Implicit discrete-time twisting controller without numerical chattering: analysis and experimental results

Olivier Huber, Vincent Acary, Bernard Brogliato, Franck Plestan

Implicit discrete-time twisting controller without numerical chattering: analysis and experimental results

Olivier Huber[a], Vincent Acary[a], Bernard Brogliato[a], Franck Plestan[b]

[a]INRIA Grenoble Rhône-Alpes, BIPOP project-team, 655 avenue de l'Europe, Inovallée, 38334 Saint-Ismier, France
[b]LUNAM Université, Ecole Centrale de Nantes - IRCCyN UMR CNRS 6597, Nantes, France

Abstract

In this paper, we present an implementation of the sliding mode twisting controller on an electropneumatic plant for a tracking control problem. To this end, implicitly and explicitly discretized twisting controllers are considered. We discuss their structure, properties and implementations, as well as the experimental results. The analysis of the performance sustains the theoretical superiority of the implicitly discretized version, as shown in previous works. The main advantages of the implicit method are better tracking performance and drastic reduction in the input and output chattering. This is achieved without modifying the structure of the controller compared to its continuous-time version. The tracking error cannot be used as the sliding variable: it has a relative degree 3 w.r.t. the control input. The tuning of the sliding surface has well as some other parameters in the control loop was instrumental in achieving good performance. We detail the selection procedure of those parameters and their influence on the closed-loop behavior. Finally we also present some results with an implicitly discretized EBC-SMC controller.

Keywords: twisting controller, sampled-data system, implicit discretization, electropneumatic actuator

1. Introduction

Implementation of control laws is almost exclusively done using microcontrollers. This implies that the controller is in discrete-time rather than in continuous-time. In sliding mode control, this can induce a degradation of the performance by contributing to the chattering phenomenon. We call this the *numerical chattering*. An intense activity over the last 30 years was devoted to the reduction of this numerical chattering, mainly for equivalent control based sliding mode control (ECB-SMC). In the early 90's, second order sliding mode control concept was introduced in Levant (1993) and sparked the development of a large wealth of literature. One of the first controllers of this kind was the twisting controller which features a discontinuous control action w.r.t. the sliding variables. However, to the best of our knowledge, few discrete-time versions of the twisting controller have been proposed. The substitution of the signum function by a saturation, common trick to reduce the chattering for first order SMC, has no straightforward extension to the twisting algorithm. It is then fair to assume that the explicit discretization was used to get a discrete-time twisting controller, like in Taleb et al. (2013).

The other discretization method we consider is the implicit method. It has been used for a long time in the nonsmooth mechanics community, but it was not applied in control theory until very recently Acary and Brogliato (2010); Acary et al. (2012); Huber et al. (2013a,b). The implicit discretization of the twisting controller was first studied in Acary et al. (2012). Roughly speaking, the difference between the explicit and implicit methods in our context is the following: given a partition $\{t_k\}$ of a time interval, with the *explicit* discretization, at the time instant t_k, the argument of the signum function is the value of the sliding variable at t_k, whereas with the *implicit* discretization it is the value at t_{k+1}. Despite its name and formulation, the implicitly discretized twisting controller is non-anticipative and induces a well-defined behavior, as we shall see in Section 2. Its main features are the drastic reduction of the output chattering and the reduction of the control input chattering, that is the control input is no more of the high frequency "bang-bang" type. In the discrete-time sliding regime, the control input is also insensitive to an increase of the gain. To simplify the nomenclature, we refer to the discrete-time twisting controller with an implicit (resp. explicit) discretization as the implicit (resp. explicit) twisting controller.

In the following, we present results from an implementation of both explicit and implicit twisting controllers on an electropneumatic plant. The control problem at hand is the tracking of a sinusoidal trajectory for the position of the end of the piston. The analysis of the gathered data supports the theoretically predicted reduction of the chattering claimed in Acary et al. (2012) and also the claim

[☆]The authors acknowledge the support of the ANR grant CHASLIM (ANR-11-BS03-0007).

that the numerical chattering can be the main source of chattering, see Huber et al. (2013a). This highlights the importance of the discretization process which is unfortunately often overlooked both in the analysis and in the implementation.

The second part of the paper is dedicated to the choice of three parameters: the first one defines the sliding variable and the two others are constants for two filtered differentiators. The influence of those parameters is only visible with the implicit controller. With an explicit one, the performance is not good enough to always see a change when their values change. It appears that with an implicit controller the differentiators become the weakest component in the control loop. Empirical data suggest that the three parameters have to be tuned simultaneously. To help with the tuning, we present the selection procedure that we used. We also analyze how the experimental tracking performance varies with the choice of the sliding surface. We hope that this presentation raises awareness for the importance of tuning to get the best possible performance for systems with similar setup.

In the remainder of this section, we introduce the notations. In Section 2 we briefly recall the twisting controller in continuous-time as well as in discrete-time. The experimental setup is presented in Section 3 as well as the control scheme. Then the experimental results are analyzed in Section 4. In Section 5, we deal with the tuning of some control parameters and the impact it has on the performance. In Section 6 an experimental comparison between the twisting and a classical first order SMC is proposed. Conclusions end the article in Section 7.

Notations: The sliding variable is denoted by σ, it is supposed to be at least twice differentiable and Σ denotes $\begin{pmatrix}\sigma & \dot{\sigma}\end{pmatrix}^T$. The control value changes at time instants t_k, defined as $t_k \coloneqq t_0 + kh$ for all $k \in \mathbb{N}$ with $t_0, h \in \mathbb{R}_+$. The scalar h is called the sampling period. Let $\sigma_k \coloneqq \sigma(t_k)$ and $\dot{\sigma}_k \coloneqq \dot{\sigma}(t_k)$ for all $k \in \mathbb{N}$. The tilded variants $\widetilde{\sigma}, \widetilde{\dot{\sigma}}$ and $\widetilde{\Sigma}$ denote variables used in the controller. Let sgn be the classical single-valued signum function: for all $x > 0$, $\operatorname{sgn}(x) = 1$, $\operatorname{sgn}(-x) = -1$ and $\operatorname{sgn}(0) = 0$.

Definition 1 (Multivalued signum function)**.** Let $x \in \mathbb{R}$. The multivalued signum function $\operatorname{Sgn}\colon \mathbb{R} \rightrightarrows [-1, 1]$ is defined as:

$$\operatorname{Sgn}(x) = \begin{cases} \{1\} & x > 0 \\ \{-1\} & x < 0 \\ [-1, 1] & x = 0. \end{cases}$$

If $x \in \mathbb{R}^n$, then the vector-valued signum function $\operatorname{Sgn}\colon \mathbb{R}^n \rightrightarrows [-1, 1]^n$ is defined as $\operatorname{Sgn}(x) \coloneqq (\operatorname{Sgn}(x_1), \dots, \operatorname{Sgn}(x_n))^T$.

2. The twisting controller

2.1. Continuous-time twisting

The twisting algorithm was one of the first second-order sliding mode controllers presented in the literature Levant (1993). It requires the control input u to be of relative degree 2 with respect to the sliding variable σ, that is

$$\ddot{\sigma}(x, t) = a(x, t) + b(x, t)u, \tag{1}$$

with the following bounds: for all $(x, t) \in \mathbb{R}^n \times \mathbb{R}_+$,

$$0 \leq K_m \leq |b(x, t)| \leq K_M \quad \text{and} \quad |a(x, t)| \leq K_a. \tag{2}$$

The control law for the twisting controller is

$$u \in -r_1 \operatorname{Sgn}(\sigma) - r_2 \operatorname{Sgn}(\dot{\sigma}), \tag{3}$$

and with the conditions

$$\begin{cases} (r_1 + r_2)K_m - K_a > (r_1 - r_2)K_M + K_a \\ (r_1 - r_2)K_m > K_a, \end{cases} \tag{4}$$

the state of the closed-loop system (1) and (3) converges to the origin in finite time. The solutions of the closed-loop system are defined within Filippov's framework (Filippov (1988)). Lyapunov functions for this controller have been recently investigated, see Orlov (2005); Polyakov and Poznyak (2009). In this paper, we follow the convention of using $G \coloneqq r_1$ and $\beta \coloneqq r_2/r_1$, instead of r_1 and r_2. The conditions listed in (4) impose that $0 < \beta < 1$.

It is worth noting that the controller (3) is by definition multivalued and that the control input u is a selection of the closed-loop differential inclusion formed by (1) and (3).

2.2. The two discrete-time twisting controllers

The control input obtained from a microcontroller is usually a step function, and its value is periodically updated. We model the control input function as $u(t) = u_k$ for $t \in (t_k, t_{k+1}]$. When implementing this controller, the task at hand at each time instant t_k is to select the control input value from all the possible values defined by a discretization of (1) and (3). We want the discrete-time version to keep the multivalued nature of the controller. This is achieved by using the implicit discretization, which applied on (3) gives

$$u_k \in -G \operatorname{Sgn}(\sigma_{k+1}) - \beta G \operatorname{Sgn}(\dot{\sigma}_{k+1}), \tag{5}$$

whereas the explicit discretization yields

$$u_k = -G \operatorname{sgn}(\sigma_k) - \beta G \operatorname{sgn}(\dot{\sigma}_k). \tag{6}$$

Note that the relation in (6) is not an inclusion since the right-hand side is a given singleton at time t_k. The case where either σ_k or $\dot{\sigma}_k$ is zero is clearly pathological. Hence the signum function in (6) is single-valued, contrarily to the continuous-time case. The computation of the control input value is in this case straightforward from (6).

With the implicit discretization, a discrete-time version of the dynamics (1) is required to perform the computation. We recast the closed-loop dynamics (1) and (5) as a first order system with state $\Sigma \coloneqq (\sigma\ \dot{\sigma})^T$. In the following, the discrete-time dynamics of Σ is supposed to be affine and given by

$$\widetilde{\Sigma}_{k+1} = A_k^d \Sigma_k + F_k^d + B_k^d \lambda, \tag{7}$$

where $\lambda \coloneqq (\lambda_1\ \lambda_2)^T$, with $\lambda_1 \in -\operatorname{Sgn}(\sigma_{k+1})$ and $\lambda_2 \in -\operatorname{Sgn}(\dot{\sigma}_{k+1})$. At each time instant t_k, we have $\Sigma_k = \Sigma(t_k)$ but $\widetilde{\Sigma}_{k+1}$ is in general not equal to $\Sigma(t_{k+1})$. If the dynamics (1) is LTI and exact, the discrete-time dynamics obtained using a ZOH discretization is exact and therefore $\widetilde{\Sigma}_{k+1} = \Sigma(t_{k+1})$. The control input value at time t_k is computed as

$$u_k = G(1\ \beta)\lambda,$$

and therefore requires the value of λ, which is obtained as the solution of the following generalized equation

$$\begin{cases} \widetilde{\Sigma}_{k+1} = A_k^d \Sigma_k + F_k^d + B_k^d \lambda \\ \lambda \in -\operatorname{Sgn}(\widetilde{\Sigma}_{k+1}) \end{cases} \tag{8}$$

with unknowns λ and $\widetilde{\Sigma}_{k+1}$.

2.3. The implicit twisting as a generalized equation

Let us analyze this system using tools from convex analysis and variational inequalities theory. First we introduce the normal cone, denoted by $\mathcal{N}_K(z)$, to a non-empty, closed convex set K at a point $z \in K$, and defined by $\mathcal{N}_K(z) = \{x \in \mathbb{R}^n \mid \langle x, y - z\rangle \leq 0\ \forall y \in K\}$. The equivalence $\lambda \in -\operatorname{Sgn}(\widetilde{\Sigma}_{k+1}) \iff \widetilde{\Sigma}_{k+1} \in -\mathcal{N}_{[-1,1]^2}(\lambda)$ with $[-1,1]^2 = [-1,1] \times [-1,1]$, enables us to transform (8) into the generalized equation

$$0 \in A_k^d \Sigma_k + F_k^d + B_k^d \lambda + \mathcal{N}_{[-1,1]^2}(\lambda), \tag{9}$$

which features only one unknown: λ. More precisely, this inclusion is an equivalent form of an Affine Variational Inequality (AVI), see Facchinei and Pang (2003). Solving this AVI consists in finding $\lambda \in [-1,1]^2$ such that

$$\text{for all } w \in [-1,1]^2 \quad (w - \lambda)^T L_k(\lambda) \geq 0, \tag{10}$$

with $L_k \colon \lambda \mapsto A_k^d \Sigma_k + F_k^d + B_k^d \lambda$ an affine map.

2.4. Existence and uniqueness of the control input

We study the solutions of AVI (9), denoted by $\operatorname{SOL}(L_k)$.

Lemma 1. *The AVI (10) has always a solution.*

Proof. Since the mapping L_k is continuous and $[-1,1]^2$ is a bounded convex set, we can apply Corollary 2.2.5, in Facchinei and Pang (2003). □

Definition 2. A matrix $M \in \mathbb{R}^{n\times n}$ is said to be *positive semi-definite plus* if it is positive semi-definite and $z^T M z = 0 \implies Mz = 0$ for all $z \in \mathbb{R}^n$.

Let us now tackle the uniqueness of $\widetilde{\Sigma}_{k+1}$ in (8).

Lemma 2. *If B_k^d is positive semi-definite plus, then the variable $\widetilde{\Sigma}_{k+1} = L_k(\lambda)$ is unique for all $\lambda \in \operatorname{SOL}(L_k)$.*

Proof. Let $\lambda, \lambda' \in \operatorname{SOL}(L_k)$. The inclusion (9) holds if and only if Λ is a solution to the AVI (10). In particular, we have $(\lambda' - \lambda)^T L_k(\lambda) \geq 0$ and $(\lambda - \lambda')^T L_k(\lambda') \geq 0$. Summing the two inequalities, we get:

$$\begin{aligned}(\lambda' - \lambda)^T (A_k^d \Sigma_k + F_k^d + B_k^d \lambda) \\ +(\lambda - \lambda')^T (A_k^d \Sigma_k + F_k^d + B_k^d \lambda') \geq 0.\end{aligned}$$

Rearranging terms gives

$$(\lambda' - \lambda)^T B_k^d (\lambda - \lambda') \geq 0. \tag{11}$$

Since B_k^d is positive semi-definite, we also have

$$(\lambda' - \lambda)^T B_k^d (\lambda' - \lambda) \geq 0. \tag{12}$$

Then combining (11) and (12) yields $(\lambda' - \lambda)^T B_k^d (\lambda' - \lambda) = 0$, which means that $\lambda' - \lambda \in \ker B_k^d$ by the assumption on B_k^d. Therefore $L_k(\lambda) = L_k(\lambda')$, ending the proof. □

Remark 1. This property is close to the F-uniqueness property for variational inequalities, exposed in (Facchinei and Pang, 2003, Section 2.3.1, p. 162). Since we deal with AVI, it is simpler to derive directly the result rather than checking the conditions in the aforementioned book.

Note that the properties remain valid for systems with more than two sliding variables.

2.5. Computation of the control input

Moving on to the actual computations, since λ takes values in a compact convex set, a solution to the AVI (10) with any matrix B_k^d can be computed using the algorithm proposed in Cao and Ferris (1996), implemented in the SICONOS software package Acary et al. (2014). Since the AVI (10) has dimension 2, it is also possible to find the solution by enumeration, that is since λ_1 and λ_2 take value in $\{1\}$ or $\{-1\}$ or $[-1,1]$ we can test the 9 possible cases and pick one that is satisfactory. This approach can also be heuristically refined, as we shall see later. *To sum up, the proposed controller is non-anticipative and the sliding variables $\widetilde{\Sigma}_{k+1} = (\widetilde{\sigma}_{k+1}, \widetilde{\dot{\sigma}}_{k+1})^T$ are always uniquely defined.* A Matlab implementation of the solver by enumeration can found in Appendix C in Huber (2015).

3. Experimental setup and control strategy

3.1. System dynamics, actuators and sensors

We start with a description of the physical system, actuators and sensors as shown in Fig. 1. The electropneumatic system of the IRCCyN lab (École Centrale de Nantes, France), depicted on Fig. 2, has two actuators. On the left-hand side, there is a double acting electropneumatic actuator (the "main" one) controlled by two servodistributors and composed of two chambers denoted P and N. The piston diameter is 80 mm and the rod diameter is 25 mm. With a source pressure equal to 7 bars, the maximum force developed by the actuator is 2720 N.

Fig. 1: Picture of the electropneumatic system.

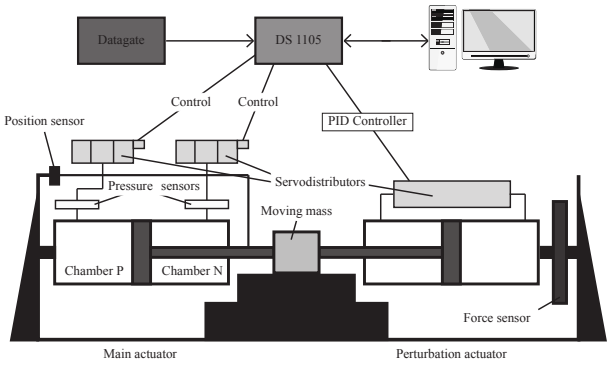


Fig. 2: Schematic of the electropneumatic system

The air mass flow rates entering the chambers are modulated by two three-way servodistributors. The pneumatic jack horizontally moves a load carriage of mass M. This carriage is coupled with the second electropneumatic actuator, the "perturbation" one, on the right-hand side. The goal of the latter is to impress a dynamic load force on the main actuator. This actuator has the same mechanical characteristics as the main one, but the air mass flow rate is modulated by a single five-way servodistributor. The control variable u is constrained to take values between -10 and 10 volts.

Under some assumptions detailed in Shtessel et al. (2012), the plant dynamics can be written as a nonlinear system affine in the control input $(u_P \ u_N)^T$, with u_P (resp. u_N) the control input of the servo distributor connected to the P (resp. N) chamber. The model is divided in two parts: the first two equations describe the pressure dynamics in each chamber, and the motion of the piston is given by the last two equations. There is a single control objective: to force the load position to track a reference trajectory. Therefore, we set $u \coloneqq u_P = -u_N$. Finally the dynamics of the electropneumatic experimental setup is

$$\begin{cases} \dot{p}_P = \dfrac{\kappa r T}{V_P(y)}[\varphi_P + \psi_P u - \dfrac{S}{rT} p_P v] \\ \dot{p}_N = \dfrac{\kappa r T}{V_N(y)}[\varphi_N - \psi_N u + \dfrac{S}{rT} p_N v] \\ \dot{v} = \dfrac{1}{M}[S\,(p_P - p_N) - b_v v - F] \\ \dot{y} = v, \end{cases} \tag{13}$$

with p_P (resp. p_N) the pressure in the P (resp. N) chamber, y and v being the position and velocity of the load. The constant κ is the polytropic index, r the ideal gas constant, T the temperature (supposed the same inside and outside the chambers) and b_v the viscous friction coefficient. The volumes in each chamber are V_P and V_N, both depending on the actuator position y. The constant piston section is S. The external force applied by the perturbation actuator is denoted by F. Finally, φ_X and ψ_X (X being P or N) are both 5^{th} order polynomial functions versus p_X as given in Sesmat and Scarvarda (1996), that characterize the mass flow rate q_X in the chamber X in the following way

$$q_X = \varphi_X(p_X) + \psi_X(p_X, \mathrm{sgn}(u_X))u_X.$$

The sources of uncertainty can be the polytropic index κ, the mass flow, the temperature T, the mass M, the viscous friction coefficient b_v and the disturbance force F. They can be modeled by additive bounded functions added to the nominal part of each parameter. As an example, the mass M can be viewed as the sum of a nominal part and an uncertain one: $M =: M_n + \Delta M$, where ΔM is a bounded uncertainty and M_n the nominal value. Also the position y, the pressures p_P, p_N are available but both the speed v and acceleration are computed using a filtered differentiator given in frequency domain by

$$D(s) = \frac{s}{1+\tau s}. \tag{14}$$

3.2. Control strategy

The presence of uncertainties motivates the use of a sliding mode control scheme, well-known for its robustness. A first study (Wang et al. (2015)) was already conducted for equivalent-based sliding mode controller, with a comparison between explicit, implicit and saturation methods. The experiments we present here were carried on with the discrete-time twisting controller presented in Section 2. Since we are interested in a trajectory tracking problem for the position of the load, y is the variable to be controlled. The desired position of the piston is y_d and the position error in the tracking problem is $e \coloneqq y - y_d$. The choice of this output leads to a relative degree 3 system. Hence, to bring the relative degree between the sliding variable and the control input to 2, so as to apply the twisting algorithm, the sliding variable is defined as

$$\sigma \coloneqq \alpha e + \dot{e}, \tag{15}$$

with $\alpha > 0$, a parameter which selection is one of the topic of Section 5. Its first and second derivatives are

$$\dot{\sigma} = \alpha\dot{e} + \ddot{e} \text{ and } \ddot{\sigma} = \alpha\ddot{e} + \dddot{y} - \dddot{y}_d,$$

where

$$\dddot{y} = \dddot{v} = \frac{1}{M}[S\,(\ddot{p}_P - \ddot{p}_N) - b_v\ddot{v} - \ddot{F}].$$

Using the relation in (13), we get

$$\dddot{y} = \frac{S\kappa rT}{M}\left(\frac{\varphi_P}{V_P} - \frac{\varphi_N}{V_N}\right) - \frac{S^2\kappa}{M}\left(\frac{p_P}{V_P} + \frac{p_N}{V_N}\right)v - \frac{b_v}{M^2}\left(S(p_P - p_N) - b_v v - F\right) - \frac{\dot{F}}{M} + \frac{S\kappa rT}{M}\left(\frac{\psi_P}{V_P} + \frac{\psi_N}{V_N}\right)u. \tag{16}$$

Let us define the following functions

$$\Phi \coloneqq \frac{S\kappa rT}{M}\left(\frac{\varphi_P}{V_P} - \frac{\varphi_N}{V_N}\right) - \frac{S^2\kappa}{M}\left(\frac{p_P}{V_P} + \frac{p_N}{V_N}\right)v - \frac{b_v}{M^2}\left(S(p_P - p_N) - b_v v\right) + \alpha\ddot{e} - \dddot{y}_d,$$

and

$$\Psi \coloneqq \frac{S\kappa rT}{M}\left(\frac{\psi_P}{V_P} + \frac{\psi_N}{V_N}\right). \tag{17}$$

Finally, the sliding variable dynamics is

$$\ddot{\sigma} = \Phi + \Delta\Phi + (\Psi + \Delta\Psi)u, \tag{18}$$

given that we consider that all the uncertainties are "additive", that is the vector fields can be written as the sum of a nominal part Φ and Ψ and uncertain terms $\Delta\Phi$ and $\Delta\Psi$. The latter include for instance the modeling errors and the action of the perturbation actuator like F and $\dot{F}$ in (16).

From Section 2.2, the control law of implicit controller is given by:

$$u_k = G(1\ \beta)\lambda \quad \text{and} \quad \lambda \in -\operatorname{Sgn}(\widetilde{\Sigma}_{k+1}), \tag{19}$$

now with $\beta = 2/3$ and $\widetilde{\Sigma}_{k+1} = (\widetilde{\sigma}_{k+1}, \widetilde{\dot{\sigma}}_{k+1})$ the value of the sliding variables given by discrete-time dynamics that we now derive. We need a relation akin to (7) for the computation of the control input value. Writing the sliding variable dynamics as a first-order ODE, we get

$$\dot{\Sigma} = A\Sigma + F + B\lambda \tag{20}$$

with $\Sigma = \begin{pmatrix}\sigma\\ \dot{\sigma}\end{pmatrix}$, $A = \begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}$, $B = \begin{pmatrix}0 & 0\\ G\Psi & \beta G\Psi\end{pmatrix}$ and $F = \begin{pmatrix}0\\ \Phi\end{pmatrix}$. We discretize the nonlinear terms Φ and Ψ using the explicit Euler scheme: we consider that $\Phi(t) = \Phi_k \coloneqq \Phi(t_k)$ and $\Psi(t) = \Psi_k \coloneqq \Psi(t_k)$ for $t \in [t_k, t_{k+1})$. For the last step in the discretization of (20), we use the ZOH method, which yields

$$\widetilde{\Sigma}_{k+1} = A^*\Sigma_k + F_k^* + B_k^*\lambda, \tag{21}$$

with $A^* \coloneqq e^{Ah} = \begin{pmatrix}1 & h\\ 0 & 1\end{pmatrix}$, $B_k \coloneqq G\Psi_k\begin{pmatrix}0 & 0\\ 1 & \beta\end{pmatrix}$ and $B_k^* \coloneqq \int_{t_k}^{t_{k+1}} e^{A\tau}B_k\mathrm{d}\tau = hG\Psi_k\begin{pmatrix}h/2 & \beta h/2\\ 1 & \beta\end{pmatrix}$, $F_k \coloneqq \begin{pmatrix}0\\ \Phi_k\end{pmatrix}$ and $F_k^* \coloneqq \int_{t_k}^{t_{k+1}} e^{A\tau}F_k\mathrm{d}\tau = \begin{pmatrix}h^2\Phi_k/2\\ h\Phi_k\end{pmatrix}$. Now we find our instance of (9) by using the relation $\widetilde{\Sigma}_{k+1} \in -\mathcal{N}_{[-1,1]^2}(\lambda)$ to get the generalized equation

$$0 \in \sigma_k + h\dot{\sigma}_k + \tfrac{h^2}{2}(\Phi_k + G\Psi_k[\lambda_1 + \beta\lambda_2]) + \mathcal{N}_{[-1,1]}(\lambda_1) \tag{22}$$

$$0 \in \dot{\sigma}_k + h\Phi_k + hG\Psi_k[\lambda_1 + \beta\lambda_2] + \mathcal{N}_{[-1,1]}(\lambda_2), \tag{23}$$

with unknowns λ_1 and λ_2. This is the problem solved to compute the control input value at each time instant t_k.

3.3. Properties of the closed-loop system

Let us check whether the results from Section 2.4 can be applied on this closed-loop system. It follows from Lemma 1 that this system always has a solution. For Lemma 2 to apply, we need the positive-definiteness of B_k^*. However this matrix does not enjoy this property: its symmetric part is

$$1/2(B_k^* + B_k^{*T}) = hG\Psi_k/2\begin{pmatrix}h & 1+\beta h/2\\ 1+\beta h/2 & 2\beta\end{pmatrix}.$$

Its determinant is $hG\Psi_k(2h\beta - (1+\beta h/2)^2)/2 = -hG\Psi_k(1-\beta h/2)^2/2$. Since Ψ_k is positive for all k (Girin, 2007, p. 48), the determinant is always negative and the matrix B_k^* is indefinite. We could try to reformulate the AVI (10) into an Linear Complementarity Problem (LCP) and see if the w-uniqueness (see Section 3.4 in Cottle et al. (2009)) property holds, but this does not work either. Nonetheless, the uniqueness of $\widetilde{\Sigma}_{k+1}$ holds for the twisting controller as we shall see with the following proposition.

Proposition 1. *The implicit twisting controller, defined by generalized equations (22) and (23), has a unique solution $\widetilde{\Sigma}_{k+1}$ and control input value u_k. Moreover if $\widetilde{\Sigma}_{k+1} \neq 0$, then the pair (λ_1, λ_2) is also unique.*

Proof. Despite not enjoying the positive semidefiniteness property, the matrix B_k^* has the following one: for any vector $x \in \mathbb{R}^2$, we have $B_k^*x = hG\Psi_k\begin{pmatrix}h/2\\ 1\end{pmatrix}(1\ \beta)x$. Therefore any vector in the range of B_k has both components of the same sign. Suppose that there exists multiple solutions to the problem AVI (10). Take two distinct solutions λ and λ' and define $\Sigma = L_k(\lambda)$, $\Sigma' = L_k(\lambda')$. Suppose that for all $i \in \{1, 2\}$, both Σ_i and Σ_i' are not zero at the same time. Let us denote by $\Delta\Sigma$, $\Delta\lambda$ the difference between any two distinct solutions and their image through L_k. From (21), we get

$$\Delta\Sigma = B_k^*\Delta\lambda = hG\Psi_k\begin{pmatrix}h/2\\ 1\end{pmatrix}(1\ \beta)\Delta\lambda. \tag{24}$$

The difference $\Delta\Sigma$ is in the range of B_k^* and we know that it implies that both its components have the same sign. Suppose that $\Delta\Sigma \geq 0$, to be understood component-wise and let $i \in \{1, 2\}$. The monotonicity of the Sgn multifunction gives us that for all $s_i \in \operatorname{Sgn}\Sigma_i$ and $s_i' \in \operatorname{Sgn}\Sigma_i'$,

$\langle \Delta\Sigma_i, s_i - s_i' \rangle \geq 0$. If $\Delta\Sigma_i > 0$, we infer that $s_i - s_i' \geq 0$. If $\Delta\Sigma_i = 0$, the fact that Σ_i and Σ_i' are not 0 at the same time prevent s_i and s_i' to both take values in $(-1, 1)$. Hence $\mathrm{Sgn}(\Sigma_i) = \mathrm{Sgn}(\Sigma_i')$ and is a singleton, implying that $s_i - s_i' = 0$. Therefore we have $s_i - s_i' \geq 0$ for all i. This implies that $\Delta\lambda \leq 0$ and $(1\ \beta)\Delta\lambda \leq 0$, which by (24) gives us $\Delta\Sigma \leq 0$. Along the same lines, starting with $\Delta\Sigma \leq 0$ gives that $\Delta\Sigma \geq 0$. Hence we infer that $\Delta\Sigma = 0$. Now suppose that for one $i \in \{1, 2\}$, both Σ_i and Σ_i' are zero. This implies $\Delta\Sigma_i = 0$ and by (24) that $\Delta\Sigma = 0$. Thus the uniqueness of Σ is established.

Now from (24), we deduce that the difference $\Delta\lambda$ between two solutions has to lie in $\ker B_k^*$. At the same time, the uniqueness of Σ means that both λ and λ' are in $-\mathrm{Sgn}\,\Sigma$. The latter is not a singleton only when either Σ_1 or Σ_2 is 0. In this case, and when $\Sigma \neq 0$, we have $\lambda - \lambda' \in \{(x, 0) \cup (0, y) \mid (x, y) \in [-2, 2]^2\} =: S_b$. From the expression of B_k^*, we know that $\ker B_k^* = \mathrm{span}\,(\beta, -1)^T \subset \{(x_1, x_2) \in \mathbb{R}^2 \mid 0 < x_1 < -x_2 \text{ or } 0 > x_1 > -x_2\} =: K$, given that $0 < \beta < 1$ due to the conditions in (2). But $K \cap S_b = \{0\}$ and thus $\ker B_k^* \cap S_b = \{0\}$, which gives the uniqueness of $\lambda = (\lambda_1, \lambda_2)$. The uniqueness of u_k comes from the expression $u_k = G(1\ \beta)\lambda$. Since any element of $\ker B_k^*$ is orthogonal to $(1\ \beta)^T$, uniqueness of the control input value follows and the proof is now complete. □

Remark 2. The same result holds in the continuous-time twisting algorithm (3), where the selections $\lambda_1 \in -\mathrm{Sgn}(\sigma)$ and $\lambda_2 \in -\mathrm{Sgn}(\dot{\sigma})$ are uniquely defined, except when $u = 0$. In this case the values lie on the segment defined by $\lambda_1 + \beta\lambda_2 = 0$ and $\lambda_1 \in [-1, 1]$. It is also noteworthy that this segment is also given by $\mathrm{span}\,(\beta\ \ -1)^T \cap [-1, 1]^2 = \ker B_k^* \cap [-1, 1]^2$. This is related to the fact that B and B_k^* have the same nullspace.

Remark 3 (Closed-loop Lyapunov stability). First remember that this paper focuses on the different behaviors observed with either the explicit or the implicit discretization. Some preliminary results for the stability analysis of the implicitly discretized twisting controller can be found in Acary et al. (2012) and in (Huber, 2015, Section 2.3). Part of the difficulty for analyzing this system is the lack of result on AVI with a matrix which does not enjoy the positive-semidefiniteness property. However, the ongoing research effort shows promising results, as the ones in the aforementioned references. For instance it is proved in Huber (2015) that a slight modification of the way the control signals λ_1 and λ_2 are computed, allows to guarantee the global asymptotic stability in a finite number of steps, of the discrete-time closed loop system. Moreover the discrete Lyapunov function is quite similar to the continuous-time Lyapunov function used in Orlov (2005), which confirms that the implicit discretization allows to remain as close as possible to the continuous-time system. However, the developments are too long to be included in this paper, which rather focuses on one two issues: the existence and uniqueness of a control signal at each sampling time, and the comparison between the explicit and the implicit methods. It is clear from the experimental results in Section 4 that the implicit twisting algorithm that we implemented, supersedes the explicit one.

4. Experimental results

This section is devoted to the analysis of the experimental results obtained on the electropneumatic setup. Recall that the control objective is to make the position of the piston track a sinusoidal trajectory. In the following, the desired trajectory is

$$y_d \coloneqq 40 \sin(0.2\pi t) \quad \text{(in cm)}.$$

The controller was implemented as a Simulink model and then transferred onto a DS1005 dSpace board. We were able to get results with the sampling period h in the range $[3, 100]$ms and with the gain G in the range $[10^{-2}, 10^7]$. The sliding surface parameter α and the two filtered differentiator time constants (in (14)) require proper tuning for each sampling period. They can drastically alter the performance of the controller. Since it appears that both have to be tuned together, preliminary values were obtained using simulations, with a selection based on the average tracking error, and were later refined on the plant. Section 5 is dedicated to the tuning of those parameters and to the analysis of their influence.

We now present results for two criteria: the tracking accuracy and the chattering magnitude on both the input and the output. In each case, we first compare the explicit and implicit controllers, before analyzing in more depth the performance of the implicit one.

4.1. Tracking accuracy

The tracking error $e = y - y_d$ is the quantity we aim to minimize through the twisting controller. Recall that due to the high relative degree of the system, the controller does not bring e to 0 in finite time, but rather $\sigma = \alpha e + \dot{e}$. Once the sliding phase $\sigma = 0$ occurs, the convergence of e to 0 is then exponentially fast if $\alpha > 0$. This parameter controls the speed of convergence: the bigger α is, the faster the error decreases, in continuous time.

To measure the accuracy of the tracking, we compute the average of the absolute value of the error over an interval of 60s. We call this quantity the average tracking error and we denote it by $\bar{e}$. Its analytical formula is

$$\bar{e} \coloneqq \sum_{k=1}^{N} \frac{|e(t_k)|}{N} \quad \text{with} \quad t_N - t_1 = 60\text{s}. \tag{25}$$

On Fig. 3, the average tracking error with both the implicit and explicit controllers is displayed for different sampling periods. The implicitly discretized controller clearly yields better performance than the explicit one, for each sampling period where the comparison is possible. Indeed it was not possible to get reliable data for large sampling

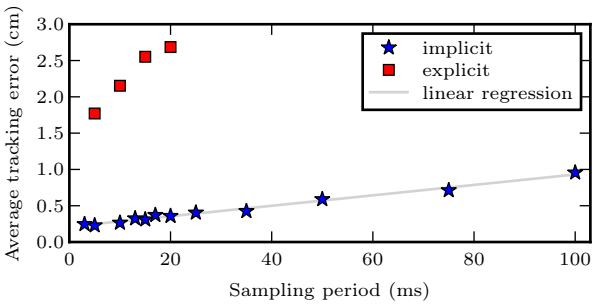


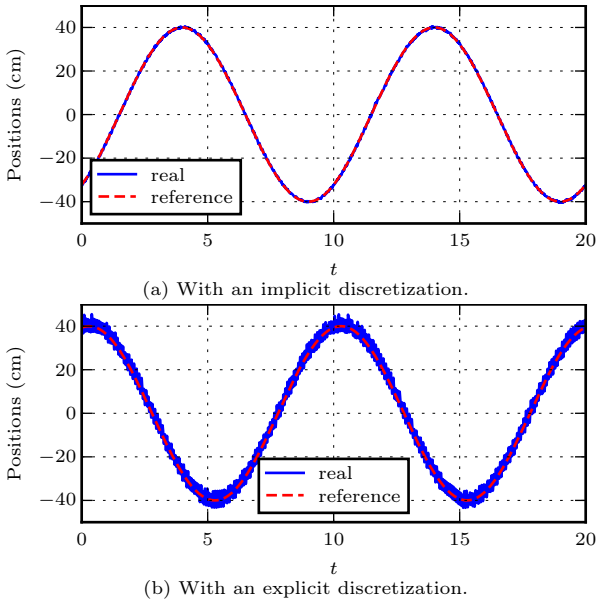
Fig. 3: Evolution of the average tracking error $\bar{e}$ with respect to the sampling time for both implicit and explicit discretizations. The gain used in every capture was $G = 10^5$.

periods with the explicit controller, since the plant was becoming unstable with sampling period larger than 20ms. The average tracking error appears to increase linearly with h, or in other word it is in $\mathcal{O}(h)$. This is underscored by the linear regression plotted on Fig. 3. This may be surprising since we use a second-order sliding mode controller and the order should be $\mathcal{O}(h^2)$. However, recall that $\sigma = \alpha e + \dot{e}$, with the derivative being computed by a simple filtered differentiator, remember (14). Looking at the C code generated using the Real-Time Workshop Toolbox, we can see that the approximated derivative $\tilde{v}$ of y is computed as the output of the following LTI system:

$$\begin{cases} a_k &= Aa_{k-1} + y_k \\ \tilde{v}_k &= Ca_k + Dy_k \end{cases},$$

with $A = -\tau^{-1}, D = \tau^{-1}$ and $C = -\tau^{-2}$, τ being the time constant in (14). This one-step approximation is of order h. Hence in (22), the term σ_k is known with a precision only in $\mathcal{O}(h)$, which can be seen as a non-matching perturbation. Most of the time, in this tracking problem, the control action tries to bring $\widetilde{\sigma}_{k+1}$ to 0, see Fig. 13 at the end of this section. The equation (22) is then the one used for the computation of the control input, propagating the error. This problem might be alleviated by the use of another differentiator, like the one proposed in Levant (1998).

Let us continue with more detailed results for a specific sampling period: $h = 10$ms. On Fig. 4a and 4b, the real and desired trajectories are depicted with, respectively, an implicit and an explicit controller. On Fig. 4a, the tracking is very accurate: at the given scale, the real position and the desired one are very close to each other. On the other hand, on Fig. 4b, the chattering of the real trajectory is visible in the form of a boundary layer around the reference trajectory. Therefore the output chattering has been drastically reduced with the use of an implicit controller. Turning our attention to the control input, Fig. 5a and 5b illustrate the evolution of this quantity in the implicit and explicit cases. In the first case, the control values are in the range $[-3, 3.3]$V, which is well inside the constraints $u \in [-10, 10]$V. Although the control is affected by the noise from the measurements, there is an underlining periodical signal, which is also seen on simulation results. The root cause of the oscillations is likely to be the approx-



(a) With an implicit discretization.

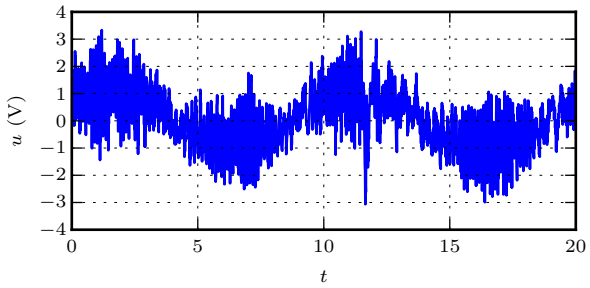
(b) With an explicit discretization.

Fig. 4: Real and desired position trajectories with $h = 10$ms and $G = 10^5$.

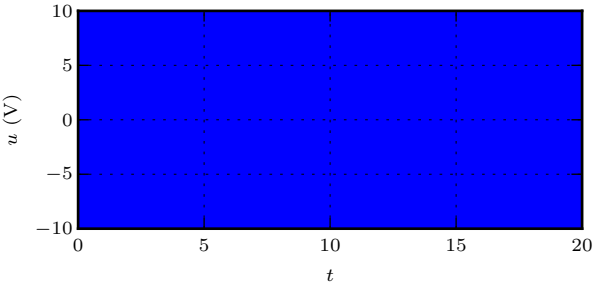
imations done to get the linearized discrete-time model in (21). It is difficult to analyze the data on Fig. 5b since the control input is switching at a very high frequency between the 2 extremal values -10 and 10 volts, sign of a chattering input. It is pretty clear that the main source of chattering is the explicit discretization of the controller.

Let us finish with the tracking error measured with the same two twisting controllers, as shown on Fig. 6. Comparing the ranges, we can see that in the implicit case (Fig. 6a), the tracking error is one order of magnitude smaller than in the explicit case (Fig. 6b). The spike on Fig. 6a around $t = 12$s is due to the action of the second actuator, which periodically switched its force acting on the moving mass from 1000N to -1000N and vice-versa. We further analyze the chattering in the second part of this section.

Having exposed the superiority of the implicit discretization with respect to the explicit one, let us further present the good performance that it yields. Firstly it is possible to increase the sampling period while keeping a good tracking and a system stable in practice. Fig. 7 illustrates this fact: even with a sampling period of **100**ms, the tracking takes place, although with degraded performance compared to the one on Fig. 4a. However the average tracking error is still better than with an explicit controller with a sampling period one order of magnitude smaller as shown on Fig. 3 and 4b. Another very nice feature of the implicit discretization is the fact that the control input value is computed as a selection of a set-valued term, being the solution of a generalized equation as in (9). One implication is that the gain just needs to be large enough with respect to the perturbation to ensure the robustness (remember (4)) but a further increase in the gain does not harm the performance. This is illustrated on Fig. 8, where we display data obtained in the following way: the exper-

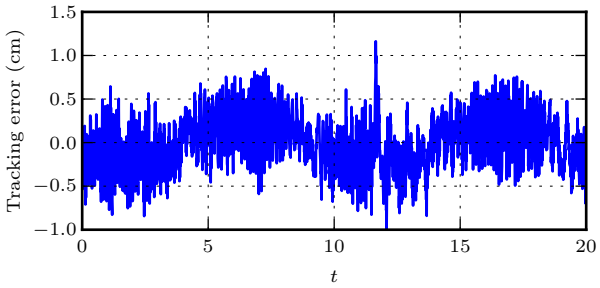


(a) With an implicit discretization.

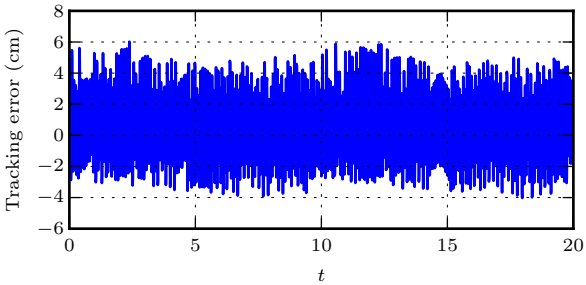


(b) With an explicit discretization.

Fig. 5: Evolution of the control input u for both implicit and explicit discretization with $h = 10$ms and $G = 10^5$.



(a) With an implicit discretization.



(b) With an explicit discretization.

Fig. 6: Evolution of the tracking error for both implicit and explicit discretization with $h = 10$ms and $G = 10^5$.

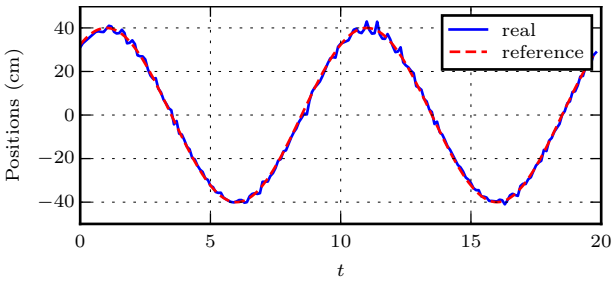


Fig. 7: Real and desired positions with an implicit discretization, $h = 100$ms and $G = 10^5$.

iment is run with the same controller 10 times, increasing the gain 10 fold each time, from 10^{-2} to 10^7. This was repeated for 3 different sampling periods. On Fig. 8, both the average tracking error $\bar{e}$ and the amplitude of the control input are plotted versus the gain G. For each sampling period, the average tracking error varies only by less than 5%, which is solely due to the noise in the plant. The random evolution, with respect to the gain further supports this claim. Regarding the amplitude of the control input, we compute it as the mean of the top 5% values of $|u_k|$, to which the 10 top values are discarded, as to remove any outlier. Again, we see only random variation when the gain is increased. To get a closer look, we have on Fig. 9 the implicit signum selections and on Fig. 10 the control inputs for the two extremal values of gain: 10^{-2} and 10^7. We call *implicit signum selection* the quantity

$$\lambda_1 + \beta\lambda_2, \tag{26}$$

where λ_1 and λ_2 are solution to (22) and (23). Multiplied by the gain G, it is equal to the control input value, see (19). The shape of the implicit signum selection is similar with both gains, however the range of the values is $[-0.08, 0.1]$ with $G_a = 10^{-2}$ whereas it is $[-0.6\ 10^{-10}, 10^{-10}]$ with $G_b = 10^7$. The ratio between the extremal values is close to G_a/G_b. Now moving on to the control input on Fig. 10, it does not change much: with both gains, the control input u is in the range $[-3, 4.5]$. As long as the gain G is large enough, the control input does not change much. The loose coupling between the control input and the gain is only possible with an implicit discretization, which enables us to compute the control input value as a selection. With an explicit discretization, it is well-known that the increase of the gain eventually leads to an increase in the control input and therefore an increase for both input and output chattering. The insensitivity of the discontinuous controller with respect to the increase in the gain has also been verified for the ECB-SMC controller in Wang et al. (2015). This is an expected property given the use of Filippov's framework. Let us switch focus on the chattering for the rest of this section.

4.2. Input and output chattering

We propose to characterize the chattering of a variable with the variation of the associated signal. Given a real-valued step function $f(\cdot)$, an interval $[t, T]$ of the real line and a sequence $\{t_k\}_{k\in\mathbb{N}^*}$ consisting of the time instants where the value of the function changes, its variation on

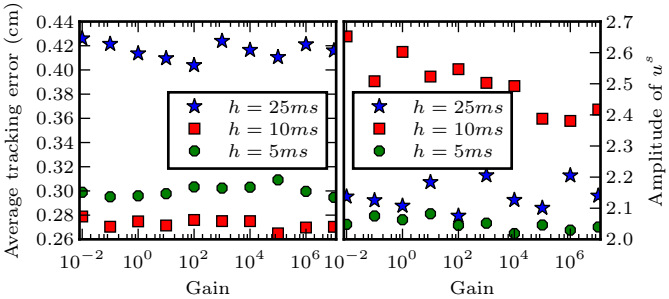


Fig. 8: Evolution of the average tracking error and the control input amplitude when the gain G in (19) varies for 3 different sampling periods.

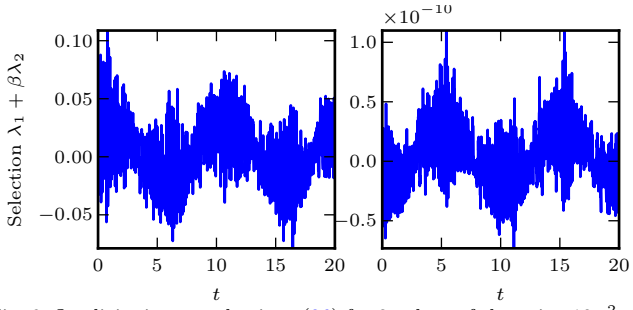


Fig. 9: Implicit signum selections (26) for 2 values of the gain: 10^{-2} and 10^{7} and with a sampling period of 10ms.

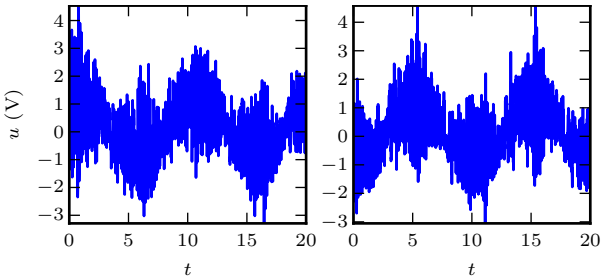


Fig. 10: Control inputs for two values of the gain: 10^{-2} and 10^{7} and with a sampling period of 10ms.

$[t, T]$ is defined as

$$\mathrm{Var}_t^T(f) \coloneqq \sum_k |f(t_k) - f(t_{k-1})|,$$

with $k \in \mathbb{N}^*$ such that $t_k \in (t, T]$ and t_k are the time instants where the control input value changes. Though this quantity is not commonly used in Control Engineering, it provides a nice characterization of the chattering on either the control input or the sliding variables. We pay attention to both input and output chattering, since the first one contributes to the second and it can also induce rapid wear of actuators, especially if they are mechanical ones. Furthermore, it may also be linked to the energy consumption of the actuator. As before, we present the

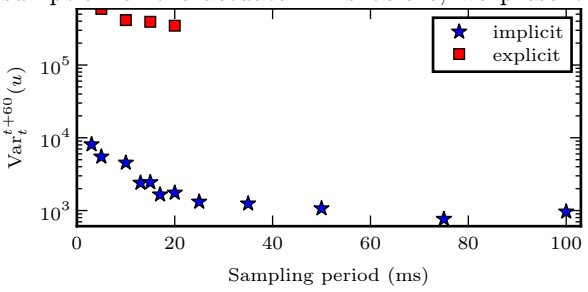


Fig. 11: Evolution of the control input variation and the error variation with respect to the sampling time for both implicit and explicit discretizations. The gain used in every capture was $G = 10^5$.

evolution of the control input chattering with respect to the sampling period for both implicit and explicit controllers. From Fig. 11, we can infer that the trend in both cases is a decrease of the variation with an increase in the sampling period. Again the implicit controller performs much better, having a control input variation two orders of magnitude smaller than the explicit one. This reduced chattering can also be assessed on site with a huge reduction of the noise made by the actuators[1].

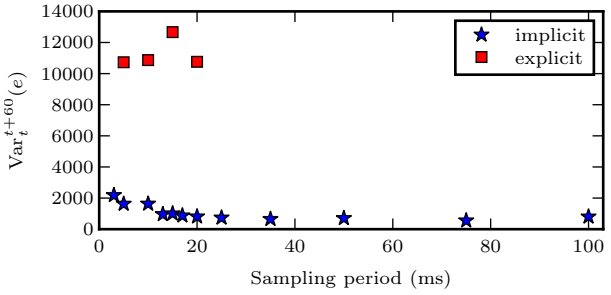


Fig. 12: Evolution of the error variation with respect to the sampling time for both implicit and explicit discretizations. The gain used in every capture was $G = 10^5$.

Moving on to the output chattering, the same conclusion follows: the implicit method performs better than the explicit one, this time by an order of magnitude (see Fig. 12). This means that the output chattering is notably reduced. Indeed a bang-bang type control input, like the one the explicit discretization yields, tends to change the sign of the sliding variable very frequently. This leads to a large variation of the error, with respect to the variation with an implicit controller. At the same time, this behavior does not yield a better tracking, as illustrated by Fig. 3.

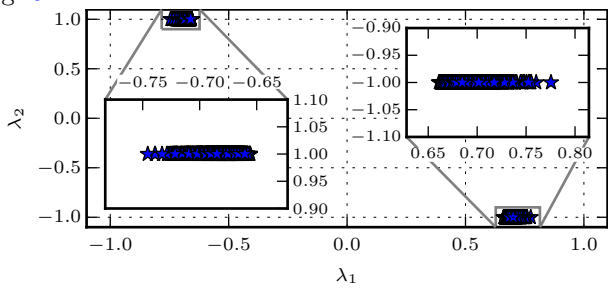


Fig. 13: Values of the two variables λ_1 and λ_2 solution to (22) and (23), with $G = 10^{-2}$ and $h = 10$ms.

Let us finish with an analysis of the values taken by λ_1 and λ_2, solutions of the generalized equations (22) and (23). They are the selections of the set-valued inputs, that is the real values taken by the controller. On Fig. 13, we can see that λ_2 is equal to either 1 or -1 whereas λ_1 takes value in $(-.745, -.656) \cup (0.660, 0.776)$. Then from equations (22) and (23) we deduce that $\widetilde{\sigma}_{k+1} = 0$ and $\dot{\widetilde{\sigma}}_{k+1} \neq 0$. Therefore the control action tries to bring $\sigma(t_{k+1})$ to 0 at each time instant t_k, in which case $\mathcal{N}_{[-1,1]}(\lambda_1) = \{0\}$. Based on this observation, one sees that (22) is in fact the equation giving its value to the implicit selection $\lambda_1 + \beta\lambda_2$. This explains the propagation of the error in the computation of $\dot{e}$ from σ_k to the control input mentioned in Section 4.1. It also provides an heuristic for the computation of the control: after a short period of time the tracking takes place and then the controller always brings $\widetilde{\sigma}_{k+1}$ to 0. Hence if we solve the AVI problem from Section 2.2 by enumeration, we can firstly try the two cases where $\widetilde{\sigma}_{k+1} = 0$.

[1]The reader is invited to watch the videos at http://nullptr.fr/pages/videos.html

5. Parameters selection

We mentioned at the beginning of Section 4 that the tuning of the sliding surface parameter α and of the two filtered differentiator constants (τ_v and τ_a) is important and may drastically affects the closed-loop behavior. Let

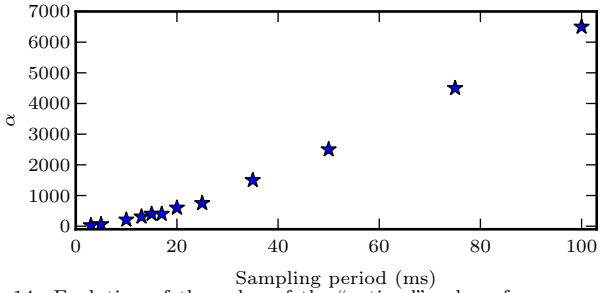


Fig. 14: Evolution of the value of the "optimal" value of α versus the sampling period.

us motivate the necessity of tuning α by looking at Fig. 14 to see how the value of α yielding the smallest average tracking error varies with the sampling period. With our experimental data, the selected values of α span from 25 for $h = 3$ms to 6500 for $h = 100$ms. With an explicitly discretized controller, it is much harder to see how a change in the value of the triplet (α, τ_v, τ_a) influences the closed-loop behavior. Those parameters also impact the performance, but the bad discretization is affecting it too much. This does not mean that those parameters should not be tuned with an explicit discretization, just that such tuning does not bring any notable improvement due the way the control input is computed. With an implicit discretization, the control input value computation is no longer the weakest component of the control loop: a change in one of the aforementioned parameters may influence the performance. Hence, the parameter tuning which follows deal only with the implicit controller. We now present the procedure used to tune those parameters and then propose a model explaining how α influences the closed-loop performance.

5.1. Parameters description

First let us recall some basic facts about the parameters α, τ_v and τ_a. In continuous time, the sliding surface parameter α influences the error dynamics once the origin is reached. In this case, the ODE in (15) becomes $\alpha e + \dot{e} = 0$ and the exponential decrease is controlled by the value of α. Therefore, the value of α impacts only the transient phase and not the steady state regime. Regarding the filtered differentiators, the constants (τ_v, τ_a) on the low-pass filter should be tuned such that the dynamics of the closed-loop system are preserved as much as possible while removing as much measurement noise as possible. It looks reasonable to assume that when the sampling period decreases, the high frequency part of the dynamics is richer since the control input changes more frequently. Hence we expect the optimal value of those coefficients to decrease with the sampling period. Note that even in simulation, the closed-loop system with the implicit controller is giving good results only for a given range of sampling period. Chattering can suddenly appear, which is not at all consistent with the theory presented in Section 2 and 3.2.

5.2. Parameters selection procedure

The procedure used to get the values for the triplet (α, τ_v, τ_a) has two steps: first we rely on simulation results to restrict the space where this triplet gives good performance. To reduce the number of possibilities, we set $\tau_v = \tau_a$. A set of simulation has to be run for each value of the sampling period. The metric used to measure the performance of each triplet is the average tracking error. The simulator is implemented in Simulink and consists of a nominal model of the plant with the same feedback loop as the one used during the experiments. Typical results (here for $h = 15$ms) are given on Fig. 15. First note the

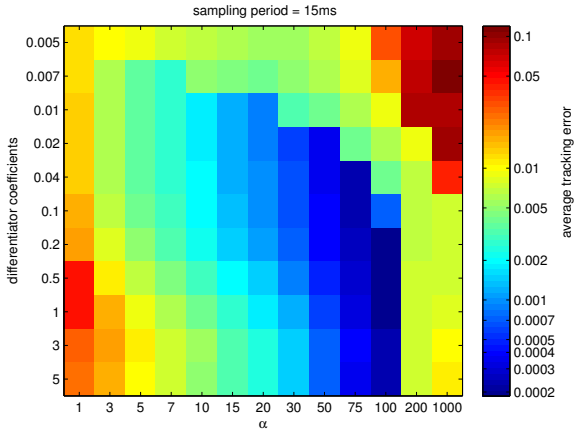


Fig. 15: Heat map of the precision for various values of the differentiator coefficients and the sliding coefficients. The sampling period was set to 15ms.

important variation of the performance, which spans over 3 orders of magnitude. An important observation for the online tuning is that α is the parameter that impacts the performance the most. Hence, we can start the online procedure by fixing τ_v and τ_a to some "large" values and tune only α. This parameter is tuned online by considering the noise produced by the plant, which can be linked to the actuator chattering, on this setup. Once we have a satisfactory value for the latter, it is then easier to change one parameter at a time and see whether it yields any improvement. To illustrate this, consider the plots on Fig. 16: with a value of α too high, the noise affects the control input and we have chattering. When α is too small, the system fails to follow the dynamics of reference signal: the tracking error does not switch sign often.

5.3. Evolution of the tracking performance w.r.t. α

Since α is the parameter whose variation impacts the most the average tracking error both in simulation and experiments, we focus on it for the rest of the section. Let us present some experimental data obtained with $h = 5$ms and $\alpha \in [10, 130]$. Two examples of the relation between the average tracking error and the input chattering are given on Fig. 17: after a quick improvement in the average tracking error, the best value is obtained. Then the

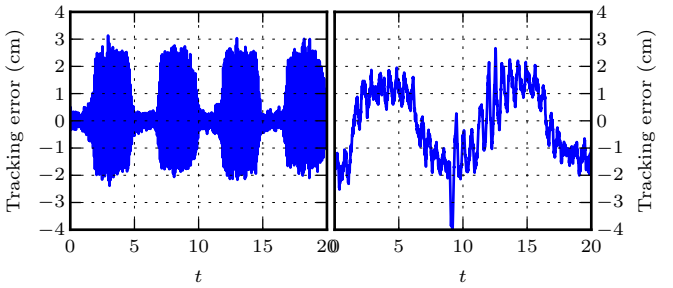


Fig. 16: Tracking error with $\alpha = 130$ (left) and $\alpha = 10$ (right), with parameters $\tau_v = \tau_a = .5$ and $h = 5$ms.

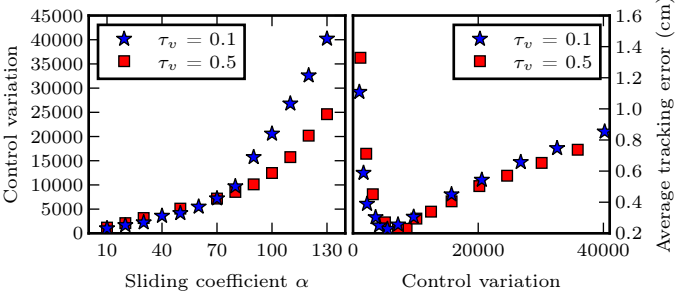


Fig. 17: Control variation relationship with the sliding parameter α and the average tracking error.

tracking error increases with the control input variation. It is also apparent on the left plot on Fig. 17 that the control variation is increasing with the sliding coefficient α. Thus a good tuning strategy is to increase α until the average tracking error seems to deteriorate and the chattering increases. Having narrowed the interval for the optimal α, we can then try to track it. Finally, looking at the evolution of the average tracking error with respect to α as displayed on Fig. 18, we see that there exists a value of α which gives the smallest average tracking error. Also note the two asymptotic behaviors: for small values of α, the evolution of the average tracking error looks like $1/\alpha$ and for large values of α the tracking error increases quasi-linearly with α. We also display the tracking error with an explicit controller: for any value of α we tested, it underperforms w.r.t. an implicit controller. The almost flat

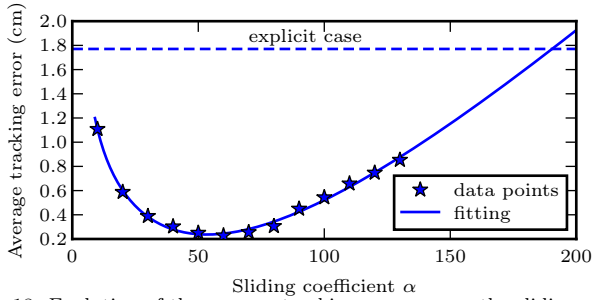


Fig. 18: Evolution of the average tracking error versus the sliding coefficient α, with filtered coefficient values: $\tau_v = 0.1, \tau_a = 0.05$. The line is the graph of a function fitting the data.

part of the curve around the best value of α shows the robustness of the overall scheme with respect to α, a desired property.

5.4. Analysis of the influence of α on the tracking error

Let us provide some analytical basis for the two trends we just mentioned and a proposal for further improvements. To do so, the experimental data is used to fit the function $\frac{ax^3+bx^2+cx+d}{x^2+ex}$ which captures the two asymptotical behaviors, as shown on Fig. 18.

Now that our perceived trends are backup by this fitting, let us formulate an explanation for them. Firstly, we focus on the behavior when α is small: in this case, we can see on Fig. 19 that the mean absolute value of σ is constant

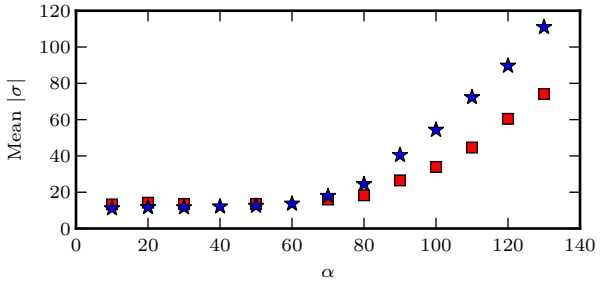


Fig. 19: Evolution of the mean absolute value of the sliding variable σ versus the sliding coefficient.

for α between 10 and 60. This is also true for the mean absolute value of the control input on Fig. 20. We also claim that the system is in the discrete-time sliding phase since the control bounds are never hit (remember the right plot on Fig. 17). Therefore, the system is trying to bring the sliding variable value to 0 in one sampling period. This gives the relation: $\sigma_{k+1} = \alpha e_{k+1} + \dot{e}_{k+1} = P_k$ where P_k accounts for all the noise and unmodeled dynamics effects. We approximate $\dot{e}_{k+1}$ by $\frac{e_{k+1}-e_k}{h}$, which leads to

$$e_{k+1} = \frac{e_k}{1+\alpha h} + \frac{h}{1+\alpha h} P_k. \tag{27}$$

This relation implies that the error e_k forms an arithmetico-geometric sequence, with the common ratio $r = P_k/\alpha$. In the following the expected value of the error e_k is computed under the hypothesis that the expected value of P_k is independent of k and is denoted by $\mathbb{E}[|P|]$. The quantity we display on Fig. 18 is $\sum_k |e_k|/N$. From (27), we get the following bounds:

$$\frac{h}{1+\alpha h} P_k - \frac{1}{1+\alpha h}|e_k| \leq |e_{k+1}| \leq \frac{1}{1+\alpha h}|e_k| + \frac{h}{1+\alpha h} P_k.$$

Summing N times this relation, ignoring the terms at the power N, and working with the expectation of the error yields

$$N\frac{\mathbb{E}[|P|]}{1+\alpha+h^{-1}} - \left(|e_0| + \frac{\mathbb{E}[|P|]}{1+\alpha+h^{-1}}\right)\left(\frac{1+\alpha h}{2+\alpha h}\right) \leq \sum_k |e_k| \leq \left(|e_0| - \frac{\mathbb{E}[|P|]}{\alpha}\right)\left(\frac{1+\alpha h}{\alpha h}\right) + N\frac{\mathbb{E}[|P|]}{\alpha}.$$

Neglecting the constant terms divided by N, we finally get

$$\frac{\mathbb{E}[|P|]}{1+\alpha+h^{-1}} \leq \sum_k \frac{|e_k|}{N} \leq \frac{\mathbb{E}[|P|]}{\alpha}$$

at the limit. This small derivation corroborates the data on Fig. 18 for small values of α, where the asymptotical behavior is like $1/\alpha$.

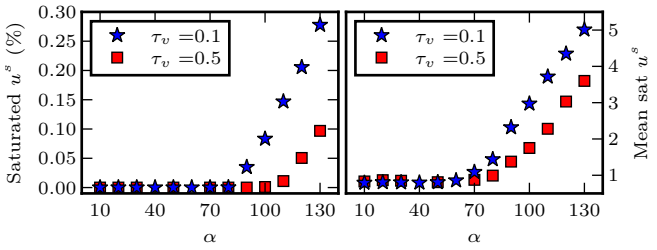


Fig. 20: Evolution of the mean absolute value of the control input and the percentage of control input values that are saturated versus the sliding parameter α.

For large values of α, the average tracking error looks like a linear function of the sliding parameter. This appears to be also the case for the mean absolute value of the control input, see the left plot on Fig. 20. Note also the saturation of the control input: on the right plot on Fig. 20, for values of α greater than 90 (or 110 for the other differentiator parameters), u is hitting its bound, and it occurs more frequently as α increases. We link this to the loss of homogeneity in the discrete-time controller, which is far beyond the scope of this paper. It is noteworthy that this degradation of performance was already seen in simulation (Huber, 2015, Section 4.1.3). Hence we suspect that this deterioration is related to the approximation on either the computation of the derivatives or the discrete-time model (21). Further investigations could include using a better differentiator and a more accurate discrete-time model to enable the use of higher values of α and if the average tracking error is improved.

6. Comparison to the classical first-order sliding mode controller

Let us present some results with an implicit equivalent control based sliding mode controller (ECB-SMC) instead of a twisting controller. For a comparison between explicitly and implicitly discretized controller for the ECB-SMC, see Wang et al. (2015), where it is shown that the implicit controller gives much better results than the explicit one. The implicit controller in the first-order sliding mode case has the following structure:

$$u_k \in -G\,\mathrm{Sgn}(\sigma_{k+1}).$$

The relative degree between the output y and the input u forces us to define the following sliding surface:

$$\sigma = \ddot{e} + 2\xi\omega\dot{e} + \omega^2 e, \tag{28}$$

with two design parameters: ξ and ω. To keep the search of the best sliding surface trackable, we fixed $\xi = 0.7$ in an analogy to the second-order ODE analysis, to theoretically ensure the fastest convergence within a 5% boundary layer of the sliding manifold. The value of ω was then tuned online to provide the best performance. However for the sake of brevity we do not discuss here in depth: the same procedure as presented in Section 5.2 has been applied. The necessity of the tuning is backed up by Fig, 2 in Wang et al. (2015). In this reference, the values of ξ and ω stay the same for all sampling periods: the performance quickly degrades with a change in the sampling period. In our experiments, only ω changes; the values for different sampling periods are given in Fig. 21. We are able to

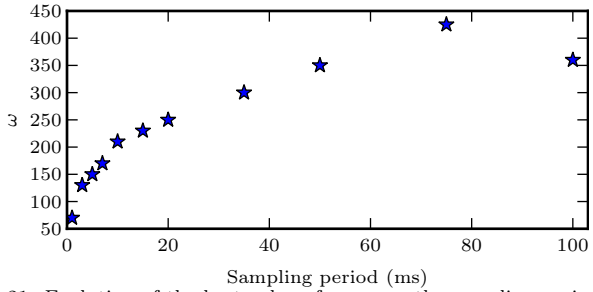


Fig. 21: Evolution of the best value of ω versus the sampling period for an implicit sliding mode controller.

provide very good performance for a substantially larger range of sampling periods: $[1, 100]$ms versus $[2, 15]$ms in the previous work.

The control scheme is as follows: the sliding surface (28) has the dynamics

$$\dot{\sigma} = \dddot{y} - \dddot{y}_d + 2\xi\omega\ddot{e} + \omega^2\dot{e},$$

which leads to dynamics close to the twisting case (18). The nominal version of this dynamics is:

$$\dot{\sigma} = \Phi' + \Psi u,$$

where $\Phi' := \dddot{y} - \Psi u + 2\xi\omega\ddot{e} + \omega^2\dot{e}$ and Ψ is the same as in (17). The discrete-time model is then derived using the same procedure as for the twisting controller. The nonlinear terms are approximated by constant terms over $[t_k, t_{k+1}]$ and hence we get the system

$$\begin{aligned}\sigma_{k+1} &= \sigma_k + h\Phi'(t_k) + h\Psi(t_k)u_k\\ u_k &\in -\mathrm{Sgn}(\sigma_{k+1}),\end{aligned}$$

which is also an AVI. Therefore the existence and uniqueness properties of the control input value can be checked by using the tools from Section 2. For the implementation of the controller, we used the code given in AppendixA, which turns out to be very simple since the sliding variable is scalar.

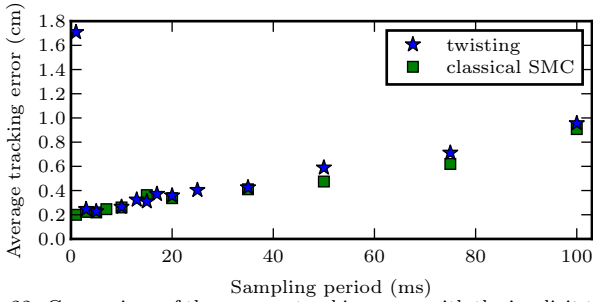


Fig. 22: Comparison of the average tracking error with the implicit twisting and the implicit sliding mode controller, for sampling periods in the range $[1, 100]$ms.

The resulting performance, again in term of the average tracking error (25) is displayed on Fig. 22, alongside the results obtained with the implicit twisting controller.

As with the latter, the controller is able to provide good performance. The relationship between the average tracking error and the sampling period appears to be linear. The implicit twisting and the ECB-SMC controllers yield very similar results on this experimental setup. Amongst the differences, one of the most prominent was the tuning of the sliding surface. Indeed on Fig. 22, for a sampling period of 1ms, the closed-loop system with the implicit twisting controller performs poorly. The average tracking error is one order of magnitude worse than with the sampling period 3ms. This is in our opinion due to the fact that we could not find a good set of parameter values (τ_v, τ_a, α) such that the system behaves well. The behavior of the closed-loop system was similar to the case where the parameters were not properly set. This illustrates the fact that with the twisting controller, the online tuning of the parameters was getting harder as the sampling period h decreased, to the point that we failed to tune them for $h = 1$ms. For the same sampling periods, the tuning with the implicit classical sliding mode controller was much easier. However for the largest sampling periods, the situation was reversed: the implicit ECB-SMC controller was harder to tune than the twisting one.

The data presented here have to be put into perspective: the performance of the closed-loop system is usually limited by the weakest component in the control loop. Our interpretation of the results obtained from this experiment is that the "limiting" component is not the controller, but rather the ones that generate the data used to feed it, like the filtered differentiators and the linearization scheme. Enhancing those parts of the controller scheme may yield better performance and might enable us to see a clear difference between the two controllers.

7. Conclusion

In this article we presented the results of a study of two discrete-time twisting controllers: the implicit and the explicit one. Extensive experiments were conducted in the context of a position tracking problem. The analysis of the data reveals that on this electropneumatic setup, the implicit twisting controller outperforms the explicit one on 3 criteria: the tracking error and both the input and output chattering. Despite the complexity of the control loop arising from the high relative degree, meaningful illustrations of theoretical results are provided, like the insensitivity with respect to an increase in the control gain. The implicit discretization allows to drastically reduce both the output and the input chattering, without modifying the controller structure compared to its continuous-time version. The other important contribution is the emphasis put on the choice of some design parameters. Tuning those greatly helps improving the results. We singled out the parameter α, introduced to cope with the relative degree 3 in the system, since this investigation may pertain to closed-loop systems sharing the same property. We also illustrate that a similar tuning has to be done with the EBC-SMC. This analysis also shows future directions to further enhance the performance: using Levant's differentiator and a better discrete-time dynamics. Video recordings of the experiments can be found online[1].

Bibliography

V. Acary and B. Brogliato. Implicit Euler numerical scheme and chattering-free implementation of sliding mode systems. *Systems & Control Letters*, 59(5):284–293, 2010.

V. Acary, B. Brogliato, and Y.V. Orlov. Chattering-free digital sliding-mode control with state observer and disturbance rejection. *Automatic Control, IEEE Transactions on*, 57(5):1087–1101, 2012.

V. Acary, B. Brémond, O. Huber, and F. Pérignon. An introduction to Siconos. Rapport Technique RT-0340, INRIA, 2014. URL http://hal.inria.fr/inria-00162911.

M. Cao and M. C. Ferris. A pivotal method for affine variational inequalities. *Mathematics of Operations Research*, 21(1):44–64, February 1996.

R.W. Cottle, J.-S. Pang, and R.E. Stone. *The Linear Complementarity Problem*. Number 60 in Classics in Applied Mathematics. Society for Industrial Mathematics, 2009.

F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer Series in Operations Research. Springer, 2003.

A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides*, volume 18 of *Mathematics and Its Applications*. Kluver Academic Publishers, 1988.

A. Girin. *Contribution à la commande non linéaire d'un système électropneumatique pour une utilisation aéronautique: application sur un benchmark dédié*. PhD thesis, Ecole Centrale de Nantes, France, December 2007. URL https://tel.archives-ouvertes.fr/tel-00207714v2.

O. Huber. *Analyse et implémentation du contrôle par modes glissants en temps discret*. PhD thesis, Université Grenoble Alpes, Mai 2015. URL https://hal.inria.fr/tel-01194430.

O. Huber, V. Acary, and B. Brogliato. Analysis of explicit and implicit discrete-time equivalent-control based sliding mode controllers. Rapport de Recherche RR-8383, INRIA, 2013a. URL http://hal.inria.fr/hal-00875209.

O. Huber, V. Acary, and B. Brogliato. Comparison between explicit and implicit discrete-time implementations of sliding-mode controllers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 2870–2875. IEEE, December 2013b.

A. Levant. Sliding order and sliding accuracy in sliding mode control. *International Journal of Control*, 58(6):1247–1263, 1993.

A. Levant. Robust exact differentiation via sliding mode technique. *Automatica*, 34(3):379–384, 1998.

Y.V. Orlov. Finite time stability and robust control synthesis of uncertain switched systems. *SIAM Journal on Control and Optimization*, 43(4):1253–1271, 2005.

A. Polyakov and A. Poznyak. Lyapunov function design for finite-time convergence analysis: "twisting" controller for second-order sliding mode realization. *Automatica*, 45(2):444–448, 2009.

S. Sesmat and S. Scarvarda. Static characteristics of a three way electropneumatic servovalve. In *Proceedings of the 12th Conference on Fluid Power Technology*, pages 643–645, Aachen, Germany, 1996.

Y. Shtessel, M. Taleb, and F. Plestan. A novel adaptive-gain *supertwisting* sliding mode controller: Methodology and application. *Automatica*, 48(5):759–769, 2012.

M. Taleb, A. Levant, and F. Plestan. Pneumatic actuator control: Solution based on adaptive twisting and experimentation. *Control Engineering Practice*, 21(5):727–736, 2013.

B. Wang, B. Brogliato, V. Acary, A. Boubakir, and F. Plestan. Experimental comparisons between implicit and explicit implementations of discrete-time sliding mode controllers: Toward input and output chattering suppression. *Control Systems Technology, IEEE Transactions on*, 23(5):2071–2075, September 2015.

AppendixA. MATLAB code for SMC

We present here the code used in the implementation of the implicit first order sliding mode controller in the case where the sliding variable is scalar. The controller is created in Simulink inside a "Matlab function" block, with the code written in the Matlab language. It is then translated into C and compiled for the targeted microcontroller by the real-time workshop toolbox. The code used to implement the discrete-time twisting controller can be found in Appendix C in Huber (2015).

```
function u_k = fcn(G, h, s_k, CBk)
toProj = -s_k/(G*CBk*h);
if toProj > 1.0, u_k = G;
elseif toProj < -1.0, u_k = -G;
else, u_k = G*toProj;
end
end
```

AppendixB. MATLAB code for the twisting algorithm

```
function [lambda1,lambda2] =
fcn(sigma, sigmaDot, K, beta, h, Psi, Phi, pL1, pL2)
%construct matrices
mat(2,:) = K*Psi*[h h*beta]; mat(1,:) = h/2*mat(2,:);
q = [sigma + h*sigmaDot + h*h/2*Phi; sigmaDot + h*Phi];
% first try the previous value
lambda = zeros(1, 2); lambda1 = 0; lambda2 = 0;
lambda(1) = pL1; lambda(2) = pL2;
if abs(lambda(1)) ~= 1.0
    lambda(1) = 1.0;
end
if abs(lambda(2)) ~= 1.0
    lambda(2) = 1.0;
end

nbIter = 0; nposIdxOld = 0; idxZero = 0;
nbIterMax = 9; alreadyDone = zeros(9, 1);
alreadyDone(lambda(1) + 2*lambda(2)+6) = 1;
posIdxOld = zeros(1, 2); oldLambdaV = lambda;

while nbIter < nbIterMax
% see if we can reach the origin
if (lambda(1) == 0) && (lambda(2) == 0)
  lambda = oldLambdaV; nposIdx = 1;
  posIdx = abs(lambda(1) + lambda(2))/2 + 1;
  posIdxOld = zeros(1, 2); nposIdxOld = 0;
else
if idxZero > 0
  if idxZero == 1 % try lambda(1) in [-1, 1]
    valU1 = -(q(1) + mat(1, 2)*lambda(2));
    if (abs(valU1) < mat(1, 1))
      lambda(1) = valU1/mat(1, 1);
      Sigma = q + mat*lambda';
      eps0 = abs(Sigma) < eps; Sigma(eps0) = 0.0;
      if sign(Sigma(2)) == -sign(lambda(2))
        lambda1 = lambda(1); lambda2 = lambda(2);
        return;
      end
    end
    lambda = oldLambdaV; nposIdxOld = 0;
    posIdx = abs(lambda(1) + lambda(2))/2 + 1;
    nposIdx = 1; posIdxOld = zeros(1, 2);
  else % try lambda(2) in [-1, 1]
    valU2 = -(q(2) + mat(2, 1)*lambda(1));
    if abs(valU2) < mat(2, 2)
      lambda(2) = valU2/mat(2, 2);
      Sigma = q + mat*lambda';
      eps0 = abs(Sigma) < eps; Sigma(eps0) = 0.0;
      if sign(Sigma(1)) == -sign(lambda(1))
        lambda1 = lambda(1); lambda2 = lambda(2);
        return;
      end
    end
    lambda = oldLambdaV; posIdxOld = zeros(1, 2);
    posIdx = abs(lambda(1) + lambda(2))/2 + 1;
    nposIdx = 1; nposIdxOld = 0;
  end
else % lambda is one of the vertex of K
    Sigma = q + mat*lambda';
    eps0 = abs(Sigma) < eps; Sigma(eps0) = 0.0;
    sLambda = sign(lambda)';
    sProd = sLambda.*sign(Sigma);
    posIdx = [-1, -1]; nposIdx = 0;
    if sProd(1)>0
      posIdx(1) = 1; nposIdx = 1;
    end
    if sProd(2)>0
      if nposIdx == 0
        posIdx(1) = 2;
      else
        posIdx(2) = 2;
      end
      nposIdx = nposIdx + 1;
    end
    if nposIdx == 0 % if this is true, we are done
      lambda1 = lambda(1); lambda2 = lambda(2);
      return;
    end
  end
end

% prepare next iteration
idxZero = 0; oldLambdaV = lambda; hasZero = 0;
for ii=1:nposIdx
  idx = posIdx(ii);
  if (nposIdxOld == nposIdx) && (((nposIdxOld >= 1) ...
    && (idx == posIdxOld(1))) || ((nposIdxOld >= 2) ...
    && (idx == posIdxOld(2))))
    lambda(idx) = 0; idxZero = idx; hasZero = 1;
  elseif hasZero == 0
    lambda(idx) = lambda(idx) - 2*sign(lambda(idx));
    lambda(idx) = lambda(idx)/abs(lambda(idx));
  end
end
nbIter = nbIter + 1; nposIdxOld = nposIdx;
posIdxOld(1:nposIdx) = posIdx(1:nposIdx);
if hasZero == 0 && (lambda(2) ~= 0)
  idxConfig = lambda(1) + 2*lambda(2) + 6;
elseif (lambda(1) == 0) && (lambda(1) == 0)
  idxConfig = 1;
```

```
else
  idxConfig = 6 + 2*(lambda(1)-1);
end
if (alreadyDone(idxConfig) == 0)
  alreadyDone(idxConfig) = 1;
else
  oldLambdaV = lambda;
  % find next available config
  if nbIter == nbIterMax
    break
  end
  jj = mod(idxConfig, nbIterMax) + 1;
  while 1
    if (alreadyDone(jj) == 0)
      alreadyDone(jj) = 1; break;
    else
      jj = mod(jj, nbIterMax) + 1;
    end
  end
  idxZero = 0;
  switch(jj)
    case 1, lambda = [0 0];
    case 2, lambda = [-1 0]; idxZero = 2;
    case 3, lambda = [-1 -1];
    case 4, lambda = [0 -1]; idxZero = 1;
    case 5, lambda = [1 -1];
    case 6, lambda = [1 0]; idxZero = 2;
    case 7, lambda = [-1 1];
    case 8, lambda = [0 1]; idxZero = 1;
    case 9, lambda = [1 1];
  end
end
end

end
```