Departement of Biorobotics, Institute of Movement Science (ISM), UMR 7287, CP910, 163 av. de Luminy,F-13009, Marseille France

Master's thesis

Biomimetic trajectory tracking by means of event-based control

Lorris DOLA

08/07/2013 - 31/12/2013

Laboratory Supervisors: Thibaut Raharijaona

KTH Supervisor : Dimos Dimarogonas

Esisar Supervisor : Damien Koenig

Abstract

Flying insects are able to accomplish unprecedented flight by regulating their optic flow which is the velocity to which the environment scrolls in front of their eyes. This approach can be correlated to an event-based control where an event is generated according to the change of the environment. The event-based control allows to use a very few updates of the control to reach an objective. In order to mimic the insect behavior, we propose to study and apply an event-based approach. The event-based control is simulated on a miniature direct current motor linked to a propeller and experiment on a real one. We also study different controllers: a event-based PI controller and a state-feedback controller. A special attention is given to the power consumption of the control in term of energy and computational resources. We propose to lower the sampling frequency of the direct current motor during the experimentation to reduce the power consumption and we estimate the propeller velocity in order to get rid of the velocity sensor.

Keywords: <event-based control>, <consumption cost>, <optic-flow>, <performance set>, <velocity estimation>

Les insectes volants sont capable d'accomplir de remarquables prouesses de vol en régulant leurs flux optiques qui correspond à la vitesse de défilement de l'environnment devant leurs yeux. Cette approche peut être correlée à une commande événementielle où les événements sont générés par le changement de contraste de l'environnement. La commande événementielle permet de limiter la mise-à-jour de la commande afin d'atteindre un objectif. Afin d'imiter le comportement des insectes, nous proposons d'étudier et d'appliquer l'approche événementielle. La commande événementielle est simulé sur un moteur à courant continu lié à une hélice puis expérimentée avec un vrai moteur. Nous étudions plusieurs contrôleurs: un contrôleur PI par événement et un retour d'état par événement. Une attention spéciale est accordée à la consommation du système en terme d'énergie et de ressources calculatoires. Nous proposons de diminuer la fréquence d'échantillonnage du moteur pendant l'expérimentation afin de réduire la consommation et nous estimons la vitesse de l'hélice afin d'éviter l'utilisation d'un capteur de vitesse.

Mots clés: <commande événementielle>, <coût de la consommation>, <flux optique>, <ensemble de performence>, <estimation de vitesse>

Acknowledgments

I would like to thank my supervisor, Thibaut Raharijanona, for giving me the opportunity to do my internship and for his continuous support, guidance and patience. It has been a pleasure to have a supervisor who cared so much about my work and took the time to respond my questions despite his extremely busy schedule.

My gratitude also goes to Stéphane Viollet who welcomed me in the research group so cordially. I thank Franck Ruffier for all his useful remarks and suggestions about my master's thesis.

I wish to thank Marc Boyron, Julien Diperi and Lucas Derderian for the time they spent into the test bench and for answering all my questions.

I would also like to thank the Fabien Colonnier, Stefano Mafrica, Roman Goulard and Augustin Manecy for their friendships and precious advices.

I am also thankful to Damien Koenig and Dimos Dimarogonas for supervising me and reading my master's thesis.

Last but not least, I would like to specially thank my family for their huge support and for everything they have done for me.

Contents

Chapter 1

Introduction

Some flying insects are able to achieve unprecedented flight capabilities like obstacle avoidance, prey following, landing or take-off in a unknown environment. Those exceptional capacities lead scientists to study their behavior and try to explain how they proceed to perform their flights. Several studies have shown that insects are able to navigate swiftly by regulating their optic flow, i.e. the angular velocity to which the environment scrolls in front of their eyes. It is therefore logical that the robotic community is highly interested in these flight capacities and try to understand and adapt them to the robotic world. The bio-robotic team has been created in order to accomplish these objectives.

Issue and specifications

The aim of my internship is to design an efficient control law inspired by the insect behavior. Recently, control strategies based on event (so called event-based control) were developed [8][2][3]. This approach consists in updating the control only when some event occurs instead of periodically. This strategies has shown very good results with very few updates (and therefore computation) of the control law. The event-based control can be correlated to natural behaviors from insects where a control action could be decided only when an event comes out and forces insects to react. Indeed, the optic flow which is a contrast scrolling in front of the insect's eyes can be correlated to an event-based approach where the changes of the scrolling speed can be considered as the events.

Based on this observation, the aim of my internship is to propose an event-based control which could mimic the flight of a flying insect. The control should minimize the computational cost, i.e the number of control update. Then we will see if it is possible to reduce the power consumption with this kind of control. In order to achieve this, we will simulate and experiment the event-based control with a miniature direct current motor. A reference tracking and a disturbance rejection will be perform on the direct current motor. Furthermore the event-based control will be test with an proportional integral controller and with a feedback control.

In the first part, we summarize the different event-based approaches in the existing literature and explain briefly their methods. Then we model a miniature direct current motor and simulate the event-based control on this model. Finally we ex-

periment the event-based control on a real miniature direct current motor where the system has been identified and the velocity of the propeller linked to the rotor has been estimated.

Chapter 2

Laboratory presentation

In order to analyze the flight capacities of the insects, N. Franceschini created the bio-robotic team. The bio-robotics belongs to the Institute of Movement Science Etienne-Jules Marey (ISM) which is a joint research center (UMR 7287) with Aix-Marseille University and the National Center for Scientific Research (CNRS). The objective of the laboratory is to model the sensory motor treatment of the flying insects by bringing together the knowledge from the biology and the robotic. The merger of these two different subjects contributes to have a better understanding of the flight mechanism of the insects and to apply it to the robots. There are two main interests of this merger: on the one hand, the study of the flying insects can help for the design of new flying methods for robots. On the other hand, the design of micro-robots based on flying insects helps to understand the flying mechanism of these insects. It is in this perspective that the bio-robotics team has built 7 land-based and aerial neuron mimetic robots in 20 years. We present in this part some realizations of the laboratory based on the optic flow.

2.1 Definition of the optic flow

Bees are able to navigate in a complex environment by regulating their optic flows. The optic flow corresponds to the angular velocity to which a contrasted environment scrolls in front of the bee's retina. The optic flow can be divided into two components: the translational optic flow and the rotational optic flow. However bees tend to stabilize their gaze by compensating for their body rotations. Thus the rotational optic flow can be neglect and we can only focus on the translational optic flow. Then the optic flow generated by a contrast at the point P in a wall during a pure translation motion is defined by:

$$\omega = \frac{V}{D}.sin\phi \tag{2.1}$$

where V is the bee's speed, D is perpendicular distance between the wall and the bee and ϕ is the angle between the direction of the velocity and the gaze direction. These different parameters are represented on the figure 2.1.

Since the optic flow depends of the velocity of the bee and its distance to a contrast, the bee can control its speed and distance to a contrasted wall by regulating

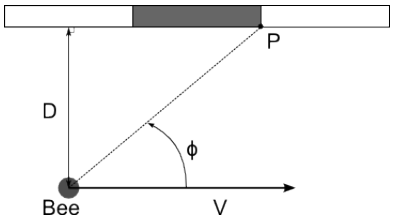

Figure 2.1: Parameters used for the computation of the optic flow during a pure translation movement

the perceived optic flow without having a direct measure of its speed or distance.

2.2 Autopilot based on the optic flow

By keeping a constant optic flow, the bee can perform obstacle avoidance, take-off and landing in a complex environment. From this statement, the bio-robotic team has developed several autopilots for micro-robots based on the measurements and the regulation of the optic flow.

The autopilot OCTAVE (Optic flow based Control sysTem for Aerospace VEhicles) has been developed in the laboratory and presented in [7]. This autopilot allows a MAV (Micro Air Vehicle) weighing 100g to automatically take off, land, cruise and react appropriately to wind disturbances. These actions are performed by only measuring and regulating the downward optic flow at a constant value. The autopilot uses only an optic flow regulation loop to navigate.

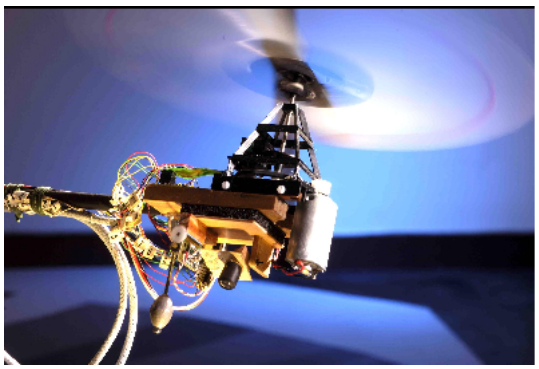

Figure 2.2: MAV using the OCTAVE autopilot. Picture from F. Ruffier and N. Franceschini, Optic Flow regulation: the key to aircraft automatic guidance, Robotics and Autonomous Systems. 50, 177-194, (2005).

In [9], the autopilot LORA (Lateral Optic flow Regulation Autopilot) is introduced. This autopilot allows a miniature hovercraft to travel along a corridor by controlling its speed and its clearance to the walls. The hovercraft is equipped with two lateral eyes that measure the right and left optic flows. Then LORA is composed of two regulation loops working with two different set points. The first loop controls the forward speed of the hovercraft based on the sum of the right and the left optic flows. The second loop controls the distance of the hovercraft from the

walls based on the maximum value between the right and the left optic flow. Then the miniature hovercraft can control its speed and distance from the walls without a direct measure of its current speed and distance from the walls but only by measuring and regulating the lateral optic flows. These control approaches are correlated to the behavior of bees since bees tend to keep constant the perceived optic flow corresponding to the nearest wall to control their distance to the walls. They also try to keep constant the larger sum of the two opposite optic flows in the horizontal and vertical planes to control their speed [5].

Figure 2.3: Miniature hovercraft equipped with the LORA autopilot. Picture from J. Serres, D. Dray, F. Ruffier and N. Franceschini, A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance, Autonomous Robot. 25:103-122, (2008).

2.3 Simulator

The team has created the autopilot ALIS (AutopiLot using an Insect based vision System) which involves the regulation of the optic flow from OCTAVE and LORA by adding the dorsal optic flow [6]. This autopilot allows a speed control in the three dimensions and a lateral, ventral and dorsal obstacle avoidance. The ALIS autopilot is used with Matlab/Simulink simulator. ALIS allows a "simulated bee" to travel along a tunnel and perform obstacle avoidance by controlling the bee's speed and its distance from the right wall, left wall, ground and roof. The user can design the tunnel to obtain different kinds of tunnel such that a straight, tapered or complex tunnel. The tunnel is then lined with grayscale natural scenes on each of the walls. The natural scenes provide a natural contrast for the measure of the optic flow. The simulated bee is equipped with four sensors of the optic flow placed at the head of the bee. The bee's head orientation is supposed to be locked to the x-axis of the tunnel and any rotation is compensated. Then each optic flow sensor receives a purely translational optic flow. Since the sensors point toward the right wall, left wall, ground and floor with an angle of 90°, the optic flows are defined by:

$$\omega_i = \frac{V_x}{D_i} \; with \; i \in \{Rght, Left, Vtrl, Drsl\} \tag{2.2}$$

where V_x is the bee's forward speed, D_{Rght}, D_{Left}, D_{Vtrl} and D_{Drsl} are respectively the distances to the right wall, the left wall, the ground and the roof. Then

each sensor receives its own optic flow which is the right optic flow ω_{Rght}, the left optic flow ω_{Left}, the ventral optic flow ω_{Vtrl} and the dorsal optic flow ω_{Drsl}. The different optic flows and a example of a tapered tunnel are shown on the figure 2.4.

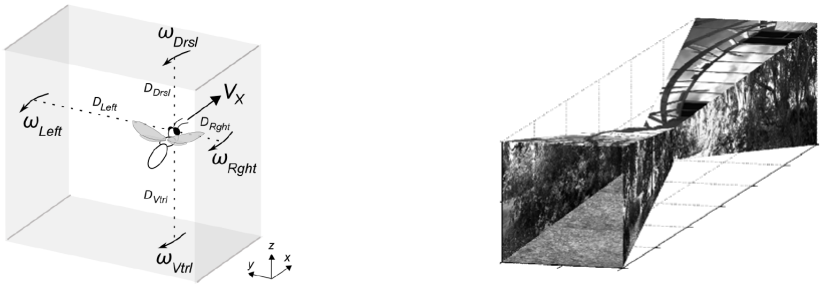

Figure 2.4: Simulated bee with the measured optic flows and tapered tunnel

Chapter 3

Literature review

Many systems are regulated by synchronous controllers that run at a constant sampling time. The problem analysis and the computation of the controller are then solved with the vast literature on system theory for sampled system. However in the some cases, it is interesting to consider event-based control in order to reduce the controller computation and reduce the update of the control without degrading the performance of a system under a certain level. With the event-based control the sampling time is no longer constant, but depends on a event which will trigger the computation of a new control. The events can have several causes e.g. a event can be generated when the output or the state signal exceeds a defined threshold or when the controller receives a data packet through a network.

Despite the fact that event-based control is a relatively new approach and only a few theories can be found with this control, there exists several articles related to this field. In [8], a event-based PID controller is presented with also the simulation results with a double tank process. This event-based PID controller is divided in two parts: an event detection part that runs at constant sampling and generates the event and a PID controller which computes a new control every time it receives a event from the event detector. The events are generated when the absolute value of the difference of the errors at time t_k and at time t_{k-1} exceeds a defined threshold e_{lim} or when the time between two samples h_{act} goes beyond a limit h_{max}: $|e(t_k) - e(t_{k-1})| > e_{lim}$ or $h_{act} > h_{max}$. Then the simulations compare a classical PID and the event-based PID with the same parameters with a double tank process. It is shown that it is possible to reduce the CPU utilization with only minor control performance degradation.

In [2], some improvements are made on the previous controller. First the discretization of the PID controller is corrected. In [8], the discretization method is based on a constant sampling period using a forward approximation which can not be apply to the event-based approach since it is not possible to know the time between the event at time t_k and the next event. By only using a backward approximation, the performance of the event-based PID controller are immediately improved. It is also suggested to remove the condition $h_{act} > h_{max}$ (named as a safety condition) which forces the controller to update the control even if the system has reached its steady-state. To do so, different algorithms are proposed to obtain a better performance level. The resulting event-based PID controller is then simulated with a simple first order system and compared with a classical PID and the PID derived in [8]. The simulation results shows that the resulting PID can use less than 50% of

sample and have better performance than the PID in [8].

Another method for a event-based approach is described in [4] where the controller is divided in two parts: a control input generator and an event generator. The event generator sends an event to the control input generator whenever the measured state of the plant $x(t)$ leaves a surrounding $\Omega(x_s) = \{x | ||x - x_s|| \leq \bar{e}\}$ where x_s is the state computed by the event generator, $\bar{e}$ is a defined threshold and $||.||$ symbolizes the vector norm operation. When a event is generated at time t_k, the control input generator re-computes the state $x_s(t_k)$ and apply the control $u(t) = -Kx_s(t_k)$ where K is the gain of the state feedback. The control is maintained until a next event is sent by the event generator. It is shown that this event-based control ensures that the state $x(t)$ remains in the surrounding $\Omega(x_s)$ even when a bounded disturbance is added and also that the error between the states of the event-based control loop x_s and the state of the plant x is bounded. Then the boundary depends of the threshold $\bar{e}$ which guarantees a certain level of performance.

The paper in which we are the most interested in concerns the description of a event-driven controller given in [3]. With this controller, the control update is computed at a constant sample rate T_s when the state of the system lies outside a set defined by the user. When the state is inside this set the control keeps its previous value. For example, we consider the system $\dot{x} = A\,x(t) + B\,u(t)$, and design the set where the control is held by $\beta := \{x \epsilon \mathbb{R}^n \mid |x| < e_T\}$. The discretized state feedback is computed as follows: $u_k = \begin{cases} Kx_k \, if \, |x_k| \geqslant e_T \\ u_{k-1} \, if \, |x_k| < e_T \end{cases}$ where $x_k = x(\tau_k)$, $u_k = u(\tau_k)$ where τ_k denotes the event times by using a zero-order hold: $u(t) = u_k$ for all $t \in [\tau_k, \tau_{k+1})$. Then as long as the state is outside β the control is computed in a normal way by a state feedback and a constant sampling time. When the state reaches the set β the control is held to its last value i.e. the value just before entering the set β. There are two ways to calculate the event times τ_k: a uniform sampling and a non-uniform sampling.

With the non-uniform sampling, τ_{k+1} is computed as follow:
$\begin{cases} \tau_{k+1} = \tau_k + T_s \quad if \quad x(\tau_k) \notin \beta \\ \tau_{k+1} = \tau_{exit} \quad if \quad x(\tau_k) \in \beta \end{cases}$ with τ_{exit} the time when the state leaves the set β. With this sampling, the control is updated at a constant rate T_s when the state lies outside β. Once the state enters the set β, the controller does not check constantly whether the state is outside the set B but compute instead the time τ_{exit} when the state should leave the set. This time can be defined as $\tau_{exit} = inf\{t > \tau_k \mid x(t) \notin \beta\}$.

With the uniform sampling, the controller checks whether the state is inside or outside the set β at a constant rate. This rate is chosen to be equal to the sampling rate of the controller T_s. Then we compute τ_{k+1} with $\tau_{k+1} = \tau_k + T_s$. Every sample time T_s, the controller checks if the state is inside or outside the set β and decides to keep or update the control.

The non-uniform sampling appears to be hard to implement whereas the uniform sampling is easier to perform. In order to keep simplicity of the event-based approach we will only focus on the uniform case for the following.

Chapter 4

Method and simulation of the event-based controller: propeller speed control of a direct current motor model

It has been decided to study a miniature direct current motor (DC motor) in order to highlight the event-based approach. The rotor axis is linked to a propeller via a reduction gear. The aim is to regulate the angular velocity of the propeller. Two different controls will be used: a PI controller and a state-feedback control. Both controls are computed for a synchronous controller then they will be adapted for a event-based control where we will define a performance set (set β) seen in [3]. The synchronous and event-based control will be then compared during the simulation of the DC motor model.

4.1 Modeling

The figure 4.1 shows the model of our DC motor.

The equations of the DC motor are:

$$\begin{aligned} u(t).V_{bat} &= R.i(t) + E + L\frac{di(t)}{dt} \\ J\frac{d\omega_r(t)}{dt} &= c_r(t) - c_p(t) + d_0(t) \end{aligned} \tag{4.1}$$

With:

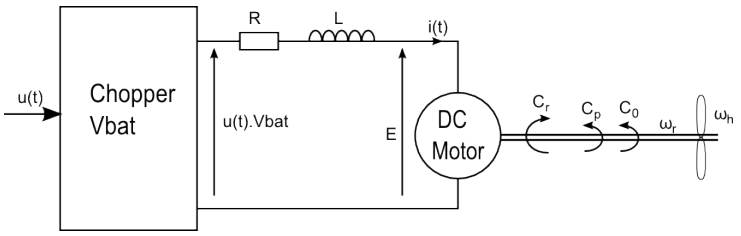

Figure 4.1: Model of the DC motor

- $u(t)$: the duty cycle [0;1]
- V_{bat}: the supply voltage (V)
- R : the armature resistance (Ω)
- $i(t)$: the armature current (A)
- E : the counter electromotive force (V)
- L : the inductance (H)
- J : the inertia of the motor (kg.m^2)
- $\omega_r(t)$: the rotor's angular velocity (rad/s)
- $c_r(t)$: the rotor's torque (N.m)
- $c_p(t)$: the resisting torque introduced by the propeller's drag (N.m)
- $c_0(t)$: a bounded disturbance that we can add to the system

Moreover, we have the following equations:

$\omega_r(t) = \omega_h(t).M$ with $\omega_h(t)$ the propeller's angular velocity (rad/s) and M the reduction ratio between the velocities of the propeller and the rotor's axis.

$E = K_e.\omega_r(t)$ with K_e the back-EMF voltage constant (V.s/rad)

$c_r(t) = K_t.i(t)$ with K_t the torque constant (N.m/A)

$c_p(t) = \frac{D}{M}.\omega_h^2$ with D the drag coefficient of the propeller (N.m.s/rad)

By inserting the different equations into 4.1, we find:

$$\begin{array}{l} \frac{di(t)}{dt} = -\frac{R}{L}.i(t) - \frac{K_e.M}{L}.\omega_h + \frac{1}{L}.u(t) \\ \frac{d\omega_h(t)}{dt} = \frac{K_t}{J.M}.i(t) - \frac{D}{J.M^2}.\omega_h^2 - \frac{1}{J.M}.c_0(t) \end{array} \tag{4.2}$$

The states of the our system are defined by the current and the propeller's velocity: $x(t) = [i(t)\ \omega_h(t)]^T$. At the equilibrium point, we obtain the following equations:

$i_{eq} = \frac{D}{K_t.M}\omega_{h_{eq}}^2$

$u_{eq} = \frac{1}{V_{bat}}(R.i_{eq} + K_e.M.\omega_{h_{eq}}) = \frac{1}{V_{bat}}(R.\frac{D}{K_t.M}\omega_{h_{eq}}^2 + K_e.M.\omega_{h_{eq}})$

The linearization of the system around the equilibrium gives the following state space representation:

$$\begin{pmatrix} \frac{d\Delta i(t)}{dt} \\ \frac{d\Delta\omega_h(t)}{dt} \end{pmatrix} = \underbrace{\begin{pmatrix} -\frac{R}{L} & -\frac{K_e.M}{L} \\ \frac{K_t}{J.M} & -\frac{2.D.\Omega_{h_{eq}}}{J.M^2} \end{pmatrix}}_{A}.\begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} + \underbrace{\begin{pmatrix} \frac{V_{bat}}{L} \\ 0 \end{pmatrix}}_{B}.\Delta u(t) + \underbrace{\begin{pmatrix} 0 \\ -\frac{1}{J.M} \end{pmatrix}}_{E}.c_0(t)$$

$$\begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C}.\begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix}$$

With $\Delta i(t) = i(t) - i_{eq}$, $\Delta\omega_h(t) = \omega_h(t) - \omega_{h_{eq}}$ and $\Delta u(t) = u(t) - u_{eq}$.

We denote $\Delta U(s)$ and $\Delta\Omega_h(s)$ the respective Laplace transforms of $\Delta u(t)$ and $\Delta\omega_h(t)$. The transfer function $H(s)$ between $\Delta U(s)$ and $\Delta\Omega_h(s)$ is given by:

$H(s) = \frac{\Delta\Omega_h(s)}{\Delta U(s)} = C.(sI - A)^{-1}.B$ with I the identity matrix of 2x2 dimension.

$$H(s) = \frac{V_{bat}.K_t.M}{J.M^2.L.s^2 + (2.D.\omega_{h_{éq}}.L + R.J.M^2).s + 2.D.\omega_{h_{éq}}.R + K_e.K_t.M^2} \tag{4.3}$$

For the simulation we use the following parameters:

- $V_{bat} = 11.1V$
- $R = 0.15\Omega$
- $L = 0.3mH$
- $J = 6.37.10^{-6} kg.m^2$
- $M = 5.6$
- $K_e = 4.139.10^{-3} V.s/rad$
- $K_t = 4.139.10^{-3} N.m/A$
- $D = 7.5.10^{-7} N.m.s/rad$

We set the equilibrium of propeller's angular velocity to $\omega_{h_{éq}} = 200rad/s$. Then we obtain:

$i_{eq} = 1.2943A$ and $u_{eq} = 0.4351V$.

After the numerical application, we have:

$$\begin{pmatrix} \frac{d\Delta i(t)}{dt} \\ \frac{d\Delta\omega_h(t)}{dt} \end{pmatrix} = \begin{pmatrix} -500 & -77.26 \\ 115.89 & -1.5 \end{pmatrix} \cdot \begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} + \begin{pmatrix} 37000 \\ 0 \end{pmatrix} .\Delta u(t) + \begin{pmatrix} 0 \\ -28033 \end{pmatrix} .c_0(t)$$
$$\begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} \tag{4.4}$$

Which gives the following transfer function:

$$H(s) = \frac{\Delta\Omega_h(s)}{\Delta U(s)} = \frac{4,288.10^6}{(s + 481.4).(s + 20.16)} \tag{4.5}$$

4.2 Indexes of performance

The number of control update used for the synchronous controller and the event-based controller is an intuitive criteria to compare the two approaches. This index shows the reduction of the computational cost with the event-based approach. We also compute the power consumption of the system with the two approaches with the following formula :

$$P = \frac{1}{T} \int_0^T u(t).i(t).V_{bat}.dt \tag{4.6}$$

with P the electrical power consumption in Watt, T the simulation time, $u(t)$ the applied duty-cycle, $i(t)$ the current and V_{bat} the power supply.

In order to compare the performance of the system, we will use another index. This index is presented in [1] and called the Integral Absolute Error (IAE):

$$IAE = \int_0^\infty |\omega_{ref}(t) - \omega_h(t)|.dt \tag{4.7}$$

This index shows how far is the output ω_h compared to the given reference ω_{ref}. Thus the lower the IAE index, the better is the control.

4.3 Event-based PI controller

4.3.1 Method

To synthesize the PI controller, we choose to apply a dominant pole compensation. The equation of the PI controller is: $C(s) = K_p.(1 + \frac{1}{T_i.s})$. We approximate the model $H(s)$ with a first order and neglect the electrical time constant introduced by the inductance L. We obtain the following transfer function : $H_{order1}(s) = \frac{K}{1+\tau s}$ with $K = \frac{V_{bat}.K_t.M}{2.D.\omega_{h_{eq}}.R+K_e.K_t.M^2}$ and $\tau = \frac{R.J.M^2}{2.D.\omega_{h_{eq}}.R+K_e.K_t.M^2}$. With the pole compensation, we have $T_i = \tau = 0.0515$. Then we adjust the gain K_p to obtain the same settling time at 95% for the open loop and the closed loop. For a settling time at 95% equals to 0.280s, we have $K_p = 0.0021$. Furthermore we add a feed-forward gain to the PI controller to bring the system at the equilibrium for a given reference. Then the feed-forward gain u_{eq} depends on the reference velocity: we have $u_{eq}(t) = \frac{1}{V_{bat}}(R.\frac{D}{K_t.M}.\omega_{ref}^2(t) + K_e.M.\omega_{ref}(t))$ with ω_{ref} the reference velocity in rad/s.

We discretize the controller by using the backward approximation in order to not degrade the signal when we use the event-based control as it is explain in [2]. The sampling time T_s is chosen to $T_s = 1ms$. Then the control computed by the discretized PI is given by:

$$\begin{cases} u_{éq}(k) = \frac{1}{V_{bat}}(R.\frac{D}{K_t.M}.\omega_{ref}^2(k) + K_e.M.\omega_{ref}(k)) \\ u_i(k) = u_i(k-1) + \frac{K_p}{T_i}.T_s.(\omega_{ref}(k) - \omega_h(k)) \\ u_p(k) = K_p.(\omega_{ref}(k) - \omega_h(k)) \\ u_{PI}(k) = u_{eq}(k) + u_i(k) + u_p(k) \end{cases} \tag{4.8}$$

We propose to adapt the event-based method described in [3] and apply it with a PI controller. The event-based control is then described by $u_{event}(k) = \begin{cases} u_{PI}(k)\, if\, x(k) \notin \beta \\ u_{event}(k-1)\, if\, |x(k) \in \beta \end{cases}$. The event-based control is computed in the same way as the synchronous PI when $x_k \notin \beta$ but it keeps its previous value if $x(k) \in \beta$. Now we need to define the performance set (set β). The set is determined according to the current i and the propeller's angular velocity ω_h such that $\beta := \begin{cases} \omega_{ref} - \omega_{tolerance} < \omega_h < \omega_{ref} + \omega_{tolerance} \\ i_{eq} - i_{tolerance} < i < i_{eq} + i_{tolerance} \end{cases}$

We choose the tolerance relative to the velocity $\omega_{tolerance}$ equals to 10 rad/s.i_{eq} corresponds to the value of the current at the equilibrium, $i_{eq} = 1.3A$. The tolerance relative to the current is chosen such that the controller can ensure a speed reference tracking from 200 rad/s to 260 rad/s, thereby we set $i_{tolerance} = 1.3A$.

4.3.2 Intuitive stability

The stability is an important aspect of the control theory but it is difficult to show the stability of a system with an event-based approach based on a performance set. However the stability of the event-based control can be intuitive in the way that the performance sets are defined around the equilibriums. When the states are inside the set β, the control is held to the value computed just before entering the set. The held control was then computed to ensure the convergence of the states toward the

set. When the states lies outside the set, the controller is updating the control at a constant rate with a stabilizing control. This ensures the stability of the system in a intuitive way.

4.3.3 Speed reference tracking

The simulations are performed on the modeled system in 4.4. The system is controlled with two controllers: the synchronous controller and its event-based equivalent described in the previous section. We propose to perform a speed reference tracking: the reference is set to the equilibrium $\omega_{h_{eq}} = 200 rad/s$ at time t=0s and change to 230 rad/s and 260rad/s at time t=1s and t=2s. The propeller speed can be seen on the figure 4.2:

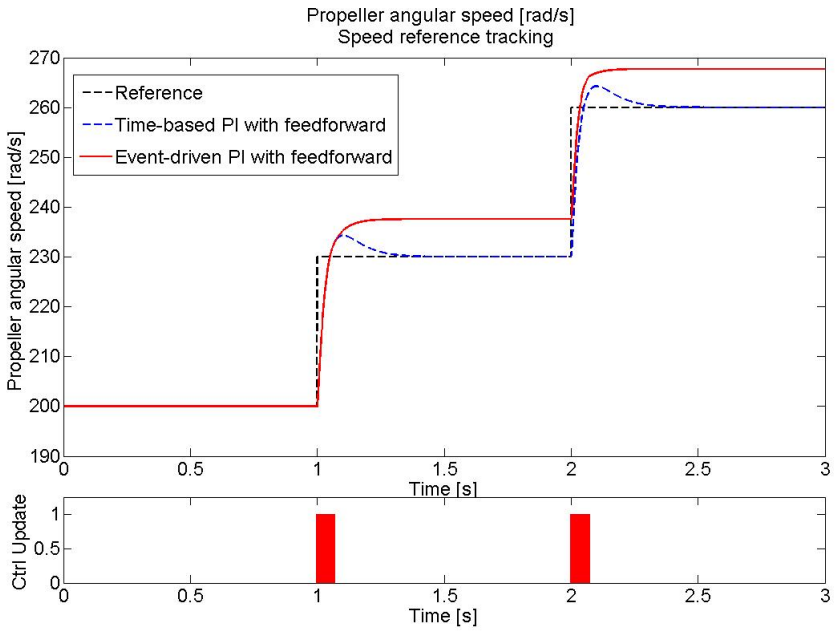

Figure 4.2: Speed reference tracking: comparison between the synchronous PI and the event-based PI. The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

One can see on the figure 4.2 that the tracking reference is performed with the event-based control with a static error inferior to the defined tolerance of 10 rad/s. The control update are only performed during the change of reference at the time t=1s and at the time t=2s.

The control and the current are shown on the figure 4.3. We can see that the control is only updated during the change of reference. Then it is held until the next change of reference. The IAE index for the synchronous controller is only 2.4 when it is 15.1 in the case of the event-based. In the case of the synchronous controller the number of control update is 3000 when it is only 144 with the event-based control that is to say a reduction of 95.2% of the control updates. The power consumption is 11.87W with the event-based control and 11.08W with the synchronous control.

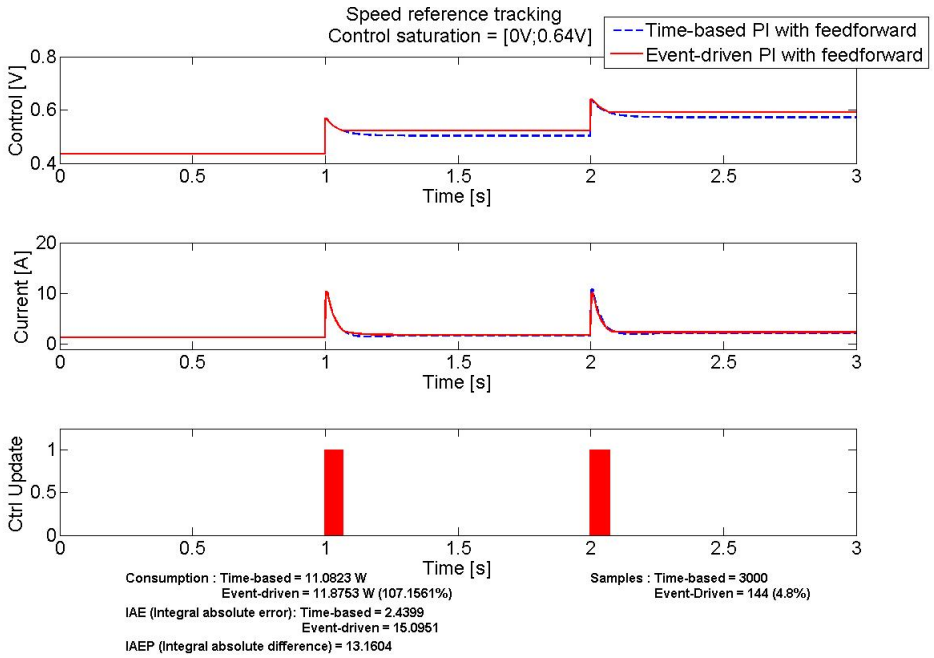

Figure 4.3: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous PI and the event-based PI. The bottom plot shows the control update in the case of the event-based PI.

The reduction of the control update is then ensured however the power consumption is still higher with a event-based control than with the synchronous control.

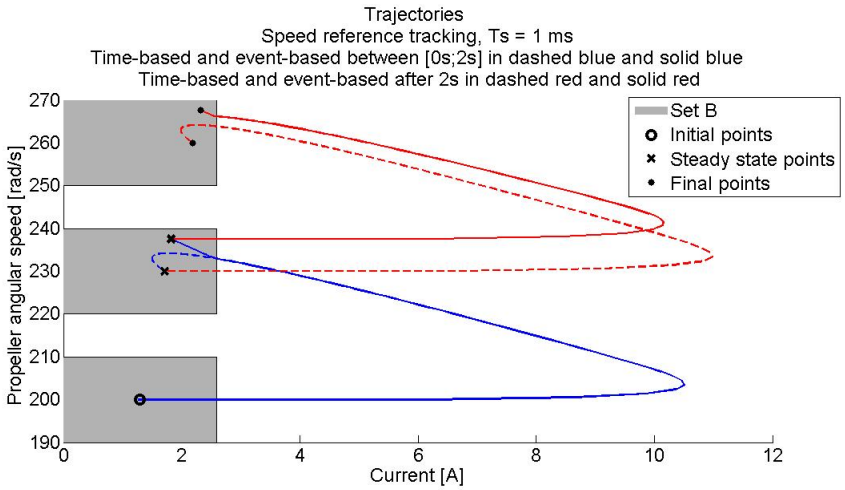

Figure 4.4: Speed reference tracking: comparison between the synchronous PI and the event-based PI. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in dashed and solid lines.

On the figure 4.4 the phase plane of the states and the performance sets (sets β) are plotted. There are 3 different sets because the speed reference changes between 3 different values. Once the reference changes, the set β is also changed by definition

of the set. At every reference changes, the states denoted by the current and the propeller's velocity remain outside the set β. Then the update of the control occurs at every sample time T_s, which will lead the output to converge toward the reference. As soon as the states reach the set β, the control is held to its previous value and it is not updated until the states leave the set. This event-based strategy ensures the convergence of the trajectory toward the performance sets.

4.3.4 Disturbance rejection

Now we simulate the system when a constant disturbance is applied to the system. The added disturbance is a constant torque $c_0(t)$ as defined in the equation 4.4. The system is set to the equilibrium speed at time t=0s i.e. $\omega_{h_{eq}} = 200 rad/s$, the disturbance is added for $t \in [1s; 2s]$ with an amplitude of -0.01 N.m. The propeller velocity is shown on the figure 4.5:

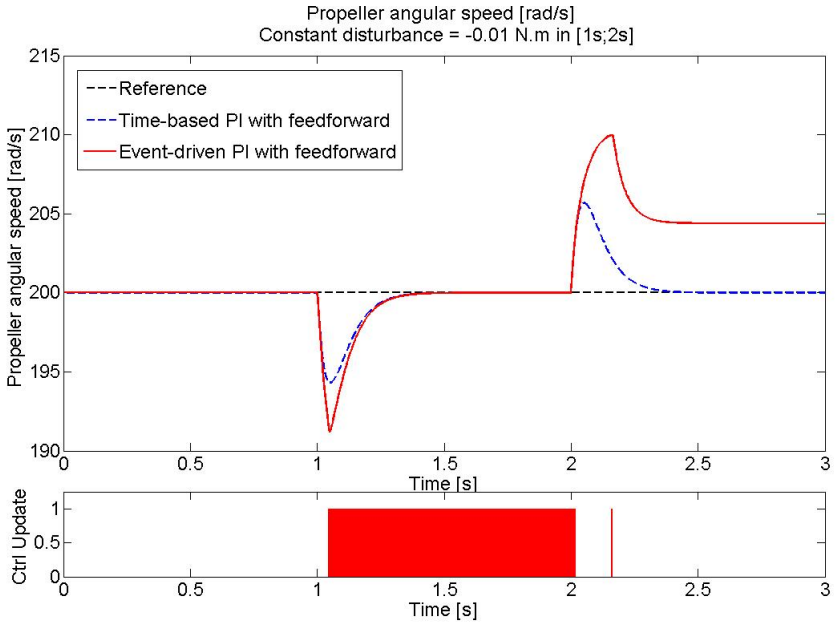

Figure 4.5: Disturbance rejection: comparison between the synchronous PI and the event-based PI. The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

One can see on the figure 4.5 that the disturbance is rejected with the event-based approach. When the disturbance is applied, the event-based controller updates constantly the control. This allowing the system to reject the disturbance. When the disturbance is removed, the propeller speed remains close to the reference with a static error less than the defined tolerance of 10 rad/s.

The control and the current are shown on the figure 4.6:

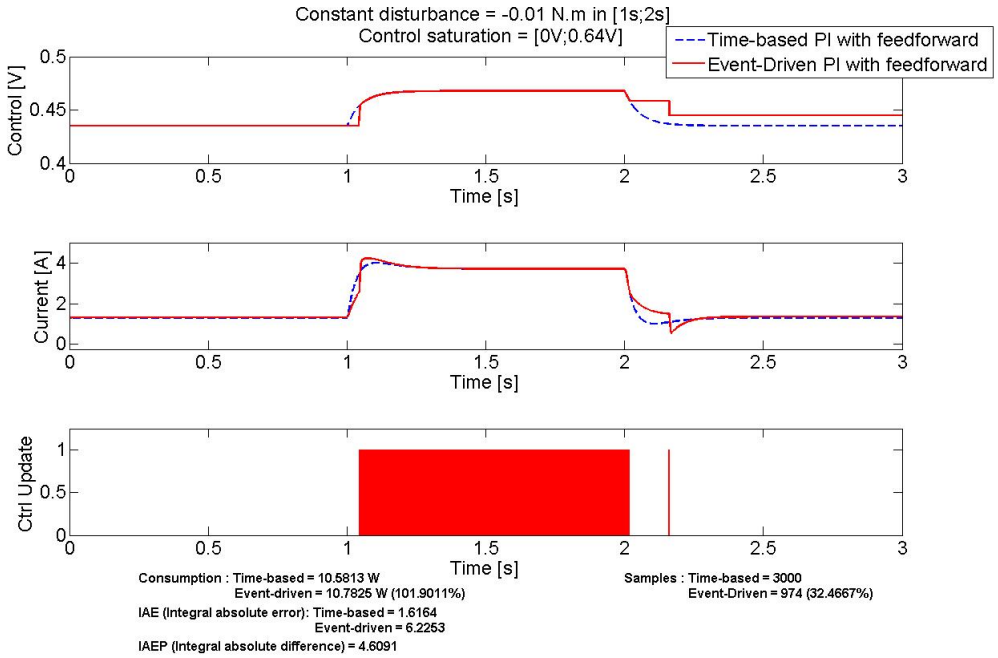

Figure 4.6: Disturbance rejection: The top plot and middle plot show respectively the control and the current with the synchronous PI and the event-based PI. The bottom plot shows the control update in the case of the event-based PI.

We can see on the figure 4.6 that the control is constantly updated when the disturbance is applied. At time t=2.2s one update is performed, the control is then held to the updated value. For t>2.2s, the control value differs from the control at the equilibrium that is why the velocity can not reach the equilibrium but the static error remains lower than the tolerance of 10 rad/s. The IAE index for the synchronous controller is 1.6 when it is 6.2 in the case of the event-based. The difference can be explained by the fact that the propeller velocity does not come back to the equilibrium when the disturbance is removed in the case of the event-based approach. This phenomenon impacts on the power consumption since a higher velocity induces a higher voltage across the motor. That is why the power consumption with the event-based controller is higher (10,8W) than the time-based controller (10,6W). In the case of the synchronous controller the number of control update is 3000 when it is only 974 with the event-based control. It is equivalent of a reduction of 67,5% of the control update.

The phase plane of the state and the performance set are shown on the figure 4.7. The added disturbance allows the trajectory to go out the set β. This implies that the event-based controller have to update the control constantly as it is shown on the figure 4.5 and 4.6. When the disturbance is removed, the trajectory converges toward the set β. However when it reaches the set, the held control does not allow the trajectory to stay inside the set. This is why we can observe a control update at time t=2.2s on the figures 4.5 and 4.6. This last held control allows the trajectory to reach the set and remain inside until the end of the simulation.

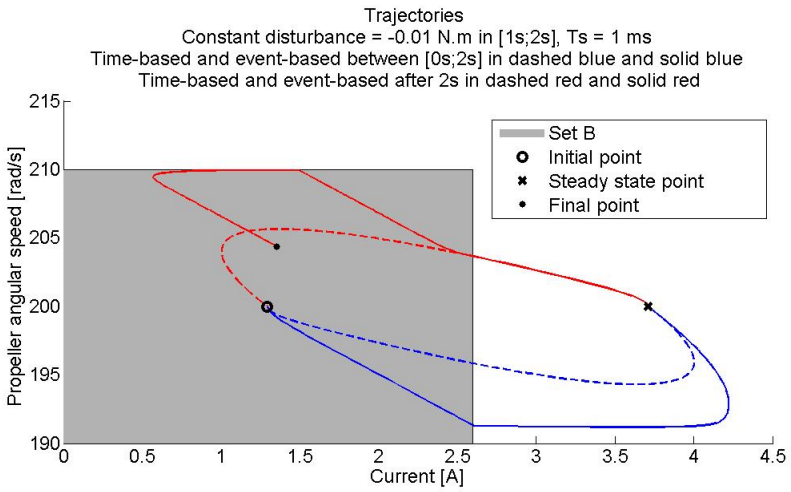

Figure 4.7: Disturbance rejection: comparison between the synchronous PI and the event-based PI. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in dashed and solid lines.

4.4 State feedback controller

4.4.1 Method

We propose to apply a linear quadratic regulator (LQ) in order to minimize the power consumption of the system. The aim of the LQ control is to attenuate the current peaks that one can see with the PI controller. Indeed these peaks increase the power consumption of the motor. The state-space representation has been given by: $\begin{cases} \Delta\dot{x}(t) = A.\Delta x(t) + B.\Delta u(t) + E.c_0(t) \\ \Delta\omega_h(t) = \underbrace{\begin{pmatrix} 0 & 1 \end{pmatrix}}_{C_1}.\Delta x(t) \end{cases}$ with $\Delta x(t) = x(t) - x_{eq}$. We add an integral action in the state feedback control $\Delta z(t)$ in order to have a reference tracking for the propeller's angular velocity: $\Delta\dot{z}(t) = \Delta\omega_{ref}(t) - \Delta\omega_h(t) = \Delta\omega_{ref}(t) - C_1.\Delta x(t)$. The augmented state-space representation is given by: $\begin{pmatrix} \Delta\dot{x}(t) \\ \Delta\dot{z}(t) \end{pmatrix} = \begin{pmatrix} A & 0 \\ -C_1 & 0 \end{pmatrix} . \begin{pmatrix} \Delta x(t) \\ \Delta z(t) \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} .\Delta u(t) + \begin{pmatrix} E \\ 0 \end{pmatrix} .c_0(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} .\Delta\omega_{ref}(t)$. We denote the augmented state by: $\Delta x_a(t) = \begin{pmatrix} \Delta x(t) & \Delta z(t) \end{pmatrix}^T$.

The LQ controller minimizes the quadratic cost function: $J = \int_0^\infty \Delta x_a(t)^T.Q.\Delta x_a(t) + \Delta u(t)^T.R.\Delta u(t).dt$ where Q and R are respectively a 3-by-3 matrix and a scalar that we have to define. We use the Matlab function *lqrd* that computes a discretized state feedback K_d such that $\Delta u(k) = -K_d.\Delta x_a(k)$ minimizes the cost J. The weights Q and R have been specified such that the current peaks of the DC motor are minimized and such that the rise time is kept to an acceptable value. The matrices are then: $Q = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 1.9 & 0 \\ 0 & 0 & 435 \end{bmatrix}$ and $R = 150000$ which gives a gain $K = [0.0048; 0.0032; -0.0486]$. The poles of the discretized closed loop with a sam-

pling time $T_s = 1ms$ are: $[0.4871; 0.9830; 0.9815]$.

The event-based control is computed by: $\Delta u_{event}(k) = \begin{cases} -K_d.\Delta x_a(k)) \, if \, x(k) \notin \beta \\ \Delta u_{event}(k-1) \, if \, |x(k) \in \beta \end{cases}$.
The set β is defined in the same way as the PI control with a tolerance on propeller's angular velocity equals to 10 rad/s and the tolerance relative to the current of 1.3A.

4.4.2 Reference tracking

We perform the reference tracking on the modeled system defined in 4.4 with the synchronous LQ controller and its event-based equivalent. The reference signal is set to the same value as with the PI controller. The starting set point is 200 rad/s and it is changed to 230 rad/s and 260 rad/s at time t=1s and t=2s. The propeller speed with the two LQ controllers are shown on the figure 4.8:

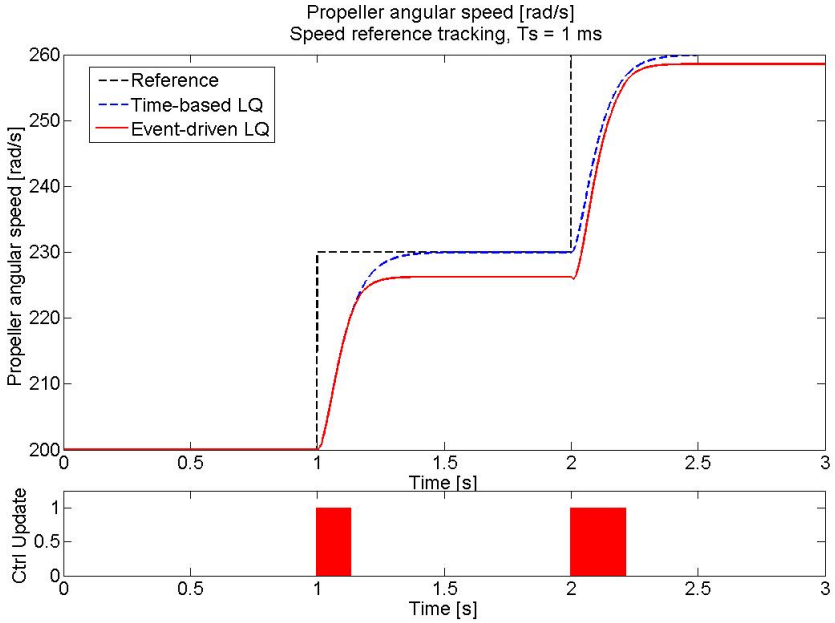

Figure 4.8: Speed reference tracking: comparison between the synchronous LQ control and the event-based LQ control. The figure shows the evolution of the propeller speed with the two types of control and the update of the event-based control: the update is performed when the indicator is at 1.

As expected the event-based approach succeeds to perform the reference tracking with a static error less than 10 rad/s. The controls are only performed during the changes of reference.

The figure 4.9 shows the control and the current with the two approaches. The IAE index for the time-based controller is 6.83 while it is 11.3 for the event-based controller. The difference is explained by the error static that we can tolerate with the event-based control. The number of control update are 3000 with the synchronous regulator and only 349 with the event-based LQ regulator. This is a reduction of the control update of 88.3%. The power consumption is 10.4W with the event-based LQ control and 10.6W with the synchronous LQ regulator. The reduction of the control

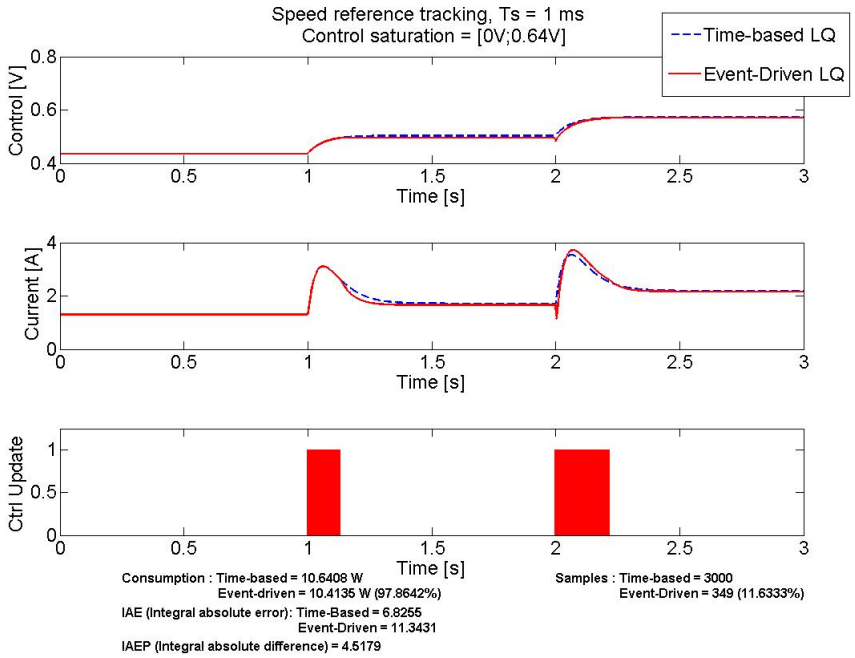

Figure 4.9: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous LQ control and the event-based LQ control. The bottom plot shows the control update in the case of the event-based LQ control.

update is still ensured and the power consumptions of the system are almost the same with the two approaches. As expected the LQ control minimizes the current peaks and thus the consumption of the system.

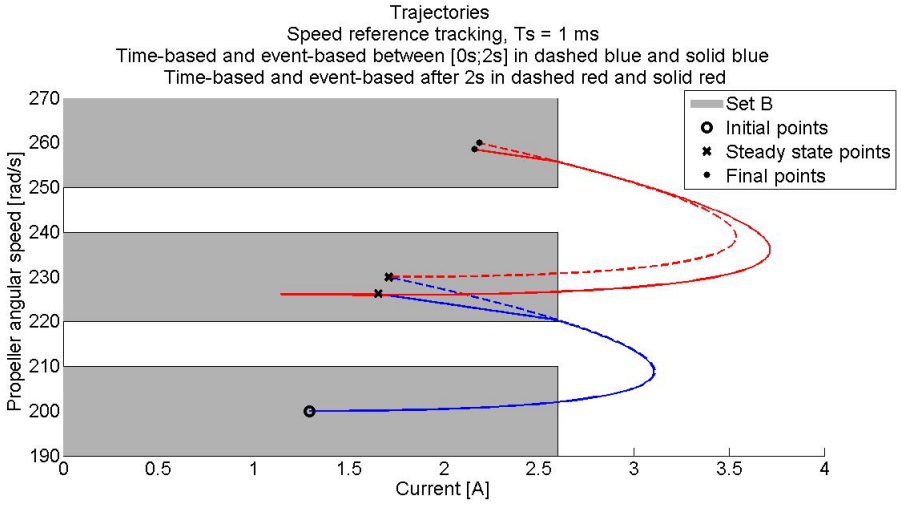

Figure 4.10: Speed reference tracking: comparison between the synchronous LQ control and the event-based LQ control. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in dashed and solid lines.

On the figure 4.10 the phase plane of the state and the performance sets (sets β)

are plotted. The convergence of the trajectories are still ensured with event-based LQ control.

However one can observe a control peak at the time t=2s in the case of the event-based, this is due of the event-based approach when it is applied to a state-feedback control. Indeed the discretized feedback control with the sampling time T_s is given by : $\Delta u(k) = -K_d.\Delta x_a(k) = -K_p.\Delta x(k) - K_i.\Delta z(k) = u_p(k) + u_i(k)$ where $u_p(k) = -K_p.\Delta x(k)$ and $u_i(k) = -K_i.\Delta z(k)$ and K_p and K_i denote respectively the state feedback for the state $\Delta x(k)$ and $\Delta z(k)$. By using a backward approximation, the control is computed by:

$$\begin{cases} u_p(k) = -K_p.\Delta x(k) \\ u_i(k) = u_i(k-1) - K_i.T_s\Delta z(k) \\ \Delta u(k) = u_p(k) + u_i(k) \end{cases} \tag{4.9}$$

When the trajectory enters into the set β at the instant $k = k_1$, the part control $u_p(k_1)$ is computed from the states $\Delta x(k_1)$ and it will be held. The trajectory will converge to a value inside the set β corresponding to the control input $\Delta u(k_1)$. When the reference changes at the instant $k = k_2$, the control will be updated and $u_p(k_2)$ is computed by $u_p(k_2) = -K_p.\Delta x(k_2)$. Since $\Delta x(k_1)$ and $\Delta x(k_2)$ can be far away depending of the size of the set β, the resulting control $u_p(k_2)$ is different from $u_p(k_1)$. Then the update of the control can result in a brutal change of the control and causes the observed peak in the figure 4.9.

4.4.3 Disturbance rejection

We apply a constant torque $c_0(t)$ with the LQ controller to disturb the system. The disturbance is applied in the same way as for the PI controller i.e. with an amplitude of -0.01 N.m for $t \in [1s; 2s]$ when the system is at its equilibrium. The propeller velocity is shown on the figure 4.11:

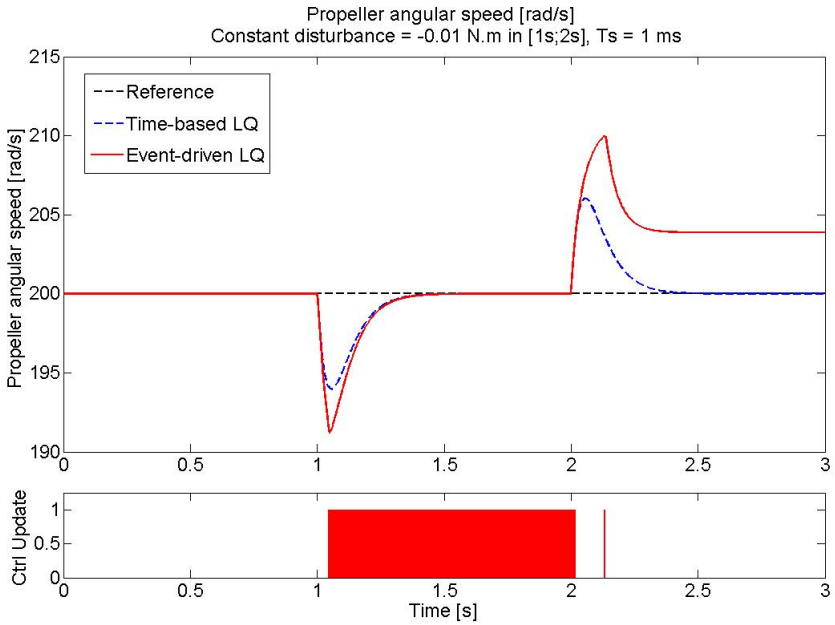

Figure 4.11: Disturbance rejection: comparison between the synchronous LQ control and the event-based LQ control. The figure shows the evolution of the propeller speed with the two types of control and the update of the event-based control: the update is performed when the indicator is at 1.

The disturbance is rejected with the event-based LQ controller. The control is only updated when the disturbance is applied.

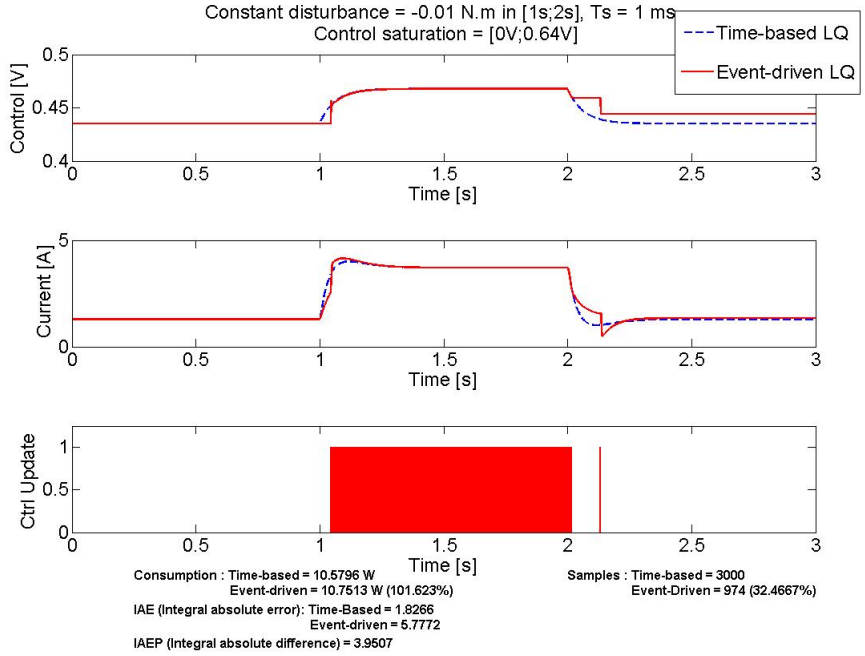

Figure 4.12: Disturbance rejection: The top plot and middle plot show respectively the control and the current with the synchronous LQ control and the event-based LQ control. The bottom plot shows the control update in the case of the event-based LQ control.

The control and the current are shown on the figure 4.12. As one can see the control is constantly updated when the disturbance occurs. Then the control is held when the disturbance is removed. The IAE index are respectively 1.8 and 5.8 for the time-based and the event-based control. With the LQ controller the number of update is 974 instead of 3000 with the synchronous LQ controller which is a reduction of 32.5%. Concerning the power consumption, the synchronous controller consumes less energy (10.6W) than the event-based control (10.8%).

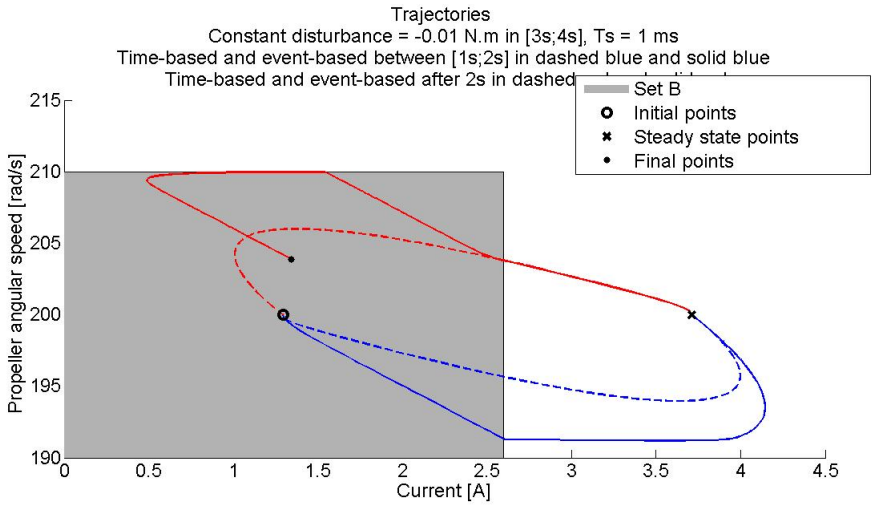

Figure 4.13: Disturbance rejection: comparison between the synchronous LQ control and the event-based LQ control. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in dashed and solid lines.

The trajectories are shown on the figure 4.13. When the disturbance is applied, it forces the trajectories to go out the set β. By doing this the control is automatically updated at a constant rate. This allows the controller to reject the disturbance. When the disturbance is removed the controller still continues to update at a constant rate until the trajectory reaches the set β. When the states remain inside the set, the control is held to its previous value and the event-based trajectory is stabilized inside the set.

Chapter 5

Experimental work and results

5.1 Propeller's speed control of a direct current motor

We propose to experiment the event-based approach with a real miniature DC motor. To do this, we need to write the specifications of the system to build a test bench. The aim is to realize a test bench with a DC motor, a power stage, a power supply and a speed sensor to catch the propeller's angular velocity.

1. The propeller is fixed to the motor. The propeller's velocity will be regulated through the power stage. The duty cycle of a Pulse Width Modulation (PWM) is sent to the power stage. The duty cycle is generated by the dSpace tool.

2. The DC motor will be regulated with a synchronous control and with a event-based control.

3. The aim is to regulate the propeller's velocity and to measure the power consumption of the motor.

4. A second motor is placed in front of the regulated motor in order to disturb the propeller's angular velocity. The second motor puts a gust of wind which disturbs the propeller's velocity of the regulated motor.

The block diagram of the test bench is represented on the figure 5.1:

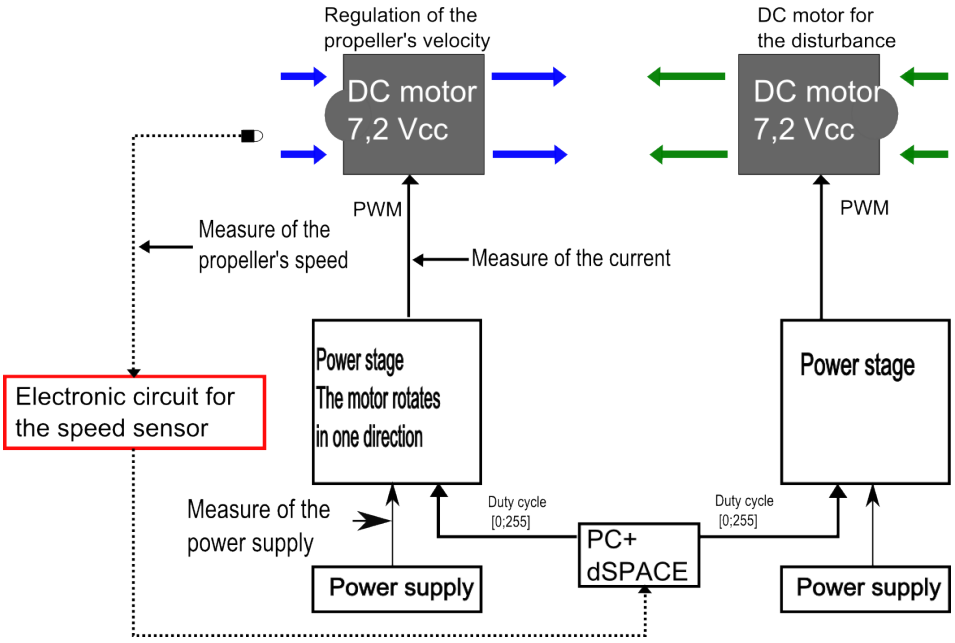

Figure 5.1: Block diagram of the test bench

The test bench uses two DC motors GWS EDF-50 (`www.gws.com.tw`, reference: DC CN12-RXC).

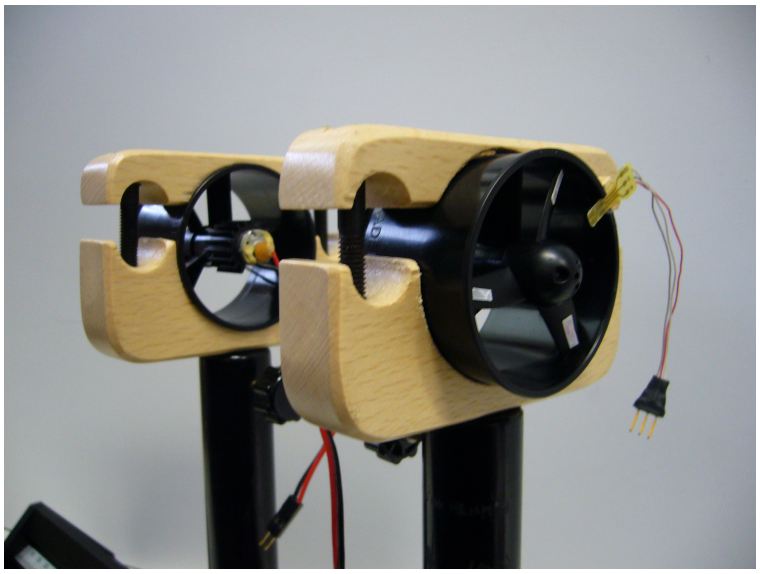

Figure 5.2: Miniature DC motors used for the test bench

In order to analyze the performance given by the event-based control, we need to measure several physical quantities:

Measured quantities	Remarks
Propeller's angular velocity	Put reflective stickers on each propeller (3M paper) and place a reflective sensor (LED + photo-transistor) in order to count each time a propeller is passing in front of the sensor.
Current	Use of a shunt
Voltage of the supply	
Voltage of the DC motor	Known since it is the control signal

Table 5.1: Measured quantities

Once the test bench has been built, we need to acquire the different signals. The speed sensor delivers a pulse every time a propeller is in front of the sensor. Thus we use an edge detector and count the pulse sent by the sensor. The motor has three propellers, so we count four pulses for one full revolution of the motor. From the time between the extreme pulses, we can deduce the propeller's angular velocity.

The sampling rate is determined from the maximum propeller's angular velocity. At this speed, the pulses given by the sensor have a minimum length of 140 µs. In order to catch all the pulses with a sufficient accuracy, we set the sampling period at 50 µs. This gives the sampling frequency $F_s = 20\,kHz$.

The current is measured by taking the voltages across a shunt R_{shunt} of 0.1Ω. The voltages are filtered with an anti-aliasing filter of order two with a cutoff frequency of $338Hz$. The voltages measured can be seen on the figure 5.3.

The current is deduced from the measured voltages V_1 and V_2. To exploit the current we need to filter it with a Kalman filter. The Kalman filter has the advantage to not induce a delay on the measure compared to a low pass filter. To apply it we need to define the process noise covariance Q and the measurement noise covariance R. We estimated Q by calculating the covariance of the measured signal and set R to get an exploitable estimate of the measured current. The values are then equal to $Q = 0.0034$ and $R = 20$. The measured current and the estimated current using the Kalman filter are shown on the figure 5.4.

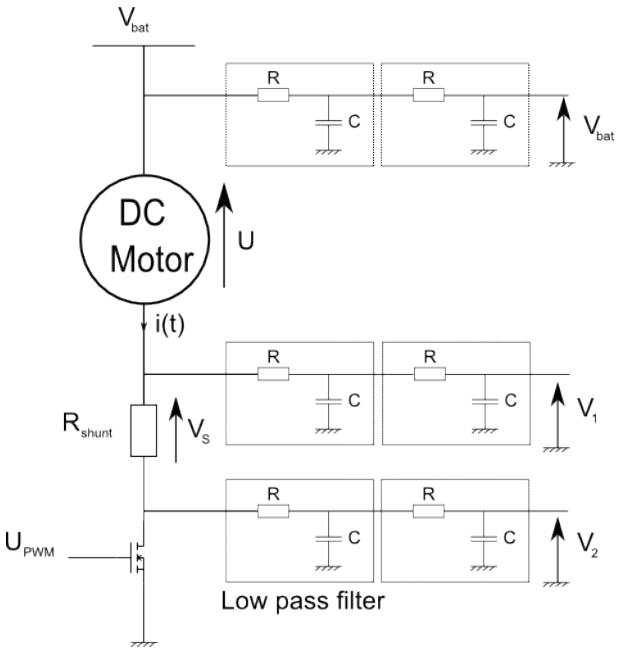

Figure 5.3: Electric diagram of the DC motor. The current is deduced by measuring the voltages V_1 and V_2. The voltages are filtered with a low-pass filter. The voltage U_{PWM} controls the transistor which controls the velocity of the motor.

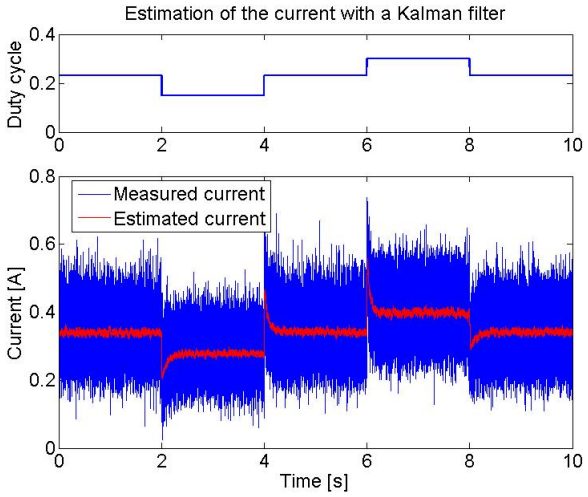

Figure 5.4: Estimation of the current with a Kalman filter: the top plot is the duty-cycle applied in open-loop and the bottom plot shows the measured current in blue and the estimated current with a Kalman filter in red.

As one can see on the figure 5.4, the measured current is highly noisy and it is not possible to use it directly. The estimated current can now be used to compute the power consumption and to identify the system dynamic.

Now we need to determine the linear region where the DC motor will be used. To obtain this region, we measure the propeller's velocity for different values of the duty cycle and determine where the transfer function from the duty cycle to the

propeller's velocity is linear. From this analysis, the linear region corresponds of the region where the duty cycle varies between 0.1 and 0.6.

5.2 System identification of the DC motor

In order to implement a PI controller and a state feedback control with the LQ control, we need to identify a state-space representation of the system with the current and the velocity of the propeller as the states of the state-space representation. To identify the process we use the System Identification Toolbox given by Matlab. In order to have a good approximation we use two sets of data: one set is used to identify the system and obtain a model while the second set is used for the comparison of the model and for the validation of the identification.

Firstly we identify the transfer function $H_1(s) = \frac{\omega_h(s)}{u(s)}$ between the duty cycle and the propeller's velocity. This transfer function can be approximate with a first order transfer function since the dynamic of the current is faster than the dynamic of the propeller's speed. We set the duty cycle to three successive values to identify the system [0.2; 0.3; 0.5]. We use the ARX (Auto Regressive model with eXternal inputs) method to identify the process since this method is the easiest way to identify a first order system. By doing this, the discretized model is given by:

$$A(q).\omega_h(k) = B(q).u(k) + e(k) \tag{5.1}$$

with $A(q) = 1 - 0.9989.q^{-1}$, $B(q) = 3.853.q^{-110}$ and $e(t)$ the white noise. We convert the obtained model into the continuous time domain by using a zero-order hold method in order to get the transfer function $H_1(s)$ and its state-space model:

$$\frac{d\omega_h(t)}{dt} = -22.65.\omega_h(t) + 7.71.10^4.u(t) \tag{5.2}$$

Now we need to identify the transfer function $H_2(s) = \frac{i(s)}{u(s)}$ from the duty cycle to the current. To do this, we recall the equation of the DC motor: $u(t).V_{bat} = R.i(t) + K_e.M.\omega_h(t) + L\frac{di(t)}{dt}$. Clearly, we can not use directly the input signal $u(t)$ and the output $i(t)$ to identify the model since the propeller's velocity appears in the equation. To identify the dynamic of the current, we create another input signal $u_2(t) = u(t).V_{bat} - K_e.M.\omega_h(t)$ that will be used with the identification toolbox. With the new input $u_2(t)$, the equation becomes : $u_2(t) = R.i(t) + L\frac{di(t)}{dt}$. From this, we can identify the transfer function $H_{2'}(s) = \frac{i(s)}{u_2(s)}$ from the created input $u_2(t)$ to the current $i(t)$ as a first order transfer function. To create the new input $u_2(t)$, we need to know the values of the parameters V_{bat} and $K_e.M$. The supply voltage V_{bat} used to operate the DC motor is already known and set to $7,2V$. The armature resistance R has been identify during the conception of the test bench and measured at $0,43\Omega$. From the value of R, we can deduce the value of $K_e.M$ by applying a constant control $u(t)$ and measure the current $i(t)$ and the propeller's velocity $\omega_h(t)$ during the steady-state since the term $L\frac{di(t)}{dt}$ equals zero. Then we find $K_e.M = 0.0016$. From these values, we create the artificial input signal $u_2(t)$ and identify the transfer function $H_{2'}(s)$. With the ARX method, the discretized result with the sampling frequency $F_s = 20kHz$ is given by:

$$A(q).i(k) = B(q).u_2(k) + e(k) \tag{5.3}$$

with $A(q) = 1 - 0.9908.q^{-1}$, $B(q) = 0.003538.q^{-20}$ and $e(t)$ the white noise. We neglect the delay and convert the model to a continuous transfer function by using the zero-order hold method in order to obtain $H_{2'}(s)$:

$$H_{2'}(s) = \frac{i(s)}{u_2(s)} = \frac{0.3855}{1 + 0.0054.s} \tag{5.4}$$

From the equation 5.4 and the expression of $u_2(t)$, we can deduce the transfer function $H_2(s)$:

$$\frac{di(t)}{dt} = -185.1852.i(t) + 71.3889.u_2(t)$$

$$\frac{di(t)}{dt} = -185.1852.i(t) + 71.3889.\left(7.2.u(t) - 0.0016.\omega_h(t)\right)$$

$$\Leftrightarrow \frac{di(t)}{dt} = -185.1852.i(t) - 0.1142.\omega_h(t) + 514.u(t) \tag{5.5}$$

From the equation 5.2 and 5.5, we build the state-space representation of the identified system:

$$\begin{cases} \begin{pmatrix} \frac{d\Delta i(t)}{dt} \\ \frac{d\Delta\omega_h(t)}{dt} \end{pmatrix} = \underbrace{\begin{pmatrix} -185.1852 & -0.1142 \\ 0 & -22.65 \end{pmatrix}}_{A} . \begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} + \underbrace{\begin{pmatrix} 514 \\ 7.71.10^4 \end{pmatrix}}_{B} .\Delta u(t) \\ \begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C} . \begin{pmatrix} \Delta i(t) \\ \Delta\omega_h(t) \end{pmatrix} \end{cases} \tag{5.6}$$

To validate the model, we build a Simulink model from the state-space representation given by 5.6 and apply the same control input signal used with the validation data. Then we can compare the output signals given by the real process and our model. These signals are shown in the figure 5.5:

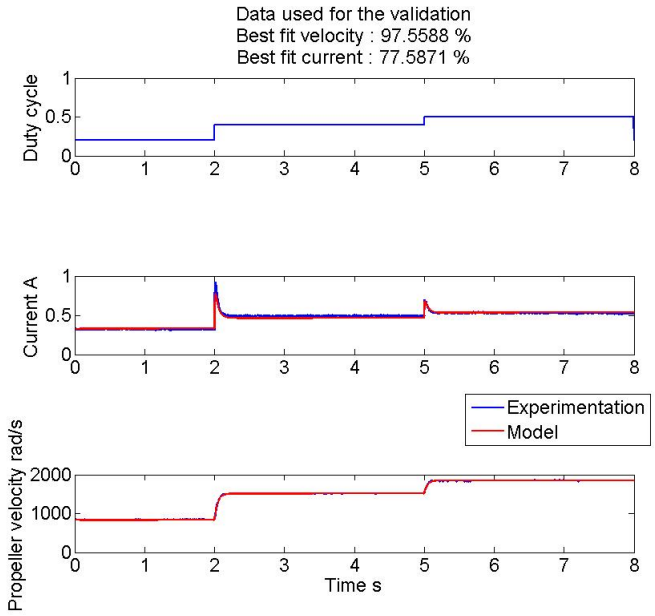

Figure 5.5: Comparison between the measured outputs and the simulated model outputs with the same input. The measured outputs and the simulated model outputs are respectively plotted in blue and red. The best fit for the velocity and the current are respectively equal to 97.6% and 77.6%

As one can see the model reproduces accurately the behavior of the system. Furthermore, we compute the percentage of the output that the model reproduces (so-called Best Fit). It is computed by:

$$Best\ fit = 100. \left(1 - \frac{|y - \hat{y}|}{|y - \overline{y}|}\right) \tag{5.7}$$

where y is the measured output, $\hat{y}$ is the simulated model output and $\bar{y}$ is the mean of y. When applying the formula 5.7, the best fit corresponding to the propeller's velocity is 97,6% and 77,6% for the current. These values correspond to the maximum best fit that we obtained for a first order transfer function with the ARX method. According to the best fit values, we will keep the model defined in 5.6, the model uncertainty will be reduce during the design of the controller.

5.3 Regulation of the DC motor without velocity sensor

We propose to regulate the DC motor without measuring the velocity of the propeller but by estimating its velocity. This method has two main advantages:

- we avoid the use of a sensor to measure the velocity of the propeller. The avoidance of a speed sensor is a real advantage concerning the energy consumption since the sensor was supplied by a voltage of 5V.

- we can significantly reduce the sampling frequency F_s since the high value value of the frequency was set to measure correctly the velocity of the propeller. By reducing the sampling frequency, we also reduce the energy consumption and the computational cost.

By estimating the velocity of the propeller, the energy consumption and the computational cost will be greatly reduced. This fact can play an important role in the case of an embedded control system where the consumption cost needs to be as small as possible.

The estimation of the velocity is already presented in [10] and we recall the main principles. We recall the equation 4.2 of DC motor in the Laplace domain with the measured values shown on the figure 5.3:

$$\begin{array}{c} U(s) = K_e.M.\Omega_h(s) + (L.s + R).I(s) \\ J.M.\Omega_h(s) = K_t.I(s) - \frac{D}{M}.\Omega_h(s) - C_o(s) \end{array} \tag{5.8}$$

The voltage $U(s)$ and the current $I(s)$ are measured by taking the voltages V_{bat}, V_1 and $V_2$with R_{shunt} the shunt resistance:

$$\begin{array}{c} U(s) = V_{bat} - V_1. \\ I(s) = \frac{V_1 - V_2}{R_s} = \frac{V_s}{R_{shunt}} \end{array} \tag{5.9}$$

The dynamical constant of the velocity τ_{meca} is very large compare to the dynamical constant of the current τ_{elec}. Then we can measure the armature current during the steady-state to estimate the velocity of the propeller during its transient. To do this, we need to measure the voltage V_{bat} of the power supply and the voltages across the shunt V_1 and V_2 as shown on the figure 5.3. Indeed the voltage U across the DC motor is defined by $U(s) = K_e.\Omega_h(s) + (L.s + R).I(s)$ and we can measure it since $U(t) = V_{bat}(t) - V_1(t)$. Moreover the current is measured and expressed in 5.9.

Then we can obtain the estimated velocity in volt $\Omega_{E_{[V]}}$ with:

$$\begin{array}{c} \Omega_{E_{[V]}}(s) = U(s) - V_s(s) = U(s) - R_s.I(s) \\ = K_e.M.\Omega_h(s) + (R - R_s).I(s) \end{array} \tag{5.10}$$

From the equation 5.9 and by neglecting the drag coefficient D and assuming no disturbance $C_o(s)$, we can express the current by:

$$I(s) = \frac{J.s.M.\Omega_h(s)}{K_t} \tag{5.11}$$

From the equations 5.10 and 5.11 we can now express the estimated velocity in volt by:

$$\Omega_{E_{[V]}}(s) = K_e.M.[1 + \tau.s].\Omega_h(s) \; with \; \tau = \frac{J.(R - R_{shunt})}{K_t K_e} \tag{5.12}$$

Then by measuring the voltages U and V_s we can have an image of the velocity in volt. To obtain an estimated of the velocity in rad/s, we need to multiply $\Omega_{E_{[V]}}$ by the factor $\frac{1}{K_e.M}$. Moreover the product $K_e.M$ has been identify during the identification of the motor and equal to $K_e.M = 0.0016$. We can now estimate the velocity of the

propeller and compare it with the measured velocity. The result is shown on the figure 5.6:

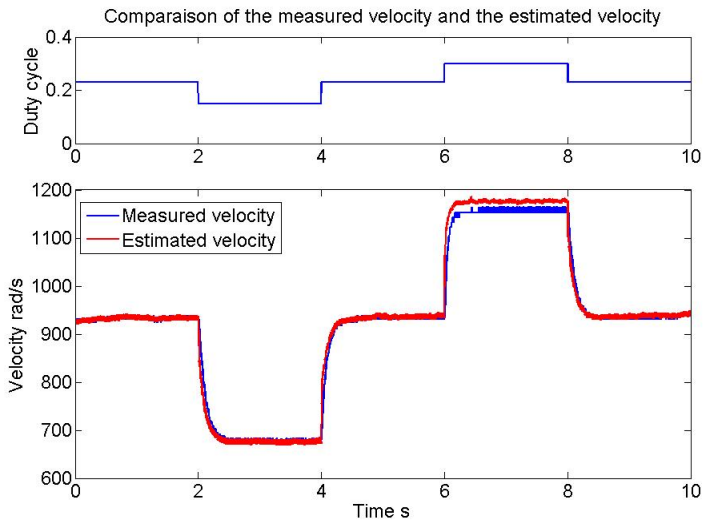

Figure 5.6: Comparison between the measured velocity and the estimated velocity: the top plot is the duty-cycle applied in open-loop and the bottom plot shows the measured velocity in blue and the estimated velocity in red.

One can observe that the estimated velocity matches with the measured velocity. We can now use the estimated velocity to regulate the DC motor and lower the sampling frequency to $F_s = 5kHz$ instead of $F_s = 20kHz$. Then the new sampling frequency $F_s = 5kHz$ will be used for the design of the controllers.

5.4 PI controller

5.4.1 Discrete time

Now that the system has been identified around an equilibrium, we implement a PI controller to regulate the propeller's velocity. To do this, we consider the transfer function $H_1(s) = \frac{\omega_h(s)}{u(s)}$. From the equation 5.2, we have $H_1(s) = \frac{3403}{1+0.0442.s}$. The PI controller has the form $C(s) = K_p.(1 + \frac{1}{T_i.s})$ and the parameters K_p and T_i are determined by pole compensation. Then $T_i = 0.0442$ and K_p is determined such that the settling time at 95% is the same for closed-loop and the open-loop system i.e. $\tau_{settling} = 0.173s$. This gives $K_p = 2.9377.10^{-4}$. The tuning of the PI controller gives a gain margin of 64,9 dB and a phase margin of 90°. The discretization of the PI controller is made by using the backward difference approximation in order to respect [2] when using the event-based approach with $F_s = 5kHz$. Then the control is computed by:

$$\begin{cases} u_P = K_p.(\omega_{ref}(k) - \omega_h(k)) \\ u_I(k) = u_I(k-1) + \frac{K_p}{T_i}.T_s.(\omega_{ref}(k) - \omega_h(k)) \\ u_{PI}(k) = u_p(k) + u_i(k) \end{cases} \tag{5.13}$$

5.4.2 Event-based PI controller

Trajectory tracking

We perform the event-based approach described by [3] and apply the method to a PI controller. We define the set $\beta := \begin{cases} \omega_{ref} - \omega_{tolerance} < \omega_h < \omega_{ref} + \omega_{tolerance} \\ i_{set} - i_{tolerance} < i < i_{set} + i_{tolerance} \end{cases}$ with $\omega_{tolerance} = 40 rad/s$, $i_{set} = 0.3A$ and $i_{tolerance} = 0.1A$. i_{set} and $i_{tolerance}$ are chosen such that the current remains inside the set when the reference tracking is performed. The chosen $\omega_{tolerance}$ corresponds to error on the velocity corresponding to 3% to 6% of the reference. We apply this event-based approach to the system and compared it with the synchronous PI controller. The results are shown below:

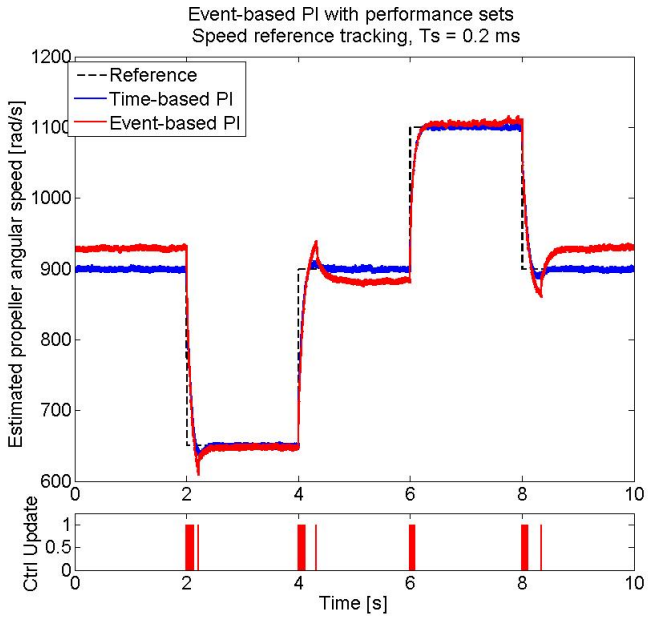

Figure 5.7: Speed reference tracking: comparison between the synchronous PI and the event-based PI. The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

Figure 5.7 shows that the reference tracking is well performed with the event-based approach. The tracking is achieved with an velocity error less than 40 rad/s. The control update takes place only when a change of reference happens.

The control and the current are shown on the figure 5.8:

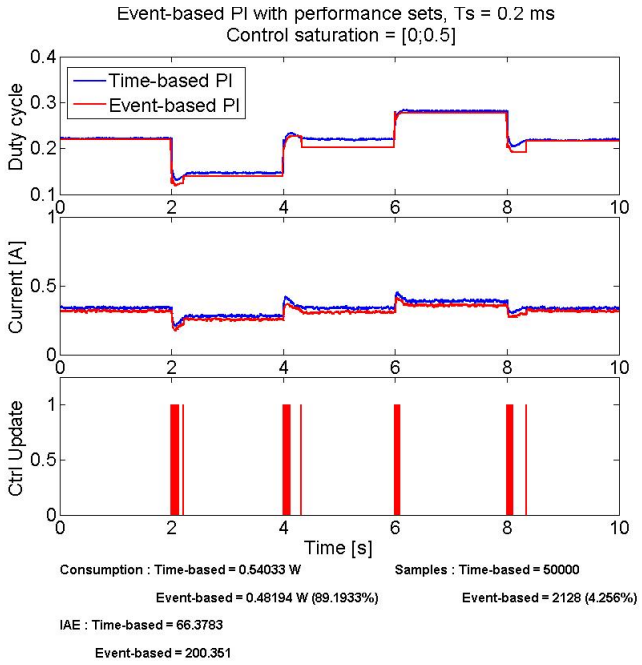

Figure 5.8: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous PI in blue and the event-based PI in red. The bottom plot shows the control update in the case of the event-based PI.

As one can see on the figure 5.8, the control is maintained constant when there is no change of reference. In this case, the control is kept to a lower value than the time-based PI. This fact allows the event-based controller to consume less energy than the synchronous PI: 0.54W for the time-based PI and 0.48W for the event-based control. This corresponds to a reduction 10% of electric consumption. Moreover the event-based control uses 95% less updates than the time-based control: 50000 updates with the synchronous controller and 2128 with the event-based controller. However, the reduction of computational cost and energy imply a degradation of the control which is represented with an IAE of 200 with the event-based control and only 66 with the synchronous controller.

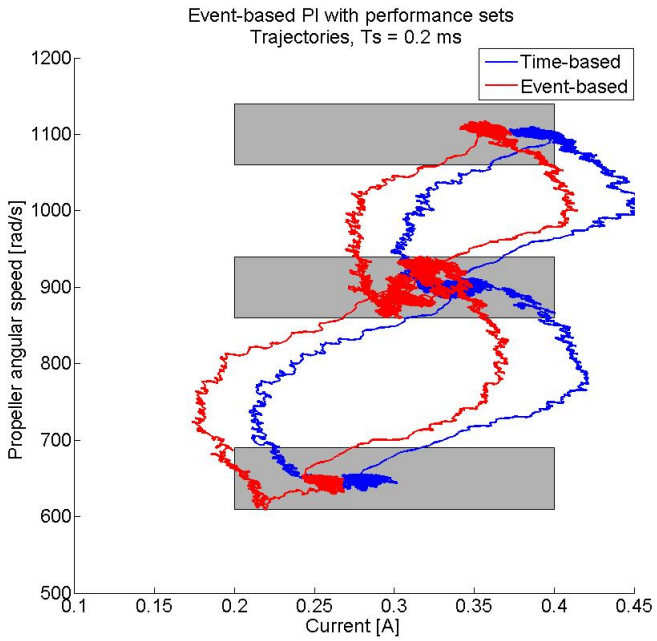

Figure 5.9: Speed reference tracking: comparison between the synchronous PI and the event-based PI. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in blue and red. The performance sets are represented by the grey rectangles.

Figure 5.9 shows the trajectories of the state and the performance sets when the reference tracking is performed. There are three performance sets since the velocity varies between three different values. As soon as the states enter in the set the control is held. Then the states converge to a value inside the set and remain in the set until a change of reference.

Disturbance rejection

To analyze the disturbance rejection, we set the regulated motor to 1100 rad/s and disturb the motor with a series of gust of wind induced by the second motor. The latter is controlled in open-loop with different duty cycle and activated at different moments. The disturbances are performed between the time $t \in [3s; 4s]$, $t \in [5s; 6s]$ and $t \in [7s; 8s]$ with a duty cycle respectively equals to 0.5, 0.6 and 0.6 otherwise the motor is turn off with a duty cycle of 0. We modify the performance set in order to have the smaller performance set possible while having a disturbance rejection with minimal control update. Then the new parameters are $\omega_{tolerance} = 40rad/s$, $i_{set} = 0.365A$ and $i_{tolerance} = 0.03A$.

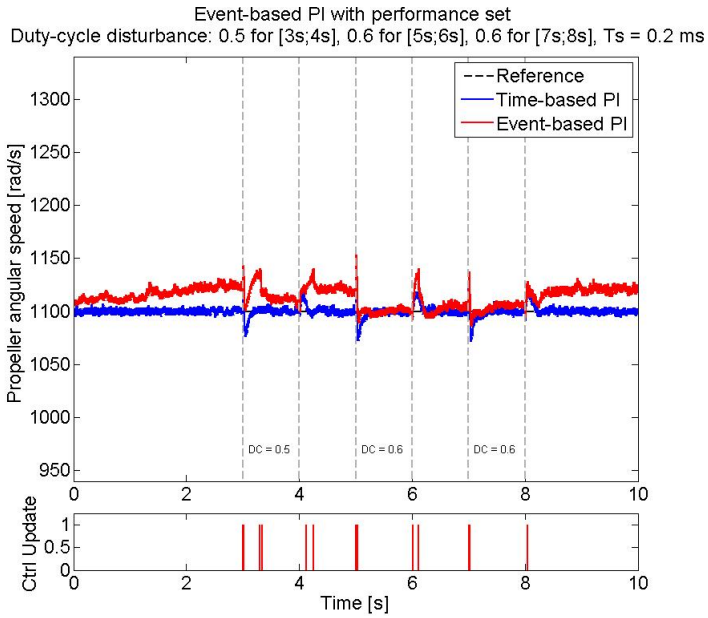

Figure 5.10: Disturbance rejection. The top plot shows the velocity of the propeller with a synchronous PI in blue and with the event-based PI in red. The perturbation is activated for $t \in [3s; 4s]$, $t \in [5s; 6s]$ and $t \in [7s; 8s]$ with a duty cycle of 0.5, 0.6 and 0.6. The bottom plot shows the control update in the case of the event-based PI.

As one can see on the figure 5.10, the event-based control rejects the disturbance. When the first disturbance is activated at time t=3s, several updates are performed to reject the disturbance. However the velocity is increasing around the time t=3.2s and leave the performance set. Then the control is updated to maintain the velocity around the reference. When the first disturbance is removed at time t=4s, the control needs to be updated since the velocity increases. When the second disturbance is activated, the states are pushed outside the performance set which induces a control update. Then the states stay inside the set until the second disturbance is removed. The same scenario happens with the third disturbance. Then it is interesting to notice that the velocity stays around the reference with an error lower than 40 rad/s without control update. When the error exceeds this threshold, the control is immediately updated to reach the performance set which leads to a error lower than 40 rad/s.

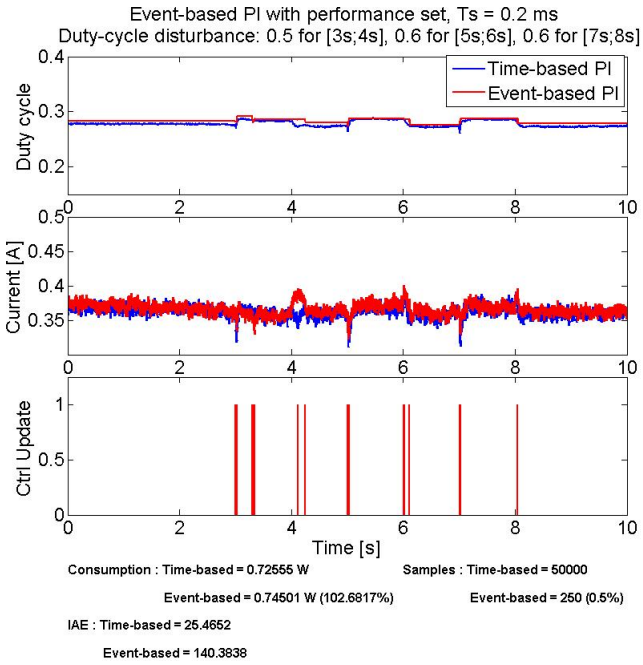

Figure 5.11: Disturbance rejection. The top plot and middle plot show respectively the control and the current with the synchronous PI in blue and the event-based PI in red. The bottom plot shows the control update in the case of the event-based PI.

Figure 5.11 shows the control and the current in the case of the disturbance rejection. The control is updated when a disturbance takes place or is removed. The event-based controller only performs 250 control updates when the synchronous PI has 50000 control updates which a reduction of 99.5% of the control update. However the power consumption is higher with the event-based PI, 0.75W, when it is only 0.73W with the time-based PI. The IAE index is 140 with the event-based PI when it is only 25 with the time-based PI. This due to the fact that we allow an error of 40 rad/s around the reference.

The trajectories of the states are shown on the figure 5.12:

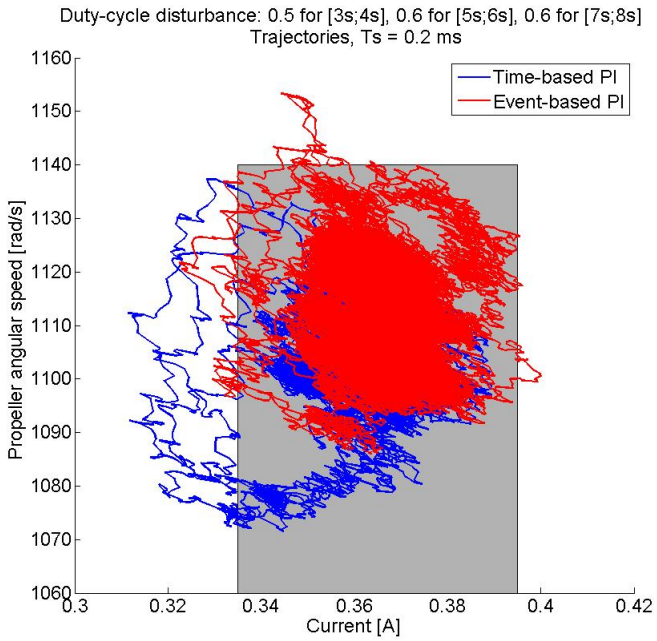

Figure 5.12: Disturbance rejection: comparison between the synchronous PI and the event-based PI. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The performance set is represented by the grey rectangle.

The states remains most of the time in the set of performance. As soon as the states leave the set, a new control is performed which leads to the convergence of the states inside the set. Thus the event-based PI can reject the disturbances with a significant reduction of the control update.

In the appendix A, it is proposed to study the effects of a reduction of the performance set while performing a reference tracking. Indeed, by reducing the performance sets, the event-based control gets close to a synchronous control which leads to an increase of the control update and a reduction of the IAE index. Then one can design an appropriate set by finding a compromise between the number of control update and the accuracy of the event-based control.

To go a step further, we propose in the appendix B to study the PI controller with the same gains with the event-based approach given in [2]. The results are shown with a reference tracking and a disturbance rejection.

5.5 Event-based state-feedback controller in [3]

5.5.1 Discrete time

Since the identification of the system takes into account the current and the propeller velocity, we propose to apply the event-based with the performance sets with a state-feedback control as in [3] instead of a PI controller. To ensure the trajectory tracking, we add an integral action $\dot{z}(t) = \omega_{ref}(t) - \omega_h(t)$ just like it is described in the section 4.4.1. We place the discrete poles of the propeller velocity ω_h, the current i and the new state z in $[0.9955; 0.9945; 0.9935]$ in the z-plane. Then the state-feedback

$u(k) = -[K_i\, K_{\omega_h}\, K_z].[i(k)\, \omega_h(k)\, z(k)]^T$is defined with $K_i = -0.2270$, $K_{\omega_h} = -0.0001$ and $K_z = -0.0015$. By using a backward discretization, the synchronous control is computed as follow:

$$\begin{cases} u_p(k) = -[K_i\ K_{\omega_h}].[i(k)\ \omega_h(k)]^T \\ u_i(k) = u_i(k-1) - K_i.T_s.(\omega_{ref}(k) - \omega_h(k)) \\ u(k) = u_p(k) + u_i(k) \end{cases} \tag{5.14}$$

We keep the same sampling frequency $F_s = 5kHz$ as for the PI controller.

Trajectory tracking

To perform the reference tracking, we define the performance set β in the same way as previously i.e : $\beta := \begin{cases} \omega_{ref} - \omega_{tolerance} < \omega_h < \omega_{ref} + \omega_{tolerance} \\ i_{set} - i_{tolerance} < i < i_{set} + i_{tolerance} \end{cases}$ with $\omega_{tolerance} = 40rad/s$, $i_{set} = 0.3A$ and $i_{tolerance} = 0.1A$.

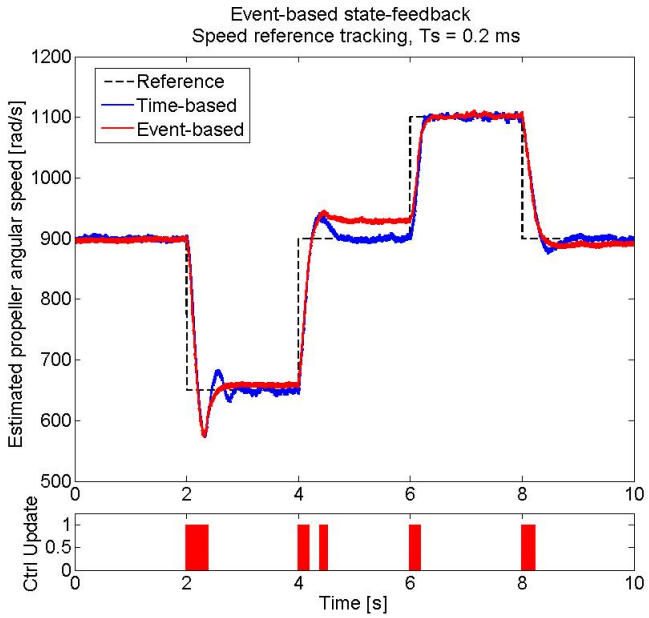

Figure 5.13: Speed reference tracking: comparison between the synchronous state-feedback (blue) and the event-based state-feedback (red). The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

Figure 5.13 shows that the event-based state-feedback can perform the reference tracking with a static error lower than $\omega_{tolerance} = 40rad/s$. The updates take place only during the changes of reference which leads to a significant reduction of the control update. One can notice that the control is updating around the time t=4.2s because the maintained control does not allow the states to stay inside the set β. Thus the controller needs to update the control around the time t=4.2s to reach again the set β.

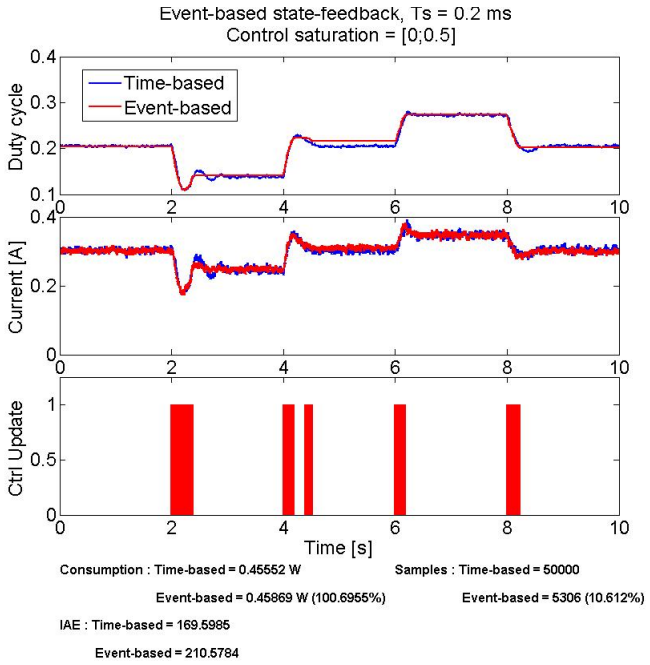

Figure 5.14: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous state-feedback in blue and the event-based state-feedback in red. The bottom plot shows the control update in the case of the event-based control.

As one can see on the figure 5.14, the control is only updated during the changes of reference and keep its previous value during the steady-state. With an event-based state-feedback the reduction of the control update is about 91%. The IAE index is 170 for the synchronous state-feedback while it is 211 with the event-based controller. The power consumptions are equivalent with the two approaches and equal to 0.46W.

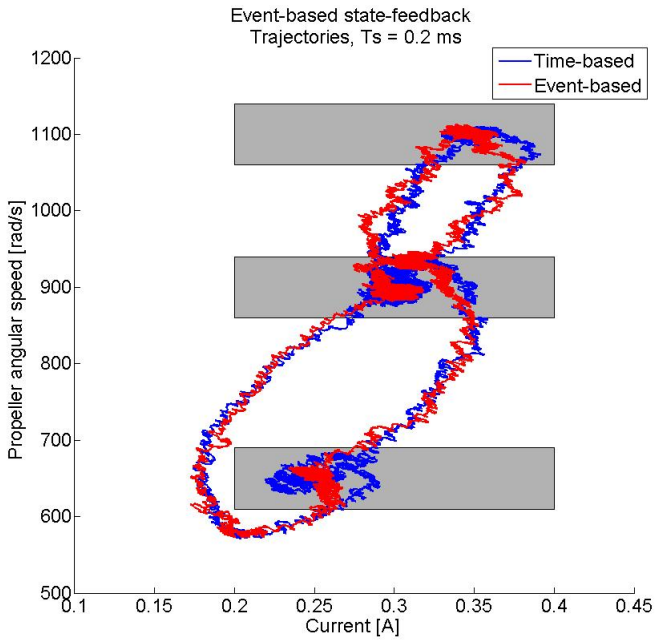

Figure 5.15: Speed reference tracking: comparison between the synchronous state-feedback and the event-based state-feedback. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in blue and red. The performance sets are represented by the grey rectangles.

Figure 5.15 shows the trajectories of the states with the two kind of controllers. One can see that the trajectory of the states converges to the different performance sets.

Disturbance rejection

The disturbance rejection is realized with the same condition as previously. We define the same performance set as for the event-based PI controller i.e. $\omega_{tolerance} = 40rad/s$, $i_{set} = 0.365A$ and $i_{tolerance} = 0.03A$.

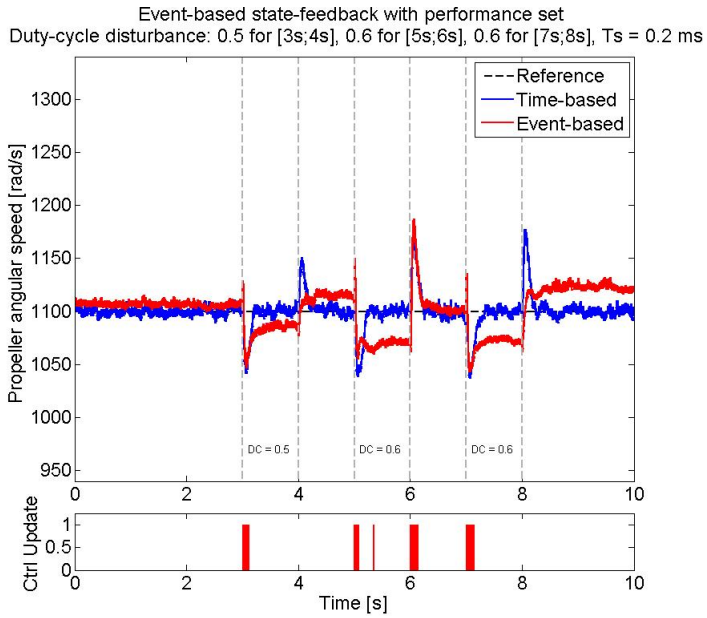

Figure 5.16: Disturbance rejection. The top plot shows the velocity of the propeller with a synchronous state-feedback in blue and with the event-based state-feedback in red. The perturbation is activated for $t \in [3s; 4s]$, $t \in [5s; 6s]$ and $t \in [7s; 8s]$ with a duty cycle of 0.5, 0.6 and 0.6. The bottom plot shows the control update in the case of the event-based state-feedback.

As one can see on the figure 5.16, the event-based state-feedback rejects the disturbances. At time t=3s the first disturbance is activated and the controller needs to update the control to stay inside the performance set. At time t=4s, when the first disturbance is removed, the propeller velocity increases but does not leave the set β then no control update is performed. When the second disturbance occurs at time t=5s, the control needs to be updated since the velocity is going out of the set β. When the disturbance is removed at time t=6s, the velocity is highly increased which leads to a control update to stabilize the velocity around the reference. Finally when the third disturbance is added, the control needs to be updated but when the disturbance is removed the states stay inside the set β and no more updates are performed.

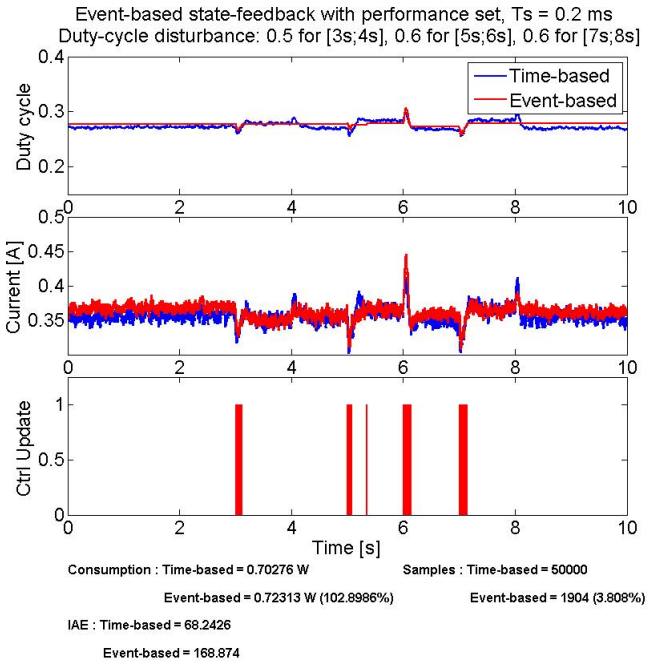

Figure 5.17: Disturbance rejection. The top plot and middle plot show respectively the control and the current with the synchronous state-feedback in blue and the event-based state-feedback in red. The bottom plot shows the control update in the case of the event-based state-feedback.

Figure 5.17 shows the control and the current during the disturbance rejection with the event-based state-feedback control. As one can see the control are only updated when the disturbances are added or removed. This leads to a minimization of the control update up to 96%. The IAE index is however deteriorated and equals to 169 with the event-based control and 68 with the time-based control. The power consumption are slightly the same: 0.70W with the synchronous control and 0.72W with the event-based control.

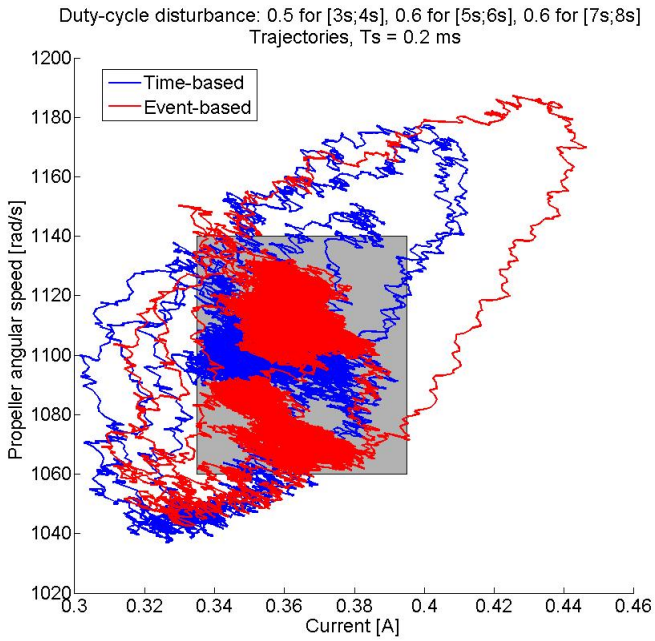

Figure 5.18: Disturbance rejection: comparison between the synchronous state-feedback and the event-based state-feedback. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The performance set is represented by the grey rectangle.

Figure 5.18 shows the trajectories of the states during the disturbance rejection. The disturbances affect the trajectories and push the states outside the set β . As soon as the states are outside the set, the control is computed and updated such that the states reach again the set. Once the control is held, the states converge inside the set. In this example, we can see two different convergence areas. The states reach these areas depending of the last control update.

5.6 Conclusion

We summarize the results with the different event-based approaches in the table 5.2:

The event-based approach with performance sets with a PI controller or a state-feedback control shows good results with a large reduction of the control update up to 90% compare to a synchronous controller. It allows to perform a reference tracking and a disturbance rejection with a few control update but it degrades the error as we can see with the IAE index. However we tolerate this error by defining the performance set that fits to the control objective. The power consumption is more or less the same as for the synchronous controller. The difference of the consumption between the event-based approach with performance set and the synchronous controller depends on where the states will converge inside the performance set. If the held control is higher than the required control to reach exactly the reference then the power consumption will be higher than the consumption with a synchronous controller. Then by defining smaller performance set, the control will get closer to the required control to reach the reference. This will lead to have a smaller error while having almost the same consumption as for a synchronous controller. However

			Number of control update	IAE	Consumption (W)
PI controller	Trajectory tracking	Synchronous	50 000	66,4	0,54
		Performance set	2 128 (4,3%)	200,4	0,48
		Reduced performance set	2542 (5,1%)	134,5	0,5
		N. Marchand [2]	1 778 (3,6%)	92	0,53
	Disturbance rejection	Synchronous	50 000	25,5	0,73
		Performance set	250 (0,5%)	140,4	0,75
		N. Marchand [2]	893 (1,8%)	41,8	0,73
State-feedback controller	Trajectory tracking	Synchronous	50 000	169,6	0,46
		Performance set	5 306 (10,6%)	210,6	0,46
	Disturbance rejection	Synchronous	50 000	68,2	0,70
		Performance set	1 904 (3,8%)	168,9	0,72

Table 5.2: Results of the event-based controller

by doing this, the control update will increase since if we reduce the performance set then the event-based control gets close to a synchronous control. Then the event-based approach with the performance set can be really efficient if one can find a performance set that fits exactly to the objectives in terms of control update, allowable error and power consumption.

The event-based presented in [2] shows also good results (refer to appendix B). With this approach, the error is significantly reduced and the reduction of the control update is around 94%. The power consumption is almost the same as for the synchronous controller. This approach is only described with a PI controller however one can slightly modify the method and adapt it to an state-feedback control. In this case, the generation of the event can stay the same but one needs to modify the treatment of the integral state.

Chapter 6

Conclusion and prospects

6.1 Conclusion

My internship aimed at studying event-based control approaches by means of experimental works. The test bench was a ducted fan which consists in a miniature DC motor. In previous work in the lab, it was embedded in the LORA robot. LORA is bioinspired miniature robot [9]. The purpose of my work was to investigate a bioinspired way of controlling miniature robots taking into account consumption considerations together with performances. It was shown that the event-based control can perform a reference tracking and a disturbance rejection on a miniature DC motor. To regulate the DC motor, we proposed to estimate the propeller speed by only measuring three voltages instead of using a velocity sensor. The event-based approach was then applied on the estimated velocity of the propeller. The event-based approach was performed with a PI controller and a state-feedback controller where the performances of each controllers were defined by finding an appropriate set of performance where the control is held. The event-based approach allowed to reduce significantly the number of control update while having a similar energy consumption compared to a synchronous controller.

Furthermore, we applied another event-based strategy where the event was generated when the difference between two consecutive errors exceeded a specified threshold [2]. With this approach, the performance of the event-based can also be defined by adjusting the threshold parameters. During the experimentation part, it was shown that this method can also reduce the control update while performing a reference tracking and a disturbance rejection.

These results will be used in the future where an event-based control will be apply to model of bee. The aim is then to regulate the optic flow that perceives the bee and perform a trajectory tracking on the simulated bee.

6.2 Prospects

Further results can be obtained with the event-based control. An interesting approach is to keep a held control when the states lie inside a certain set and to compute this set a priori. This set could be endowed with properties of invariance. This approach could lead to interesting stability properties.

Bibliography

[1] S. Durand. *Reduction of the energy consumption in embedded electronic devices with low control computational cost.* PhD thesis, Université de Grenoble, 2011.

[2] S. Durand and N. Marchand. Further results on event-based pid controller. 2009.

[3] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. Van Den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81:4:571–590, 2012.

[4] J. Lunze and D. Lehmann. A state-feedback approach to event-based controller. *Automatica*, 46:211–215, 2010.

[5] G. Portelli, F. Ruffier, F. Roubieu, and N. Franceschini. Honeybee's speed depends on dorsal as well as lateral, ventral and frontal optic flows. *Plos One*, 2011.

[6] G. Portelli, J. Serres, F. Ruffier, and N. Franceschini. Modelling honeybee visual guidance in a 3-d environment. *Journal of Physiology*, pages 104:27–39, 2010.

[7] F. Ruffier and N. Franceschini. Optic flow regulation: the key to aircraft automatic guidance. *Robotics and Autonomous Systems*, pages 177–194, 2003.

[8] K.-E. Årzén. A simple event-based pid controller. *In Preprints of the 14th World Congress of IFAC, Beijing, P.R. China,*, 1999.

[9] J. Serres, D. Dray, F. Ruffier, and N. Franceschini. A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance. *Autonomous Robot*, 25:103–122, 2008.

[10] S. Viollet, L. Kerhuel, and N. Franceschini. A 1-gram dual sensorless speed governor for micro-air vehicles. *Control and Automation, 2008 16th Mediterranean Conference on 2008*, pages 1270–1275, 2008.

Appendix A

Reduction of the performance sets

We propose to reduce the performance sets in order to highlight the effects of the sets concerning the number of control update and the IAE index. To do this, we reduce the tolerance on the velocity $\omega_{tolerance}$, we set $\omega_{tolerance} = 20rad/s$ instead of 40 rad/s. We keep the same tolerance for the current, i.e. $i_{tolerance} = 0.1A$, such that the current remains in the set for the given reference. Then the set is defined by $\beta := \begin{cases} \omega_{ref} - \omega_{tolerance} < \omega_h < \omega_{ref} + \omega_{tolerance} \\ i_{set} - i_{tolerance} < i < i_{set} + i_{tolerance} \end{cases}$ with $\omega_{tolerance} = 20rad/s$, $i_{set} = 0.3A$ and $i_{tolerance} = 0.1A$. The sets can be seen on the figure A.3. Then we perform a reference tracking with the PI controller described in the section 5.4.2 with the same gains with the new performance sets:

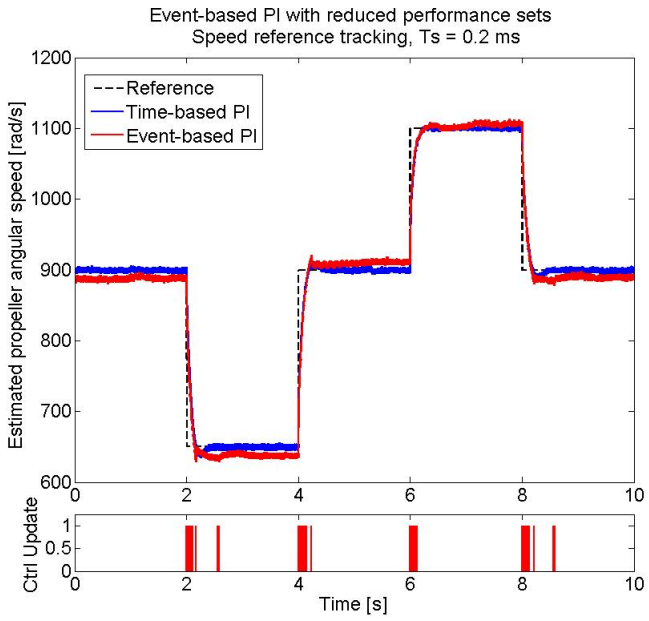

Figure A.1: Speed reference tracking: comparison between the synchronous PI and the event-based PI with a reduced set. The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

Figure A.1 shows the velocity of the propeller and the control update with the event-based control with the reduced sets. As one can see the reference tracking is perform with a smaller performance set and the control updates take only place during the change of reference.

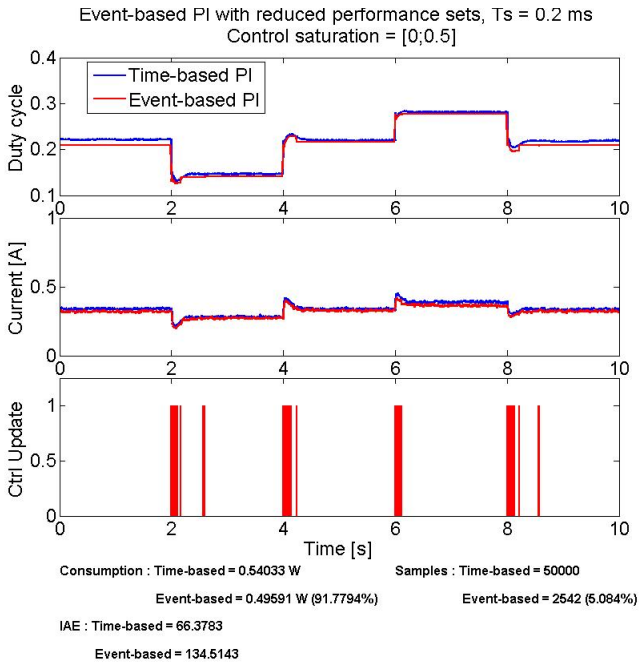

Figure A.2: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous PI in blue and the event-based PI with a reduced set in red. The bottom plot shows the control update in the case of the event-based PI.

The current and the control are plotted on the figure A.2. The event-based control with reduced performance sets uses 2542 control updates which is a reduction of 95% of the control update compared to a synchronous controller. By using the reduced sets, the IAE index is 134.5 while it is 66.4 with the time-based PI. We recall that in the case of the event-based control with the tolerance on the velocity $\omega_{tolerance} = 40 rad/s$, the number of control update was 2128 with an IAE index of 200.4. By reducing the performance sets, we increase the number of control update and decrease the IAE index. This can be deduced by the fact that if one reduces the performance set, the control gets closer than a synchronous controller (if the set tends to an empty set, then the resulting control tends to a synchronous controller). The power consumption in the case of the event-based control is 0.50W when it is 0.54W with the time-based controller. This reduction of power consumption is explained by the fact that the held control has a lower value than the control with a time-based PI during the steady-state.

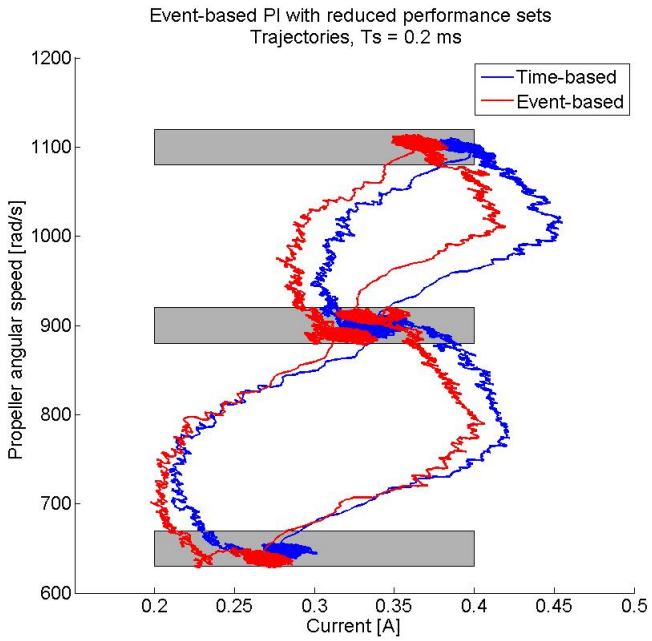

Figure A.3: Speed reference tracking: comparison between the synchronous PI and the event-based PI. The figure shows the phase plane of the states: the current i and the propeller velocity ω_h. The trajectories of the synchronous controller and the event-based one are respectively plotted in blue and red. The reduced performance sets are represented by the grey rectangles.

Figure A.3 shows the trajectories of the states with the event-based and the time-based controller. The reduced sets of performance are also plotted. As one can see, the states remain in the sets during the steady-state even if they are reduced. If the sets are too much reduced, it can seen that the controller will need to perform more control update to reach each sets. However if the sets are too large, the error will significantly increase. Then one needs to find a compromise between the number of control update and the accuracy of the control by designing a appropriate set that fits to the requirements.

Appendix B

Event-based PI controller described in [2]

We propose to apply the event-based method described in [2]. With this method, the event occurred when $|e(t_{k-1}) - e(t_k)| > e_{lim}$ where e is the error $e(t_k) = \omega_{ref}(t_k) - \omega_h(t_k)$ and e_{lim} is a threshold to define. Several algorithms are introduced in [2] in order to remove the safety condition firstly introduced in [8]: $h_{act} > h_{max}$ where h_{act} is the interval between two consecutive events and h_{max} is a threshold to define. The safety condition forces to update the control even if no control action is needed in order keep the stability of the system. To understand this, it is necessary to study the integral term of the PI controller $u_I k) = u_I(k-1) + \frac{K_p}{T_i}.h_{act}.e(t_k)$. During the steady-state, we have $\omega_h \simeq \omega_{ref}$ i.e $e \simeq 0$ and the time h_{act} is increasing since no control update are performed during the steady state. The product $h_{act}.e$ remain small because the error e is close to 0. However when the reference changes or when a disturbance occurs, the error e become high and the product $h_{act}.e$ is even higher since h_{act} has increased during all the steady-state. This results in a huge overshoot on the control. To prevent this phenomenon, four algorithms are presented in [2] to limit this effect. We propose to implement the hybrid algorithm which mixes an exponential forgetting factor with a saturation of the product $h_{act}.e$. The algorithm is given below:

Algorithme B.1 Hybrid algorithm from [2]

```
% Inputs
ysp = u(1);
y = u(2);
e = ysp - y

% calculate control signal
hact = hact + Ts;
if ( abs(e-e_old)>elim )
    up = Kp*e;
    if ( hact >= hmax )
        hact_i = hact*exp(hnom-hact);
        he = (hact_i - hnom)*elim + hnom*e;
    else
        he = hact*e;
    end
    ui = ui + Kp/Ti*he;
    u = ui + up;

    % Update
    e_old = e;
    y_old = y;
    hact = 0;
end
```

B.1 Trajectory tracking

To implement the new event-based control we need to define the parameters e_{lim} and h_{max}. We set $e_{lim} = 4$ and $h_{max} = 0.1$. These parameters are chosen such that the control update are significantly reduced while the control allows a reference tracking. In order to compare the new approach with the synchronous PI, it is important to keep the same PI gains: $K_p = 2.9377.10^{-4}$ and $T_i = 0.0442$. The results for the reference tracking are shown below:

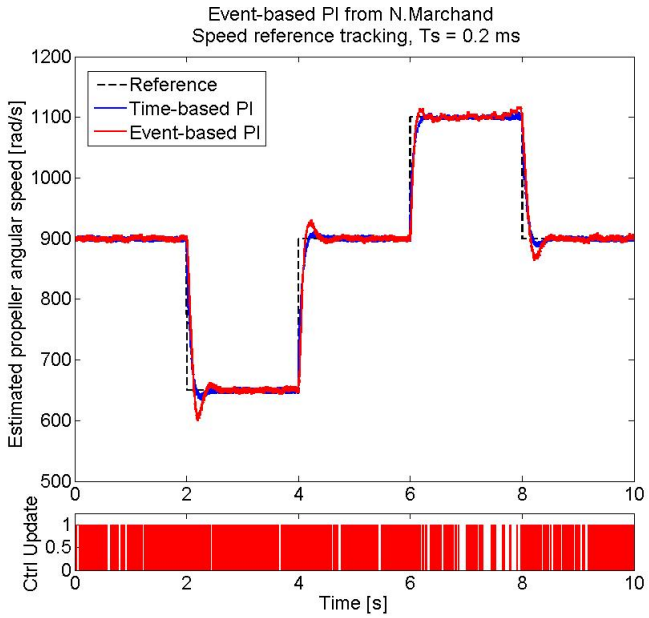

Figure B.1: Speed reference tracking: comparison between the synchronous PI (blue) and the event-based PI described in [2] (red). The figure shows the evolution of the propeller speed with the two types of control and the control update of the event-based control: the update is performed when the indicator is at 1.

On the figure B.1, the velocity of the propeller are shown with the event-based described in [2]. This event-based approach can perform a reference tracking without static error. One can observe that this event-based approach increases the overshoots observed during the changes of reference.

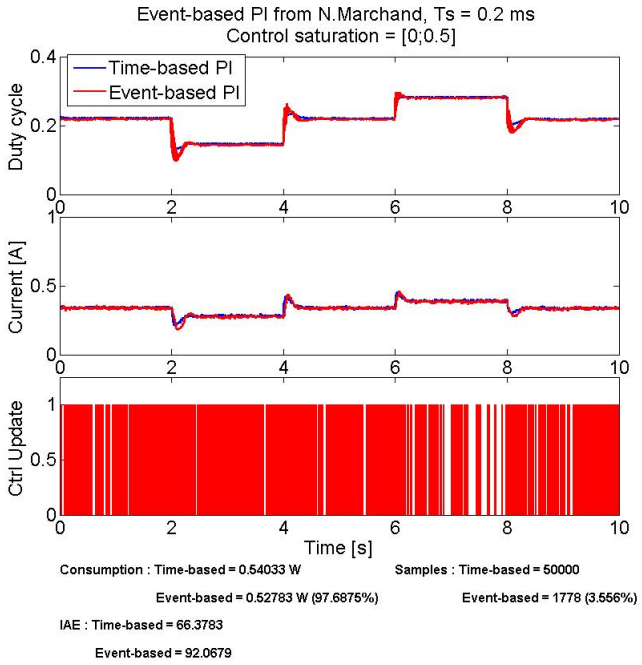

Figure B.2: Speed reference tracking. The top plot and middle plot show respectively the control and the current with the synchronous PI in blue and the event-based PI described in [2] in red. The bottom plot shows the control update in the case of the event-based PI.

Figure B.2 shows the control and the current with the event-based method described in [2]. With this event-based approach the controls are updated even when the reference does not change unlike the event-based with the performance sets. However the reference tracking is performed with only 1778 updates which is a reduction of 96.5% of the control update. This method does not imply a large static error which leads to an IAE index of 92 close to IAE index of the synchronous PI equals to 66.

B.2 Disturbance rejection

We analyze the disturbance rejection with the event-based described in [2] with the same procedure as previously. The results are shown below:

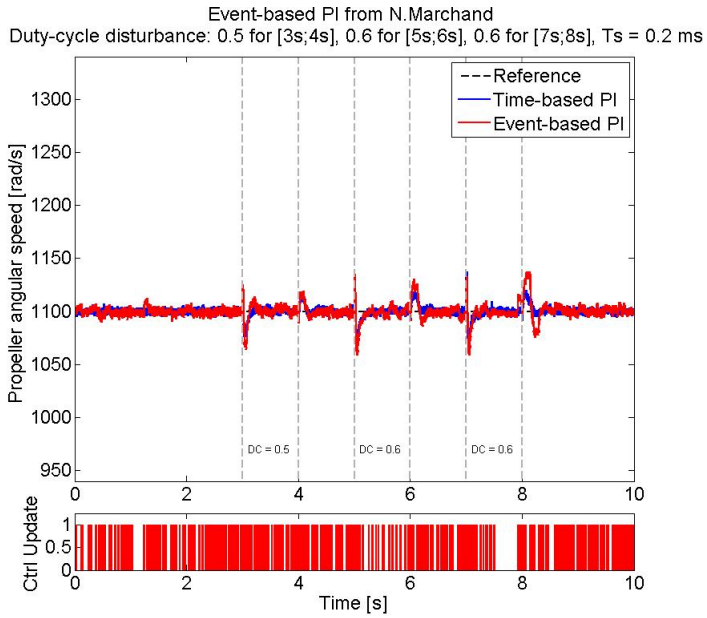

Figure B.3: Disturbance rejection: comparison between the synchronous PI (blue) and the event-based PI described in [2] (red). The perturbation is activated for $t \in [3s; 4s]$, $t \in [5s; 6s]$ and $t \in [7s; 8s]$ with a duty cycle of 0.5, 0.6 and 0.6. The top plot shows the velocity of the propeller and the bottom plot shows the control update in the case of the event-based PI.

Figure B.3 shows the velocity of the propeller with the event-based control en red and the time-based control in red. We can see that the event-based control rejects the disturbances. This control induces a larger overshoot than the synchronous PI.

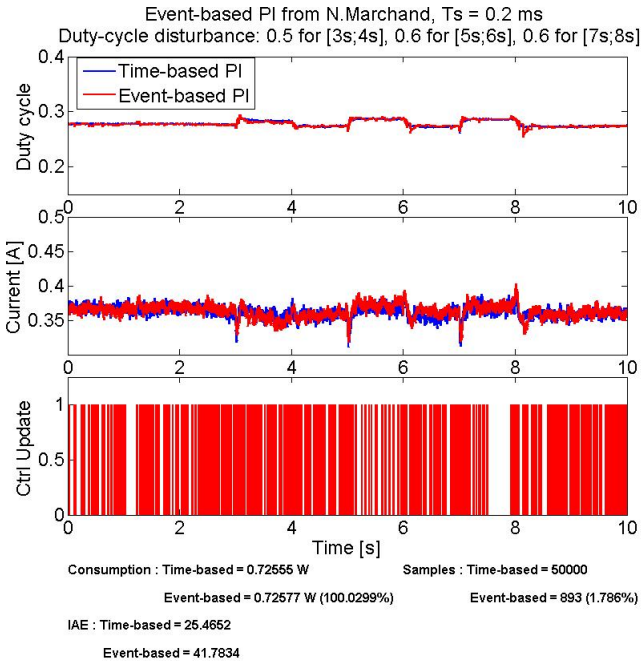

Figure B.4: Disturbance rejection: The top plot and middle plot show respectively the control and the current with the synchronous PI in blue and the event-based PI described in [2] in red. The bottom plot shows the control update in the case of the event-based PI.

Even if this event-based control updates the control during the steady-state, the reduction of the control update is about 92% as it is shown on the figure B.4. It confers an IAE index of 42 when it is 25 for the time-based control. Since the current and the control are quite similar, the power consumption with both controllers are almost equal to 0.72W.

The event-based approach described in [2] is really efficient and allows to perform a reference tracking and a disturbance rejection with a very few control updates and without large static error. One can increase the value of e_{lim} in order to minimize the update of the control specially during the steady-state when there is no change of reference. However by doing this, the reference tracking and the disturbance rejection are not insured. Here we have set $e_{lim} = 4$ to be sure that the control will be updated even if it is updated because of the noise on the estimated velocity during the steady-state.