Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 5757–5763

Fast Hildreth-based Model Predictive Control of Roll Angle for a Fixed-Wing UAV

V.T.T. Lam * **A. Sattar** ** **L. Wang** *** **M. Lazar** *

* *Department of Electrical Engineering, Control Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands, (email: v.t.t.lam@student.tue.nl)*

** *School of Engineering Cluster, Electrical and Computer Engineering, RMIT University, Melbourne, Australia*

*** *School of Engineering Cluster, Electrical and Biomedical Engineering, RMIT University, Melbourne, Australia*

Abstract: In this paper we consider model predictive control (MPC) design for roll angle control for a fixed-wing unmanned aerial vehicle (UAV) with multiple segmented control surfaces. The challenge of roll angle control for a fixed-wing UAV consists of switching between inner and outer aileron pairs with hard constraints due to safety, energy saving and switching actuators. The novelty consists of formulating a hybrid control problem as a switched linear constrained MPC-QP problem and switched state observer design for fixed-wing UAV. A fast novel QP-solver based on the active-set QP-solver Hildreth is developed to meet the real-time implementation sampling time of $T_s = 10$ ms. The designed MPC controllers are simulated using MATLAB. Simulations and the CPU-time from the improved QP-solvers show MPC to be a very good solution for real-time roll angle control of fixed-wing UAVs.

Keywords: Model Predictive Control, Quadratic Programming, Fixed-Wing Unmanned Aerial Vehicle, Active Set Methods

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) or drones have become very popular in the past years (Marris, 2013). UAVs are becoming more popular for various reasons. First, they are flexible and suitable for a broad range of applications. Second, they are of low cost. Finally, less risk for humans is involved compared to manned aerial vehicles (MAVs). Due to their versatility, they are used in civil (Shakhatreh et al., 2019), geomorphic mapping and imaging (Hackney and Clayton, 2015) and military applications (Carapau et al., 2017). These applications demand fast and large manouvring coming from the pilot commands, which require fast and optimal control.

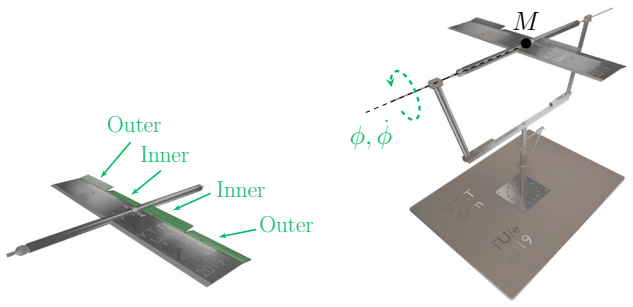
There are many types of UAVs and they come in different shapes, but the two main categories are fixed-wing and multi-rotor UAVs (Boon et al., 2017), (Thamm et al., 2015), and each type has its own benefits and disadvantages. For instance, the multi-rotor UAVs do not need runways for take-off and landing and are easier to manoeuvre and control, while it is a challenge to land fixed-wing UAVs such that hardware damage is minimized. However, the fixed-wing configuration has better flight-time endurance and can map larger coverage and therefore can glide in the air even when engines are switched off, which is not the case for the multi-rotor configuration. For these reasons, control of roll angle for fixed-wing UAVs is considered in this paper.

In the past, automatic tuning of proportional integral derivative (PID) controllers for fixed-wing UAVs (Poksawat et al., 2017) and multi-rotor hexacopter UAVs has been proposed. Furthermore, rate-based MPC for multi-rotor hexacopter UAVs with fault-tolerance has been proposed (Ligthart et al., 2017).

Model predictive control (MPC) is adopted in this paper, because it has several advantages. First, it can anticipate for future reference changes. Second, safety or actuator constraints can be handled a priori. Finally, non-minimum phase systems require predictive control. These three aspects are what PID is lacking (Aström and Hägglund, 2006). The main challenge of MPC is, however, to solve a constrained optimization problem in real-time, very fast (samping time $T_s = 10$ ms).

In this work, we are dealing with a fixed-wing UAV with multiple segmented control surfaces (MSCS). This means that the fixed-wing UAV has inner and outer aileron pairs that are influencing the roll rate and thus the roll angle. Each of the aileron pairs have hard actuator constraints and in order to save energy consumption, MPC is a suitable control strategy. Since *either* the inner aileron pair *or* the outer aileron pair is allowed to be active every time instance, we could formulate the roll angle control problem as a mixed integer quadratic program (MIQP) (Stellato et al., 2018), where usually requires solving more than one quadratic programs (QP). However, due to the challenging $T_s = 10$ ms sampling time constraint, we

10.1016/j.ifacol.2020.12.1608



(a) Considered UAV with two aileron pairs.

(b) The roll-rig and considered roll axis.

Fig. 1. 3D depiction of a fixed-wing UAV with MSCS.

propose a simpler solution, which is to switch between two linear QP MPC algorithms with linear constraints and between two state observers, each corresponding to the identified models of the aileron pairs. State observers estimating the state vector $\hat{x}$ are necessary because only the output y can be measured. This switching strategy has several benefits and is similar to a gain-scheduling approach. First, the controller can switch between mild and aggressive control actions as the outer aileron pairs have a larger gain in comparison with the inner aileron pair. Second, the difference in gain requires the outer aileron pair to deflect less than the inner aileron pair thus saving energy consumption. The reasoning of different gains is similar to switching between different gears in a cars gearbox.

The contribution of this paper consists of: *(i)* formulating the roll angle control problem for a fixed-wing UAV as a switched linear MPC problem with switched state observer design and *(ii)* developing an improved fast QP-solver for linear MPC based on the active-set Hildreth QP-solver. There are many other fast QP-MPC solvers, such as active-set QP-solver qpOASES (Ferreau, 2011), (Ferreau, 2006) and OSQP (Stellato et al., 2019), (Ferreau et al., 2014). However, we will focus on Hildreth's algorithm because it allows a simple implementation in C-code and it involves simple operations that can be executed efficiently on inexpensive microcontrollers. Since currently we are considering box inequality constraints only, fast gradient MPC solvers, such as `qpdunes` (Frasch et al., 2013), (Frasch et al., 2014) could also be applied. However, we focus on active set QP-solvers because we would like to include affine inequality constraints in the future. The main challenge is to solve the QP-MPC problem for roll angle control in less than $T_s = 10$ms.

2. IDENTIFIED MODELS USED AS PLANT

A fixed-wing aircraft has six degrees of freedom (DOF), which are positions x, y, z and orientations axes pitch θ, roll ϕ and yaw ψ angles. Their respective angular velocities are $\dot{\theta}$, $\dot{\phi}$ and $\dot{\psi}$. These three orientation axes are controlled by three actuators, i.e. elevators, ailerons and rudder, respectively. However, in this paper, we consider a fixed-wing UAV with MSCS together with a roll rig, where the three positions are fixed and every orientation axis is fixed except for the roll axis. The plant 3D model is depicted in Fig. 1. Only the ailerons are considered as control input. In fact, the ailerons consist of *inner* and *outer* aileron pairs. We first consider the roll axis only, because the fixed-wing UAV has only inner and outer aileron pairs. Also, controlling the roll axis (roll angle) is important for stability and it plays an important role in manoeuvring the fixed-wing UAV.

System identification by means of relay experiments (Poksawat, 2018) has been carried out to estimate a mathematical model for the UAV, since it is more practical than using first principles modeling. Suppose the inner and outer aileron pairs are denoted by control inputs u_1 and u_2, respectively. Then let the roll rates as result of deflection in inner and outer ailerons be denoted by $s\phi_1$ and $s\phi_2$ respectively, which correspond to derivative for zero initial conditions. Here, s denotes the Laplace variable. The continuous-time transfer functions from aileron deflection u_j to roll rate $s\phi_j$ have been identified as

$$G_1(s) = \frac{-74.15s + 8892}{s^3 + 59.11s^2 + 1599s + 7936}$$

$$G_2(s) = \frac{4.746s^3 - 392.5s^2 + 2.443 \cdot 10^4 s + 2.064 \cdot 10^5}{s^4 + 80.26s^3 + 3026s^2 + 2.829 \cdot 10^4 s + 1.21 \cdot 10^5}$$

with $j \in \{1, 2\}$ denoting the inner and outer aileron pairs, respectively. However, we are interested in controlling the *roll angle* ϕ_j. Integrating G_1 and G_2 in s-domain gives the two SISO plant models as

$$\phi = y = \frac{G_j(s)}{s} u_j \quad \text{if mode } j \text{ is active.} \tag{1}$$

This is a hybrid system that switches between two SISO plant models, i.e. G_1 is active when $u_1 \neq 0, u_2 = 0$ and G_2 is active when $u_1 = 0, u_2 \neq 0$ and u_1, u_2 can never both be non-zero.

Next, we convert each of the transfer functions in Eq. (1) to a continuous-time state-space model suitable for MPC. Realizing that controller hardware is dealing with discrete-time digital signals, we then convert the continuous-time state-space model into a discrete-time one, with sampling time $T_s = 10$ ms, using MATLAB's `c2d`:

$$\Sigma^j_{\text{disc}} : \begin{cases} x_j(k+1) & = A_j x_j(k) + B_j u_j(k) \\ y(k) = \phi(k) & = C_j x_j(k), \end{cases} \tag{2}$$

where k is the discrete-time index and we assume no feedthrough term of input u_j in output y_j. In Eq. (2), it should hold that (A_j, B_j) is controllable and (A_j, C_j) is observable, which is the case. The switching condition is defined by:

$$\left.\begin{array}{l} \text{Outer ailerons} \\ \text{active, i.e.} \\ u_1 = 0, u_2 \neq 0, \end{array}\right\} \Leftarrow \left\{\begin{array}{c} \text{if } \rho(k) < \text{lower} - \epsilon \\ \text{or} \\ \text{upper} + \epsilon < \rho(k) \end{array}\right.$$
$$\left.\begin{array}{l} \text{Inner ailerons} \\ \text{active, i.e.} \\ u_1 \neq 0, u_2 = 0, \end{array}\right\} \Leftarrow \left\{\begin{array}{c} \text{if lower} + \epsilon < \rho(k) \\ \text{and} \\ \rho(k) < \text{upper} - \epsilon \end{array}\right. \tag{3}$$

Above, inner ailerons are active when the value of roll rate $\rho(k)$ is between upper and lower bounds and outer ailerons are active otherwise. A margin ϵ is implemented, which is equivalent to dwell time to prevent frequent switching that may lead to undesired oscillations.

3. MPC DESIGN

The control strategy adopted in this paper is constrained MPC by means of switching between two linear MPC-QP based algorithms and two state observers with roll

rate $\rho(k)$ value as switching condition. To define the MPC problem, let Y_k denote the vector of future outputs $y_{i|k}$ with $i = 1, \dots, N$, and let U_k denote the vector of future inputs $u_{i|k}$ with $i = 0, \dots, N-1$. Let $x(k) = x_{0|k}$ denote the current state. By expressing every element of Y_k in terms of $x(k)$ and U_k, we can predict future outputs as

$$Y_k = \Phi_j x_j(k) + \Gamma_j U_k \tag{4}$$

where

$$\Phi_j = \begin{bmatrix} C_j A_j \\ \vdots \\ C_j A_j^N \end{bmatrix}, \Gamma_j = \begin{bmatrix} C_j B_j & \cdots & 0 \\ \vdots & \ddots & \vdots \\ C_j A_j^{N-1} B_j & \cdots & C_j B_j \end{bmatrix},$$

$$Y_k = \begin{bmatrix} y_{1|k}^T & \cdots & y_{N|k}^T \end{bmatrix}^T, U_k = \begin{bmatrix} u_{0|k}^T & \cdots & u_{N-1|k}^T \end{bmatrix}^T.$$

Here, N is the *prediction horizon* which determines how many samples we look ahead in the future.

3.1 MPC-QP formulation

Let future reference vector $\mathcal{R}_k$ consist of $r_{i|k}$ with $i = 1, \dots, N$. Some weight matrices Q_j and R_j penalize the error between future output and future reference $e_{i|k} = y_{i|k} - r_{i|k}$ with $i = 0, \dots, N$ and control input $u_{i|k}$ respectively. These are designed by the user to meet the design criteria. Terminal cost matrix P_j is chosen by the user, typically $P_j \succeq Q_j$, to enhance tracking performance. Then we can specify an MPC cost function quadratic in the decision variable U_k as follows

$$\begin{aligned} &J_j\left(x_j(k), U_k\right) \\ &= e_{N|k}^T P_j e_{N|k} + \sum_{i=0}^{N-1} \left(e_{i|k}^T Q_j e_{i|k} + u_{i|k}^T R_j u_{i|k} \right) \\ &= e_{0|k}^T Q_j e_{0|k} + E_k^T \Omega_j E_k + U_k^T \Psi_j U_k \end{aligned}$$

with

$$\Omega_j = \operatorname{diag}\left\{Q_j, \cdots, Q_j, P_j\right\}, \quad \Psi_j = \operatorname{diag}\left\{R_j, \cdots, R_j\right\}$$

$$E_k = \begin{bmatrix} e_{1|k}^T & \cdots & e_{N|k}^T \end{bmatrix}^T.$$

Substitution of Eq. (4) and removing the constant term gives us

$$\tilde{J}_j\left(x_j(k), U_k\right) = \frac{1}{2} U_k^T G_j U_k + U_k^T F_j \left(\Phi_j x_j(k) - \mathcal{R}_k\right) \tag{5}$$

where

$$G_j = 2\left(\Psi_j + \Gamma_j^T \Omega_j \Gamma_j\right) \succ 0, \qquad \text{if } \Psi_j \succ 0, \Omega_j \succeq 0,$$

$$F_j = 2\Gamma_j^T \Omega_j.$$

Next, we formulate the *hard actuator constraints* on the future aileron pair deflections $u_{i|k}$ and *hard safety constraints* on future roll angles $y_{i|k}$ for all i as:

$$u_{\min} \le u_{i|k} \le u_{\max}, \qquad y_{\min} \le y_{i|k} \le y_{\max}.$$

We can write these constraints into the form

$$M_i y_{i|k} + Z_i u_{i|k} \le b_i, \quad M_N y_{N|k} \le b_N$$

with $i = 0, \dots N-1$ and

$$M_N = \left[-I_q, I_q\right]^T, b_N = \left[-y_{\min}, y_{\max}\right]^T,$$

$$M_i = \left[0, 0, -I_q, I_q\right]^T, Z_i = \left[-I_m, I_m, 0, 0\right]^T,$$

$$b_i = \left[-u_{\min}, u_{\max}, -y_{\min}, y_{\max}\right]^T,$$

where m and q are dimensions of $u(k)$ and $y(k)$, respectively. If we recall $y(k) = C_j x_j(k)$, then we can rewrite constraints in compact form as

$$\mathcal{D}_j x_j(k) + \mathcal{M} Y_k + \mathcal{E} U_k \le c$$

where

$$\mathcal{D}_j = \begin{bmatrix} (M_0 C_j)^T & \cdots & 0 \end{bmatrix}^T, c = \begin{bmatrix} b_0^T & \cdots & b_N^T \end{bmatrix}^T,$$

$$\mathcal{M} = \begin{bmatrix} 0 & \cdots & 0 \\ M_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M_N \end{bmatrix}, \mathcal{E} = \begin{bmatrix} Z_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Z_{N-1} \\ 0 & \cdots & 0 \end{bmatrix}$$

Substitution of Eq. (4) gives linear constraints

$$L_j U_k \le c + W_j x_j(k) \tag{6}$$

with

$$L_j = \mathcal{M}\Gamma_j + \mathcal{E}, \quad W_j = -\mathcal{D}_j - \mathcal{M}\Phi_j.$$

Assuming that $G_j \succ 0$, then $\tilde{J}$ is convex, yields a convex quadratic program that needs to be solved to compute the MPC control action. Solving the QP would yield a minimizer U_k^* consisting of optimal future control inputs. Selecting its first element gives the next control input to the plant as

$$u(k) = u_{0|k}^* = \begin{bmatrix} I_m & 0 & \cdots & 0 \end{bmatrix} U_k^*.$$

The receding horizon principle is then applied: for each discrete-time instant we measure $y(k)$ and estimate $x(k)$ with an observer, compute U_k^* by solving the MPC-QP, and then extract $u(k)$ as the first element of U_k^*.

To estimate state vector $x_j(k)$, let $\hat{x}_j$ denote the estimated state and define the observer dynamics as

$$\begin{aligned} \hat{y}_j(k) &= C_j \hat{x}_j(k) \\ \hat{x}_j(k+1) &= A_j \hat{x}_j(k) + B_j u_j(k) + L_j\left(y(k) - \hat{y}_j(k)\right) \end{aligned}$$

where we assume no feedthrough term of input u_j in estimated output $\hat{y}_j$. If we define the estimation error as $e_j(k) = x_j(k) - \hat{x}_j(k)$ we can write the error dynamics as

$$e_j(k+1) = \left(A_j - L_j C_j\right) e_j(k).$$

Therefore, we need to design observer gains L_j such that $\left|\lambda\left(A_j - L_j C_j\right)\right| < 1$ holds for all eigenvalues. Then it is guaranteed that $e_j(k) \to 0$ and $\hat{x}_j(k) \to x_j(k)$ as $k \to \infty$ (in simulation we observe that $\hat{x}$ converges to x within 41 samples or 0.41s). This can be done using MATLAB's `L_j = place(A_j',C_j',p)`, where `p` are the desired observer poles inside the unit circle.

The switched linear MPC-QP algorithm can be summarized as follows:

1. Meausure roll angle output $y(k) = \phi(k)$ from the plant (with IMU sensor).
2. Measure roll rate $\rho(k)$ from the plant (with IMU sensor) to decide the switching mode.
3. Based on roll rate $\rho(k)$, use observer j to estimate $\hat{x}_j(k)$ and solve MPC-QP j to obtain U_k^*.
4. Extract $u_j(k)$ and apply to the plant.

Next, we will present an algorithm for solving MPC-QP problems.

3.2 Hildreth+

Let a general QP problem be denoted as

$$\min_\theta J(\theta) = \frac{1}{2}\theta^T E \theta + \theta^T F, \text{ s.t. } M\theta \le \gamma$$

where θ is the decision variable, M, E are static and $F = F_j\left(\Phi_j x_j(k) - \mathcal{R}_k\right)$ and $\gamma = c + W_j x_j(k)$ are dynamic.

The original Hildreth algorithm, see, e.g., (Wang, 2009), is recalled in Alg. 1.

Algorithm 1. Hildreth algorithm
Input: E, F, M, γ, $\bar{m}$, δ
Output: θ
Step 1: Unconstrained solution $\theta = -E^{-1}F$
Step 2: Check constraint violation $M\theta \leq \gamma$
if *no violation* **then**
  stop
else
  Step 3: Compute $H = ME^{-1}M^T$, $K = \gamma + ME^{-1}F$
end
 Set $\lambda^m = 0$
Step 4:
 for $m = 1 : \bar{m}$ **do**

$$w_i^m = -\frac{1}{h_{ii}}\left[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^m + \sum_{j=i+1}^{n} h_{ij}\lambda_j^{m-1}\right] \quad (7)$$

$$\lambda_i^m = \max\left(0, w_i^m\right) \quad (8)$$

  if $\left|\left|\lambda^m - \lambda^{m-1}\right|\right| < \delta$ **then**
    stop
  else
    continue
  end
end
Step 5: Return $\theta = -E^{-1}F - E^{-1}M^T\lambda^{m+1}$

Above $\bar{m}$ denotes the maximum number of iterations and δ denotes a small number. Let the modified version of Hildreth be denoted by Hildreth+, which consists of the following changes:
Step 2: check constraint violation in a vectorized fashion in the form of $K = \gamma - M\theta \geq 0$, instead of using a for loop and scalar operations.
Step 3: warm-start by initializing λ based on the solution of the previous sample. Here, we use a `persistent` variable for λ in MATLAB, so it can be deleted as input and output arguments of the QPsolver-function. However, when disturbances come into play, cold-start (i.e. initialize $\lambda = 0$ as in Step 3) would be wiser because the active set could be different from the previous sample.
Step 4: improving rate of convergence (ROC) and reducing number of iterations using successive over-relaxation (SOR) (Mittal, 2014). Inspired by the Jacobi and Gauss-Seidel (GS) method for an iterative solution (Roberts, 2010) to $H\lambda = -K, \lambda \geq 0$, we obtained a parameter $\omega = 1.08 > 1$ by trial and error for optimal ROC. The ω appears in the equations as follows and SOR is a weighted average of previous iterate λ_i^{m-1} and GS version of original Hildreth iterate

$$w_i^m = (1-\omega)\,\lambda_i^{m-1} + \omega \cdot \underbrace{\frac{1}{D_{ii}}\left[-k_i - (L_i + U_i)\,\lambda^{m-1}\right]}_{\text{original}}$$

$$\lambda_i^m = \max\left(0, w_i^m\right)$$

If $\omega = 1$, then we obtain the original equations from Eq. (7) and Eq. (8), but in a more compact form, realising structures of L and U. The reason we cannot use $\lambda = -H^{-1}K$ directly is because H may not always be invertible (singular) if active constraints are linearly dependent. Hence vectorization is not possible and for-looping through scalar elements is necessary to avoid ill-conditioned problems. The matrix $H = U + D + L$ is decomposed into upper-U, lower-triangular L and diagonal matrices D offline.
Step 4: precompute H, E^{-1} offline as they are static. K is dynamic as it depends on $\mathcal{R}, \hat{x}$ and can be reused from Step 2 in the form of $K = \gamma - M\theta$, with $\theta = -E^{-1}F$ in Step 1, which is faster than using `theta=E\F`.
Step 4: for the termination condition, we check whether $\left(\lambda^{m+1} - \lambda^m\right)^T\left(\lambda^{m+1} - \lambda^m\right) < \delta$. This is changed into $e = \lambda^{m+1} - \lambda^m$, $e^T e < \delta$. We set $\delta = 1{\cdot}10^{-13}$ and maximum number of iterations to $\bar{m} = 1 \cdot 10^6$ for all Hildreth-solvers.
Step 5: extract non-zero elements (larger than $1 \cdot 10^{-5}$) of λ^{m+1} to be $\tilde{\lambda}$ as well as corresponding rows of M to be $\tilde{M}$ to compute the correction in a vectorized fashion. Because of decreasing dimensions in $\tilde{\lambda}, \tilde{M}$, this should reduce CPU time.
Step 5: implement correction to unconstrained solution as $\theta = \theta - E^{-1}\tilde{M}^T\tilde{\lambda}^{m+1}$, where we reuse the unconstrained solution θ from step 1. Using parenthesis for priority from right to left (`theta=theta-invE*(M'*lambda)`) reduces CPU-time.

Let a further modified version of Hildreth+ be denoted by Hildreth+'. Here we introduce the concept of *reduced mini-Hildreth*, which includes the following changes:
Step 4: we extract the indices of violated constraints from step 2 in the form of $K = \gamma - M\theta < \sigma$. Here, $\sigma = 0.2 > 0$ is a tunable margin because barely unviolated constraints may be active. If $\sigma = 0$ then the bound would be too strict. Then we take rows and columns of H and K corresponding to these indices and compute reduced $\tilde{H}, \tilde{K}$. Then we solve a *reduced mini-Hildreth* algorithm iteratively as in Step 4 using $\tilde{H}, \tilde{K}$ instead and repeat until reduced $\tilde{\lambda}$ converges. Because of decreasing dimensions in $\tilde{H}, \tilde{K}$ and reduction of ROC and number of iterations, this could save CPU time. Directly computing $\tilde{\lambda} = -\tilde{H}^{-1}\tilde{K}$ is not possible if H is not invertible (singular), which could happen if active constraints are linearly dependent. Hence vectorization is not possible and for-looping through scalar elements is necessary to avoid ill-conditioned problems. Logically, CPU-time should decrease as the number of violated and active constraints decrease as well.
Step 5: then we extract rows of M to be $\tilde{M}$ corresponding to the indices of violated constraints. This should reduce CPU time as $\tilde{M}, \tilde{\lambda}$ are of smaller dimensions and vectorized operations are employed.

4. SIMULATION EXPERIMENTS

4.1 MPC performance

The simulations in MATLAB R2018b were performed on Microsoft Surface 3 Pro (2013) with Windows 8 64-bit, Intel(R) Core(TM) i7-4650U CPU 1.70GHz 2.30GHz (100% CPU speed), 8 GB RAM. In simulation, plant output roll angle ϕ is subject to control and plant input is the inner or outer aileron pair. For plant simulation, the original transfer function is discretized and used directly, while for prediction matrices the corresponding discrete-time state-space representations are used. Here, $j \in \{1, 2\}$ denotes the active inner aileron pair and the active outer aileron

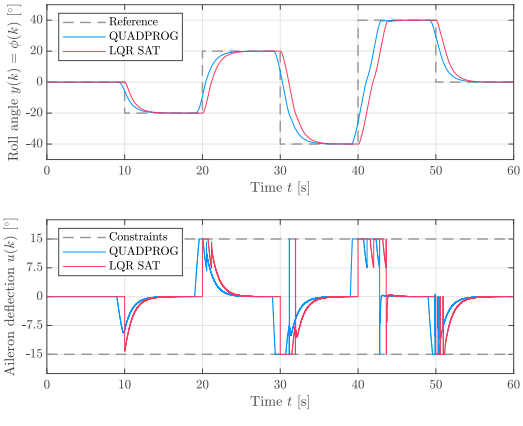
Table 1. MPC parameters

MPC parameter	Symbol	Value	Unit
Input constraints	$[u_{\min}, u_{\max}]$	$[-15, 15]$	°
Output constraints	$[y_{\min}, y_{\max}]$	$[-270, 270]$	°
Prediction horizon	N	100	–
Stage tracking error penalty cost	Q_j	10	–
Input penalty cost	R_j	10	–
Terminal tracking error penalty	P_1	$1 \cdot 10^6$	–
cost of system 1 (inner) and 2 (outer)	P_2	$1 \cdot 10^9$	–
Lower, upper bounds in Eq. (3)	[lower, upper]	$[-16, 16]$	°/s
Sampling time	T_s	10	ms

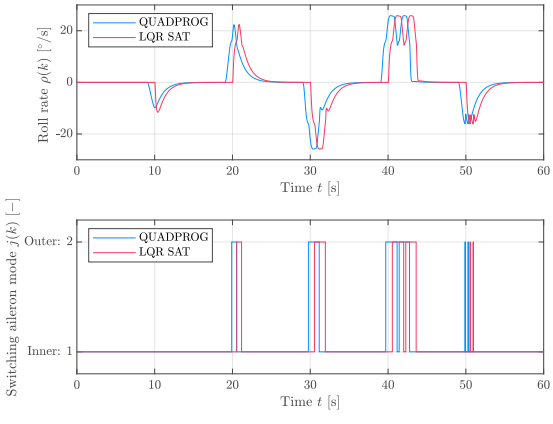
pair, respectively. In the SISO plant models, we assume roll angle to be the output, while in reality the UAV has an IMU sensor which measures both roll angle $\phi(k)$ and roll rate $\rho(k)$. The constraints on aileron deflection (hard actuator constraint) and roll angle (soft safety constraint) and other MPC parameters are summarized in Table 1.

The reference to track is a piecewise constant trajectory as 0°, −20°, 20°, −40°, 40°, 0°, each constant part lasting for 10 seconds. In total, the simulation lasts for 60 s. With sampling time $T_s = 10$ ms this is equivalent to $6 \cdot 10^3$ samples.

Fig. 2 presents results for MATLAB's `quadprog` and a linear quadratic regulator (LQR) controller with saturation, where the latter is scripted as `m`-function and converted into MEX-file with C/C++ code to exclude compiling time. It can be observed that both control strategies achieve offset-free tracking with similar tracking performance, without overshoot and same switching sequence. However, MPC anticipates future reference changes, starts earlier with switching and control input is smoother prior each jump in reference change, compared to the LQR with saturation.



(a) Tracking output $y(k) = \phi(k)$ to reference $r(k)$ and control input $u(k)$



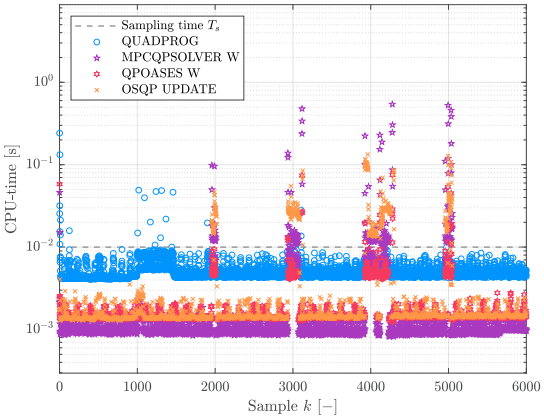
(b) Roll rate $\rho(k)$ and switching mode $j(k) \in \{1, 2\}$

Fig. 2. MPC controller performance.

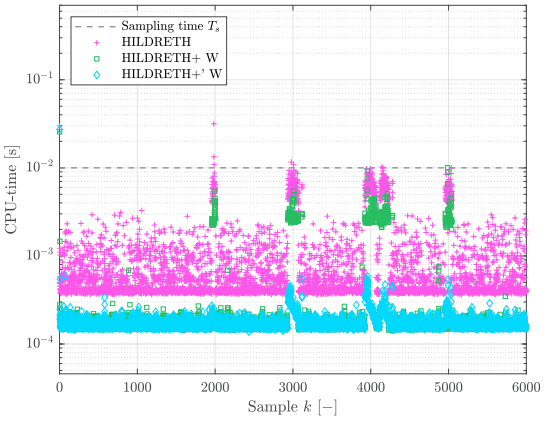
4.2 CPU-time performance

The Hildreth-based QP-solvers were scripted as `m`-function and converted into MEX-file with C/C++ code to exclude compiling time using MATLAB Coder's `codegen`. `mpcqpsolver`, `qpOASES` and `OSQP` are MEX-files too, where the former solver is MATLAB's fastest default QP-solver. Although `quadprog` is the slowest, we regard its solution as optimal. Its black box code is protected, hence, it was not converted into MEX-files.

In Fig. 3, CPU-time (excluding observer-time) among the QP-solvers are compared. 'Warm-start' is denoted as 'W'. First, we can observe that computing the unconstrained solution (Steps 1 and 2 of Hildreth) is faster than computing the constrained solution, where λ needs to be updated (Hildreth Step 4). Hildreth+ and Hildreth+' perform equally fast when constraints are inactive, but Hildreth+'



(a) CPU-time other QP-solvers



(b) CPU-time Hildreth-based QP-solvers

Fig. 3. CPU performance and QP-solver accuracy.

Table 2. Average and worst CPU-time of QP-solvers, ordered from fastest to slowest on average CPU-time. Accuracy QP-solvers solution $u(k)$ w.r.t. `quadprog` in final two columns. ✓= yes, ✗* = slightly no, ✗= no

QP-solver	Average CPU-time [s]	Maximum CPU-time [s]	Max CPU $< T_s$ (10 ms)?	Average absolute val. error $u(k)$ [°] w.r.t. quadprog	Accurate w.r.t. quadprog?
HILDR+' W	$1.7408 \cdot 10^{-4}$	$5.9128 \cdot 10^{-4}$	✓	$2.6603 \cdot 10^{-11}$	✓
HILDR+ W	$4.0940 \cdot 10^{-4}$	$1.01 \cdot 10^{-2}$	✗*	$2.5219 \cdot 10^{-11}$	✓
HILDR	$1.1 \cdot 10^{-3}$	$3.16 \cdot 10^{-2}$	✗	$4.9292 \cdot 10^{-11}$	✓
QPOASES W	$1.9 \cdot 10^{-3}$	$7.53 \cdot 10^{-2}$	✗	$1.5410 \cdot 10^{-11}$	✓
MPCQPSLV W	$2.9 \cdot 10^{-3}$	$5.417 \cdot 10^{-1}$	✗	$1.4642 \cdot 10^{-11}$	✓
OSQP UPD	$4.4 \cdot 10^{-3}$	$1.330 \cdot 10^{-1}$	✗	$2.630 \cdot 10^{-1}$	✗
QUADPR	$5.1 \cdot 10^{-3}$	$4.92 \cdot 10^{-2}$	✗	–	–

is faster when constraints are active. Second, only the worst-case CPU-time of Hildreth+' stays below sampling period $T_s = 10$ ms, while other QP-solvers violate this. Third, CPU-time of original Hildreth seems to have a bigger variance, i.e. is more spreaded out than Hildreth+ and Hildreth+', which are faster than the original Hildreth algorithm.

Table 2 summarizes Fig. 3 concisely and provides an additional accuracy comparison. 'Warm-start' is denoted as 'W'. Here, we consider only samples 33 - 6000 (0.55% - 100%) to omit transient peaks at the start, which is fair as real-world experiments has zero reference for starting up. Overall, CPU-time of Hildreth+ and Hildreth+' are reduced compared to original. Hildreth+' has the best average, minimum and worst-case CPU-time, where the latter always stays below the sampling period of $T_s = 10$ ms. The absolute value of error w.r.t. `quadprog` is about 10 times the tolerance of termination criterion. From the table, we can also see that there is a trade-off between accuracy and CPU-speed. Finally, it can be observed that `OSQP` is not very accurate compared to the other QP-solvers. We followed the manual from `OSQP` (Stellato et al., 2019) and checked the syntax is matching and used the default settings, such as tolerances. It may be possible to tune some specific solver-parameters for better accuracy, however this is out the scope of this paper. Despite the lack of accuracy, the tracking performance was still good, anticipating for future reference changes and offset-free, however with some overshoot. Hildreth+ and Hildreth+' are both more accurate than the original. All Hildreth-based solvers and most other solvers have similar error w.r.t. `quadprog` in the same order of magnitude of 10^{-11}.

It is worth to mention that the developed Hildreth-based solvers were also tested on the chain of masses benchmark example of (Kögel et al., 2012). For that example, Hildreth+' was still the fastest solver and qpOASES was the second fastest solver (and similar accuracy of same magnitude). For that example, it could be that Hildreth+ and original Hildreth were slower than qpOASES because of the disturbance present and relative low number of decision variables of 40 ($N = 10$ and 4 control inputs). In this paper, Hildreth+ and the original Hildreth were faster than qpOASES, possible because the number of decision variables is much higher, i.e., 100 (corresponding to $N = 100$ and one control input). Parameters for the improved Hildreth-based QP-solvers were $\bar{m} = 1 \cdot 10^6$, $\delta = 1 \cdot 10^{-13}$, $\sigma = 0.2$ and $\omega = 1.2$.

5. CONCLUSIONS

This paper presented a fast Hildreth-based MPC design for roll angle control of a fixed-wing UAV with inner and outer aileron-pairs. A switched linear MPC-QP and switched state observer design for fixed-wing UAVs were formulated and a fast novel QP-solver based on Hildreth's algorithm was designed. Reducing worst-case CPU-time to be below sampling period of $T_s = 10$ ms was the main objective. This was achieved with the developed Hildreth+' algorithm. MATLAB-simulations of the designed MPC-QP solvers showed MPC to be very suitable for real-time control of UAVs.

ACKNOWLEDGEMENTS

The research presented in this paper on fixed-wing UAV modeling and MPC control design was carried out at RMIT Melbourne under supervision of Prof. L. Wang and MSc. A. Sattar. The research presented in this paper on fast QP solvers for MPC based on Hildreth was carried out at the Eindhoven University of Technology under supervision of Dr. M. Lazar.

The first author would like to thank Prof. P.M.J. Van den Hof and Prof. L. Wang for organising the visit to RMIT Melbourne and MSc Y.H. Yuen for his contribution in improving the Hildreth algorithm and suggesting use of MEX-files.

REFERENCES

Aström, K.J. and Hägglund, T. (2006). Advanced PID Control. *International Society of Automation (ISA)*.

Boon, M.A., Drijfhout, A.P., and Tesfamichael, S. (2017). Comparison of a Fixed-wing and Multi-rotor UAV for Environmental Mapping Applications: a Case Study. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII(2), 47–54.

Carapau, R.S., Rodrigues, A.V., Marques, M.M., and Lobo, V. (2017). Unmanned Aerial Systems in Military Environments: the Benefits of Interoperability. *Naval Academy Scientific Bulletin*, XX(1), 43–47.

Ferreau, H.J. (2006). An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications for Predictive Engine Control. *Ruprecht-Karls-Universität Heidelberg, MSc thesis*.

Ferreau, H.J. (2011). Model Predictive Control Algorithms for Applications with Millisecond Timescales. *Universiteit Leuven, Department of Electrical Engineering, PhD thesis*.

Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6, 327–363.

Frasch, J.V., Vukov, M., Ferreau, H.J., and Diehl, M. (2013). A dual Newton strategy for the efficient solution of sparse quadratic programs arising in SQP-based nonlinear MPC. *Optimization Online*, 1–8.

Frasch, J.V., Vukov, M., Ferreau, H.J., and Diehl, M. (2014). A new quadratic programming strategy for efficient sparsity exploitation in SQP-based nonlinear MPC and MHE. *19th World Congress, IFAC*, 47, 2945–2950.

Hackney, C. and Clayton, A.I. (2015). Unmanned Aerial Vehicles (UAVs) and their applications in geomorphic mapping, chapter 2. *Geomorphological Techniques*, (1.7), 1–12.

Kögel, M., Zometa, P., and Findeisen, R. (2012). On Tailored Model Predictive Control for Low Cost Embedded Systems with Memory and Computational Power Constraints. *Technical report: Otto-von-Guericke-University Magdeburg, Magdeburg, Germany*, 1–8.

Ligthart, J.A.J., Poksawat, P., and Wang, L. (2017). Experimentally validated fault tolerant model predictive control design for a hexacopter. *International Journal of Control*, 50(1).

Marris, E. (2013). Fly, and bring me data. *Nature*, 498, 156–158.

Mittal, S. (2014). A Study of Successive Over-relaxation (SOR) Method Parallelization Over Modern HPC Languages. *International Journal of High Performance Computing and Networking, Inderscience.*

Poksawat, P. (2018). Automatic Controller Design and Turbulence Mitigation System for Fixed-wing Unmanned Aerial Vehicles. *RMIT University, School of Engineering, PhD thesis.*

Poksawat, P., Wang, L., and Mohamed, A. (2017). Gain Scheduled Attitude Control of Fixed-Wing UAV with Automatic Controller Tuning. *IEEE Transactions on Control Systems Technology*, 1–12.

Roberts, S. (2010). Topic 3, Iterative methods for $Ax = b$, 3.2 Jacobi method, 3.3 Gauss-Seidel method. *University of Oxford, Machine Learning Research Group.*

Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A., and Guizani, M. (2019). Unmanned Aerial Vehicles (UAVs): A survey on Civil Applications and Key Research Challenges. *IEEE Access*, 7, 48572–48634.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2019). OSQP: An Operator Splitting Solver for Quadratic Programs. *arXiv: 1711.08013.*

Stellato, B., Naik, V.V., Bemporad, A., Goulart, P., and Boyd, S. (2018). Embedded Mixed-Integer Quadratic Optimization Using the OSQP Solver. *2018 European Control Conference (ECC).*

Thamm, H.P., Brieger, N., Neitzke, K.P., Meyer, M., Jansen, K., and Mönninghof, M. (2015). Songbird: an innovative UAS combining the advantages of fixed wing and multi rotor UAS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL(1), 345–349.

Wang, L. (2009). Model Predictive Control System, Design and Implementation using Matlab. *Springer.*