Implementation of Massé et al.'s $\mathcal{H}_\infty$ Controller

Jérémie X. J. Bannwarth

30 November 2018

1 Introduction

This document describes the implementation of the controller presented by Massé et al. [1] on the quadcopter model introduced in our AIAA Journal paper [2]. The main goal behind this task is to gain a better understanding of the $\mathcal{H}_\infty$ controller design process.

In their paper, Massé et al. present a simulation model for a standard quadcopter frame, making use of the propeller aerodynamic model developed by Khan and Nahon [3, 4]. They subsequently describe and evaluate the performance of two linear station keeping controllers; a LQR controller and a $\mathcal{H}_\infty$ controller.

Initially, an attempt at replicating the multirotor model used by Massé et al. was made. Replicating the model would enable to ensure the controller has been implemented properly by comparing results with those presented in the paper. In addition, the propeller model could be reused in the 'Aerodynamic Modelling' section of the thesis as a point of comparison. However, several issues listed in Section 3 arose. As a consequence, it was decided to instead implement the $\mathcal{H}_\infty$ controller on the model presented in our AIAA Journal paper [2], which will be described in the first part of this report.

2 Structure of the Controller

The controller block diagram is shown in Figure 1.

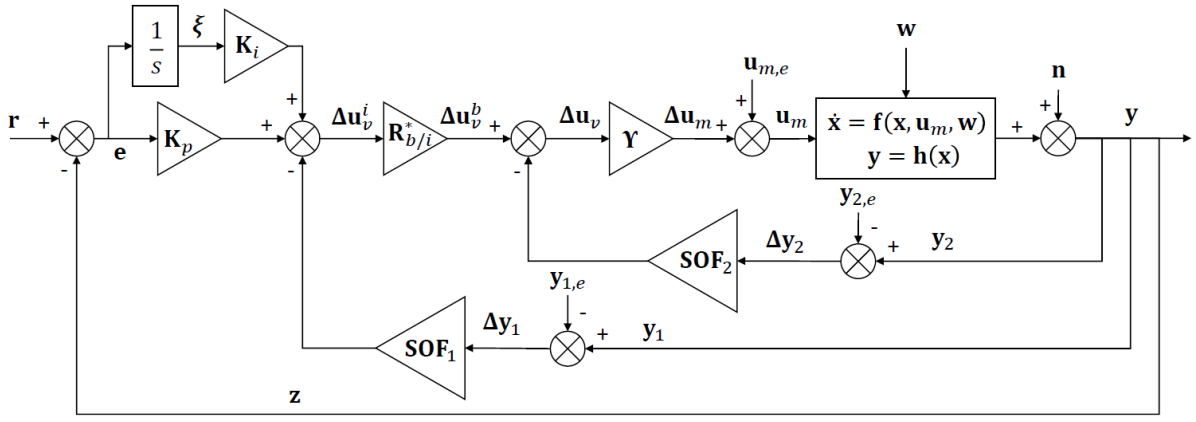

Figure 1: Block diagram of Massé et al.'s $\mathcal{H}_\infty$ controller.

The state vector is defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_q{}^\top & \mathbf{x}_m{}^\top \end{bmatrix}^\top \tag{1}$$

where $\mathbf{x}_q$ represents the states of the motors and $\mathbf{x}_q$ represents the rest of the states of the UAV:

$$\mathbf{x}_q = \begin{bmatrix} u & v & w & p & q & r & x & y & z & \phi & \theta & \psi \end{bmatrix}^\top. \tag{2}$$

where u, v, and w are the linear velocities of the UAV in the body frame, and p, q, r are the angular velocities of the UAV in the body frame. The remaining states are the standard pose of the UAV in the world frame.

The input vector is $\mathbf{u}_m = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\top$, where u_j represents the control signal sent to motor j, where motor 1 is located to the front left, motor 2 to the back left, etc. It is obtained by multiplying the virtual control input $\mathbf{u}_v = \begin{bmatrix} F_{z,d} & M_{x,d} & M_{y,d} & M_{z,d} \end{bmatrix}^\top$ by the mixing matrix $\boldsymbol{\Upsilon}$. As our current simulation model still uses the north-west-up convention, the mixing matrix is different from that presented in the original paper and is defined as

$$\boldsymbol{\Upsilon} = \frac{1}{4} \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \tag{3}$$

The structure is similar to that of a reference tracking controller. $\mathbf{SOF}_1$ and $\mathbf{SOF}_2$ are state feedback matrices. However, $\mathbf{SOF}_2$ acts on states within the body frame while $\mathbf{SOF}_1$ acts on states within the world frame. This is achieved by defining the vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ as

$$\mathbf{y}_1 = \begin{bmatrix} \dot{y} & \dot{x} & y & x \end{bmatrix}^\top, \tag{4}$$

$$\mathbf{y}_2 = \begin{bmatrix} \dot{z} & p & q & r & z & \phi & \theta & \phi \end{bmatrix}^\top. \tag{5}$$

The output of the state feedback component in the world frame is added to that of the reference tracking component and then converted to the body frame through the coordinate transformation matrix $\mathbf{R}^*_{b/i}$.

The reference tracking component of the controller tries to minimise the error between the reference states $\mathbf{r} = \begin{bmatrix} x_d & y_d & z_d & \psi_d \end{bmatrix}^\top$ and the output states $\mathbf{z} = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\top$ in order to track a reference position and yaw angle. The gain matrix $\mathbf{K}_i$ acts on the integrated error while $\mathbf{K}_p$ acts on the error.

In order to take advantage of the decoupling between the axes of the UAV, Massé et all set most elements of the gain matrices to 0. We go further and force gains pertaining to the x/roll axis to be the same as those pertaining to the y/pitch axis in order to both ensure the UAV performs equally well regardless of its yaw angle and to simplify the gain tuning procedure. In effect, this was almost already the case for the gain matrices presented by Massé et al. and as such should not make a large difference. The gain matrices thus take the following forms:

$$\mathbf{SOF}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ A_1 & 0 & A_2 & 0 \\ 0 & -A_1 & 0 & -A_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{SOF}_2 = \begin{bmatrix} B_1 & 0 & 0 & 0 & B_4 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 & 0 & B_5 & 0 & 0 \\ 0 & 0 & -B_2 & 0 & 0 & 0 & -B_5 & 0 \\ 0 & 0 & 0 & B_3 & 0 & 0 & 0 & B_6 \end{bmatrix}, \tag{6}$$

$$\mathbf{K}_p = \begin{bmatrix} 0 & 0 & C_2 & 0 \\ 0 & C_1 & 0 & 0 \\ -C_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 \end{bmatrix}, \mathbf{K}_i = \begin{bmatrix} 0 & 0 & D_2 & 0 \\ 0 & D_1 & 0 & 0 \\ -D_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & D_3 \end{bmatrix}. \tag{7}$$

These matrices are tuned using the `systune` algorithm in MATLAB, through the use of the Simulink Control System Tuner. This algorithm takes as inputs a list of control objectives, in this case wind disturbance rejection, closed-loop pole confinement, open-loop stability margin, and step tracking performance. It then linearises the system and uses structured $\mathcal{H}_\infty$ techniques to compute the gains. This is an optimisation problem that can have multiple local minima, and as a consquence the starting point will have an influence on the outcome. Figure 2 shows an example of the Control System Tuner software, with plots of the various control system objectives.

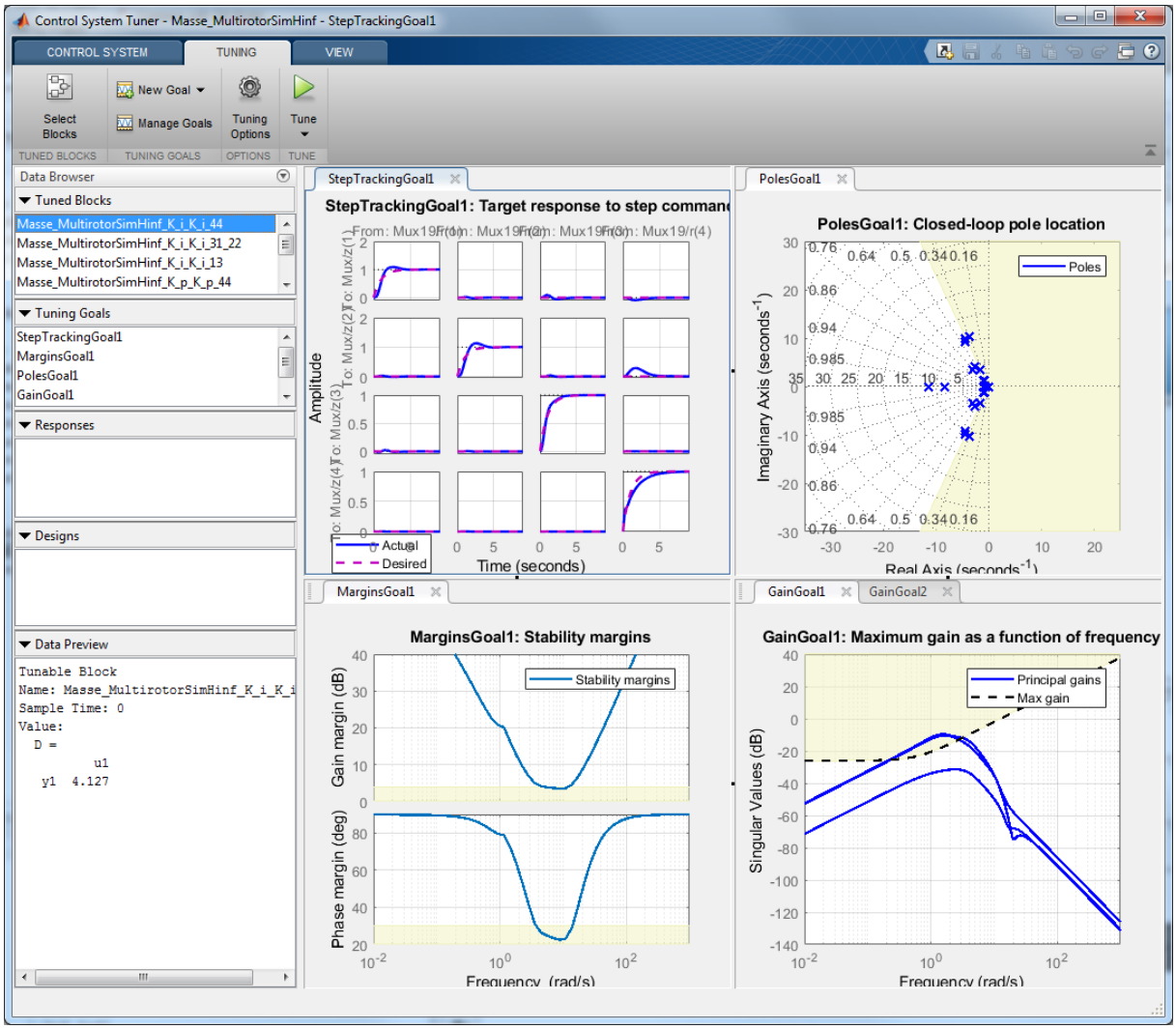

Figure 2: Simulink Control System Tuner window.

3 Issues with Model Implementation

Several issues arose while implementing the quadcopter model presented by Massé et al. They are listed in this section as a matter of record.

3.1 Khan and Nahon's Propeller Model

Khan and Nahon's propeller model predicts three-dimensional forces and moments generated by a propeller given the propeller parameters, its rotational speed, and an inflow wind vector. It uses a combination of blade-element and momentum theory and accounts for assymetric blade effects.

The first step of the process is to compute the blade aerodynamic coefficients as a function of the effective angle of attack on the blade. The results presented in the paper were successfully replicated, as shown in Figure 3.

The second step consists of computing the induced axial velocity distribution on the blade as a function of the angle of the blade and distance of the element from the centre of the propeller. This was also successfully replicated.

The third step involves the computation of the three primary forces, F_x, F_y, and F_z, and moments, M_x, M_y, and M_z. These equations to compute these forces and moments can be divided

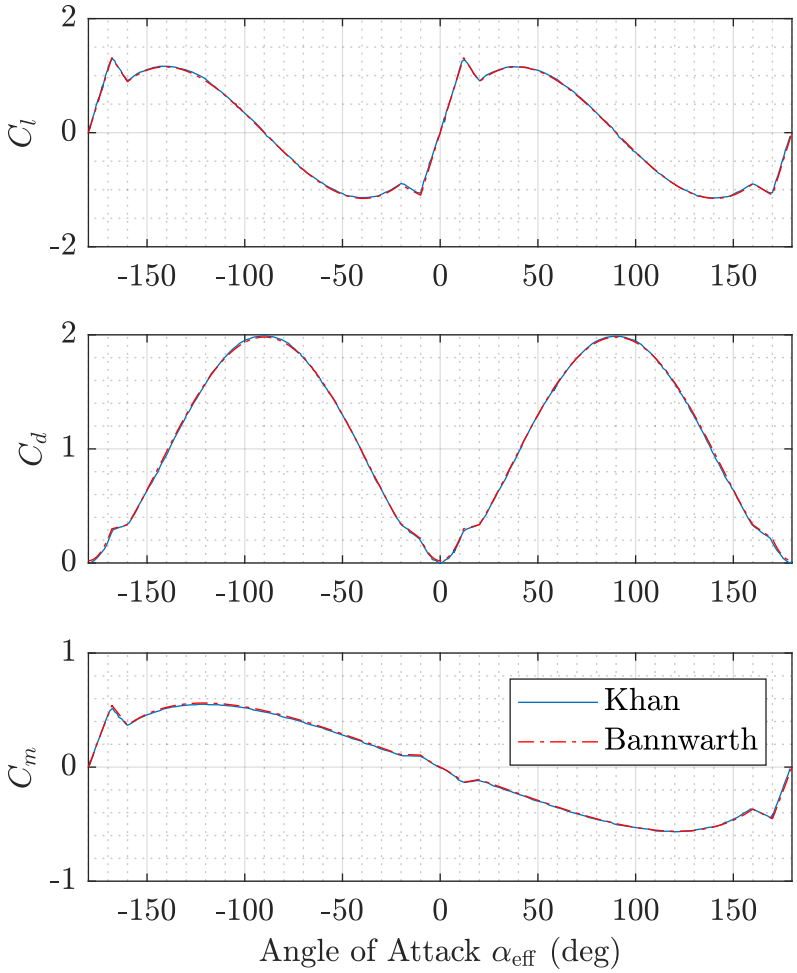

Figure 3: Aerodynamic coefficients variations with angle of attack

in two groups; those derived from Equation (8) and those derived from Equation (9):

$$\mathrm{d}T = \frac{N}{4\pi}\rho c \int_{\psi=0}^{2\pi} V_R^2 \left(C_l \cos\phi - C_d \sin\phi\right) \mathrm{d}\psi \mathrm{d}r, \tag{8}$$

$$\mathrm{d}H = \frac{N}{4\pi}\rho c \int_{\psi=0}^{2\pi} V_R^2 \left(C_l \sin\phi + C_d \cos\phi\right) \mathrm{d}\psi \mathrm{d}r. \tag{9}$$

The equations for F_x, M_y, and M_z are derived from Equation (8) and those for F_y, F_z, and M_x are derived from Equation (9). The former were able to be matched relatively closely, while the latter were not, as shown in 4 While it is possible this is due to an implementation mistake, the equations for all forces and moments are relatively similar, and were re-derived and checked several times.

Finally, a small mistake was found in the forces and moment equations. As an example, the equation for the axial thrust generated by the rotor is given as

$$T = \frac{N}{4\pi}\rho c \sum_{r=R_h}^{R_p} \sum_{\psi=0}^{2\pi} V_R^2 \left(C_l \cos\phi - C_d \sin\phi\right) \Delta\psi \Delta r. \tag{10}$$

However, the chord length c is a function of the distance of the element from the centre of the propeller, r. As a consequence, it should by inside the first summation sign rather than outside,

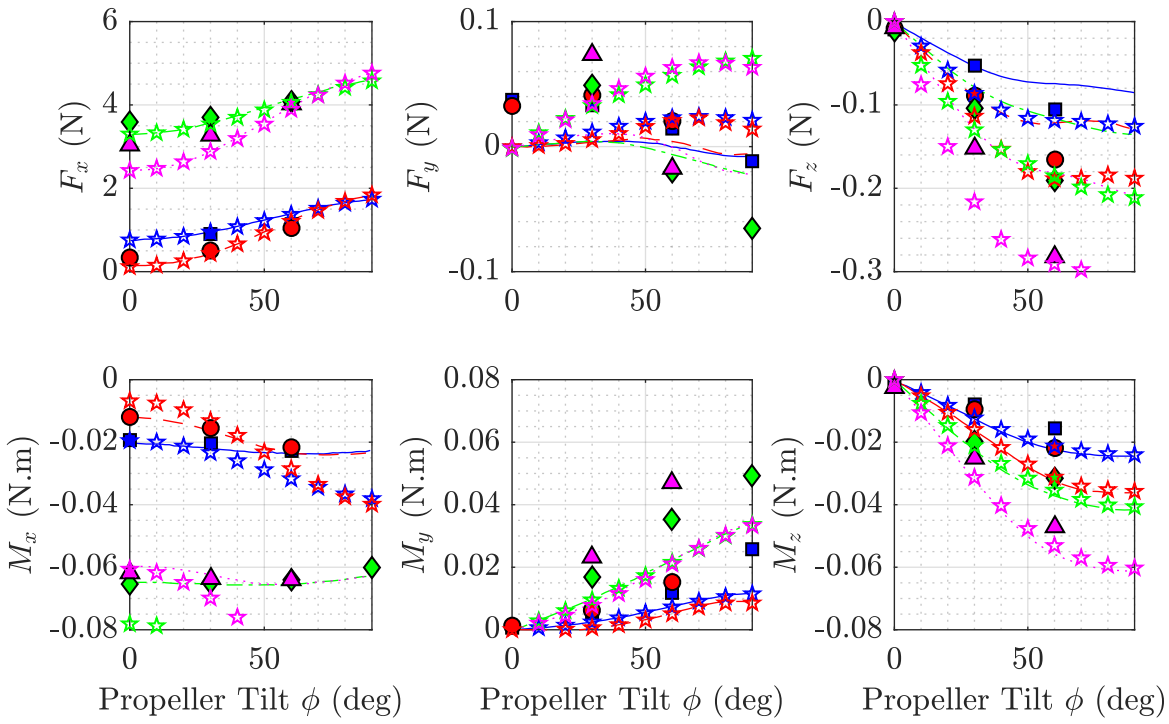

Figure 4: Aerodynamic forces and moments predicted by the propeller model. The solid markers and lines represent Khan et al.'s experimental and simulation results respectively, and the stars represent Bannwarth et al.'s results. These results are expressed in the UAV frame with an incident angle of the wind of 90°, which means that the y and z axes are inverted compared to the propeller frame.

as follows:

$$T = \frac{N}{4\pi}\rho \sum_{r=R_h}^{R_p} c \sum_{\psi=0}^{2\pi} V_R^2 \left(C_l \cos\phi - C_d \sin\phi\right) \Delta\psi \Delta r. \tag{11}$$

This small correction applies to all six force and moment equations.

3.2 Massé et al.'s Implementation of the Propeller Model

There are two main issues with Massé et al.'s implementation of Khan and Nahon's propeller model.

First of all, in their paper, Massé et al. state that they have used generated look-up tables using Khan and Nahon's propeller model. They claim the look-up tables were generated over the following ranges of input rotor speed ω_j, axial wind speed u_a^p, and in-plane wind speed $|| \begin{bmatrix} v_a^p & w_a^p \end{bmatrix} ||_2$:

$$\begin{aligned} \omega_j &\in [0, 8600] \text{ RPM} \\ u_a^p &\in [-12, 12] \text{ m s}^{-1} \\ || \begin{bmatrix} v_a^p & w_a^p \end{bmatrix} ||_2 &\in [0, 20] \text{ m s}^{-1} \end{aligned}$$

The range for the angular speed and in-plane velocity is appropriate. On the other hand, Khan and Nahon's model is for forward flight, which means that it is not proven to work for negative axial wind speeds. Indeed, Khan and Nahon do not present results in the negative range in any of their papers. As a consquence, the range of axial wind speeds presented is dubious.

The second issue lies in the computation of the forces and moments. Khan and Nahon's model computes all forces and moments generated by a propeller. However, Massé et al. add the output of

a standard thurst model $\propto k_t\omega_j^2$ to that of their propeller model, which should result in a doubled amount of thrust. In addition, the forces and moments generated by the propeller model are scaled from 0 to 1 depending on the orientation of the propeller relative to the wind and the UAV. No details or rationale for this implementation are provided.

3.3 Controller Implementation Details

There is a number of small mistakes in the controller implementation details presented in the paper. For example, some of the gain matrices presented for the $\mathcal{H}_\infty$ controller have incorrect signs, and the given state feedback $\mathbf{SOF}_2$ matrix has a form indicating it should be applied to the vector $\mathbf{y}_2 = \begin{bmatrix}\dot{z} & z & p & \phi & q & \theta & r & \psi\end{bmatrix}^\top$, whereas the paper previously states that $\mathbf{y}_2 = \begin{bmatrix}\dot{z} & p & q & r & z & \phi & \theta & \psi\end{bmatrix}^\top$.

In addition, the paper provides a state-space model for the system and states it is used for the controller synthesis procedure. However, the state-space model does not have a disturbance input, which is necessary to synthesise a $\mathcal{H}_\infty$ controller. It is likely that Massé et al. used Simulink's Control System Tuner which automatically performs a linearisation of the system dynamics, however they do not explicitely say so. This means that there is no need to replicate the state-space model provided in the paper, as numerically linearising the non-linear model using Simulink's tools is necessary anyway.

References

[1] C. Massé, O. Gougeon, D. Nguyen, and D. Saussié, "Modeling and Control of a Quadcopter Flying in a Wind Field: A Comparison Between LQR and Structured $\mathcal{H}_\infty$ Control Techniques," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2018, pp. 1408–1417.

[2] J. X. J. Bannwarth, Z. J. Chen, K. A. Stol, B. A. MacDonald, and P. J. Richards, "Aerodynamic Force Modeling of Multirotor Unmanned Aerial Vehicles," *AIAA Journal*, 2018.

[3] W. Khan and M. Nahon, "Toward an Accurate Physics-Based UAV Thruster Model," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1269–1279, Aug 2013.

[4] ——, "A Propeller Model for General Forward Flight Conditions," *International Journal of Intelligent Unmanned Systems*, vol. 3, no. 2/3, pp. 72–92, 2015. [Online]. Available: https://doi.org/10.1108/IJIUS-06-2015-0007