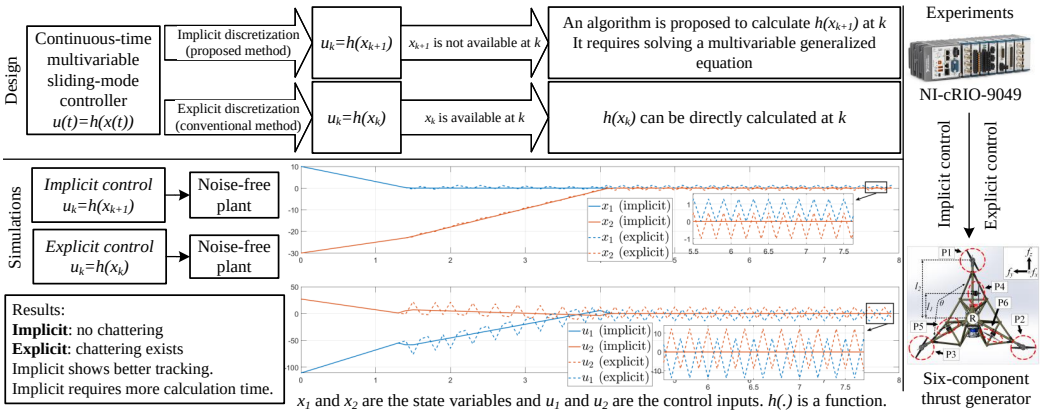
Euler implicit time-discretization of multivariable sliding-mode controllers

Mohammad Rasool Mojallizadeh, Félicien Bonnefoy, Franck Plestan, Mohamed Assaad Hamida, Jérémy Ohana

Graphical Abstract

Euler implicit time-discretization of multivariable sliding-mode controllers

Mohammad Rasool Mojallizadeh, Félicien BONNEFOY, Franck PLESTAN, Mohamed Assaad HAMIDA, and Jérémy OHANA



x_1 and x_2 are the state variables and u_1 and u_2 are the control inputs. $h(.)$ is a function.

Highlights

Euler implicit time-discretization of multivariable sliding-mode controllers

Mohammad Rasool Mojallizadeh, Félicien BONNEFOY, Franck PLESTAN, Mohamed Assaad HAMIDA, and Jérémy OHANA

- The implicit discretization of the sliding-mode controllers is extended to the multivariable case.
- An algorithm is proposed to solve the multivariable generalized equations that cannot be solved by conventional graphical interpretations.
- The proposed algorithm leads to a chattering-free discrete-time implementation without modifying the continuous-time control law.
- The finite-time convergence and gain insensitivity of the discrete-time control law have been addressed.

Euler implicit time-discretization of multivariable sliding-mode controllers

Mohammad Rasool Mojallizadeh*, Félicien BONNEFOY, Franck PLESTAN, Mohamed Assaad HAMIDA, and Jérémy OHANA

[a]*Nantes Université, École Centrale de Nantes, CNRS, LHEEA, LS2N, UMR 6598, F-44000 Nantes, France, 1 Rue de la Noe, Nantes, 44300, , France*

Abstract

This paper aims to develop the Euler implicit time-discretization of multivariable sliding-mode controllers to solve the numerical chattering problem without modifying the continuous-time control law. To this end, a continuous-time multi-input plant under a multivariable sliding-mode control is studied, and it is shown that the implicit discretization of the continuous-time sliding-mode controller leads to a multivariable generalized equation with several set-valued terms which is not possible to be solved using the graphical interpretations. Subsequently, an algorithm is proposed to solve such a multivariable generalized equation required to synthesize the implicit sliding-mode control signal at each time step. The proposed algorithm is explained through a simple example accompanied by numerical simulations. The properties of the implicit multivariable sliding-mode controller, *e.g.*, finite-time convergence, gain insensitivity, and chattering suppression, are studied analytically. Afterwards, a multivariable sliding-mode controller is implemented on a digital processor based on the developed algorithm to control a six-input six-output system, *i.e.*, six-component thrust generator, and the results are compared with the case where the continuous-time sliding-mode controller is implemented using the conventional Euler explicit discretization. In the end, the related issues and drawbacks are addressed to be considered in future works.

Keywords: Euler implicit discretization, backward discretization,

*Corresponding author

Email address: rasool.mojallizadeh@ec-nantes.fr (Mohammad Rasool Mojallizadeh)

multivariable systems, multivariable generalized equation, sliding-mode control

1. Introduction

The SMCs are initially developed and studied in a continuous time setting. Hence, a time-discretization method has to be used in order to implement them on digital computers. The effect of the time-discretization is usually underestimated in the studies and the SMCs are implemented and evaluated without addressing the discretization's effect. While in most studies the discretization method used to implement the SMC is not even mentioned, the most natural way to implement a control algorithm is the Euler forward discretization, known as explicit discretization, because of its straightforward implementation. Recent studies revealed that the chattering problem is basically caused by an inappropriate time-discretization method, *e.g.*, explicit discretization, and not the discontinuous functions (or more accurately set-valued functions), itself, that exists in the structure of the continuous time SMCs [1, 2]. As a result, an alternative discretization method, *i.e.*, Euler backward discretization known as the implicit discretization has been proposed for the SMCs to handle the set-valued terms that exist in the continuous-time structure of the SMCs in order to avoid the chattering phenomena. The implicit discretization involves the Euler backward discretization and replacing the single-valued terms with the set-valued counterparts. In the implicit discretization, it is assumed that the discontinuous terms return a set at the discontinuous points allowing to achieve a continuous discrete-time system even in the presence of discontinuities in the continuous-time setting. It should be noted that such property cannot be achieved in explicit discretization.

Following the literature, the concept of implicit discretization was introduced in [3]. However, to the best of the authors' knowledge, this discretization disappeared for two decades, and the importance and properties of implicit discretization remained unknown to the control community until the early 2010s when the implicit discretization was reintroduced for the SMC in [4, 5], and some of its basic characteristics were presented. The application of this discretization method was then studied for observers with discontinuous terms [6]. Further studies mainly dealt with the mathematical and experimental analyses of the implicit SMCs [7, 8, 9, 10, 11, 12] and observers [13, 14] such as chattering elimination, gain insensitivity in the

absence of noise, and keeping the properties of the continuous-time system such as finite-time convergence. The implicit discretization was then developed for the twisting [15, 16] and super-twisting controllers [17]. Furthermore, this discretization method has been used for the sliding-mode-based differentiators [6, 18, 19, 20, 21] and some useful properties, *e.g.*, finite-time convergence, gain insensitivity and chattering reduction have been reported by [22, 23, 24]. The main drawback of implicit discretization is that it requires relatively higher calculation resources compared to explicit ones. For instance, an implicitly implemented differentiator may need a solver to obtain the solutions of a polynomial equation at each time step [22, 23]. However, implicit discretization can still be used to implement both controllers [15] and observers [22] in practical applications. In addition, the semi-implicit schemes can be used to alleviate the calculation burden while keeping some of the advantages of the implicit discretization [25, 26, 27].

The references mentioned above only considered the single-input systems, where only one scalar set-valued function appears in the generalized equation (GE) which is not the case for the multi-input systems. The multi-input systems, in general, contain a set-valued vector function [28] and, therefore, several set-valued scalar terms appear in the GE. As a result, the graphical interpretations considered in all the references mentioned above cannot be used anymore to solve the GEs [1]. To avoid this issue, semi-implicit [27] approaches may be employed to implement some terms implicitly while others remain explicit. However, as studied in [23], semi-implicit discretizations lead to significantly different behavior than the fully implicit one and several useful characteristics, *e.g.*, gain-insensitivity cannot be ensured anymore. Hence, motivated by the fact that the implicit discretization of the multivariable sliding-mode controllers has not been yet addressed in the literature, the main contribution of this study is to address the multi-input systems under the multivariable SMCs, where several set-valued functions appear in the GE. To this end, an algorithm is proposed to solve the multivariable GEs. The algorithm will be explained through a simple example accompanied by numerical simulations. A six-input six-output thrust generator will be considered as the case study to show the practicability of the proposed algorithm.

In the remainder of this manuscript, a multivariable SMC is introduced in Sec. 2, and its implicit time-discretization is addressed in Sec. 3. To this end, an algorithm is proposed in this section to solve the multivariable GE along with a simple example and the characteristics of the multi-input

plant under the implicit multivariable SMC will be evaluated analytically and numerically. The practicability of the algorithm on a real multi-input system is investigated in Sec. 4, and the conclusions, drawbacks, and potential subjects for future studies are presented in Sec. 5.

Throughout this manuscript, the set-valued sign function is defined as follows:

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{for} \quad x \in \mathbb{R} < 0 \\ [-1, +1] & \text{for} \quad x = 0 \\ +1 & \text{for} \quad x \in \mathbb{R} > 0. \end{cases} \tag{1}$$

The functions $f_1(x)$, $f_2(x)$ and $f_3(x)$ are defined in (2) to (4), respectively. These functions map a natural number from the set $\{1, 2, 3\}$ into an integer belong to the set $\{-1, 0, 1\}$. When the input of these functions are vectors and matrices, they operate elementwise.

$$f_1(x) = \begin{cases} 1 & \text{for} \quad x = 1 \\ 0 & \text{for} \quad x = 2 \\ 1 & \text{for} \quad x = 3, \end{cases} \tag{2}$$

$$f_2(x) = \begin{cases} 0 & \text{for} \quad x = 1 \\ 1 & \text{for} \quad x = 2 \\ 0 & \text{for} \quad x = 3, \end{cases} \tag{3}$$

$$f_3(x) = \begin{cases} -1 & \text{for} \quad x = 1 \\ 0 & \text{for} \quad x = 2 \\ 1 & \text{for} \quad x = 3. \end{cases} \tag{4}$$

The matrix $\boldsymbol{R} \in \{1, 2, 3\}^{n \times 3^n}$ is called the permutation matrix. The columns of this matrix are unique and contain a permutation of n elements selected from the set $\{1, 2, 3\}$. The matrix $\boldsymbol{R}$ can be generated systematically using the script provided in the Appendix. The notation $\operatorname{diag}(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ returns a diagonal matrix where the main diagonal is composed of the elements of the vector $\boldsymbol{x} \in \mathbb{R}^n$. Moreover, the notation x_i and $x_{k,i}$, indicates the i^{th} element of $\boldsymbol{x}$ and $\boldsymbol{x}_k$, respectively. In addition, x_k is the discretization of the continuous-time variable $x(t)$, where $x_k = x(t)$ for $t \in [kh, (k+1)h)$, and $h > 0$ is the sampling time.

2. Problem formulation

The following dynamical system is considered in this study as the plant:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(\boldsymbol{x}(t))\boldsymbol{x}(t) + \boldsymbol{B}(\boldsymbol{x}(t))\boldsymbol{u}(t), \tag{5}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}(t) \in \mathbb{R}^n$ is the control vector, $\boldsymbol{A}(\boldsymbol{x}(t)) \in \mathbb{R}^{n\times n}$ is the state transition matrix, $\boldsymbol{B}(\boldsymbol{x}(t)) \in \mathbb{R}^{n\times n}$ is an invertible input matrix, and t is the time variable corresponding to the continuous-time system. In this study, it is assumed that all the system's parameters $\boldsymbol{A}$ and $\boldsymbol{B}$ are known (see Sec. 5 for further discussions). Note that the input arguments of the functions, *e.g.*, $\boldsymbol{A}(\boldsymbol{x}(t))$, may be suppressed for the compactness of the presentation. A typical multivariable sliding-mode control law to control (5) reads as:

$$\boldsymbol{u} = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\lambda}\,\mathrm{sgn}(\boldsymbol{x})), \tag{6}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{n\times n}$ is an invertible gain matrix. Substituting (6) into (5) gives the closed-loop dynamic equation:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\lambda}\,\mathrm{sgn}(\boldsymbol{x}). \tag{7}$$

Following [29, 30], the methods that have been developed for the linear control systems can be used to design $\boldsymbol{\lambda}$ according to Theorem 1.

Theorem 1 ([30]). *Assuming that* $\det(\boldsymbol{\lambda}) \neq 0$ *is bounded,* $\boldsymbol{x}(t) = 0$ *is the asymptotically stable solution of* (7) *if and only if the following system is stable:*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\lambda}\boldsymbol{x}(t). \tag{8}$$

The common way to implement the continuous control law (6) is to use the Euler explicit discretization leading to the following explicit control law:

$$\boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{\lambda}\,\mathrm{sgn}(\boldsymbol{x}_k)), \tag{9}$$

where $k = t/h$ is the discrete time step, $h > 0$ is the sampling time, and $\mathrm{sgn}(0) = 0$ is single-valued. It can be seen that the implementation of the explicit SMC is straightforward. However, according to [7, 8], such discretization leads to poor performances and numerical chattering. Alternatively, the implicit discretization of (6) reads as:

$$\boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{\lambda}\,\mathrm{sgn}(\boldsymbol{x}_{k+1})). \tag{10}$$

Comparing (9) and (10), one can see that the implementation of the implicit SMC (10) requires extra manipulations since the term $\boldsymbol{x}_{k+1}$ appears in the

control law at the time step k. To calculate $\boldsymbol{x}_{k+1}$ at the discrete time step k, the dynamic equation of the plant (5) is discretized as follows:

$$\boldsymbol{x}_{k+1} = h(\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k) + \boldsymbol{x}_k. \tag{11}$$

Substituting (10) into (11) gives the discrete form of the closed-loop equation under the implicit SMC control (see [1] for the scalar form):

$$\boldsymbol{x}_{k+1} \in h\boldsymbol{\lambda}\,\mathrm{sgn}(\boldsymbol{x}_{k+1}) + \boldsymbol{x}_k. \tag{12}$$

Note that $\in$ is written instead of the equality $=$, since (12) is a GE, *i.e.*, the left-hand side of (12) is a scalar while the right-hand side is a set because of using the set-valued sign function (1). The variable $\boldsymbol{x}_{k+1}$ may be obtained by solving the GE using the graphical interpretations when (12) is in the scalar form as proposed by [4]. However, this is not the case for the vector form. This issue was considered an open problem in [1, Section 6]. An algorithm is proposed in Sec. 3 to solve the multivariable GEs.

Remark 1. *In the case of diagonal gain matrix $\boldsymbol{\lambda}$, coupling disappears in the system and n independent controller can be designed for the system. In this case, the monovariable approach can be applied to each subsystem. However, a non-diagonal gain matrix $\boldsymbol{\lambda}$ may be necessary to achieve a certain level of performance (see the example provided in [31, Sec. III]).*

3. Solving multivariable GEs

The set-valued $\mathrm{sgn}(x)$ function (1) presents three different behaviors corresponding to the inputs $x < 0$, $x = 0$, and $x > 0$ for $x \in \mathbb{R}$. The implicit state vector $\boldsymbol{x}_{k+1}$ contains n elements, *i.e.*, $x_{k+1,i}, i = 1, 2, \ldots, n$. Hence, in general, the GE (12) contains n set-valued scalar functions $\mathrm{sgn}(x_{k+1,i})$. As a result, the GE (12) presents 3^n different cases that need to be studied separately (three permutations of n set-valued sgn functions). It should be noted that for the scalar case, there are only three cases that can be studied using graphical interpretations. An algorithm is presented in Algorithm 1 to solve the GE (12) and synthesize the implicit control signal (10) at each time step k. This algorithm contains 24 steps as follows:

- Step 1: Receiving the required parameters, *i.e.*, the system's matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ (see (5)) and the gain matrix $\boldsymbol{\lambda}$ (according to Theorem 2 and remark 4).

- Step 2: The intermediate variables used by the algorithm are initialized in the memory, where i, j, and c are arbitrary scalar variables, $\boldsymbol{k}_3$, $\boldsymbol{\zeta}_k$, $\boldsymbol{\zeta}_k^*$ and $\boldsymbol{R}^*$ are arbitrary vectors with n elements, $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ are two arbitrary $n \times n$ matrices, and $\boldsymbol{R}$ is a $n \times 3^n$ arbitrary matrix, where n is the order of the system (5).

- Step 3: A zero value is assigned to the scalar index variables i, j, and c used for the loops.

- Step 4 (measurement): The state variable $\boldsymbol{x}_k$ is measured at the time step k.

- Step 5 (generating all possible cases): As it was mentioned above, (12) contains n set-valued sign functions with three different behaviors leading to 3^n cases to be studied when solving the GE. At this step, the algorithm generates the permutation matrix $\boldsymbol{R} \in \{1, 2, 3\}^{n \times 3^n}$, based on the script given in the Appendix, where i^{th} row of this matrix corresponds to the behavior of $\text{sgn}(x_{k+1,i})$ such that the elements 1, 2, and 3 indicate $\text{sgn}(x_{k+1,i}) = -1$, $\text{sgn}(x_{k+1,i}) \in [-1, +1]$, $\text{sgn}(x_{k+1,i}) = 1$, respectively. Furthermore, each column indicates a unique permutation of different behaviors of the set-valued sign functions.

- Steps 6-12 (solving the GE for each specific case): In these steps, a column of the matrix $\boldsymbol{R}$ is selected and assigned to $\boldsymbol{R}^*$, and the GE (12) turns into the system of linear equations (13) with n equations and n variables as follows:

$$\boldsymbol{k}_1 \boldsymbol{x}_{k+1} - h\boldsymbol{\lambda}\boldsymbol{k}_2 \,\text{sgn}(\boldsymbol{x}_{k+1}) = \boldsymbol{x}_k + h\boldsymbol{\lambda}\boldsymbol{k}_3, \tag{13}$$

  where the matrices $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ are diagonal matrices and $\boldsymbol{k}_3$ is a vector which is calculated as follows:

  - if the i^{th} element of $\boldsymbol{R}^*$ is equal to 1, the value of $\text{sgn}(x_{k+1,i})$ is constant and equal to -1, and $x_{k+1,i}$ needs to be solved. As a result, the i^{th} element on the main diagonal of $\boldsymbol{k}_1$ is equal to 1 while the i^{th} element on the main diagonal of $\boldsymbol{k}_2$ is equal to zero. Moreover, the i^{th} element of the vector $\boldsymbol{k}_3$ is equal to -1.
  - if the i^{th} element of $\boldsymbol{R}^*$ is equal to 2, $\text{sgn}(x_{k+1,i}) \in [-1, 1]$ needs to be solved as a variable and clearly, according to (1), $x_{k+1,i} = 0$ is

constant. As a result, the i^{th} element on the main diagonal of $\boldsymbol{k}_1$ is 0 while the i^{th} element on the main diagonal of $\boldsymbol{k}_2$ is 1. Moreover, the i^{th} element of the vector $\boldsymbol{k}_3$ equals zero.

- if the i^{th} element of $\boldsymbol{R}^*$ is equal to 3, the value of $\text{sgn}(x_{k+1,i})$ is constant and equal to 1, and $x_{k+1,i}$ needs to be solved. As a result, the i^{th} element on the main diagonal of $\boldsymbol{k}_1$ is equal to 1 while the i^{th} element on the main diagonal of $\boldsymbol{k}_2$ is equal to zero. Moreover, the i^{th} element of the vector $\boldsymbol{k}_3$ equals 1.

The functions f_1, f_2, and f_3 are selected in (2) to (4) to calculate the appropriate values of $\boldsymbol{k}_1 \in \{0,1\}^{n\times n}$, $\boldsymbol{k}_2 \in \{0,1\}^{n\times n}$, and $\boldsymbol{k}_3 \in \{-1,0,1\}^n$, respectively.

- Steps 13-17 (checking the solutions): During these steps, a vector $\boldsymbol{\zeta}_k$, is calculated based on (13) as follows (note that $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ and $\boldsymbol{k}_3$ are selected such that the row number i of either $\boldsymbol{k}_1\boldsymbol{x}_{k+1}$ or $h\boldsymbol{\lambda}\boldsymbol{k}_2\,\text{sgn}(\boldsymbol{x}_{k+1})$ are always zero):

$$\boldsymbol{\zeta}_k = (\boldsymbol{k}_1 - h\boldsymbol{\lambda}\boldsymbol{k}_2)^{-1}(\boldsymbol{x}_k + h\boldsymbol{\lambda}\boldsymbol{k}_3), \tag{14}$$

  where $\zeta_{k,i}$ corresponds to the solution of either $x_{k+1,i}$ or $\text{sgn}(x_{k+1,i})$ (this is equivalent to the selection procedure [1] for the scalar GEs). This function is verified as follows to select the valid case number among all 3^n possible cases:

  - if i^{th} element of $\boldsymbol{R}^*$ is equal to 1 and the i^{th} element of $\boldsymbol{\zeta}_k > -1$ the selected case must be ignored.
  - if i^{th} element of $\boldsymbol{R}^*$ is equal to 2 and the i^{th} element of $\boldsymbol{\zeta}_k \notin [-1,1]$, similarly this case must be avoided.
  - if i^{th} element of $\boldsymbol{R}^*$ is equal to 3 and the i^{th} element of $\boldsymbol{\zeta}_k < 1$ this case is not valid.

  Note that at each time step k, only a unique case number (corresponding to the selected $\boldsymbol{R}^*$) is valid and selected based on the conditions above.

- Steps 18-22 (calculating the set-valued sign function): The rows of the vector $\boldsymbol{\zeta}_k$ calculated in the previous step, contain the solutions of either $x_{k+1,i}$ or $\text{sgn}(x_{k+1,i})$ depending on the case. More clearly, if

the i^{th} element of $\boldsymbol{R}^*$ is equal to 1 or 3, the i^{th} element of $\boldsymbol{\zeta}_k$ is the solution of $x_{k+1,i}$ but not $\text{sgn}(x_{k+1,i})$. Hence, the value of $\text{sgn}(x_{k+1,i})$ needs to be calculated using (1) as $\zeta^*_{k,i} = \text{sgn}(\zeta_{k,i})$. Otherwise, $\zeta_{k,i}$ directly corresponds to the solution of $\text{sgn}(x_{k+1,i})$, *i.e.*, $\zeta^*_{k,i} = \zeta_{k,i}$. Note that vector $\boldsymbol{\zeta}^*_k$ contains the solutions of the selection process in the multivariable case, *i.e.*, $\boldsymbol{\zeta}^*_k = \text{sgn}(\boldsymbol{x}_{k+1})$ (see [1] for the scalar case).

- Step 23 (synthesizing the control signal): The value of $\boldsymbol{\zeta}^*_k$ is used to synthesize the control (10) as shown in (15):

$$\boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{\lambda}\boldsymbol{\zeta}^*_k). \tag{15}$$

Since the multivariable GEs have not yet been considered in the literature, a detailed example is provided in Example 1 for $n = 2$.

Example 1. *Considering* $n = 2$, (12) *gives:*

$$\begin{bmatrix} x_{k+1,1} \\ x_{k+1,2} \end{bmatrix} \in \begin{bmatrix} h\lambda_{11} & h\lambda_{12} \\ h\lambda_{21} & h\lambda_{22} \end{bmatrix} \begin{bmatrix} \text{sgn}(x_{k+1,1}) \\ \text{sgn}(x_{k+1,2}) \end{bmatrix} + \begin{bmatrix} x_{k,1} \\ x_{k,2} \end{bmatrix}, \tag{16}$$

where λ_{11}, λ_{12}, λ_{21}, and $\lambda_{22} \in \mathbb{R}$ are four scalars. Depending on the values of $x_{k+1,i}$ and $\text{sgn}(x_{k+1,i})$, $i = 1, 2$, eq. (16) shows 3^2 different cases as listed in Table 2. According to the script provided in Appendix, the matrix $\boldsymbol{R}$, in this case, is as follows:

$$\boldsymbol{R} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \end{bmatrix}, \tag{17}$$

where nine columns correspond to nine different cases and i^{th} row corresponds to the behavior of $\text{sgn}(x_{k+1,i})$. All nine cases are studied in the following, where in each case, one column of $\boldsymbol{R}$ is considered and assigned to $\boldsymbol{R}^*$.

- Case 1 ($\boldsymbol{R}^* = [1, 1]^\intercal$):

  In this case, the first column of $\boldsymbol{R}$ is selected, *i.e.*, $\boldsymbol{R}^* = [1, 1]^\intercal$, which is decoded as $\text{sgn}(x_{k+1,1}) < 0$, $\text{sgn}(x_{k+1,2}) < 0$. In this case, according to steps 6-12 of Algorithm 1, (16) turns into (13) with the parameters

Algorithm 1 Algorithm for the implicit discretization of the multivariable SMC at each time-step k for any order n

0) Start
1) Receive $\boldsymbol{\lambda}$, $\boldsymbol{A}$, and $\boldsymbol{B}$.
2) Initialize i, j, c, $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, $\boldsymbol{\zeta}_k$, $\boldsymbol{\zeta}_k^*$, $\boldsymbol{R}$, and $\boldsymbol{R}^*$
3) $i \leftarrow 0$, $j \leftarrow 0$, and $c \leftarrow 0$
4) Measure $\boldsymbol{x}_k$
5) Generate the permutation matrix $\boldsymbol{R}$ (see the Appendix)
6) $i \leftarrow i + 1$
7) $\boldsymbol{R}^* \leftarrow$ column i of $\boldsymbol{R}$
8) $\boldsymbol{k}_1 \leftarrow \text{diag}(f_1(\boldsymbol{R}^*))$
9) $\boldsymbol{k}_2 \leftarrow \text{diag}(f_2(\boldsymbol{R}^*))$
10) $\boldsymbol{k}_3 \leftarrow f_3(\boldsymbol{R}^*)$
11) $j \leftarrow j + 1$ and $c \leftarrow 0$
12) $\boldsymbol{\zeta}_k \leftarrow (\boldsymbol{k}_1 - h\boldsymbol{\lambda}\boldsymbol{k}_2)^{-1}(\boldsymbol{x}_k + h\boldsymbol{\lambda}\boldsymbol{k}_3)$
13) If (element j of $\boldsymbol{R}^*$) $= 1$ and (element j of $\boldsymbol{\zeta}_k$) ≥ -1 then go to 6
14) If (element j of $\boldsymbol{R}^*$) $= 2$ and (element j of $\boldsymbol{\zeta}_k$) $\notin [-1, 1]$ then go to 6
15) If (element j of $\boldsymbol{R}^*$) $= 3$ and (element j of $\boldsymbol{\zeta}_k$) ≤ 1 then go to 6
16) $c \leftarrow c + 1$
17) If $c = n$ then go to 18, otherwise go to 11
18) $j \leftarrow 0$
19) $j \leftarrow j + 1$
20) If (element j of $\boldsymbol{R}^*$) $\in 1, 3$ then (element j of $\boldsymbol{\zeta}_k$) $\leftarrow$ sgn(element j of $\boldsymbol{\zeta}_k$)
21) $\boldsymbol{\zeta}_k^* \leftarrow \boldsymbol{\zeta}_k$
22) If $j < n$ then go to 19
23) Synthesize the control signal (15)
24) Terminate

(18):

$$\begin{aligned}&\underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{\boldsymbol{k}_1}\begin{bmatrix}x_{k+1,1}\\ x_{k+1,2}\end{bmatrix} - h\boldsymbol{\lambda}\underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}}_{\boldsymbol{k}_2}\begin{bmatrix}\operatorname{sgn}(x_{k+1,1})\\ \operatorname{sgn}(x_{k+1,2})\end{bmatrix}\\ &= \boldsymbol{x}_k + h\boldsymbol{\lambda}\underbrace{\begin{bmatrix}-1\\ -1\end{bmatrix}}_{\boldsymbol{k}_3}.\end{aligned} \tag{18}$$

Subsequently, step 12 of Algorithm 1 gives:

$$\begin{bmatrix}x_{k+1,1}\\ x_{k+1,2}\end{bmatrix} = \boldsymbol{\zeta}_k = (\boldsymbol{k}_1 - h\boldsymbol{\lambda}\boldsymbol{k}_2)^{-1}(\boldsymbol{x}_k + h\boldsymbol{\lambda}\boldsymbol{k}_3). \tag{19}$$

The required conditions for the validity of the solutions are $x_{k+1,1} < 0$ and $x_{k+1,2} < 0$. If the solutions do not satisfy this condition, they must be avoided and jump to the next case. Moreover, since all elements of $\boldsymbol{R}^* \in \{1, 3\}$, $\boldsymbol{\zeta}_k^*$ can be calculated according to steps 20-21 of Algorithm 1:

$$\boldsymbol{\zeta}_k^* = \operatorname{sgn}(\boldsymbol{\zeta}_k). \tag{20}$$

If this case is valid, *i.e.*, the solutions satisfy the conditions mentioned above, the value of $\boldsymbol{\zeta}_k^*$ is used to synthesize the control (15). Otherwise, the next case (indicated in the next column of the matrix $\boldsymbol{R}$) will be considered

- Case 2 ($\boldsymbol{R}^* = [1, 2]^\intercal$):

  For this case, second column of $\boldsymbol{R}$ is selected and assigned to $\boldsymbol{R}^*$, *i.e.*, $\boldsymbol{R}^* = [1, 2]^\intercal$ indicating $x_{k+1,1} < 0$ and $x_{k+1,2} = 0$. The coefficients of (13) read as:

$$\boldsymbol{k}_1 = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix}-1\\ 0\end{bmatrix}. \tag{21}$$

  These coefficients along with (19) can be used to calculate $\boldsymbol{\zeta}_k$. Moreover, according to steps 20-21:

$$\boldsymbol{\zeta}_k^* = \begin{bmatrix}\operatorname{sgn}(\zeta_{k,1})\\ \zeta_{k,2}\end{bmatrix}. \tag{22}$$

- Case 3 ($\boldsymbol{R}^* = [1, 3]^\intercal$): This case leads to $x_{k+1,1} < 0$ and $x_{k+1,2} > 0$. The coefficients $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ and $\boldsymbol{k}_3$ are listed below:

$$\boldsymbol{k}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}. \tag{23}$$

  This case is valid when the solutions satisfy $x_{k+1,1} < 0$ and $x_{k+1,2} > 0$. After finding $\boldsymbol{\zeta}_k$, $\boldsymbol{\zeta}_k^*$ can be calculated using (20) to synthesize (15).

- Case 4 ($\boldsymbol{R}^* = [2, 1]^\intercal$): In this case, $x_{k+1,1} = 0$, $x_{k+1,2} < 0$ and one has

$$\boldsymbol{k}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{24}$$

  The solutions must satisfy $|\operatorname{sgn}(x_{k+1,1})| \leq 1$, $x_{k+1,2} < 0$. In addition,

$$\boldsymbol{\zeta}_k^* = \begin{bmatrix} \zeta_{k,1} \\ \operatorname{sgn}(\zeta_{k,2}) \end{bmatrix}. \tag{25}$$

- Case 5 ($\boldsymbol{R}^* = [2, 2]^\intercal$):

  For this case $x_{k+1,1} = 0$, $x_{k+1,2} = 0$ leading to

$$\boldsymbol{k}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{26}$$

  The solutions of (19) must satisfy $|\operatorname{sgn}(x_{k+1,1})| \leq 1$, $|\operatorname{sgn}(x_{k+1,2})| \leq 1$. Otherwise, this case must be omitted when synthesizing the control signal at time step k. Moreover, in this case, one has $\boldsymbol{\zeta}_k^* = \boldsymbol{\zeta}_k$.

- Case 6 ($\boldsymbol{R}^* = [2, 3]^\intercal$):

  The mentioned $\boldsymbol{R}^*$ indicates $x_{k+1,1} = 0$, $x_{k+1,2} > 0$, which leads to

$$\boldsymbol{k}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{27}$$

  The required condition for this case is $|\operatorname{sgn}(x_{k+1,1})| \leq 1$, $x_{k+1,2} > 0$. In addition, (25) can be used to synthesize (15).

- Case 7 ($\boldsymbol{R}^* = [3,1]^\intercal$):

  This case corresponds to $x_{k+1,1} > 0$, $x_{k+1,2} < 0$ and one has:

$$\boldsymbol{k}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{28}$$

  The solutions satisfying $x_{k+1,1} > 0$ and $x_{k+1,2} < 0$ are considered for the control synthesize. Moreover, (20) can be used to obtain $\boldsymbol{\zeta}_k^*$.

- Case 8 ($\boldsymbol{R}^* = [3,2]^\intercal$):

  The vector $\boldsymbol{R}^*$ is read as $x_{k+1,1} > 0$, $x_{k+1,2} = 0$, leading to

$$\boldsymbol{k}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{29}$$

  with the condition $x_{k+1,1} > 0$ and $|\operatorname{sgn}(x_{k+1,2})| \leq 1$. Subsequently, (22) can be used to obtain $\boldsymbol{\zeta}_k^*$ and synthesize (15).

- Case 9 ($\boldsymbol{R}^* = [3,3]^\intercal$): This case corresponds to $x_{k+1,1} > 0$, $x_{k+1,2} > 0$ with

$$\boldsymbol{k}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{k}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{k}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{30}$$

  Clearly, the solutions must satisfy $x_{k+1,1} > 0$ and $x_{k+1,2} > 0$, and (20) can be used to synthesize (15).

The responses of the plant (5) under the explicit (9) and implicit (15) SMCs are studied through a numerical simulation and the results are shown in Figs. 1 and 2, respectively. The conditions of the simulation are provided in Table 1. The numerical simulation has been performed on a computer equipped with an `i5-1135G7` CPU with 8 GB of memory in `MATLAB` environment. The total required time to synthesize the control signal corresponding to the explicit and implicit discretizations as well as proceeding (11) for the amount of 8 seconds of simulations are 438 μs and 964 μs, respectively. It indicates that the implicit discretization requires almost two times more calculation resources compared to the explicit one.

For $n = 2$, the state vector $\boldsymbol{x} = [x_1, x_2]^\intercal$ contains two variables x_1 and x_2, with the input vector $\boldsymbol{u} = [u_1, u_2]^\intercal$. According to Fig. 1, as soon as one of the components of the sliding vector, *i.e.*, x_1, reaches around the origin

at $t = 1.4$ s, the system under the explicit SMC (9) starts showing the numerical chattering on all state variables which is caused by the coupling that exists in the multivariable system. The system continues to show the numerical chattering until the end of the simulations. On the other hand, the response of the same continuous-time plant (5) under the implicit SMC (15) is illustrated in Fig. 2. Compared to the explicit one, the numerical chattering is not observed anymore and both x_1 and x_2 converge exactly to the origin and stay there until the end of the simulation. For this example with $n = 2$, the system presents nine (3^2) different cases, and the evolution of the selected cases is shown at the bottom of Fig. 2, it can be seen that the system starts from case seven, and then switched to the cases four and five, and stayed in the fifth case until the end of the simulation. Note that Case 5 corresponds to the condition where the system converges to the origin, *i.e.*, $\boldsymbol{x} = 0$. The last case in the implicit discretization can be calculated using $(3^n-1)/2+1$, where n is the system's order (see the Appendix). The phase plane of the closed-loop system corresponding to the explicit and implicit discretizations is provided in Fig. 3 to see the effect of the discretization on the system's trajectory. This figure confirms that for case number seven, the system is in the reaching phase and the behavior of both explicit and implicit discretizations are the same. However, as soon as one of the components of the sliding vector hits the origin, the explicit discretization starts to show the chattering which is not the case for the implicit one. Another observation is that case number five indicates the convergence of the implicit discretization to the origin, and the system under the implicit discretization stays in the origin thereafter, which cannot be observed for the explicit discretization. The proposed algorithm can be potentially used for any n as shown by another numerical simulation given in the Appendix.

Remark 2. *The choice of* (2) *and* (3) *is such that* (13) *always leads to a system of linear equations with n equations and n variables. The reason is that for a specific* $i = 1, \ldots, n$*, only one of the variables* $x_{k+1,i}$ *or* $\operatorname{sgn}(x_{k+1,i})$*, appears in* (13)*.*

Theorem 2. *Assuming that* $\boldsymbol{\lambda}$ *is designed according to Theorem 1, the implicit SMC provided in Algorithm 1 ensures the convergence of the closed-loop system with* $n = 2$*.*

Proof. The proof is only given for a specific condition when $n = 2$ and $\lambda_{11} < 0$ and $\lambda_{22} < 0$. To this end, we consider the convergence of system

(11) under all nine cases explained in Example 1 as follows. We start the proof with Case 5, since as it will be seen, all other cases will lead to this case.

- Case 5: The fifth column of matrix $\boldsymbol{R}$, given in (17), is $[2, 2]^\intercal$ meaning that in this case one has $\boldsymbol{x}_{k+1} = 0$. It indicates that once the system enters Case 5 at k, it reaches the origin at the next time step $k + 1$. Furthermore, thereafter, one has $\boldsymbol{x}_k = 0$, indicating that the system remains in the origin.

- Cases 1, 3, 7, 9: For these cases, the system is in the reaching phase, substituting the values of $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, and $\boldsymbol{k}_3$ corresponding to these cases into (13) one can see that no coupling exists in the generalized equation. Since the parameters are obtained based on Theorem 1, the inequalities $\lambda_{11} < -|\lambda_{12}|$ and $\lambda_{22} < -|\lambda_{21}|$ always hold. Hence, the convergence of the system can be shown for these cases. For instance, in Case 1, from (19) one has:
$$\begin{bmatrix} x_{k+1,1} \\ x_{k+1,2} \end{bmatrix} = \begin{bmatrix} x_{k,1} \\ x_{k,2} \end{bmatrix} - h \begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix}. \tag{31}$$
Assuming that one step passes, this case indicates $x_{k,1} < 0$ and $x_{k,2} < 0$. The values of $x_{k+1,1}$ and $x_{k+1,2}$ increase indicating the convergence of the state variables towards the origin.

- Case 4: Substituting (24) into (13) gives:
$$\begin{bmatrix} x_{k+1,1} \\ x_{k+1,2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -\frac{\lambda_{21}}{\lambda_{11}} & 1 \end{bmatrix} \begin{bmatrix} x_{k,1} \\ x_{k,2} \end{bmatrix} + h \begin{bmatrix} 0 \\ \frac{\lambda_{12}\lambda_{21}}{\lambda_{11}} - \lambda_{22} \end{bmatrix}. \tag{32}$$
In this case, $x_{k+1,1} = 0$ always holds independent to $x_{k+1,2}$. Assuming that one step passes, one has $x_{k+2,2} = x_{k+1,2} + h(\frac{\lambda_{12}\lambda_{21}}{\lambda_{11}} - \lambda_{22})$. Since, in this case, $x_{k+1,2} < 0$, and according to Theorem 1 and the fact that $\lambda_{11} < 0$, $h(\frac{\lambda_{12}\lambda_{21}}{\lambda_{11}} - \lambda_{22}) > 0$, the value of $x_{k+2,2}$ will increase until the system reaches Case 5.

- Case 6: Substituting (27) into (13) gives $x_{k+1,1} = 0$, and similar to Case 4, one has $x_{k+2,2} = x_{k+1,2} - h(\frac{\lambda_{12}\lambda_{21}}{\lambda_{11}} - \lambda_{22})$. Compared to Case 4, the value of $x_{k+2,2}$ decreases until it reaches Case 5.

- Cases 2 and 8: The proof of convergence for Cases 2 and 8 is similar to Cases 4, and 6, respectively where it can be done by replacing $x_{k,1}$ and $x_{k,2}$.

The proof has only been provided for $n = 2$ and a special condition when $\lambda_{11} < -|\lambda_{12}|$ and $\lambda_{22} < -|\lambda_{21}|$ (see Sec. 5 for further discussions). ■

Remark 3. *According to Theorem 2, the plant* (5) *under the implicit SMC always converges to Case 5 for* $n = 2$*. In this case, as it has been shown in Theorem 2, the equality* $\boldsymbol{x}_{k+1} = 0$ *always holds, meaning that the system converges to the origin and stays there one step after satisfying Case 5. It indicates that the multivariable implicit SMC does not show the numerical chattering inherently (see [1] for the scalar case) which is confirmed by the numerical simulation illustrated in Fig. 2.*

Remark 4. *Substituting* (26) *into* (15) *one has*

$$\operatorname{sgn}(\boldsymbol{x}_{k+1}) = \boldsymbol{\zeta}_k^* = (h\boldsymbol{\lambda})^{-1}\boldsymbol{x}_k. \tag{33}$$

Moreover, from (33) *and* (15)*:*

$$\begin{array}{l} \boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x}_k - h\boldsymbol{\lambda}\boldsymbol{\lambda}^{-1}\boldsymbol{x}_k) \rightarrow \\ \boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A}\boldsymbol{x}_k - h\boldsymbol{x}_k). \end{array} \tag{34}$$

As can be seen, the gain matrix $\boldsymbol{\lambda}$ *does not appear in the implicit control signal during Case 5. It implies that the multivariable implicit SMC is insensitive to the gain during the sliding phase. It should be noted that the explicit discretization is sensitive to the gain* $\boldsymbol{\lambda}$ *since the gain matrix* $\boldsymbol{\lambda}$ *always appears in the control law* (9)*. As a result, increasing the gain amplifies the chattering for the explicit discretization which is not the case for the implicit one. Note that the mentioned gain insensitivity only occurs in noise-free conditions. Assuming that the state vector* $\boldsymbol{x}_k$ *is polluted by an additive measurement noise* $\tilde{\boldsymbol{n}}$*,* i.e.*,* $\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k + \tilde{\boldsymbol{n}}$*, where* $\bar{\boldsymbol{x}}_k$ *is the noise-free part of the state vector,* (34) *turns into:*

$$\boldsymbol{u}_k = \boldsymbol{B}^{-1}(-\boldsymbol{A} - h\boldsymbol{I}_n)(\bar{\boldsymbol{x}}_k + \tilde{\boldsymbol{n}}) = \boldsymbol{B}^{-1}(-\boldsymbol{A} - h\boldsymbol{I}_n)(\bar{\boldsymbol{x}}_k) + \boldsymbol{B}^{-1}(-\boldsymbol{A} - h\boldsymbol{I}_n)(\tilde{\boldsymbol{n}}), \tag{35}$$

where $\boldsymbol{I}_n$ *is the* $n \times n$ *identity matrix. It indicates that even the implicit discretization is affected by the measurement noise during the sliding phase. Substituting* (35) *into the discrete equation of the plant* (11)*, one can see* $\boldsymbol{x}_{k+1} = 0$ *does not hold anymore, meaning that the closed-loop system is not invariant during the sliding phase. In other words, measurement noise perturbs the system from the sliding phase and pushes it away to the other*

cases where the control gain $\boldsymbol{\lambda}$ appears in the control law. Hence, in practical applications, where the measurement noise exists, the gain $\boldsymbol{\lambda}$ must be chosen according to the noise level for both explicit and implicit discretizations.

Remark 5. *The multivariable GE* (12) *always has a unique solution. The reason is that the columns of $\boldsymbol{R}$ are unique and the validity conditions corresponding to cases do not show any overlap (see Table 2 for $n = 2$). Hence, at each time step, only one case is valid. Moreover, in each case,* (13) *turns into a system of linear equations with n equation and n variables since for $x_{k+1,i} \neq 0$ the value of* $\operatorname{sgn}(x_{k+1,i})$ *is known. Alternatively, $x_{k+1,i} = 0$ indicates that* $\operatorname{sgn}(x_{k+1,i})$ *is a variable that needs to be obtained. In other words, it is not possible to have two variables $x_{k+1,i}$ and* $\operatorname{sgn}(_{k+1,i})$*, $i = 1, \ldots, n$, simultaneously, at each time step.*

4. Practical experiments

To evaluate the practicability of the developed algorithm, a multi-input plant, *i.e.*, six component thrust generator is selected with six inputs and six outputs as represented in Figs. 4 and 5. The system is composed of six propeller actuators $P1, P2, \ldots, P6$, which can be divided into two groups. The first group of the propellers, *i.e.*, P1, P2, and P3, are located at the distance of $l_2 = 705$ mm from the center R. The second group of propellers, *i.e.*, P4, P5, and P6 are located at the same plane with the distance of $l_1 = 300$ mm from the center R. The angles between each two links are $\theta = 2\pi/3$ rad. Assuming that the thrust generated by the propellers P1, P2, P3, P4, P5, and P6 are T_1, T_2, T_3, T_4, T_5, and T_6, respectively, the output of the system is the overall forces and moments on the point R and can be calculated based on the geometry of the system as follows:

$$\begin{cases} f_x = T_1 + T_2 + T_3 \\ f_y = -T_4 - (T_5 + T_6)\cos(2\pi/3) \\ f_z = (-T_5 + T_6)\sin(2\pi/3) \\ m_x = l_1(T_4 + T_5 + T_6) \\ m_y = l_2 T_1 + (T_2 + T_3) l_2 \cos(2\pi/3) \\ m_z = l_2(T_2 - T_3)\sin(2\pi/3), \end{cases} \tag{36}$$

where f_x, f_y, and f_z, are three force components along the x, y, and z axes, respectively, and m_x, m_y, and m_z are three moments around the mentioned

axes, respectively, measured on the point R. In the matrix form, one has:

$$\begin{cases} \boldsymbol{y}(t) = \boldsymbol{M}_a \boldsymbol{T}(t), \\ \boldsymbol{y}(t) = [f_x, f_y, f_z, m_x, m_y, m_z]^T \\ \boldsymbol{T}(t) = [T_1, T_2, T_3, T_4, T_5, T_6]^T, \end{cases} \tag{37}$$

where $\boldsymbol{y}(t)$ is the output and $\boldsymbol{M}_a \in \mathbb{R}^{6\times 6}$ is the allocation matrix which can be obtained based on (36) as follows:

$$\begin{cases} \boldsymbol{M}_a = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & C_\theta & C_\theta \\ 0 & 0 & 0 & 0 & -S_\theta & S_\theta \\ 0 & 0 & 0 & l_1 & l_1 & l_1 \\ l_2 & l_2 C_\theta & l_2 C_\theta & 0 & 0 & 0 \\ 0 & l_2 S_\theta & -l_2 S_\theta & 0 & 0 & 0 \end{bmatrix} \\ C_\theta = \cos(\theta), S_\theta = \sin(\theta), \theta = 2\pi/3. \end{cases} \tag{38}$$

Following [32, 33, 34, 35], the trust generated by each individual propeller $Pi = 1, \ldots, 6$, *i.e.*, $T_i(t) \in \mathbb{R} > 0$, follows the dynamic:

$$\dot{T}_i(t) = f_i T_i(t) + g_i u_i(t), \tag{39}$$

where $f_i \in \mathbb{R} < 0$ and $g_i \in \mathbb{R} > 0$ are the constants of the propeller i, and u_i is the control input corresponding to the propeller Pi. The control objective is to design six scalar control inputs $\boldsymbol{u} = [u_1, \ldots, u_6]$ such that the system's output $\boldsymbol{y}(t)$ tracks the desired reference trajectory $\boldsymbol{y}_r = [f_{xr}, f_{yr}, f_{zr}, m_{xr}, m_{yr}, m_{zr},]$, where f_{xr}, f_{yr}, f_{zr}, m_{xr}, m_{yr}, and m_{zr}, are the desired values of f_x, f_y, and f_z, m_x, m_y, and m_z, respectively. As can be seen, the system contains six scalar control inputs $\boldsymbol{u} \in \mathbb{R}^6$ and six scalar outputs $\boldsymbol{y} \in \mathbb{R}^6$. The system's output $\boldsymbol{y}$ is measured using a six-component force sensor installed on point R. However, the generated thrust by each actuator cannot be measured directly and is calculated based on (37) as follows:

$$\boldsymbol{T}(t) = \boldsymbol{M}_a^{-1} \boldsymbol{y}(t). \tag{40}$$

Similarly, the vector of the desired thrust generated by each individual actuator $\boldsymbol{T}_r = [T_{1r}, T_{2r}, T_{3r}, T_{4r}, T_{5r}, T_{6r},]$ reads as:

$$\boldsymbol{T}_r(t) = \boldsymbol{M}_a^{-1} \boldsymbol{y}_r(t), \tag{41}$$

where $T_{ir}, i = 1, \ldots, 6$ denotes the desired thrust generated by Pi, and $\boldsymbol{y}_r = [f_{xr}, f_{yr}, f_{zr}, m_{xr}, m_{yr}, m_{zr},]$ is the desired output. The equations (40) and (41) are only valid when the allocation matrix is invertible, which is the case in the scenario considered in this study.

The control objective is to synthesize the control input $\boldsymbol{u} = [u_1, \ldots, u_6]$ such that the system's output $\boldsymbol{y}$ tracks the desired trajectory $\boldsymbol{y}_r$. The state vector $\boldsymbol{x}(t) = [x_1, \ldots, x_6]^\intercal \in \mathbb{R}^6$ is defined as the tracking error, *i.e.*,

$$\boldsymbol{x}(t) = \boldsymbol{T}(t) - \boldsymbol{T}_r. \tag{42}$$

Using (39) and (42) the dynamic of the thrust generated by the propellers can be written into the matrix form defined in (5), where $\boldsymbol{A} = \operatorname{diag}([f_1, \ldots, f_6])^T$, $\boldsymbol{u} = [u_1, \ldots, u_6]^T$, and $\boldsymbol{B} = \operatorname{diag}([g_1, \ldots, g_6])^T$.

The parameter of the continuous-time SMC (6), *i.e.*, $\boldsymbol{\lambda}$ is selected according to Theorem 1 and Remark 4 as $\boldsymbol{\lambda} = -\operatorname{diag}([10\ 10\ 10\ 3\ 3\ 3])$. However, such a gain matrix may not provide optimal results and, in real applications, the gains should be specifically tuned for each type of discretization. Note that if the components of the gain matrix $\boldsymbol{\lambda}$ are selected too small, the system presents slow transient responses. On the other hand, a gain matrix with large components amplifies the noise effect for both explicit and implicit discretizations (see Remark 4). In addition, larger components of $\boldsymbol{\lambda}$ lead to a higher numerical chattering for the explicit discretization which is not the case for the implicit one according to Remark 4. As it is known (see [1]), numerical chattering can affect the tracking performance as was seen in Fig. 1. Moreover, chattering may lead to higher energy consumption, actuator degradation, and even instability.

Both explicit and implicit controllers are implemented on `NI-cRIO-9049` computer manufactured by National Instruments. This computer contains an `Intel Atom E3940` CPU with four cores running on 1.6 GHz. The sampling time is set to 10 ms. Hence, for $n = 6$, all 729 cases must be checked within 10 ms to realize a practicable system. In other words, each case must be executed in less than 13 μs. For such a computer with four cores, this is doable. However, the algorithm may not lead to real-time implementation for other processors with fewer calculation resources (see Sec. 5 for further discussions).

To evaluate the performances of the implemented controllers, two characteristics, *i.e.*, the chattering on the control signal and the tracking error are studied. These two characteristics are quantified by two objective functions,

i.e., the variation on the control signal ($\text{var}(u_i)$) and the second norm of the tracking error ($||x_i||$) as follows:

$$\begin{aligned} &\text{var}(u_i) = \textstyle\sum_{k=2} |u_i(k) - u_i(k-1)|, \\ &||x_i|| = \sqrt{\textstyle\sum_{k=1} x_i^2(k)}, \end{aligned} \tag{43}$$

where u_i and x_i are the i^{th} components of the control $\boldsymbol{u}$ and state $\boldsymbol{x}$ vectors, respectively. A higher variation on the control signal indicates larger chattering. Moreover, a smaller $||x_i||$ shows better tracking since the state variable (42) is the tracking error.

The synthesized control signal corresponding to the explicit (9) and implicit (15) SMCs are shown in Figs. 7 and 8, respectively. It can be seen that the implicit controller shows less chattering on the control signal than the explicit one. The variation ($\text{var}(\cdot)$) corresponding to each component of the control vector is listed in Table 4. This table confirms the previous observation. The existing variations seen on the implicit control signal are caused by the measurement noise but not the numerical chattering. The tracking performances of the SMC discretized in the explicit and implicit ways are shown in Figs. 9 and 10, respectively. The references are the square waves with periods of 2s. One can see that both explicit and implicit controllers converge to the desired reference values. However, the explicit one shows a large amount of chattering caused by the numerical chattering on its control signal, which is not the case for the implicit one. The second norm ($||\cdot||$) of the tracking error for the components of the state vector is $\boldsymbol{x}$ provided in Table 4 to confirm that the implicit SMC shows a smaller tracking error than the explicit one.

Comparing the experimental results obtained for the explicit and implicit discretizations, it can be seen that while the implicit and explicit discretizations are just two different discretizations of the continuous-time control signal (6) with the same parameters, the differences are significant which indicate the importance of the time discretization which is usually underestimated. Note that to reduce the numerical chattering of the explicit discretization, it is possible to reduce the amplitude of the elements in the gain matrix $\boldsymbol{\lambda}$, which in turn affects the tracking performances. In any case, this study aims to show the impact of discretization on the continuous-time SMC with the same parameters, which seems to be significant.

5. Conclusion

An algorithm has been proposed in this work to be able to extend the implicit discretization to the multivariable sliding-mode controller. The proposed algorithm can systematically solve the multivariable generalized equations that cannot be done using graphical interpretations. An example is provided to show the validity of the algorithm applied to a multi-input system accompanied by a numerical simulation. In addition, the algorithm has been implemented on a six-component thrust generator with six inputs and six outputs to show its practicability. The proposed implicit discretization can bring the following advantages over the commonly used explicit discretization as follows:

- In the developed algorithm, the continuous-time form of the control law is not modified, meaning that the resulting discrete-time implementation shares the same behavior as its continuous-time counterpart without an approximation. For instance, there is no need to replace the sign function with its continuous approximation like the saturation function as can be seen in Sec. 3.

- The discrete-time implementation shows a chattering-free implementation (see Remark 3 and Fig. 2) and gain insensitivity (Remark 4) under noise-free conditions.

- The finite-time convergence of the discrete-time form of the implemented control law has been shown in Theorem 2.

- The feasibility of the algorithm has been studied from the computational (existence and uniqueness of the solutions shown in Remark 5) and implementation points of view (the calculation burden was reasonable to implement the algorithm on a typical industrial computer `NI-cRIO-9049`).

In addition, since the implicit discretization of the multivariable systems has not been considered in the literature before, the proposed algorithm is still at a very early stage of development. As a result, the following topics are suggested to be considered in future studies for further developments:

- While the algorithm can be used for any system's order n, the stability analysis and convergence *e.g.*, Theorem 2, have been only given for a

system with two inputs, and it is not clear whether such properties are valid for $n > 2$ or not. Hence extension of Theorem 2 for an arbitrary n would be necessary.

- This work aims to develop the implicit discretization for multi-input systems. Hence, the design and properties of the continuous-time controller are out of the scope of this work. Throughout the manuscript, it is assumed that all system elements are known without presenting any perturbation. In this case, it is not clear how to tackle the implicit discretization when the system's parameters, *i.e.*, $\boldsymbol{A}$ and $\boldsymbol{B}$, are unknown. The authors believe that this issue should be addressed in future works.

- The algorithm presented in this work concerns the implicit discretization of the continuous-time SMC presented in (6). The extension of the Algorithm for other multivariable SMCs, *e.g.*, multivariable twisting and super-twisting types [27, 28] of SMC remains for future studies.

- The practicability of the algorithm has been shown on an `NI-cRIO-9049` computer equipped with an `Intel Atom E3940` CPU with four cores running on 1.6 GHz. As it was seen, the algorithm leads to 3^n cases that need to be checked within a time step. Such a calculation may not be executable within a time step for all types of computers. The authors believe that the calculation resources required by the algorithm should be studied in the future. In addition, it was seen that the implicit SMC always stays in the case number $(3^n - 1)/2 + 1$ (see Remark 3, Figs. 2, 3 and 11 and the Appendix) during the sliding phase. Hence, the configuration of the permutation matrix can be modified such that the algorithm verifies this case and the cases nearby before the others to reduce the calculation burden. In this context, the learning algorithms may be employed to identify the most probable configurations to be checked before the others [36, 37].

Acknowledgments

This work was supported by the ANR project CREATIF (ANR-20-CE05-0039). The authors are indebted to the colleagues in the test tank at LHEEA, ECN, France, who helped with the experiments presented in this work.

Appendix

A simple script in MATLAB to generate the permutation matrix $\boldsymbol{R}$ is given below:

```
n=6; % n indicates the system's order
for i=0:3^n-1
    x=dec2base(i,3);
    while strlength(\boldsymbol{x})<n
        x=append('0',x);
    end
    for j=1:n
        R(j,i+1)=str2double(x(j))+1;
    end
end
```

Algorithm 1 can be potentially used for any order of systems. For instance, a simulation has been made for $n = 10$, $h = 100$ ms and randomly generated $A, B, \boldsymbol{\lambda}$ and initial conditions leading to $3^{10} = 59049$ cases. The algorithm led to the convergence for the state vector $\boldsymbol{x}_k$, and the selected cases are shown in Fig. 11. An interesting observation is that the final case can be calculated using $(3^n - 1)/2 + 1$. In this example, the final case is equal to $(3^{10} - 1)/2 + 1 = 29525$ while for the example provided in Example 1 it was $(3^2 - 1)/2 + 1 = 5$. This is the case where the state vector converges to the origin and stays there one step after. All the software developments including the proposed algorithm written in MATLAB are available upon request to the first author.

Table 1: Condition of the numerical simulation

$h = 100$ ms, $x_1(0) = 10$, $x_2(0) = -30$
$\boldsymbol{A} = \begin{bmatrix} -1 & -2 \\ 2 & -7 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \boldsymbol{\lambda} = -\begin{bmatrix} 10 & 3 \\ 5 & 10 \end{bmatrix}$

Table 2: Different configurations of the GE (12) when $n = 2$

Case number	$x_{k+1,1}$	$\text{sgn}(x_{k+1,1})$	$x_{k+1,2}$	$\text{sgn}(x_{k+1,2})$	variables to be solved	conditions to be verified	
1	< 0	-1	< 0	-1	$x_{k+1,1}$, $x_{k+1,2}$	$x_{k+1,1} < 0$	$x_{k+1,2} < 0$
2	< 0	-1	0	$[-1, 1]$	$x_{k+1,1}$, $\text{sgn}(x_{k+1,2})$	$x_{k+1,1} < 0$	$\lvert\text{sgn}(x_{k+1,2})\rvert \leq 1$
3	< 0	-1	> 0	1	$x_{k+1,1}$, $x_{k+1,2}$	$x_{k+1,1} < 0$	$x_{k+1,2} > 0$
4	0	$[-1, 1]$	< 0	-1	$\text{sgn}(x_{k+1,1})$, $x_{k+1,2}$	$\lvert\text{sgn}(x_{k+1,1})\rvert \leq 1$	$x_{k+1,2} < 0$
5	0	$[-1, 1]$	0	$[-1, 1]$	$\text{sgn}(x_{k+1,1})$, $\text{sgn}(x_{k+1,2})$	$\lvert\text{sgn}(x_{k+1,1})\rvert \leq 1$	$\lvert\text{sgn}(x_{k+1,2})\rvert \leq 1$
6	0	$[-1, 1]$	> 0	1	$\text{sgn}(x_{k+1,1})$, $x_{k+1,2}$	$\lvert\text{sgn}(x_{k+1,1})\rvert \leq 1$	$x_{k+1,2} > 0$
7	> 0	1	< 0	-1	$x_{k+1,1}$, $x_{k+1,2}$	$x_{k+1,1} > 0$	$x_{k+1,2} < 0$
8	> 0	1	0	$[-1, 1]$	$x_{k+1,1}$, $\text{sgn}(x_{k+1,2})$	$x_{k+1,1} > 0$	$\lvert\text{sgn}(x_{k+1,2})\rvert \leq 1$
9	> 0	1	> 0	1	$x_{k+1,1}$, $x_{k+1,2}$	$x_{k+1,1} > 0$	$x_{k+1,2} > 0$

Table 3: Parameters of the propellers $i = 1, \ldots, 6$

$$\boxed{f_i = -226\ ,\ g_i = 1100}$$

Table 4: Results obtained for the experiments

$\text{var}(\cdot)$	Implicit	Explicit	$\lvert\lvert \cdot \rvert\rvert$	Implicit	Explicit
$\text{var}(u_1)$	1202	9922	$\lvert\lvert x_1 \rvert\rvert$	26.9	30.6
$\text{var}(u_2)$	719	9877	$\lvert\lvert x_2 \rvert\rvert$	23.0	33.3
$\text{var}(u_3)$	589	10465	$\lvert\lvert x_3 \rvert\rvert$	25.2	25.6
$\text{var}(u_4)$	3654	6656	$\lvert\lvert x_4 \rvert\rvert$	20.8	43.5
$\text{var}(u_5)$	5408	9482	$\lvert\lvert x_5 \rvert\rvert$	14.5	14.6
$\text{var}(u_6)$	5082	9603	$\lvert\lvert x_6 \rvert\rvert$	13.4	13.5

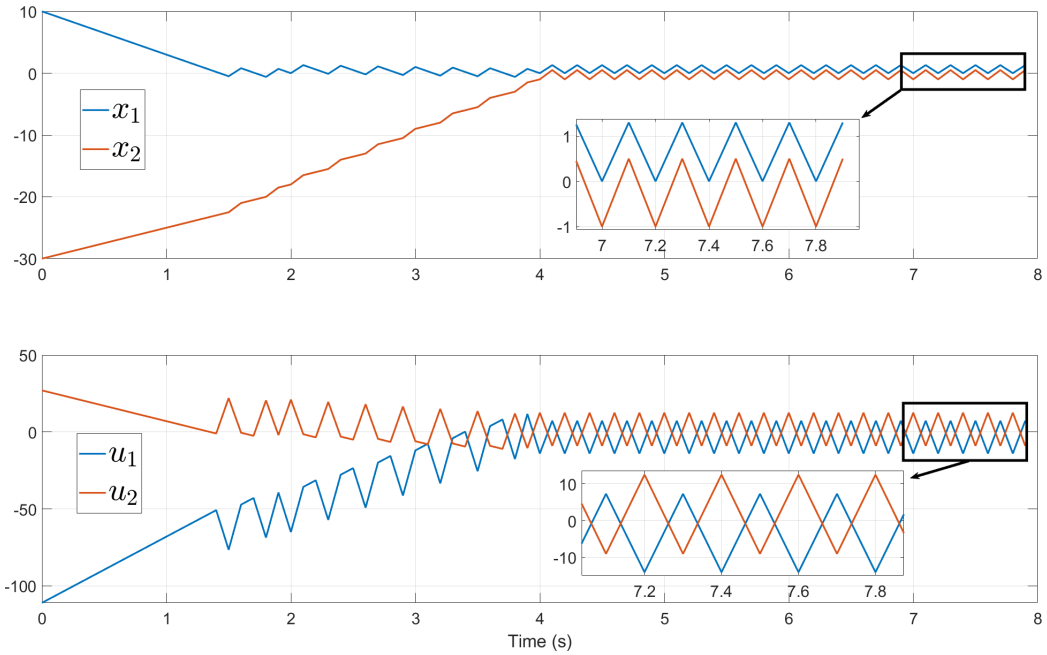


Figure 1: Result of the multivariable explicit SMC. The system starts to show the numerical chattering as soon as one of the state variables hits the origin.

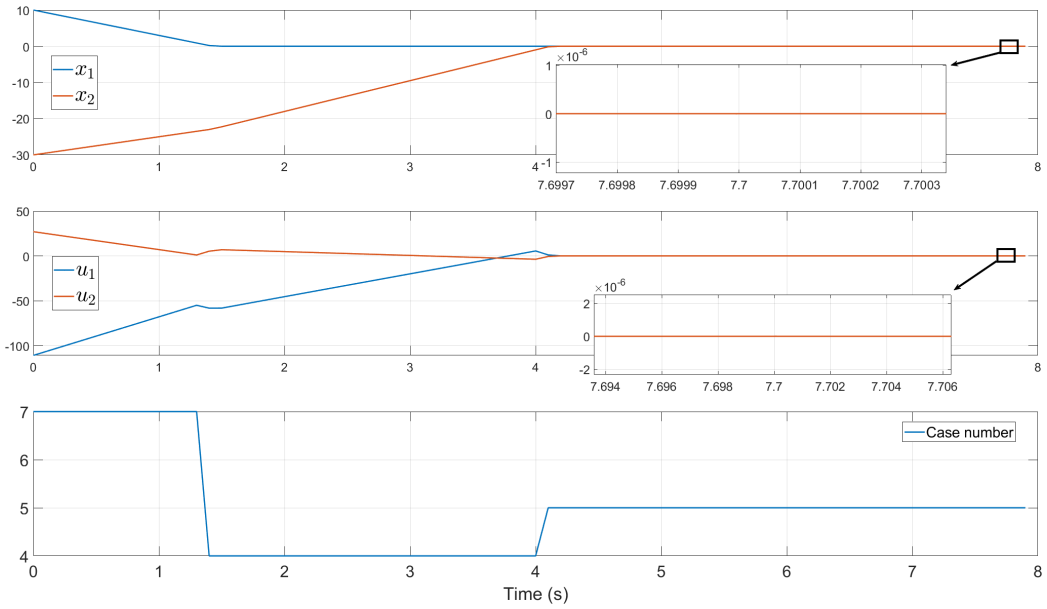


Figure 2: Result of the multivariable implicit SMC. The numerical chattering is not observed.

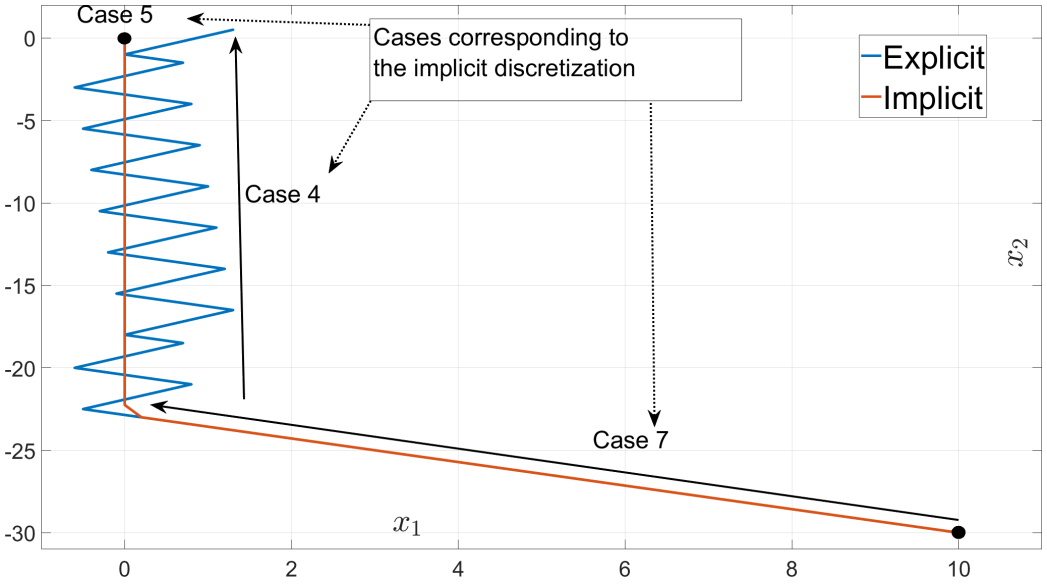


Figure 3: Phase plane of the plant (5) under explicit (9) and implicit (15) SMCs. The implicit discretization directly converged to the origin which is not the case for the explicit one.

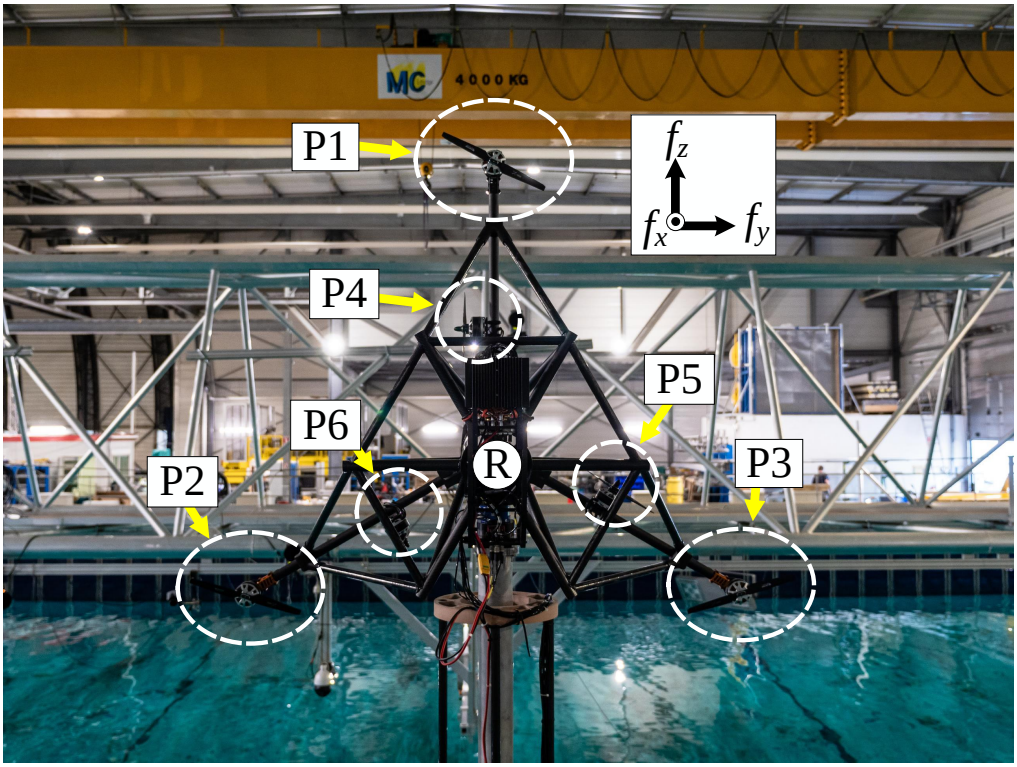


Figure 4: Photo of the six-component thrust generator installed on the head of a floating wind turbine in the test tank to study the aerodynamic effects (front view)

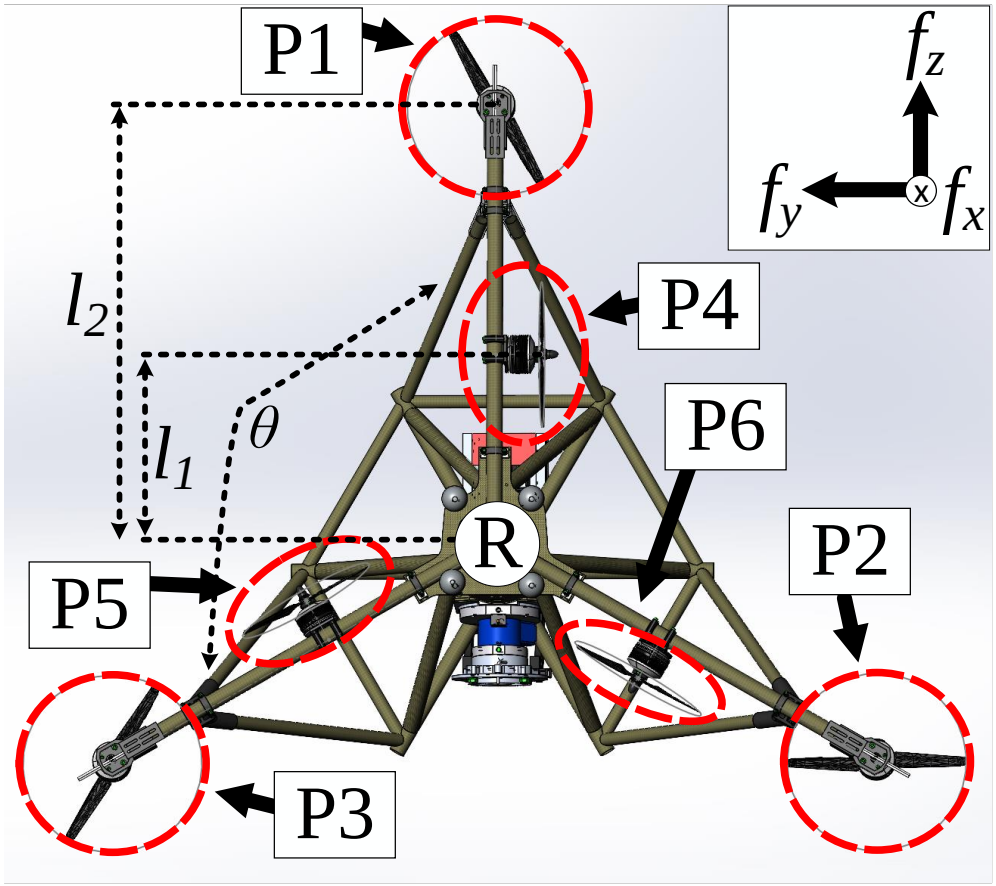


Figure 5: The position and direction of the propellers in the six-components actuator (rear view)

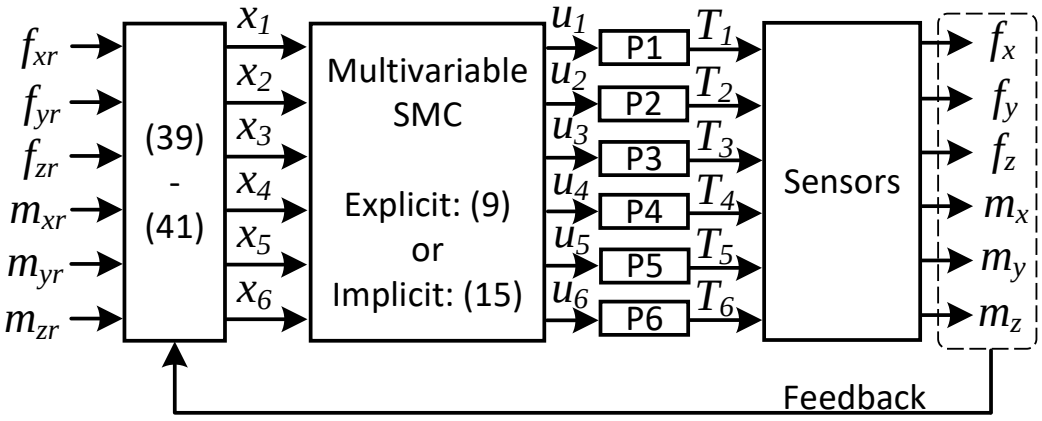


Figure 6: The control loop, where u_i are the control signals, Pi are the propeller actuators, T_i are the thrusts generated by each propeller, and the outputs of the control loop are f_x, f_y, f_z, m_x, m_y and m_z which are the forces and moments on point R.

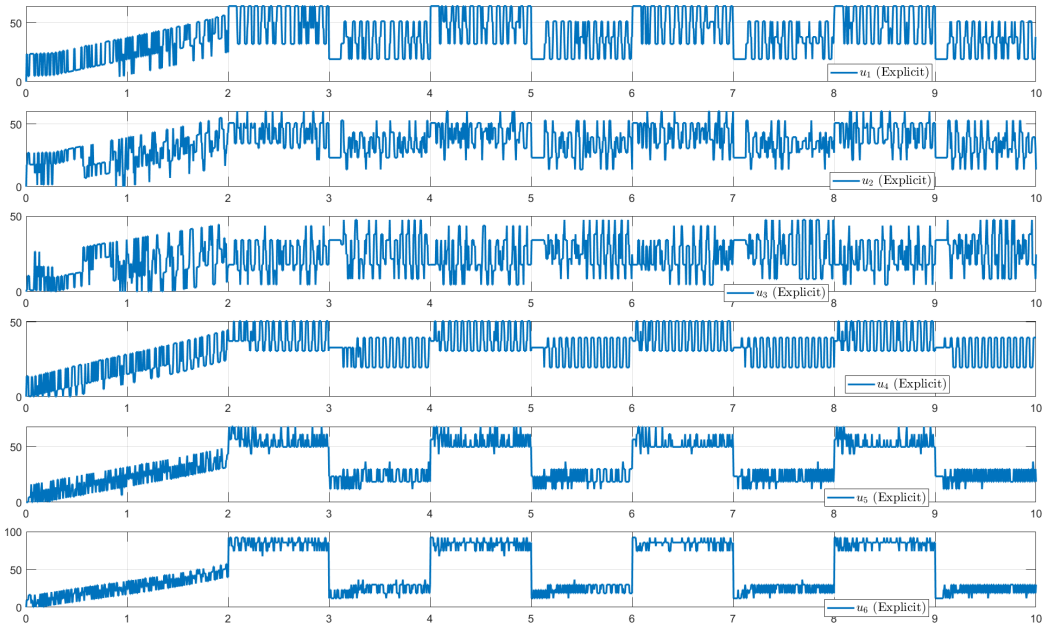


Figure 7: Control signal $\boldsymbol{u}$ of the explicit SMC. The chattering can be observed on the control signal. Horizontal axis: Time (s).

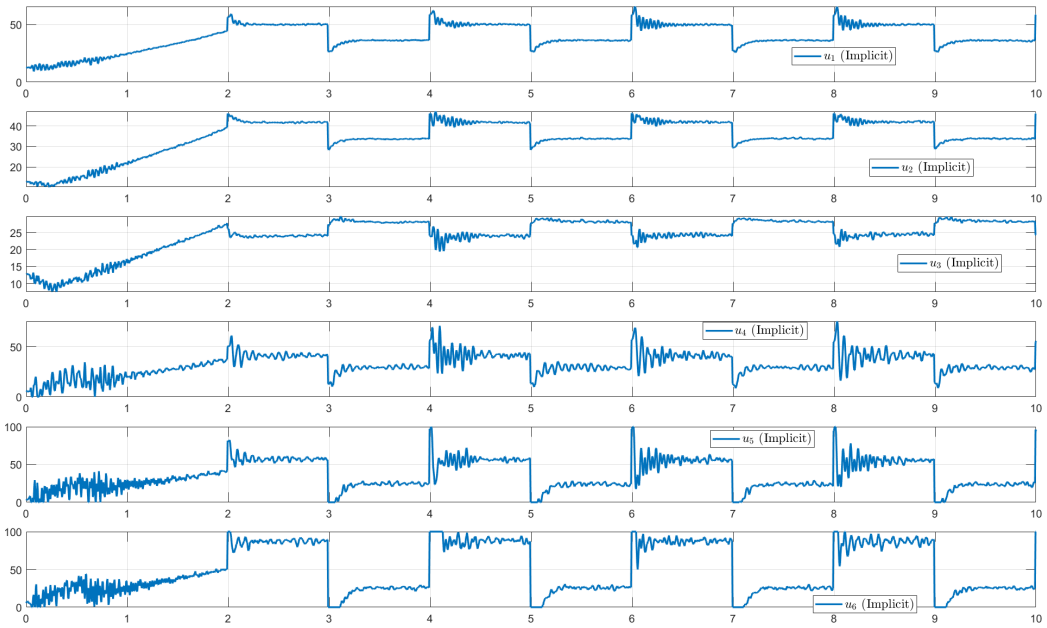


Figure 8: Control signal $\boldsymbol{u}$ of the implicit SMC. The implicit control signal is much smoother than the explicit one. Horizontal axis: Time (s).

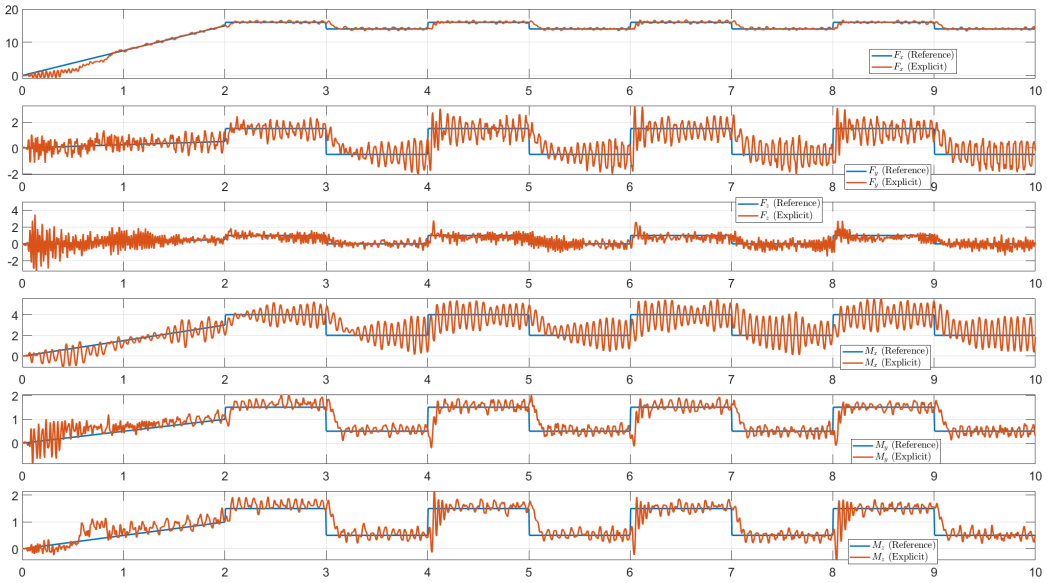


Figure 9: Tracking performance corresponding to the explicit SMC. The tracking is affected by the chattering. Horizontal axis: Time (s).

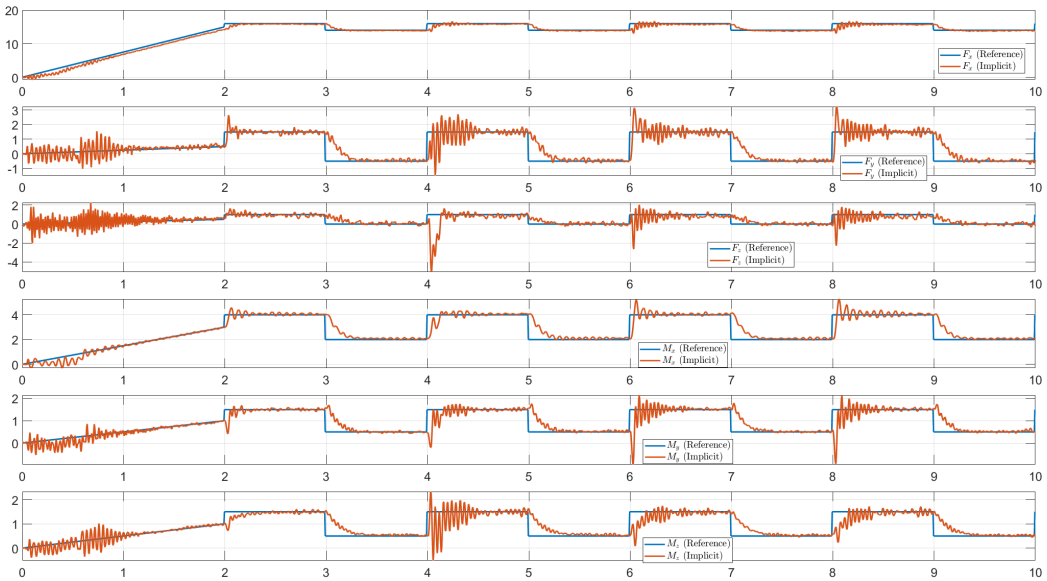


Figure 10: Tracking performance corresponding to the implicit SMC. The implicit implementation shows better tracking than the explicit one since it is not affected by numerical chattering. Horizontal axis: Time (s).

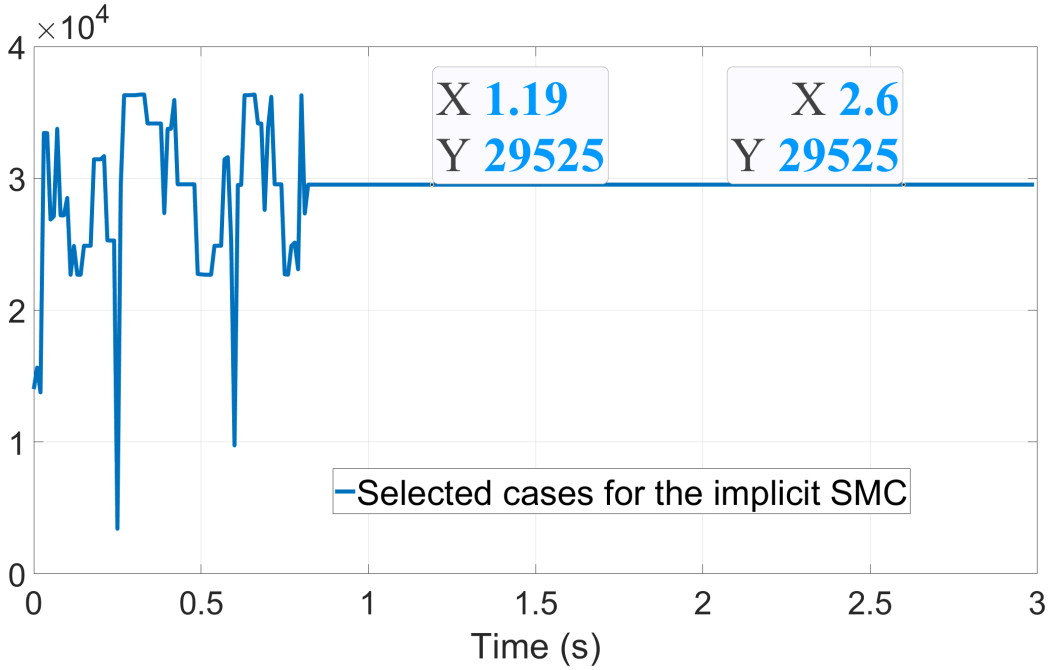


Figure 11: Selected cases for a simulation with $n = 10$ leading to $3^{10} = 59049$ cases. It can be seen the last case number for $n = 10$ can be obtained by $(3^n - 1)/2 + 1 = 29525$ meaning that the system is converged to the origin.

References

[1] B. Brogliato, A. Polyakov, Digital implementation of sliding-mode control via the implicit method: A tutorial, International Journal of Robust and Nonlinear Control 31 (9) (2021) 3528–3586. doi:10.1002/rnc.5121.

[2] Q. Zhang, X. Song, S. Song, V. Stojanovic, Finite-time sliding mode control for singularly perturbed PDE systems, Journal of the Franklin Institute 360 (2) (2023) 841–861. doi:10.1016/j.jfranklin.2022.11.037.

[3] S. Drakunov, V. Utkin, On discrete-time sliding modes, IFAC Proceedings Volumes 22 (3) (1989) 273–278. doi:/10.1016/S1474-6670(17)53647-2.

[4] V. Acary, B. Brogliato, Implicit Euler numerical scheme and chattering-free implementation of sliding mode systems, Systems & Control Letters 59 (5) (2010) 284–293. doi:10.1016/j.sysconle.2010.03.002.

[5] R. Kikuuwe, S. Yasukouchi, H. Fujimoto, M. Yamamoto, Proxy-based sliding mode control: A safer extension of PID position control, IEEE Transactions on Robotics 26 (4) (2010) 670–683. doi:10.1109/TRO.2010.2051188.

[6] S. Jin, R. Kikuuwe, M. Yamamoto, Real-time quadratic sliding mode filter for removing noise, Advanced Robotics 26 (8-9) (2012) 877–896. doi:10.1163/156855312X633011.

[7] O. Huber, V. Acary, B. Brogliato, Lyapunov stability and performance analysis of the implicit discrete sliding mode control, IEEE Transactions on Automatic Control 61 (10) (2016) 3016–3030. doi:10.1109/TAC.2015.2506991.

[8] O. Huber, B. Brogliato, V. Acary, A. Boubakir, F. Plestan, B. Wang, Experimental results on implicit and explicit time-discretization of equivalent control-based sliding mode control, in: L. Fridman, J. Barbot, F. Plestan (Eds.), Recent Trends in Sliding Mode Control, Institution of Engineering and Technology, 2016, pp. 207–235. doi:10.1049/PBCE102E_ch3.2.

[9] F. A. Miranda-Villatoro, B. Brogliato, F. Castaños, Multivalued robust tracking control of Lagrange systems: Continuous and discrete-time algorithms, IEEE Transactions on Automatic Control 62 (9) (2017) 4436–4450. doi:10.1109/TAC.2017.2662804.

[10] F. A. Miranda-Villatoro, F. Castaños, B. Brogliato, A set-valued nested sliding-mode controller, IFAC-PapersOnLine 50 (1) (2017) 2971–2976, 20th IFAC World Congress. doi:10.1016/j.ifacol.2017.08.662.

[11] Y. Yamamoto, J. Qiu, Y. Munemasa, T. Doi, T. Nanjo, K. Yamashita, R. Kikuuwe, A sliding-mode set-point position controller for hydraulic excavators, IEEE Access 9 (2021) 153735–153749. doi:10.1109/ACCESS.2021.3128215.

[12] O. Huber, Experimental results on implicit and explicit time-discretization of equivalent control-based sliding mode control, Control,

Robotics and Sensors, Institution of Engineering and Technology, 2016, pp. 207–235. doi:10.1049/PBCE102E_ch3.2.
URL https://digital-library.theiet.org/content/books/10.1049/pbce102e_ch3.2

[13] V. Acary, B. Brogliato, Y. V. Orlov, Chattering-free digital sliding-mode control with state observer and disturbance rejection, IEEE Transactions on Automatic Control 57 (5) (2012) 1087–1101. doi:10.1109/TAC.2011.2174676.

[14] X. Xiong, R. Kikuuwe, M. Yamamoto, Backward-Euler discretization of second-order sliding mode control and super-twisting observer for accurate position control, in: ASME 2013 Dynamic Systems and Control Conference, Palo Alto, USA, 2013, pp. 1–8. doi:10.1115/DSCC2013-3872.

[15] O. Huber, V. Acary, B. Brogliato, F. Plestan, Implicit discrete-time twisting controller without numerical chattering: Analysis and experimental results, Control Engineering Practice 46 (2016) 129–141. doi:10.1016/j.conengprac.2015.10.013.

[16] O. Huber, V. Acary, B. Brogliato, Lyapunov stability analysis of the implicit discrete-time twisting control algorithm, IEEE Transactions on Automatic Control 65 (6) (2019) 2619–2626. doi:10.1109/TAC.2019.2940323.

[17] B. Brogliato, A. Polyakov, D. Efimov, The implicit discretization of the supertwisting sliding-mode control algorithm, IEEE Transactions on Automatic Control 65 (8) (2020) 3707–3713. doi:10.1109/TAC.2019.2953091.

[18] Z. Lv, S. Jin, X. Xiong, J. Yu, A new quick-response sliding mode tracking differentiator with its chattering-free discrete-time implementation, IEEE Access 7 (2019) 130236–130245. doi:10.1109/ACCESS.2019.2940262.

[19] J. Carvajal-Rubio, J. Sánchez-Torres, M. Defoort, A. Loukianov, On the discretization of robust exact filtering differentiators, IFAC-PapersOnLine 53 (2) (2020) 5153–5158, 21st IFAC World Congress. doi:10.1016/j.ifacol.2020.12.1179.

[20] R. Kikuuwe, R. Pasaribu, G. Byun, A first-order differentiator with first-order sliding mode filtering, IFAC-PapersOnLine 52 (16) (2019) 771–776. doi:10.1016/j.ifacol.2019.12.056.

[21] G. Byun, R. Kikuuwe, An improved sliding mode differentiator combined with sliding mode filter for estimating first and second-order derivatives of noisy signals, Int. J. Control Autom. Syst. 18 (2020) 3001–3014. doi:10.1007/s12555-019-0688-y.

[22] M. R. Mojallizadeh, B. Brogliato, A. Polyakov, S. Selvarajan, L. Michel, F. Plestan, M. Ghanes, J.-P. Barbot, Y. Aoustin, A survey on the discrete-time differentiators in closed-loop control systems: Experiments on an electro-pneumatic system, Control Engineering Practice 136 (2023) 105546. doi:10.1016/j.conengprac.2023.105546.

[23] M. R. Mojallizadeh, B. Brogliato, V. Acary, Time-discretizations of differentiators: Design of implicit algorithms and comparative analysis, International Journal of Robust and Nonlinear Control 31 (16) (2021) 7679–7723. doi:10.1002/rnc.5710.

[24] M. R. Mojallizadeh, B. Brogliato, Effect of Euler explicit and implicit time discretizations on variable-structure differentiators, in: Sliding-Mode Control and Variable-Structure Systems: The State of the Art, Vol. 490, Springer, 2023, pp. 165–180. doi:10.1007/978-3-031-37089-2_7.

[25] M. Wetzlinger, M. Reichhartinger, M. Horn, L. Fridman, J. A. Moreno, Semi-implicit discretization of the uniform robust exact differentiator, in: 2019 IEEE 58th Conference on Decision and Control (CDC), Nice, France, 2019, pp. 5995–6000. doi:10.1109/CDC40024.2019.9028916.

[26] M. R. Mojallizadeh, B. Brogliato, V. Acary, Discrete-time differentiators: design and comparative analysis, Tech. rep., INRIA Grenoble-Alpes, ⟨hal-02960923⟩, hal.inria.fr (08 2020).
URL https://hal.science/hal-02960923/

[27] X. Yang, X. Xiong, Z. Zou, Y. Lou, S. Kamal, J. Li, Discrete-time multivariable super-twisting algorithm with semi-implicit Euler method, IEEE Transactions on Circuits and Systems II: Express Briefs 69 (11) (2022) 4443–4447. doi:10.1109/TCSII.2022.3182772.

[28] I. Nagesh, C. Edwards, A multivariable super-twisting sliding mode approach, Automatica 50 (3) (2014) 984–988. doi:10.1016/j.automatica.2013.12.032.

[29] L. Hsu, J. P. V. S. da Cunha, R. R. Costa, F. Lizarralde, Multivariable output-feedback sliding mode control, in: X. Yu, J.-X. Xu (Eds.), Variable Structure Systems: Towards the 21st Century, Vol. 274, Springer, Berlin, Heidelberg, 2002, pp. 283–313. doi:10.1007/3-540-45666-X_12.

[30] S. V. BAIDA, Unit sliding mode control in continuous- and discrete-time systems, International Journal of Control 57 (5) (1993) 1125–1132. doi:10.1080/00207179308934434.

[31] J. F. García-Mathey, J. A. Moreno, MIMO super-twisting controller: A new design, in: 2022 16th International Workshop on Variable Structure Systems (VSS), 2022, pp. 71–76. doi:10.1109/VSS57184.2022.9901971.

[32] S. Bouabdallah, Design and control of quadrotors with application to autonomous flying, Ph.D. thesis, Faculté des STI, EPFL Lausanne (2007). URL https://infoscience.epfl.ch/record/95939?ln=en

[33] R. Martinez-Alvarado, F. J. Ruiz-Sanchez, A. Sanchez-Orta, O. Garcia-Salazar, Dynamic response of bldc-thruster for small scale quadrotors under aerodynamic load torque, in: 2014 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), 2014, pp. 1–6. doi:10.1109/ROPEC.2014.7036341.

[34] D. Muzar, E. Lanteigne, J. McLeod, Dynamic characterization of brushless dc motors and propellers for flight applications, Unmanned Systems 05 (03) (2017) 159–167. doi:10.1142/S2301385017400027.

[35] V. Arnal, Modélisation expérimentale d'une éolienne flottante par une approche sofware-in-the-loop., Ph.D. thesis, LHEEA, ECN (2020). URL https://www.theses.fr/2020ECDN0037

[36] R. Wang, Z. Zhuang, H. Tao, W. Paszke, V. Stojanovic, Q-learning based fault estimation and fault tolerant iterative learning control for mimo systems, ISA Transactions 142 (2023) 123–135. doi:10.1016/j.isatra.2023.07.043.

[37] Z. Zhuang, H. Tao, Y. Chen, V. Stojanovic, W. Paszke, An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints, IEEE Transactions on Systems, Man, and Cybernetics: Systems 53 (6) (2023) 3461–3473. doi:10.1109/TSMC.2022.3225381.