Event-Triggered Saturating Attitude Controller
Problem Statement

Diogo Almeida

February 21, 2014

The following is a draft of the problems and proposed solutions for my thesis project, namely the control of a quadcopter attitude using event triggered strategies. It is proposed the implementation of a saturating controller that fully exploits the available control torques, together with a Lyapunov based event triggering function.

1 The attitude controller

Proposed by Fritsch, Henze and Lohmann in [1], the attitude controller implements a control law that stabilizes a quadcopter attitude in two steps. Firstly, the thrust axis of the quad is aligned with the reference direction. Secondly, the yaw angle of the vehicle is corrected. The control variables here are the displacement angle of the thrust axis, φ, and the yaw error, ϑ. Prioritizing the thrust axis direction has advantages if the primary concern of the higher level control revolves around the translational movement of the robot. Another advantage lies on the controller's hability to take advantage of the actuating power available by explicitly modelling the maximum torques the system can provide and exploiting that knowledge by saturating the control torques whenever possible. This is achieved by adopting an energy shaping approach, in which a desired energy for the system is designed and a damping strategy that penalizes movements that go against the reference equilibrium, while boosting the ones that go in the desired way, is applied. The resulting control torques are given by

$$\boldsymbol{\tau}(\mathbf{x}) = \mathbf{T}(\mathbf{q}) - \mathbf{D}(\mathbf{x})\boldsymbol{\omega} \,, \tag{1}$$

where $\mathbf{q}$ is the quaternion that represents the attitude error of the quadcopter, $\boldsymbol{\omega} = [\omega_x \;\; \omega_y \;\; \omega_z]^\top$ are the angular velocities of the quad around its three body frame axis, $\mathbf{T}(\mathbf{q})$ is the torque field that the desired energy would generate and $\mathbf{D}(\mathbf{x})$ is a positive semi-definite damping matrix that allows for the control saturation as well as the almost global assymptotic stability of the controlled system. Finally, the state of the system with respect to the attitude is given by $\mathbf{x} = [\mathbf{q} \;\; \boldsymbol{\omega}]^\top$.

The control law (1) is designed to ensure that a Lyapunov function (LF) consisting of the sum of the system kinectic energy and an artificial potential

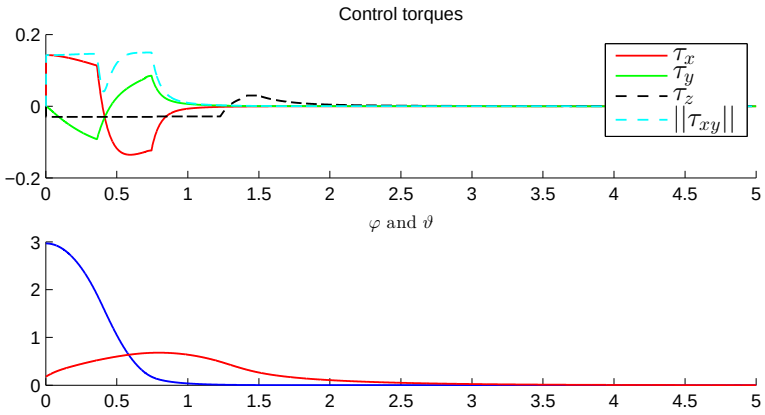

Figure 1: The applied torques and resulting behavior of the control variables for the saturating controller [1]

energy has a time derivative given by

$$-\boldsymbol{\omega}^\top \mathbf{D}(\mathbf{x})\boldsymbol{\omega} \leq 0. \tag{2}$$

The authors of [1] were able to prove assymptotic stability based on LaSalle's Stability theorem. The baseline result for this controller, under the initial conditions in [1] are given in figure 1.

2 Event-triggered strategy

2.1 Lyapunov based scheduling

The main problem in event-triggered implementations is to find an execution rule that allows to generate the sampling instants t_k in such a way that the stabilization of the system is ensured, while avoiding accumulation points, that is, having the inter sampling time decreasing to zero. One systematic approach to the problem is given by [2], where a sampling instant is generated every time the time derivative of the LF, $V(\mathbf{x})$, of the system violates an inequality condition. This strategy requires the LF to be bounded above and below by strictly increasing functions of the state and for its time derivative to be upper bounded by an 'error term' $-\alpha(|\mathbf{x}|) + \gamma(|\mathbf{e}|)$, where $\mathbf{e}$ is the difference between the state at the current time t and at the last sampling instant t_k

$$\begin{aligned} \underline{\alpha}(|\mathbf{x}|) \leq \quad V(\mathbf{x}) \quad &\leq \overline{\alpha}(|\mathbf{x}|) \\ \dot{V}(\mathbf{x}) \quad &\leq -\alpha(|\mathbf{x}|) + \gamma(|\mathbf{e}|) \end{aligned} \tag{3}$$

and, by ensuring

$$\gamma(|\mathbf{e}|) \leq \sigma\alpha(|\mathbf{x}|)\,, \quad 0 < \sigma < 1 \tag{4}$$

we get $\dot{V}(\mathbf{x}) \leq 0$ at all times.

2.2 Threshold based scheduling

A simpler alternative to use as an execution rule is to define a threshold $\overline{e}$ for the difference between the control variable at the time t_k [3], *i.e.* the last instant the control signal was updated, and the current value at the time t, over which the control signal is recomputed. However, it does not offer the guarantee that the system will remain stable when enforcing this update rule. As such, it probably needs to be tuned experimentally to obtain acceptable results without degrading performance too much.

3 Possible approaches

What follows are some ideas for event-triggering rules that can be implemented together with the saturating controller.

3.1 Lyapunov based Saturating Event Triggered control

This is the approach discussed in [2]. The event triggering rule is obtained from the time derivative of the LF of the controlled system, (2). For that, a $\alpha(|\mathbf{x}|)$ and $\gamma(|\mathbf{e}|)$ class-$\mathcal{K}$ functions need to be found that comply with (3) and allows for the definition of the update rule (4). Fritsch et al. [1] define $\mathbf{D}(\mathbf{x})$ as

$$\mathbf{D}(\mathbf{x}) = \begin{bmatrix} \kappa_{xy}(\mathbf{x})\mathbf{D}_{xy}(\mathbf{x}) & \mathbf{0} \\ \mathbf{0} & \kappa_z(\mathbf{x})d_z(\mathbf{x}) \end{bmatrix} \tag{5}$$

where

$$\mathbf{D}_{xy}(\mathbf{x}) = \frac{d_\varphi(\mathbf{x})}{1-q_p^2}\begin{bmatrix} q_x^2 & q_xq_y \\ q_xq_y & q_y^2 \end{bmatrix} + \frac{d_\perp}{1-q_p^2}\begin{bmatrix} q_y^2 & -q_xq_y \\ -q_xq_y & q_x^2 \end{bmatrix}$$

hence, the time derivative of the LF, (2) becomes

$$-\frac{\kappa_{xy}(\mathbf{x})}{1-q_p^2}\left[\omega_x^2\left(d_\varphi(\mathbf{x})q_x^2 + d_\perp q_y^2\right) + \omega_y^2\left(d_\varphi(\mathbf{x})q_y^2 + d_\perp q_x^2\right)\right] - \omega_z^2\kappa_z(\mathbf{x})d_z(\mathbf{x}). \tag{6}$$

One can easily see that, since in the equilibrium $q_p = \pm 1$ and, by multiplying the first element in the sum (6) by $1-q_p^2$, we obtain a function that is always bigger than (6). That can be used to find the event rule (4) while ensuring $\dot{V}(\mathbf{x}) \leq 0$ for all $\mathbf{x}$ different than $\mathbf{0}$.

3.2 Heuristic approach

Another alternative is to discard altogether the Lyapunov based approach and to generate an event-triggering mechanism based on either the state error as defined above, or the error with respect to the reference. In the first case, the control signal is updated every time the state deviates more than a certain value

from the state at the previous sampling time [3]. In the second case, one can do periodic sampling when outside a 'tolerance region' specified by the system designer [4]. Moreover, if the purpose of the event triggering system is to save network resources, as in [3], the control algorithm may run periodically over an input error that is updated according to the update rule, or the control algorithm may be the limited to run only at event times, and the control signal is kept constant in the meanwhile. All these approaches lack stability results *a priori* and require an analysis case by case.

4 Lyapunov based event triggering results

Given a controller that stabilizes the system, if we fulfil the conditions in [2, Theorem 3.1], we can get a control update rule that preserves the stability of the system and does not pose the risk of accumulation points. The main problem with this approach is that it is not easy to find an $\alpha(|\mathbf{x}|)$ that is stricly increasing and satisfies the inequality. Remembering that a desired equilibrium has two corresponding points in the quaternion space, $\boldsymbol{\omega} = \mathbf{0}$ and $\mathbf{q} = [0 \;\; 0 \;\; 0 \;\; \pm 1]$, we have that

$$\alpha(\mathbf{x}) = -\boldsymbol{\omega}^\top D_\alpha(\mathbf{x})\boldsymbol{\omega} \tag{7}$$

where

$$D_\alpha(\mathbf{x}) = \begin{bmatrix} \kappa_{xy}(\mathbf{x})(1-q_p^2)\mathbf{D}_{xy}(\mathbf{x}) & \mathbf{0} \\ \mathbf{0} & \kappa_z(\mathbf{x})d_z(\mathbf{x}) \end{bmatrix}$$

satisfies (3), though it is not a class-$\mathcal{K}$ function, nor a function of the norm of the state. The need for the function to be strictly increasing with the norm of the state is to ensure that there are no accumulation points. This can be solved by imposing a minimum inter sampling time[1]. The stability of the resulting system is ensured by (4). Indeed, the implementation of the update rule with $\alpha(\mathbf{x})$, as defined in (7), allows for the convergence of the controlled variables without any noticeable performance issues, see figure 2 for the results with $\sigma = 0.9$.

4.1 How to deal with accumulation points

The main problem with this result, is the decrease in the inter-sampling time as the simulation goes on. Eventually, when the state of the system stabilizes, inter-sampling time coincides with the sampling time used for the simulations. It seems as if $\gamma(|\mathbf{e}|)$ is decreasing slower than $\alpha(\mathbf{x})$, and this results in the inequality (4) being violated every time it is checked. I assume this is a direct consequence of not enforcing $\alpha(\mathbf{x}) \in \mathcal{K}$. My proposal is to add a very small ϵ_x to (4) so as to become

$$\gamma(|\mathbf{e}|) \leq \sigma\alpha(|\mathbf{x}|) + \epsilon_x\,, \;\; 0 < \sigma < 1. \tag{8}$$

With this new update rule we introduce a dead zone for the state-error that may yield stability problems, but at the same time allows for the inter-sampling

[1] But will that grant stability?

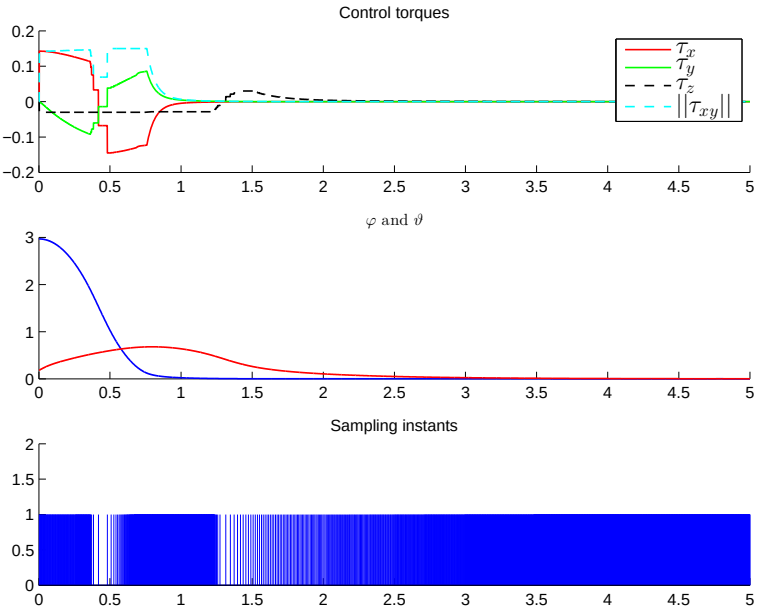

Figure 2: Results using the update rule (4) with $\alpha(\mathbf{x})$ defined as in (7)

time to grow unbounded if the state is in the desired equilibrium, see figure 3 with $\sigma = 0.9$ and $\epsilon_x = 0.05$.

5 Error based methods

Besides tying the event triggering to the evolution of the time derivative of the LF of the system, one can try to implement rules based purely on the state error, that is, on how much the state has changed since the last time the control was updated or even by defining an area of admissible error, where the control is kept constant while the state of the system remains inside it.

5.1 State error method

Here, the update rule is based on the evolution of the state since the last time the control was updated. Given the error

$$\mathbf{e} = \mathbf{x}(t_k) - \mathbf{x}(t),$$

the control is updated every time the norm of the state error grows beyond a certain threshold

$$|\mathbf{e}| > \sigma_e. \tag{9}$$

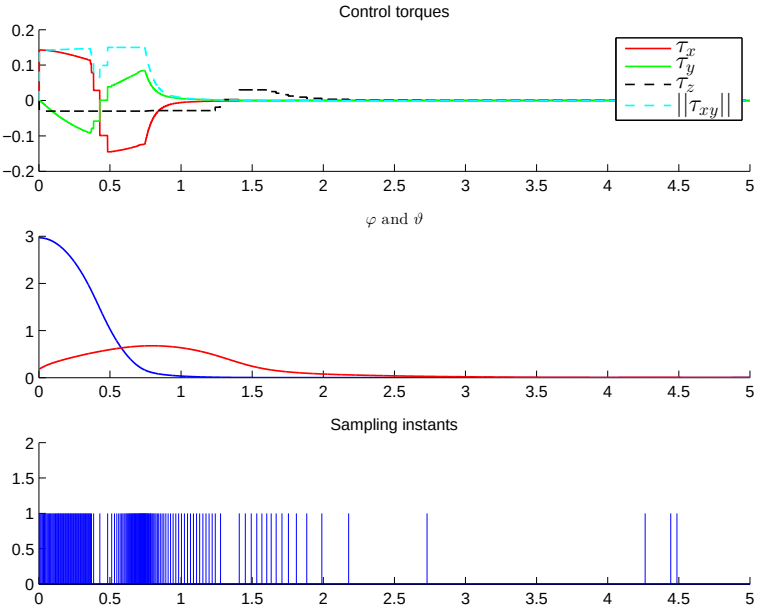

Figure 3: Results for the execution rule (8)

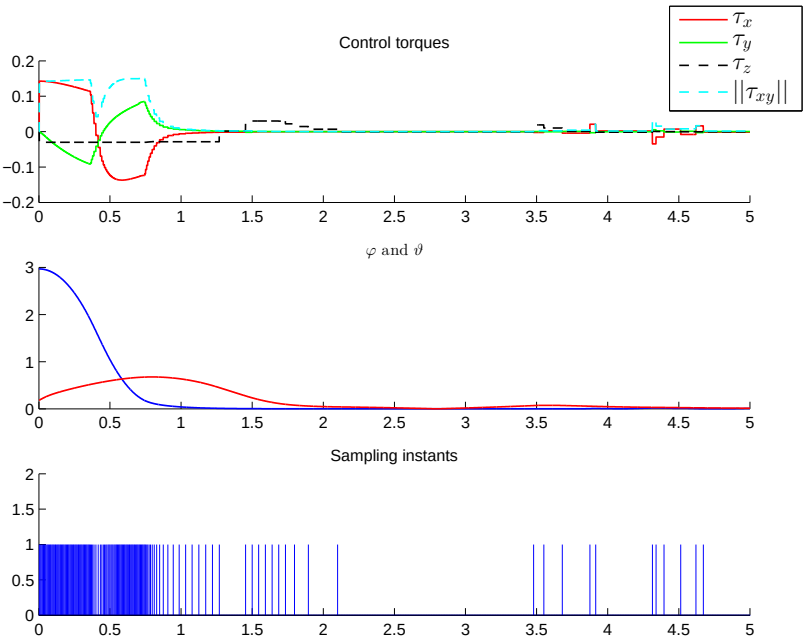

Figure 4: Results for the execution rule (9)

One advantage of this rule relies on its simplicity. If the state is diverging too much from the one we had the last time the control was updated, then it is time to do it again. Stability may be a concern here, at least when near the equilibrium, the control may be so agressive that the state doesn't stay where it is meant to be, resulting in poor performance and extra control updates. The value of the threshold σ_e impacts the results significantly as well. For $\sigma_e = 0.1$ the observed results are the ones in figure 4. With this threshold value, it is noticeable that the controller lets the state move away from the desired equilibrium and reacts later in the simulation to correct it. This results in a diverging behavior of the state when it should be resting in the reference value (figure 5). A similar behavior is observed when using the rule (8) for larger values of σ_e, case where there are less updates to the control signal, but results in larger tolerance for positive $\dot{V}(\mathbf{x})$.

5.2 Error with respect to the reference

Another simple idea is to update the control value every time the state leaves an admissible set [4]. This is similar to having a relay in the control loop that closes the feedback loop only when the error grows above a certain threshold.

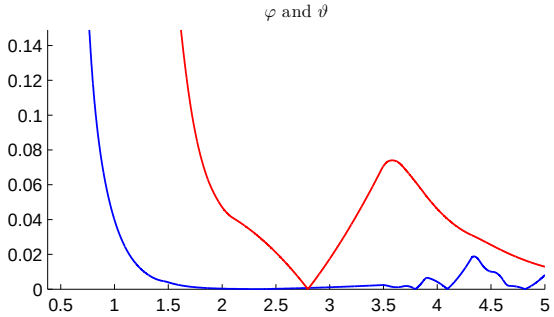

Figure 5: Details of the diverging behavior when using execution rule (9)

Here, the error may be given as

$$\mathbf{e} = \mathbf{x}(t) - \mathbf{x}_d,$$

where $\mathbf{x}_d$ is the state in the equilibrium. Since the proposed controller works on the error between the current state and the reference, the desired equilibrium is always the same, which makes it easy to implement the update rule. In figure 6 are depicted the results for an admissible deviation of 1 percent of the state norm. For every value of the admissible error experimented with, the oscillating behavior appeared, sooner or later. That seems reasonable, since the system is unstable while in open-loop. Changing the error value has an impact in the overall performance of the system while near the equilibrium, as well as on the number of control updates while near the admissible set.

6 Comparison between approaches

With the exception of the update rule (4)[2], all the approaches analysed here do not ensure that the assimptotic stability of the system is preserved. That may be acceptable or lead to unwanted behavior. There may be a tradeoff between performance and number of time events. Finally, some techniques may be objectively better than others in some aspects. Some comparison rules are proposed to benchmark the event rules.

6.1 Number of updates

The trivial measurement of performance is the number of updates of the control signal during the same time period. Less updates means that the control algorithm is called less often, resulting in computational savings. The simulation results for the cases above are presented in table 1.

[2]Or even with the inclusion of the update rule (4). Is the Lyapunov based analysis (3) only valid for continuous systems?

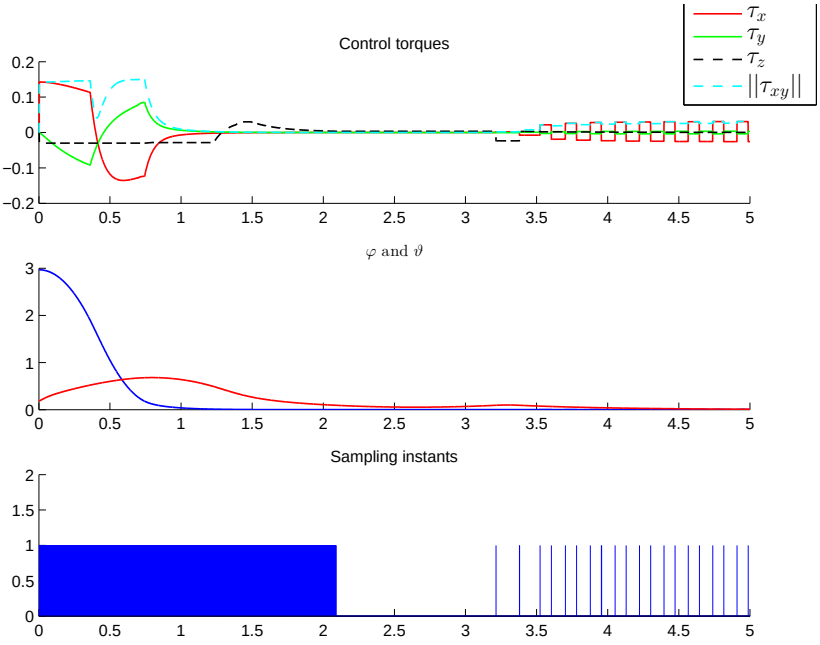

Figure 6: Results for the admissible set rule

Strategy	Updates
Baseline	5000
Rule (4)	2801 (56%)
Rule (8)	439 (9%)
Rule (9)	155 (3%)
Admissible set rule	2109 (42%)

Table 1: Number of control updates

6.2 Control signal energy

One of the purposes of using event-triggered control is to reduce the energy consumption of the overal system, by minimizing the CPU usage. Those gains may be lost if the resulting control signals uses more energy, defined as

$$E_i = \frac{1}{T}\sum_{t_k=0}^{T} \tau_i^2 \quad , \; i = \{x, y, z\}\,.$$

The respective results are presented in table 2.

Strategy	τ_x	τ_y	τ_z
Baseline	2.3333	0.53697	0.25928
Rule (4)	2.4573 (105%)	0.5833 (109%)	0.26245 (101%)
Rule (8)	2.477 (106%)	0.60163 (112%)	0.26369 (102%)
Rule (9)	2.3949 (102%)	0.54307 (101%)	0.29017 (112%)
Admissible set rule	2.539 (108%)	0.54029 (101%)	0.28066 (108%)

Table 2: Torques energy

7 Work plan

The parameters used for each approach were not tuned to obtain the best results. This may have a significant impact in the comparisons that were made. There may be worth it to look deeply into the definition of (5) to try to find a better $\alpha(\mathbf{x})$ function, or even one that fulfills the theorems in [2]. Furthermore, these approaches are only applied in simulations where I assume full access to the system state. Another issue concerns the implementation. Getting to know the system and the low level control code that is already implemented in the Arducopter may take some time. There is a need to estimate the maximum available torques that the system can output, to ensure that the proposed controller will behave properly. Finally, setting up a systematic way of testing and comparing these approaches in the real system is something that is important to do. A possible approach to the project is listed bellow:

- Get to know the system: where is the current attitude control implemented; how and where in the code is the state being measured/estimated; how does the embedded controller in the quad communicates with an external computer over the motes.
- Implement a framework that can be used to test attitude controllers and to benchmark them: set a program that allows to measure and process data required for the analysis of the controllers and event-triggering techniques.
- Assemble an experimental setup that ensures that multiple tests can be carried away in the same conditions, for example, by only allowing one degree of freedom of the quad to be available at any time.
- Depending on how to work goes on, iterate between experimental and theoretical work to try and cover a good array of event-triggering techniques.

References

[1] O. Fritsch, B. Henze, and B. Lohmann, "Fast and saturating attitude control for a quadrotor helicopter," 2013.

[2] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," 2007.

[3] D. Lehmann and K. H. Johansson, "Event-triggered pi control subject to actuator saturation," 2012.

[4] K. J. Aström, "Event based control," in *Analysis and Design of Nonlinear Control Systems* (A. Astolfi and L. Marconi, eds.), pp. 127–147, Springer Berlin Heidelberg, 2008.