlinear 3-DoF Helicopter with Data-driven Model Predictive Control is possible. However, improvements for different setpoints, such as the landing setpoint, are still needed in terms of trajectory quality or in the control scheme with its respective parameters. Also, the condensed algorithm still has to show good performance results in order to implement the `qpOASES` solver on the microcontroller. Further steps are data generation from the real plant the direct real-time implementation of the controller on the hardware and respective tests.

References

[1] "Solution to the practical work in the lecture *Concepts of Control Theory* of the Institute for Systems Theory and Automatic Control at the University of Stuttgart."

[2] J. Berberich, J. Koehler, M. A. Mueller, and F. Allgoewer, "Data-driven tracking MPC for changing setpoints," *IFAC-PapersOnLine*, 2020, 21st IFAC World Congress.

[3] ——, "Linear tracking MPC for nonlinear systems—part II: The data-driven case," *IEEE Transactions on Automatic Control*, sep 2022.

[4] J. Willems, P. Rapisarda, I. Markovsky, and B. De Moor, "A note on persistency of excitation," *Systems and Control Letters*, vol. 54(4), 04 2005.

[5] F. Wegner, "Data-enabled predictive control of robotic systems," Master's thesis, ETH Zuerich, 2021.

[6] Hans Joachim Ferreau et al., "qpOASES User's Manual," ABB Corporate Research, Switzerland, support@qpOASES.org, Apr. 2017.