Virtual reference feedback tuning with robustness constraints

Citation for published version (APA):

DOI:
10.1016/j.engappai.2023.106490

Document status and date:
Published: 01/08/2023

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Virtual reference feedback tuning with robustness constraints: A swarm intelligence solution

Luan Vinícius Fiorio a,b,*, Chrystian Lenon Remes c,d, Patrick Wheeler e, Yales Rômulo de Novaes b,e

a Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands
b Department of Electrical Engineering, Santa Catarina State University, 89219-710, Joinville, Brazil
c Department of Electrical Engineering, Federal University of Paraná, 80060-000, Curitiba, Brazil
d Department of Electrical Engineering, Federal University of Rio Grande do Sul, 90035-190, Porto Alegre, Brazil
e Department of Electrical and Electronic Engineering, University of Nottingham, NG7 2RD, Nottingham, United Kingdom

A R T I C L E   I N F O
Keywords:
Data-driven control
Robust control
Swarm intelligence algorithms
Virtual reference feedback tuning

A B S T R A C T
The simplified modeling of a complex system allied with a low-order controller structure can lead to poor closed-loop performance and robustness. A feasible solution is to avoid the necessity of a model by using data for the controller design. The Virtual Reference Feedback Tuning (VRFT) is a data-driven design method that only requires a single batch of data and solves a reference tracking problem, although with no guarantee of robustness. In this work, the inclusion of an $H_{\infty}$ robustness constraint to the VRFT cost function is addressed. The estimation of the $H_{\infty}$ norm of the sensitivity transfer function is extended to maintain the one-shot characteristic of the VRFT. Swarm intelligence algorithms are used to solve the non-convex cost function. The proposed method is applied in two real-world inspired problems with four different swarm intelligence algorithms, which are compared with each other through a Monte Carlo experiment of 50 executions. The obtained results are satisfactory, achieving the desired robustness criteria.

1. Introduction

The complexity of certain processes usually requires simplification in the mathematical modeling to allow proper controller design (Nise, 2000), e.g., for power systems (Chaudhuri et al., 2012; Xie et al., 2021). For dc-dc converters, for example, the majority of the control techniques assume the existence of an accurate model (Kazimierzuk, 2008; Kobaku et al., 2017), which can be difficult to obtain since power converters have nonlinear dynamics. Low-order controllers are commonly used in industry, such as Proportional Integral (PI) and Proportional Integral Derivative (PID) controllers (Aguiar et al., 2018; Tharanidharan et al., 2022; Tudon-Martinez et al., 2022; van Tan et al., 2022), but a robust design can still be a challenge. Oversimplified modeling due to the process complexity, and/or a limited performance of the chosen controller structure are possible reasons for achieving poor performance and robustness of the controlled system (Keel and Bhattacharya, 2008).

On the other hand, data-driven control design techniques are used to overcome common problems related to system models, such as the dilemma on representativity and complexity, or even the unavailability of those (Remes et al., 2021; Zenelis and Wang, 2022; Huang et al., 2022). Some of the data-driven approaches require several plant experiments and iterative acquisition of data, like Iterative Feedback Tuning (IFT) (Hjalmarsson et al., 1998), Correlation-based Tuning (CbT) (Karimi et al., 2004), and more recent methods such as the deep neural network-based implementation of data-driven Iterative Learning Control (ILC) (Li et al., 2020), and data-driven optimization-based ILC (Meng and Zhang, 2022). While other methods like Virtual Reference Feedback Tuning (VRFT) (Campi et al., 2002), Direct Iterative Tuning (DIT) (Kammer et al., 2000), Optimal Controller Identification (OCI) (Campestrini et al., 2017), Virtual Disturbance Feedback Tuning (VDFT) (Eckhard et al., 2018), Data-Driven Linear Quadratic Regulator (DD-LQR) (Gonçalves Da Silva et al., 2019), and data-driven predictive model control (Carlet et al., 2020) only require a single batch of data (one-shot) in order to tune the controller parameters. If data acquisition is difficult, its repetition is prevented by a tight deadline, or memory constraints are an issue, one-shot solutions are usually preferred over iterative solutions. For more details, a recent and systematic review of data-driven control methods can be found at Prag et al. (2022).

Robustness considering low-order controllers is a frequent topic of discussion (Pérez and Llopis, 2018; Alcántara et al., 2013) since specific processes can present uncertainties, as well as disturbances that might occur over time. Another point to be observed is that a poor choice of

* Corresponding author at: Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands.
E-mail address: l.v.fiorio@tue.nl (L.V. Fiorio).
https://doi.org/10.1016/j.engappai.2023.106490
Received 16 June 2022; Received in revised form 3 April 2023; Accepted 17 May 2023
Available online xxxx
0952-1976/© 2023 Published by Elsevier Ltd.
reference model or limited controller class in, e.g., the VRFT design, may result in poor performance or robustness (Bazanella et al., 2014). Since robustness can be measured by the $H_\infty$ norm of the sensitivity transfer function $S(z)$ of a closed-loop system (Skogestad and Postlethwaite, 2005), its inclusion in the data-driven design of controllers can be considered, allowing for a more robust design when necessary. A recent methodology proposed in Chiluka et al. (2021) has suggested the inclusion of a robustness criterion in the VRFT design, at the expense of:

(i) more experiments since the proposed design procedure iterates in a trial-and-error fashion until the desired robustness is achieved and, essentially removes one of the greatest advantages of the VRFT — being a one-shot method; and

(ii) this type of iterative procedure requiring more background knowledge from the designer for choosing reference models and specifying requirements.

Also, a data-driven one-shot approach for multivariable systems regarding robust solutions of $H_\infty$ and $H_\infty$ criteria, and loop-shaping specifications, has been recently addressed (Karimi and Kammer, 2017). But the method relies on an initial solution that influences the final one since it is convexified by linearization around the initial stabilizing controller, which is undesirable. In the case of data-driven one-shot approaches for Single-Input Single-Output (SISO) systems, considering an $H_\infty$ robust performance criteria, a recent work (Nicoletti et al., 2019) proposes a technique where the order of the controller is increased as the solution converges and only deals with noise-free data. Both methods (Karimi and Kammer, 2017; Nicoletti et al., 2019) use only frequency-domain data. Other data-driven robust solutions are achieved in an online fashion and require higher computational processing than offline techniques, and also present the demand of measuring signals in real-time. Some of those techniques are the use of the modified Riccati equation with online data-driven learning (Na et al., 2021) and the application of a data-driven Model Predictive Control method with robustness guarantees (Berberich et al., 2021).

Among the aforementioned data-driven design techniques, the VRFT has been more broadly applied to several classes of problems, a fact that can be useful to attest to the feasibility of the proposed method of this paper, as it requires less computational cost than, e.g., OCl and online methods. Therefore, this work is based on the VRFT technique, as presented in Campi et al. (2002) and Campestri et al. (2011), proposing the inclusion of a robustness constraint, in terms of the $H_\infty$ norm of the sensitivity transfer function of the system, in the VRFT design, but maintaining one of its most attractive features: the necessity of only one batch of data. The constraint is included in the VRFT cost function as a penalty (Luenberger and Ye, 2015), which compromises its convex behavior. To deal with the non-convex optimization procedure, the proposed method is tackled in two main steps: (i) the design, in a data-driven fashion, of a controller using the VRFT approach; if a previous controller is inexistent; and (ii) the application of a metaheuristic optimization algorithm to minimize the cost function considering an $\|S(z)\|_\infty$ norm constraint, using the controller from the previous step as an initial solution.

Notice that metaheuristics, more specifically the Particle Swarm Optimization (PSO) algorithm, have been used on the VRFT design for enhancing the performance of the controller in terms of reference tracking on hybrid systems (Jiaqi et al., 2017). Also, more closely related to this work, PSO has been used for solving the VRFT with a modified cost function which includes the optimization of the reference model based on a reference tracking metric (Selvi et al., 2018). The work focuses on reducing the mismatch between the reference model and the actual controlled system but does not mention robustness.

Since metaheuristic algorithms may work well for a certain class of problems, but might fail over other problems, according to the No Free Lunch (NFL) theorems (Wolpert and Macready, 1997), more than a single metaheuristic optimization algorithm should be evaluated. Looking over the available types of metaheuristics, three can be highlighted: evolutionary algorithms (Mirjalili, 2019); physics-based algorithms (Alatas and Can, 2015); and swarm intelligence algorithms.

Although some authors group evolutionary algorithms with multiple agents and swarm algorithms together (Ab Wahab et al., 2015), this paper considers the two classes separately, as done by other authors in the metaheuristic optimization subject (Mirjalili et al., 2014). This work focuses on swarm intelligence algorithms since they usually have fewer parameters to be tuned by the user or designer (Ab Wahab et al., 2015). Four swarm intelligence algorithms are considered: Particle Swarm Optimization (Kennedy and Eberhart, 1995) and Artificial Bee Colony (ABC) (Karaboga, 2005) since those are two of the most used in literature; and the two swarm intelligence algorithms with the least number of hyperparameters, Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014), and its most recent version, the Improved Grey Wolf Optimizer (I-GWO) (Nadimi-Shahrafi et al., 2021).

In summary, the main contributions of this work are:

- in theoretical terms, the derivation of the expressions of the signals for the data-driven estimation of the impulse response of the sensitivity transfer function of a system, in one-shot, at Section 4.1.1;
- the proposal of a modified cost function to the VRFT method regarding the inclusion of a robustness constraint, at Section 5, and the use of swarm-intelligence algorithms for solving the proposed cost function;
- at Section 6, the use and comparison of four different swarm-intelligence algorithms – PSO, ABC, GWO, and I-GWO – in the proposed method for solving two real-world inspired problems based on the structure of widely used DC-DC converters.

This paper is structured as follows: Section 2 describes the system that is considered in this paper for the theoretical formulation; Section 3 presents the basis of data-driven controller design and details the basic VRFT design procedure; Section 4 explains the method for the $H_\infty$ norm estimation of the sensitivity transfer function; Section 5 presents the proposed method and details the four considered swarm intelligence algorithms; Section 6 illustrates and validates the method by showing its application in two real-world inspired examples; and finally, Section 7 concludes this work.

2. Preliminaries: description of the system

The system considered in this paper for the theoretical formulation is a discrete-time, causal, linear time-invariant, and Single-Input Single-Output (SISO) system $G(z)$. It is considered that $z$ is the forward discrete time-shift operator such that $zx(k) = x(k + 1)$. The output $y(k)$ of the system can be described as

$$y(k) = G(z)u(k) + v(k),$$

where $u(k)$ is the input signal and $v(k)$ is the process noise — stochastic effects that are not represented by $G(z)$, i.e., not captured by the input–output relation of $u(k)$ and $y(k)$. The closed-loop system taken into account in this work is regarded as a controller $C(z)$ with the process $G(z)$ and a unit gain feedback, as shown in Fig. 1, where $y(k)$ is the reference signal and $e(k)$ is the error signal. The closed-loop control law is

$$u(k) = C(z, \rho)(r(k) - y(k)), \quad (2)$$

with

$$C(z, \rho) = \rho \hat{C}(z)$$

being a controller with parameter $\rho \in \mathbb{R}^d$, $\hat{C}(z)$ a vector of transfer functions belonging to the controller class $C$ (e.g., PI or PID controller classes) and $r(k)$ the reference signal.

The output of the closed-loop system is given as

$$y(k) = T(z)e(k) + S(z)u(k), \quad (4)$$
where the reference signal \( r(k) \) is applied to the transfer function from the reference \( r(k) \) to the output \( y(k) \), \( T(z) \), with

\[
T(z) = \frac{C(z)G(z)}{1 + C(z)G(z)}
\]

and \( S(z) \) is the sensitivity transfer function such that \( S(z) + T(z) = 1 \) and

\[
S(z) = \frac{1}{1 + C(z)G(z)}
\]

In the next section, the Model Reference Control (MRC) is introduced. The VRFT method is described for the cases where the process has minimum or non-minimum phase.

3. Virtual Reference Feedback Tuning (VRFT)

The Model Reference Control (MRC), which is the basis for the VRFT, more generally called model matching control (Goodwin et al., 2000), concerns the problem of reference tracking of the closed-loop system’s response, disregarding the effects of noise at the output (Bazanella et al., 2014).

In order to obtain a controller, the MRC requires the designer to elaborate a target transfer function for the controlled closed-loop system, called a reference model \( (T_d(z)) \), which generates the output \( y_d(k) = T_d(z) r(k) \). A reference tracking performance criterion is then obtained by solving the optimization problem

\[
\text{minimize } J^{MR}(\rho) = \| (T(z) - T_d(z)) r(k) \|_2^2 ,
\]

which can be solved by considering (5), resulting in the solution controller for the MRC, called the ideal controller \( C_i(z) \).

The VRFT is a one-shot optimization data-driven controller design technique based on the MRC. It is defined as one-shot since only a single batch of input-output data is required to solve the model reference control problem, which can be done by the use of least squares when the controller is linearly parameterized as in (3), resulting in the parameter \( \rho \) of a controller with predefined class. The VRFT design depicted in this paper follows the procedures of Bazanella et al. (2014), Remes et al. (2022).

Consider an experiment, in open-loop or closed-loop, that results in a batch of collected data \( \{ u(k), y(k) \}_{k=1}^{N} \). A virtual reference signal \( \hat{r}(k) \) is defined such that \( T_d(z) \hat{r}(k) = y(k) \). A virtual error can be obtained as \( \hat{e}(k) = \hat{r}(k) - y(k) = (T_d^{-1}(z) - 1) y(k) \). In summary, a controller \( C(z, \rho) = \rho \hat{C}(z) \) is considered satisfactory if it generates \( u(k) \) when fed by \( \hat{e}(k) \). The closed-loop of such block diagram for the VRFT controller design is illustrated in Fig. 2.

The VRFT solves the optimization problem

\[
\text{minimize } J^{VR}(\rho) = \| u(k) - C(z, \rho) (T_d^{-1}(z) - 1) y(k) \|_2^2 ,
\]

having the same minimum as if the ideal controller \( C_i(z) \) in (5) belongs to the same controller class \( C = \{ C(z, \rho), \rho \in \mathbb{R}^p \} \) as \( C(z, \rho) \).

To compensate for the fact that the ideal controller rarely belongs to the chosen controller class, a filter \( L(z) \) is applied to the data to approximate the minimum of \( J^{VR} \) to the minimum of \( J^{MR} \), where the amplitude should satisfy (Bazanella et al., 2014):

\[
| L(e^{j\Omega}) | = | T_d(e^{j\Omega}) | (1 - | T_d(e^{j\Omega}) |)^2 \frac{\Phi_x(e^{j\Omega})}{\Phi_y(e^{j\Omega})}, \quad \forall \Omega \in [-\pi, \pi] .
\]

where \( \Phi_x(e^{j\Omega}) \) with \( x \) representing any signal or system, represents the Discrete Fourier Transform of \( x(k) \), \( \Phi_y(e^{j\Omega}) \), \( \Phi_x(e^{j\Omega}) \) are, respectively, the power spectra of the signals \( r(k), u(k) \).

Instrumental variables can be used in order to suppress estimation bias caused by the noise in data (Ljung, 1999), requiring the use of a second data batch. In practice, the input signal can be formed by two identical sequences in the same experiment, if memory restrictions do not impose a problem. Then, the signals can be synced together afterward, resulting in two batches of data from one single experiment.

In the presence of a Non-Minimum Phase (NMP) zero at the process, a flexible reference model can be used, as presented in Bazanella et al. (2014). The optimization problem, then, becomes

\[
\text{minimize } J^{VR}(\rho) = \| \eta F(z)(u(k) + \rho \hat{C}(z) y(k)) - \rho \hat{C}(z) y(k) \|_2^2 ,
\]

where \( \eta \in \mathbb{R}^n \), and \( F(z) \) is a vector of transfer functions such that \( T_d(z, \eta) = \eta F(z) \). The step-by-step design for the VRFT with a flexible reference model, from data collection to the algorithm design, is detailed in Remes et al. (2021).

To be able to include a robustness criterion at the VRFT cost function, a means of evaluating this parameter is required. Nonetheless, this is approached in the next section.

4. Robustness index estimation

Depending on the choice of \( T_d(z) \) or the controller class \( C \), as well as the response of the plant \( G(z) \), the VRFT-designed controller can result in poor robustness of the controlled process. For such cases, a robustness constraint can be included in the form of the \( H_\infty \) norm of \( S \), also called maximum sensitivity (\( M_S \)), which can be used as a measure of robustness (Skogestad and Postlethwaite, 2005).

Typically, a system that presents \( M_S > 2 \) is considered to have poor robustness (Skogestad and Postlethwaite, 2005). In this context, and considering the use of data-driven design approaches, it is necessary to estimate the value of \( \| S(z) \|_{H_\infty} \) in a data-driven way since it is assumed that no plant model is available to the designer. By this reasoning, the estimation of \( M_S \) is explained in the following subsection.

4.1. Estimation of \( M_S \)

The \( H_\infty \) norm estimation procedure developed in this work is based on the Impulse Response (IR) of the system, as presented in Fiorio (2022), modified from (Gonçalves da Silva et al., 2020) to a SISO impulse response identification procedure, which allows for a regularized estimation according to the existing literature (Chen et al., 2012). Also, in order to maintain the one-shot characteristic of the VRFT, the estimation of the \( H_\infty \) norm of \( S \) based on impulse response is addressed as follows.
Consider the linear discrete-time causal and SISO system $S$, represented by its transfer function $S(z, \rho)$, such that its output signal $\psi(k)$ is given by the equation

$$S : \psi(k) = z(k) \ast \zeta(k) = \sum_{n=0}^{\infty} z(k-n)\zeta(n),$$

where $\zeta(k)$ is the input signal of $S$, whose impulse response is $z(k)$.

Since (11) requires infinite data to be obtained, an order $M$ is defined such that it is assumed that any IR greater than $M$ is negligible, which is valid for stable systems since $\lim_{n \to \infty} z(k) = 0$. Nevertheless, Eq. (11) can be truncated to $M$ terms, leading to:

$$S : \psi(k) = \sum_{n=0}^{M} z(k-n)\zeta(n) = \sum_{n=0}^{M} s(k-n)\zeta(n).$$

(12)

On the other hand, the definition of the $H_\infty$ norm, when applied to the system $S$, can be written as (Skogestad and Postlethwaite, 2005)

$$H_\infty : \|S\|_{H_\infty} = \max_{\|z\|_{\infty}} \frac{\|s(k) + \zeta(k)\|_2}{\|\zeta(k)\|_2},$$

(13)

which requires the whole set of possible inputs $\{\zeta(k) \neq 0\}$. Therefore, expression (13) cannot be directly calculated. An alternative strategy is to obtain a matrix relation for $S$, which allows for the use of induced norm properties.

Expanding (12) to the $M$ first terms gives

$$\begin{align*}
\psi(0) &= s(0)\zeta(0) \\
\psi(1) &= s(1)\zeta(0) + s(0)\zeta(1) \\
&\vdots \\
\psi(M) &= s(M)\zeta(0) + \cdots + s(0)\zeta(M),
\end{align*}$$

(14)

and the following matrix relation truncated at $M$ elements is obtained:

$$\begin{bmatrix} 
\psi(0) \\
\psi(1) \\
\vdots \\
\psi(M) 
\end{bmatrix} = 
\begin{bmatrix} 
s(0) & 0 & \cdots & 0 \\
s(1) & s(0) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
s(M) & s(M-1) & \cdots & s(0) 
\end{bmatrix}
\begin{bmatrix} 
\zeta(0) \\
\zeta(1) \\
\vdots \\
\zeta(M) 
\end{bmatrix}.$$  

(15)

From the assumption that the order $M$ is sufficiently high, it can be said that matrix $S_M$ characterizes the IR $z(k)$ of $S$.

A useful matrix property is the induced norm (Skogestad and Postlethwaite, 2005, A.5), which can be applied to (15), such that

$$\|S_M\|_p = \max_{z_M \neq 0} \frac{\|S_M Z_M\|_p}{\|Z_M\|_p},$$

(16)

where the subscript $i$ stands for induced. In short, (16) is a matrix form of representing the system gain considering a set of possible input signals $Z_M$. From the induced-2 norm, the following is obtained

$$\|S_M\|_2 = \hat{\sigma}(S_M) = \sqrt{\lambda_{\max}(S_M^T S_M)}.$$  

(17)

where $\hat{\sigma}$ and $\lambda_{\max}$ stands for largest singular value and largest eigenvalue, respectively, and comparing (13) with (16), it can be seen that

$$\|S\|_2 \approx \max_{z_M \neq 0} \frac{\|S_M Z_M\|_2}{\|Z_M\|_2} = \sqrt{\lambda_{\max}(S_M^T S_M)}.$$  

(18)

Since the $\|S(z, \rho)\|_{H_\infty}$ norm can be estimated based on its IR via (18), an expression for the input signal $\{\zeta(k)\}$ and the output signal $\psi(k)$ of $S(z, \rho)$ has to be derived in order to estimate its impulse response in the VRFT design context.

### 4.1.1. Input–output signals of the sensitivity transfer function

Considering the system presented in Fig. 2, its sensitivity transfer function in (6) can be rewritten as

$$1 + C(z, \rho)G(z) = S^{-1}(z, \rho).$$

(19)

Assuming that $u(k)$ is sufficiently informative to capture all relevant characteristics of $S(z, \rho)$, multiplying both sides of (19) by $u(k)$ achieves

$$u(k) + C(z, \rho)G(z)u(k) = S^{-1}(z, \rho)u(k).$$

(20)

It is known that $G(z)u(k) = y(k)$, Substituting such relation in (20):

$$u(k) + C(z, \rho)y(k) = S^{-1}(z, \rho)u(k).$$

(21)

From (21), the signals

$$\zeta(k) = u(k), \quad \zeta(k) = u(k) + C(z, \rho)y(k),$$

(22)

can be derived. Expressions (21) and (22) mean that when a signal $\zeta(k)$ formed by $u(k) + C(z, \rho)y(k)$ is applied to $S(z, \rho)$, an output $\zeta(k)$ is obtained. Therefore, the impulse response of $S(z, \rho)$ can be estimated considering the data set $\{\zeta, \zeta^N\}$. In this work, the IR estimation is made through identification with regularization techniques since: (i) the variance of the estimates increases with the order $M$, which is suppressed with the use of regularization (Chen et al., 2012); and (ii) knowing that IR is a sparse signal for sufficiently high $M$, the use of regularization is known to improve sparse signal estimates (Brunton and Kutz, 2019). The algorithm for regularized estimation of the impulse response is described in Chen et al. (2012), Chen and Ljung (2013), and is available in MATLAB® (Anon, 2021), Python (Fiorio et al., 2021), and R (Yerramilli and Tangirala, 2017).

The inclusion of the $H_\infty$ constraint in the cost function spoils the convexity characteristic of the VRFT and the solution can no longer be obtained through the least squares algorithm. A strategy to deal with local minima and other characteristics that may arise from a non-convex cost function is to use metaheuristic optimization (Du and Swamy, 2016). Nevertheless, the VRFT with robustness constraints is formally proposed in the next section, which also addresses the use of swarm intelligence algorithms to achieve a feasible solution.

### 5. VRFT with robustness constraints

The proposed method regards a two-step procedure. The first step follows the design of a controller using the VRFT, as commented in Section 3. The same data from the first step is used in the second step. The need for a second experiment is avoided since the estimation of $M_S$, represented in this context as $\tilde{M}_S(\rho)$, does not require new data, as derived in Section 4.1. In this case, the cost function $J^{VR}$ is modified by the addition of a robustness constraint, regarding the value of the estimated ($\tilde{M}_S(\rho)$) and desired ($M_{SD}$) $\|S(z, \rho)\|_\infty$ norm, leading to a new optimization problem:

$$\begin{align*}
\begin{array}{l}
\text{minimize} \\
\rho
\end{array}
J^{VR}(\rho)
\text{subject to} \\
\tilde{M}_S(\rho) \leq M_{SD},
\end{align*}$$

(23)

which can be applied directly to the cost function as a penalty (Luemberger and Ye, 2015), resulting in the Swarm Intelligence optimization cost function:

$$\begin{align*}
\begin{array}{l}
\text{minimize} \\
\rho
J^{SI}(\rho) = \|u(k) - C(z, \rho)(T_k^{-1}(z) - 1)y(k)\|_2^2 + cH(\rho),
\end{array}
\end{align*}$$

(24)

where $c$ is a positive constant, usually with $c \gg 1$, and the $H_\infty$ penalty can be written as

$$H(\rho) = \frac{1}{2}(\max([0, \tilde{M}_S(\rho) - M_{SD}])^2).$$

(25)
The controller $C(\tau, \rho)$ in (24) can be chosen as belonging to any controller class $C$, linearly parameterized in $\rho$ as $C = \{C(\tau, \rho), \rho \in \mathbb{R}^p\}$. The robustness index $\tilde{M}_S(\rho)$ is estimated at each iteration of the swarm intelligence algorithm optimization following the procedure described in Section 4.1.

Notice that an important insight can be drawn from (24) regarding the choice of $M_{sd}$. If the choice is too ambitious, i.e., $M_{sd}$ much lower than $\tilde{M}_S(\rho_0)$ estimated with an initial VRFT-obtained controller with parameter $\rho_0$, the obtained parameter $\rho$ could be found too distant from the VRFT solution. Such a difference could considerably increase the value of $J_Y(\rho)$ to a point where the performance of the swarm intelligence-obtained controller is drastically affected. On the other hand, if the constant $c$ is chosen with an exorbitantly high value, only the cases where the optimization algorithm falls into a local minimum and is not able to minimize $H(\rho)$ would be affected, resulting in a final cost much higher than with a lower value of $c$. If the penalty element $H(\rho)$ is solved by the optimization algorithm, as seen in (25), its value becomes zero, indifferent of the value of $c$, not affecting the VRFT cost $J_Y(\rho)$.

Considering a search space $\mathcal{O} \in [l_i, u_i], l_i, u_i \in \mathbb{R}$, in order to accelerate the convergence of the metaheuristic algorithm, the initialization of the search agents can inherit the first step solution $\rho_0 \in \mathbb{R}^p$ as a central point, as expressed in

$$\bar{X}(0) = R \cdot \overline{X}(0) + \rho_0, \quad R = \frac{\max(I_i, u_i)}{2},$$

with $R$ being the initial population spawn radius, and $\overline{X}(0) \in \mathbb{R}^p$ a random position vector such that $\overline{X}(0) \sim U(0, 1)$.

An inherent step of the method is to collect input–output data from the process, as suggested in Bazanella et al. (2014), Remes et al. (2021). Remember to take into account system identification theory (Ljung, 1999) for data to be sufficiently informative. Then, the two proposed design steps can be applied:

1. Use the VRFT to design a controller for the process. Use a flexible reference model if the plant is NMP, as presented in Campestrini et al. (2011). Check the obtained robustness index and proceed to the second step if it does not satisfy $\tilde{M}_S \leq M_{sd}$, or else, take the controller from Step 1 as final. Such a main step can be divided into the following specific steps:
   - acquire a data set $(u, y)^N_t$ from the closed-loop system with an initial stabilizing controller;
   - use the data set to design a controller with the VRFT method, as detailed in Section 3. Controller parameters $\rho$ are obtained after the minimization procedure of the VRFT method. In the NMP case, parameters for the reference model $\tilde{q}$ are also obtained;
   - estimate the robustness index according to the method in Section 4.1. If $\tilde{M}_S > M_{sd}$ proceed to the second step, or else, use the VRFT-obtained controller with no further modification.
2. Apply a swarm intelligence algorithm, as presented next in Section 5.1, considering the optimization problem described in (24) according to the desired value of $M_S$, with restriction applied in the form of a penalty as in (25), with the initial spawn of agents following the recommendation of (26). The second step can be divided into:
   - implement the VRFT cost function with the penalty as in (25) regarding the desired maximum value of $M_S$;
   - change the initialization procedure of the chosen swarm intelligence algorithm to consider a center spawn $\rho_0$, i.e., the VRFT-obtained solution at the first step, and a spawn radius as suggested in (26) to accelerate convergence;
   - execute the algorithm and obtain controller parameters that satisfy the robustness restriction.

The swarm intelligence algorithms considered for obtaining the results of this work are described next.

5.1. Swarm intelligence algorithms

Swarm intelligence algorithms consist of stochastic optimization algorithms that are based on the collective intelligence of groups composed of simple agents, usually based on the behavior of animals in nature (Bonabeau et al., 2001). In order to cope with the NFL theorems (Wolpert and Macready, 1997), four algorithms are chosen to be used: Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) and Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007), which are well known and widely used (Du and Swamy, 2016; Talbi, 2009); Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014) and Improved Grey Wolf Optimizer (I-GWO) (Nadimi-Shahraki et al., 2021), more recent approaches with fewer hyperparameters than the aforementioned. In the following, the four algorithms are briefly presented.

5.1.1. Particle swarm optimization

Particle Swarm Optimization (PSO) involves populations (or swarms) in which each element is called a particle that represents a form of directed mutation (Talbi, 2009; Kennedy and Eberhart, 1995).

The swarm is composed of $\ell$ particles searching in a $D$-dimensional space, which are initialized randomly within the search space with lower and upper bounds $l_i$ and $u_i$. Each particle has its own position $(\bar{X}_i)$ and velocity $(\bar{V}_i)$ and is considered as a possible solution for the problem. The best solution found locally by a particle $i$ is represented by $\bar{P}_i = (\bar{P}_{i1}, \bar{P}_{i2}, \ldots, \bar{P}_{ip})$, while $\bar{G} = (\bar{G}_{1}, \bar{G}_{2}, \ldots, \bar{G}_{P})$ is the best solution found globally. As for the standard algorithm, each particle is initialized at a random location with random velocity.

The pseudo-code for PSO is given in Appendix A, in which $\max$, $\hat{f}$ is the maximum number of iterations and $f$ is the cost function. The constant $\omega_i$ is an inertia weight, $C_1$ is the cognitive learning factor, and $C_2$ is the social learning factor. Details for the implementation are available in the literature (Kennedy and Eberhart, 1995).

5.1.2. Artificial bee colony

The Artificial Bee Colony (ABC) algorithm simulates the behavior of bees performed during their foraging process, conducting a local search in each iteration. Possible solutions are represented by food sources, while the quality of each solution is proportional to the nectar amount in each source (Du and Swamy, 2016; Karaboga, 2005; Karaboga and Basturk, 2007). There are three types of bees: scout, employed, and onlooker. At the initialization, the scout bees randomly find possible food sources (solutions). Each food source receives an employed bee. By roulette wheel selection, onlooker bees choose food sources to be exploited based on their quality, but both types perform local searches in their neighborhood.

The pseudo-code in Appendix B describes the ABC optimization procedure, where $\bar{X}_i$ with $D$ dimensions is the location of food sources. In this form, $\ell$ is the number of possible solutions (food sources) and the number of scouts, employed, and onlooker bees are taken as the same as the number of food sources. $L$ is the abandonment criteria, defined by the designer. Implementation details are available in Karaboga (2005).

5.1.3. Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) is an algorithm based on the hunting behavior of grey wolves, which have a strict social dominant hierarchy. The leaders are the alphas ($\alpha$), who are responsible for making decisions. At the second level are the betas ($\beta$), subordinates to the alphas who help in decision-making and other pack activities. The third level wolves are the deltas ($\delta$), representing scouts, sentinels, elders, hunters, and caretakers. The rest of the pack is called omega, which must submit to the higher-ranking wolves (Nadimi-Shahraki et al., 2021).

In the GWO algorithm, all wolves follow the mean position of the $\alpha$, $\beta$, and $\delta$s. At each iteration, the new three best wolves are re-defined, with the $\alpha$ position being the final solution. The pseudo-code for the GWO algorithm is presented in Appendix C. Details for the implementation can be checked at Mirjalili et al. (2014).
5.1.4. Improved Grey Wolf Optimizer

There are three main problems noted in literature around the GWO algorithm (Nadimi-Shahraki et al., 2021): (i) lack of population diversity; (ii) imbalance between exploitation and exploration; (iii) premature convergence. The Improved Grey Wolf Optimizer (I-GWO) tries to solve those issues by changing the search strategy of the GWO algorithm, including the Dimension Learning-based Hunting (DLH) strategy, which defines a new (possible) update position for each wolf based not just on the position of as, β, and δ wolves, but also on the position of its neighbors. Appendix D briefly describes the I-GWO. Details of the implementation are given at Nadimi-Shahraki et al. (2021).

6. Validation results

In order to validate and illustrate the proposed method, two real-world-inspired examples are considered. The method is applied as suggested in Section 5 with all four swarm intelligence algorithms commented in Section 5.1. The results are compared in terms of: (i) cost value obtained for best solution — best (lowest) cost; (ii) \( \|S(z, \rho)\|_{\infty} \) value obtained for best solution; (iii) convergence speed. Notice that the system model is only used to generate data in simulation. The knowledge of the model is neglected at any stage of the design, maintaining a pure data-driven fashion.

6.1. Example 1: a second-order non-minimum phase plant

The first system to be considered is

\[
G_1(z) = \frac{-0.05(z - 1.4)}{z^2 - 1.7z + 0.7325} \tag{27}
\]

with a time step of 1 s, which is similar to the discrete-time model of a Boost/Buck–Boost converter operating in continuous conduction mode, regarding the transfer function of the output voltage by the duty cycle (Erickson and Maksimovic, 2001). The presence of a non-minimum phase zero makes it necessary to use the VRFT with a flexible criterion (Bazanella et al., 2014) at the first step of the proposed method.

Assuming that the system model (27) is unknown, there is no previous knowledge about its zero being Non-Minimum Phase (NMP). In this sense, it is possible to analyze the estimated impulse response since the IR of NMP systems initially moves in the opposite direction (downwards) related to the steady-state one (Brunton and Kutz, 2019). Therefore, a Pseudo-Random Binary Signal (PRBS), which is persistently exciting of high order (Ljung, 1999), containing \( N = 2000 \) samples is applied to \( G_1(z) \) in simulation, generating an output signal. Additive white Gaussian noise with a Signal-to-Noise Ratio (SNR) of 20 dB was added to the system at the output, representing measurement noise. With the input–output dataset, the IR of \( G_1(z) \) can be identified with an IR identification algorithm available in the literature (Chen and Ljung, 2013; Fiorio et al., 2021; Anon, 2021; Yerramilli and Tangirala, 2017), resulting in the signal presented in Fig. 3. Clearly, the IR initially goes downwards, indicating the presence of an NMP zero, justifying the VRFT with a flexible reference criterion (Bazanella et al., 2014).

6.1.1. Data collection

The data for estimation is obtained in a closed-loop with a proportional stabilizing controller (Remes et al., 2021) since its presence in the system avoids signal divergence. By the small gain theorem (Skogestad and Postlethwaite, 2005), a stabilizing controller can be obtained following

\[
k_p < \frac{1}{\|G\|_{\infty}}. \tag{28}\]

Therefore, the stabilizing controller \( k_p \) is chosen as

\[
k_p = \frac{0.5}{\|G_1(z)\|_{\infty}} = 0.8039. \tag{29}\]

In order to obtain the \( H_{\infty} \) norm of \( G_1(z) \), its impulse response is estimated according to Fiorio et al. (2021) and the norm is calculated as proposed in Section 4.1.

The excitation signal considered for the data acquisition is a PRBS with \( N = 2000 \) samples. The signal is applied to the control reference of the closed-loop system formed by \( G_1(z) \) with stabilizing controller \( k_p \). The control output signal \( u(k) \) and the system output signal \( y(k) \) are acquired, forming the input–output set \( \{u, y\}_{k=1}^{\infty} \).

6.1.2. Step 1 — VRFT with flexible criterion

Assume a situation where the control requirements are: (i) null error in steady-state; (ii) settling time approximately 2.5 times faster than in closed-loop with the stabilizing controller \( k_p \); (iii) null overshoot for a step reference. A reference model that meets such requirements, following the guidelines of Remes et al. (2021), is chosen as

\[
T_d(z, \tilde{\eta}, \tilde{\rho}) = \frac{-21(z - 1.01)}{(z - 0.7)(z - 0.3)}. \tag{30}\]

Notice that the initial zero of \( T_d(z) \) is set as greater than 1, as suggested in Bazanella et al. (2014), allowing for the VRFT with a flexible criterion to identify the plant’s NMP zero. The chosen controller class chosen to be used is the PID class of controllers, which gives

\[
C(z) = \left[ \begin{array}{cc} -0.4793 & 0.6377 \\ -0.4793 & 0.6377 \end{array} \right] \tag{31}\]

After solving the cost function according to the VRFT method with flexible criterion, the following solution pair for \( \eta, \rho \) is obtained:

\[
\hat{\eta} = \begin{bmatrix} -0.4793 \\ 0.6377 \end{bmatrix}, \quad \hat{\rho} = \begin{bmatrix} [k_p & k_i & k_d] \end{bmatrix} = \begin{bmatrix} [1.1246 & 0.3124 & 6.9713] \end{bmatrix}. \tag{32a} \tag{32b}\]

resulting in a new reference model \( T_d(z, \hat{\eta}) \), and in the controller \( C(z, \hat{\rho}) \), respectively:

\[
T_d(z, \hat{\eta}) = \hat{\eta}F(z) = \frac{-0.6899(z - 1.33)}{(z - 0.7)(z - 0.2401)}; \tag{33a}\]

\[
C(z, \hat{\rho}) = \hat{\rho}\hat{C}(z) = \frac{8.4083z^2 - 1.792z + 0.8291}{2(z - 1)}. \tag{33b}\]

The non-dominant pole of the reference model, now \( T(z, \hat{\eta}) \), is updated together with the minimization of \( \eta \) and \( \rho \), as suggested in Remes et al. (2021).

By estimating the \( H_{\infty} \) norm of \( S(z, \rho) \) of the closed-loop system based on the solution (33), an \( M_S = 2.1952 \) is obtained, which may be too high for applications that require lower robustness indexes since it is greater than 2 (Skogestad and Postlethwaite, 2005). The next subsection presents the application of the second step of the proposed method to reduce \( M_S \) for the obtained VRFT solution.

Fig. 3. Identified impulse response of \( G_1(z) \).
required to parameterize the controller obtained in the previous iteration for each search agent since it is happen if \( M \) compromise substantially the performance of the system, which could sufficient value in terms of robustness, satisfies (33a). The desired \( \hat{M} \)\® the PSO parameters were chosen as the MATLAB parameters for the PSO and ABC algorithms, except for the number of analysis of results each algorithm was executed 50 times. The hyper-

\[
\text{100. In order to obtain a satisfactory number of realizations for the distribution within the suggested spawn radius in (26),}
\]

\[
\text{\( R \) = \text{\( k \)}_2 / 2 = 5, with the central point equal to the VRFT solution at Step 1 (32b).}
\]

The reference model for the cost function (24) is considered to be the VRFT-obtained controller with a flexible criterion reference model, \( T_d(z, \hat{\eta}) \), obtained in Step 1 as (33a). The desired \( M_S \) to be achieved, \( M_S \), is set to 1.8, which is a sufficient value in terms of robustness, satisfies \( M_S \leq 2 \) and should not compromise substantially the performance of the system, which could happen if \( M_S \ll 2 \).

6.1.3. Step 2 — Swarm intelligence algorithm

The use of swarm intelligence algorithms for solving the proposed problem is straightforward. At their implementation – see Appendices A–D for the pseudo-codes – the cost function \( f \) is the proposed \( J_S(r) \) (24). At each iteration, the input to \( f \) is the parameter \( r \), obtained in the previous iteration for each search agent since it is required to parameterize the controller \( C(z, \rho) \) and to estimate the \( H_{\infty} \) norm \( M_S(r) \).

The computation of \( J_S(r) \) is done by directly calculating the cost (24). The norm \( M_S(r) \) can be obtained through (18), estimating the impulse response of \( S(z, \rho) \) with the signals proposed in (22) and an IR estimation algorithm available in the literature, such as (Fiorio et al., 2021).

The upper search bound for the swarm intelligence algorithms is defined as \( u_0 = 10 \), which should be sufficient considering that the maximum \( r \) value of the VRFT-obtained controller is 6.9713 and, taking into account a choice of \( M_S \) that is not too ambitious, the resultant \( r \) should not be too distant from the initialized value. The lower search bound is chosen as \( l_0 = 0 \) to avoid negative controller gain, making the obtained controller passive (Bao and Peter, 2007).

The initialization of all agents is done randomly with a uniform distribution within the suggested spawn radius in (26), \( R = u_0 / 2 = 5, \) with the central point equal to the VRFT solution at Step 1 (32b). The reference model for the cost function (24) is considered to be the VRFT with a flexible criterion reference model, \( T_d(z, \hat{\eta}) \), obtained in Step 1 as (33a). The desired \( M_S \) to be achieved, \( M_S \), is set to 1.8, which is a sufficient value in terms of robustness, satisfies \( M_S \leq 2 \) and should not compromise substantially the performance of the system, which could happen if \( M_S \ll 2 \).

To make the comparison between algorithms possible, the number of agents was fixed to 50 and the number of iterations was limited to 100. In order to obtain a satisfactory number of realizations for the analysis of results each algorithm was executed 50 times. The hyper-parameters for the PSO and ABC algorithms, except for the number of agents and the maximum number of iterations, are presented in Table 1. The PSO parameters were chosen as the MATLAB® default parameters of the Global Optimization Toolbox (Anon, 2021), while the ABC parameters were used according to the algorithm implementation of Heris (2015). GWO and I-GWO do not contain any hyperparameter aside from the number of agents and the maximum number of iterations.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter settings</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>Cognitive learning factor (C₁)</td>
<td>1.49</td>
</tr>
<tr>
<td></td>
<td>Social learning factor (C₂)</td>
<td>1.49</td>
</tr>
<tr>
<td></td>
<td>Inertia range (range of uᵢ)</td>
<td>[0.1,1.1]</td>
</tr>
<tr>
<td>ABC</td>
<td>Abandonment criteria (L)</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>Acceleration coefficient (a)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Parameters settings for PSO and ABC.

Table 2: Time for convergence of all algorithms for example 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1-s. time (s)</th>
<th>It. to converge</th>
<th>Time to converge (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>8.04</td>
<td>41</td>
<td>329.75</td>
</tr>
<tr>
<td>ABC</td>
<td>22.63</td>
<td>19</td>
<td>430.02</td>
</tr>
<tr>
<td>GWO</td>
<td>11.25</td>
<td>14</td>
<td>157.53</td>
</tr>
<tr>
<td>I-GWO</td>
<td>24.18</td>
<td>20</td>
<td>483.66</td>
</tr>
</tbody>
</table>

Table 3: Quantitative results from the box plot in terms of best cost for example 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>median</th>
<th>σ</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>0.2032</td>
<td>0.2782</td>
<td>0.2017</td>
<td>2.1710</td>
</tr>
<tr>
<td>ABC</td>
<td>0.2334</td>
<td>0.0402</td>
<td>0.2025</td>
<td>0.4009</td>
</tr>
<tr>
<td>GWO</td>
<td>0.2489</td>
<td>0.0858</td>
<td>0.2017</td>
<td>0.4764</td>
</tr>
<tr>
<td>I-GWO</td>
<td>0.2017</td>
<td>5.6516 × 10⁻⁵</td>
<td>0.2017</td>
<td>0.2018</td>
</tr>
</tbody>
</table>

Fig. 4 shows the average convergence curve of all algorithms for 50 runs, considering the system \( G_1(z) \) as aforementioned. Table 2 presents the time that each iteration took and the number of iterations to converge, considering the average value for all 50 realizations and a convergence criterion of \( \delta = 1 \times 10^{-3} \) from an iteration cost to the subsequent one. The results were obtained with an Intel Core i5 4670 3.40 GHz processor. The I-GWO algorithm took a longer time to converge, followed by ABC, PSO, and finally, GWO. Comparing similar algorithms, I-GWO obtained a duration for one iteration that is more than twice the same duration for the GWO. But since the optimization of the proposed cost function is executed offline, where the obtained parameters (\( \rho \)) are applied to the controller without any further modification, this is not considered an issue for the current application.

Fig. 4 shows the average convergence curve of all algorithms for 50 runs, considering the system \( G_1(z) \) as aforementioned. Table 2 presents the time that each iteration took and the number of iterations to converge, considering the average value for all 50 realizations and a convergence criterion of \( \delta = 1 \times 10^{-3} \) from an iteration cost to the subsequent one. The results were obtained with an Intel Core i5 4670 3.40 GHz processor. The I-GWO algorithm took a longer time to converge, followed by ABC, PSO, and finally, GWO. Comparing similar algorithms, I-GWO obtained a duration for one iteration that is more than twice the same duration for the GWO. But since the optimization of the proposed cost function is executed offline, where the obtained parameters (\( \rho \)) are applied to the controller without any further modification, this is not considered an issue for the current application.

Considering all 50 executions per swarm intelligence algorithm, Fig. 5 presents the best (lower) cost statistics for all algorithms at example 1 in the form of a box plot. Clearly, I-GWO achieved the most desirable performance in terms of cost since it contains fewer outliers and a very low dispersion if compared to the other algorithms' solutions. PSO, ABC, and GWO, in general, result in higher cost values than I-GWO for the considered cost function. Table 3 shows the quantitative values related to the best cost of all algorithms at each run, confirming the conclusions taken from Fig. 5. The obtained values for the parameter \( \rho = [k_S k_{z,2}] \) are presented in Fig. 6 and show how the lower cost dispersion obtained with the I-GWO algorithm can influence the values of the controller parameters and their tendency. For the case

Fig. 5. Box plot of a Monte Carlo experiment with 50 executions for all algorithms in terms of best cost value obtained for example 1.

Fig. 4. Average convergence curves for all algorithms considering a Monte Carlo experiment of 50 executions for example 1.
of example 1, the I-GWO is the most robust algorithm in relation to the random initialization of the parameters. Such a characteristic is desired in a real-world scenario, reducing the number of trials in the design procedure to obtain a satisfactory solution. Also, notice that the median solution obtained with the I-GWO is not too far from the VRFT solution (32b) obtained at Step 1, which agrees with the previous hypothesis that a value of desired robustness $\hat{M}_S$ chosen as not too ambitious when compared with the initial VRFT solution (32b) should result in a final solution that is close to the initial (VRFT) one.

Fig. 7 presents the box plot for the obtained $||S(z, \hat{\rho})||_\infty$ by the best solution of each algorithm at each run, in a closed-loop with $G_1(z)$. I-GWO obtained the most desired result in terms of $\hat{M}_S$ considering the lack of outliers and the low dispersion. PSO had one outlier with $\hat{M}_S > 2$, while ABC obtained three outliers of higher $\hat{M}_S$, and the performance by the GWO algorithm for this problem was not satisfactory since there were too many solutions that achieved an $\hat{M}_S$ higher than 2. Table 4 shows the quantitative data of the box plot presented in Fig. 7, in agreement with what is commented over the results.

To briefly demonstrate that, although the robustness is increased with the proposed method, the reference tracking is not overly penalized, a step reference was applied to the controlled plant of example 1 using a controller designed by the proposed method with the I-GWO algorithm, with parameters $\hat{\rho}_{I-GWO} = [1.090, 0.2194, 5.4018]$, and the VRFT controller (32b) obtained at step 1. As shown in Fig. 8, the VRFT-controlled system achieves a settling time of 54 s, a step overshoot of 20% and an undershoot of 42%. For the proposed method, a settling time of 39 s is achieved, with 9% of overshoot and 33% of undershoot. Notice that, in this specific case, the proposed method could even enhance the reference tracking performance in terms of settling time, however, this is not expected in most of the systems since the inclusion of a robustness constraint will penalize the VRFT cost function. The reduced overshoot and undershoot are a consequence of the increased robustness, with an $\hat{M}_S$ of 2.2079 for the VRFT and 1.8030 for the I-GWO-designed controller.

6.2. Example 2: fourth-order plant

The fourth-order plant consists of

$$G_2(z) = \frac{0.1381(z - 0.95)(z^2 - 1.62z + 0.6586)}{(z^2 - 1.7z + 0.7325)(z^2 - 1.84z + 0.8564)}, \quad (34)$$

with a time step of 1 s, which has the same structure as the discrete-time transfer function of a SEPIC converter, from duty cycle to output voltage (Kassick, 2011). Since the plant’s zeros have minimum phase, which can be evaluated with data as aforementioned in Section 6.1, the VRFT method is used without the flexible reference model (Bazanella et al., 2014).
Fig. 8. Output signal for a reference step signal applied to the controlled plant with a controller designed with the proposed method using I-GWO and the VRFT-designed controller for example 1.

Fig. 9. Average convergence curves for all algorithms considering a Monte Carlo experiment of 50 executions for example 2.

Table 5

<table>
<thead>
<tr>
<th>Alg.</th>
<th>1-std. time (s)</th>
<th>It. to converge</th>
<th>Time to converge (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>7.24</td>
<td>20</td>
<td>144.72</td>
</tr>
<tr>
<td>ABC</td>
<td>22.93</td>
<td>10</td>
<td>229.32</td>
</tr>
<tr>
<td>GWO</td>
<td>7.55</td>
<td>9</td>
<td>68.00</td>
</tr>
<tr>
<td>I-GWO</td>
<td>22.34</td>
<td>9</td>
<td>201.10</td>
</tr>
</tbody>
</table>

6.2.1. Data collection

For plant $G_2(z)$, the data is obtained in the same way as described, for example 1, in Section 6.1, with a PRBS signal of $N = 2000$ samples applied to the closed-loop system with stabilizing controller $k_p = 0$.

\[
\|G_2(z)\|_\infty = 0.3828, \quad (35)
\]

considering additive white Gaussian noise with an SNR of 20 dB to represent measurement noise. The input-output set is formed by \(\{u, y\}\) 

6.2.2. Step 1 — VRFT

After the data is acquired, the next step is to use the VRFT to design a controller, which solves the cost function $f$. For this example, the following control requirements are assumed: (i) null steady-state error; (ii) settling time of approximately 6.5 times faster than the closed-loop settling time with stabilizing controller $k_p$; (iii) null overshoot for a step reference. Considering such requirements, the choice of the reference model is made as suggested in Remes et al. (2021), obtaining

\[
Td(z) = \frac{1.4(z - 0.6)}{(z - 0.3)(z - 0.2)} \quad (36)
\]

Supposing a limited situation where only a PI controller is available, e.g., because of hardware limitations on a certain product. Therefore, the controller class to be considered is the PI class of controllers, resulting in

\[
C(z) = \left[\begin{array}{c} z \\ z - 1 \end{array}\right]^T. \quad (37)
\]

The obtained VRFT solution results in the controller parameter

\[
\hat{\rho} = [k_p, k_i] = [6.6568, 3.3728]. \quad (38)
\]

Via (3), the controller

\[
C(z, \hat{\rho}) = \hat{\rho}' \hat{C}(z) = \frac{10.03(z - 0.6637)}{(z - 1)} \quad (39)
\]

is obtained.

Considering the VRFT-obtained solution (38), the robustness index of the system can be estimated according to Section 4, obtaining $M_S = 2.2767$. As aforementioned, an $M_S \leq 2$ is desired to ensure sufficient robustness (Skogestad and Postlethwaite, 2005), which leads to the application of the second step of the proposed method.

6.2.3. Step 2 — swarm intelligence algorithm

The swarm intelligence algorithms PSO, ABC, GWO, and I-GWO are applied to the problem (24) for the fourth-order plant case, according to the pseudo-codes presented in Appendices A–D, where $f$ is the cost function (24), as already commented at Section 6.1.3. The upper search bound is kept at $u_b = 10$ and the lower bound at $l_b = 0$, in order to increase the passivity of the controller as mentioned in Section 6.1. An upper bound of 10 should be sufficient, considering that the maximum desired robustness is not too far from the estimated robustness index at the end of step 1. The initial population spawn radius follows (26), $R = (|u_b| + |l_b|)/2 = 5$, with the solution of the VRFT at Step 1 (38) as the central point. The desired $\|S(z, \hat{\rho})\|_\infty$ is set to 1.5, which satisfies $M_S \leq 2$ and is not much lower than 2.

The number of agents of all algorithms is set to 50, with a maximum of 100 iterations per execution. Each algorithm is executed 50 times for different noise realizations, such that the results are sufficiently representative for further analysis. For PSO and ABC algorithms, parameters are set as presented in Table 1. Aside from the number of agents and
the maximum number of iterations, no other parameter is set by the user with the proposed GWO and I-GWO algorithm.

The average convergence curve of all algorithms for this case is presented in Fig. 9. Table 5 presents the time for one iteration and the number of iterations each algorithm took to converge, considering all executions. PSO, ABC, and I-GWO did not present far outliers, as those seen in the case of GWO. In comparison to example 1, GWO is still the fastest algorithm to converge, while ABC took I-GWO place as the slowest convergence.

Fig. 10 shows the box plot regarding the best fitness value for each algorithm, considering all executions. PSO, ABC, and I-GWO did not present far outliers, as those seen in the case of GWO. The quantitative values of the box plot are shown in Table 6. In this case, the most desirable result is obtained by the PSO algorithm. Although, notice that the scale of the y-axis (best cost) only varies at the third decimal place, which means that PSO, ABC, and I-GWO solutions should perform very similarly in practice.

The $\|S(z, \hat{\rho})\|_\infty$ norm obtained for the best solution at each run is shown in Fig. 11, with its quantitative values presented in Table 7. PSO, ABC, and I-GWO present very similar values for the obtained $H_\infty$ norm, which shows that the performance of those solutions is close, as also noticed from the cost analysis. GWO is the only algorithm that presents outliers in this case, which, in comparison to I-GWO, indicates that the local minima problems pointed out in literature (Nadimi-Shahraki et al., 2021) indeed can occur.

Additionally, similar conclusions can be drawn from the controller parameter’s values obtained with each optimization algorithm for the proposed problem. In Fig. 12, outliers are only obtained with the GWO algorithm, where the gain of the integral part of the controller, $k_i$, achieves values close to zero, which has, as a consequence, a lower performance for the controlled system, even though the robustness criteria is met. The obtained $\rho$ for PSO, ABC, and I-GWO are very similar for all executions, indicating that any of the three algorithms could be used for this example without significant loss of performance or robustness when compared to one another.

In the same way as for example 1, a step reference is applied to the controlled plant of example 2 using a VRFT-designed controller and a controller designed with the proposed method using I-GWO, as shown in Fig. 13. In this case, as one would expect, the reference tracking was penalized in terms of settling time, from 5 s (VRFT) to 14 s (I-GWO-designed). The overshoot is 42% for the VRFT controller and 29% for the proposed method with I-GWO, reflecting the reduced $\hat{M}_c$ from 2.3834 to 1.5156.

### Table 6
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>median</th>
<th>$\sigma$</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>0.49284</td>
<td>$2.2270 \times 10^{-5}$</td>
<td>0.49284</td>
<td>0.49284</td>
</tr>
<tr>
<td>ABC</td>
<td>0.49287</td>
<td>$4.7174 \times 10^{-5}$</td>
<td>0.49284</td>
<td>0.49303</td>
</tr>
<tr>
<td>GWO</td>
<td>0.49290</td>
<td>$1.0414 \times 10^{-2}$</td>
<td>0.49284</td>
<td>0.53568</td>
</tr>
<tr>
<td>I-GWO</td>
<td>0.49285</td>
<td>$6.4636 \times 10^{-6}$</td>
<td>0.49284</td>
<td>0.49286</td>
</tr>
</tbody>
</table>

### Table 7
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\rho$</th>
<th>$k_i$</th>
<th>$k_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>1.5169</td>
<td>1.4558 $\times 10^{-6}$</td>
<td>1.5169</td>
</tr>
<tr>
<td>ABC</td>
<td>1.5165</td>
<td>4.4060 $\times 10^{-4}$</td>
<td>1.5148</td>
</tr>
<tr>
<td>GWO</td>
<td>1.5133</td>
<td>8.4447 $\times 10^{-3}$</td>
<td>1.4825</td>
</tr>
<tr>
<td>I-GWO</td>
<td>1.5168</td>
<td>1.2216 $\times 10^{-4}$</td>
<td>1.5165</td>
</tr>
</tbody>
</table>

### Conclusion
This work proposed a data-driven one-shot technique to increase the robustness of a closed-loop discrete-time system by changing the controller parameters using swarm intelligence algorithms. The considered optimization problem (24) is the VRFT cost function with the addition of a penalty regarding the value of the $\|S(z, \rho)\|_\infty$ norm, which can be directly used as a measure of robustness. Such a value is estimated via an impulse response at each iteration of the metaheuristic algorithm.

Four swarm intelligence algorithms – PSO, ABC, GWO, and I-GWO – have been considered to illustrate the proposed technique with two
real-world inspired plants. For the example of a second-order non-
minimum phase plant, I-GWO obtained the best results in terms of
dispersion, outliers, and cost, with acceptable values of $∥S(z, ρ)∥_∞$. PSO
achieved similar results in this case, while ABC and GWO algorithms
had a higher occurrence of outliers — in terms of cost and $M_∞$. In the
case of the fourth-order minimum-phase plant, ABC, I-GWO, and PSO
obtained very similar results, all satisfactory in terms of cost and $H_∞$
norm of $S(z, ρ)$. GWO, on the other hand, obtained a few outliers that,
although the criterion of robustness was met, those cases achieved a
very low integral gain, which could drastically affect the performance
although the criterion of robustness was met, those cases achieved a
very low integral gain, which could drastically affect the performance
of the system. In terms of speed of convergence, I-GWO and ABC were
the slowest algorithms, while PSO converged faster and GWO was the
fastest.

As for future works, it is suggested: the inclusion of other constraints
(e.g., for control effort) simultaneously with the robustness constraints;
the use of other types of metaheuristics, as evolutionary of physics-
based algorithms; the inclusion of a robustness constraint to the OCI,
VDFT, or DD-LQR methods; extension of the current work for MIMO
systems.

CRediT authorship contribution statement

Luan Vinicius Fiorio: Conceptualization, Methodology, Software,
Formal analysis, Investigation, Writing – original draft, Visualization.
Christian Lennon Remes: Conceptualization, Methodology, Validation,
Formal analysis, Writing – original draft, Supervision, Project adminis-
tration. Patrick Wheeler: Writing – review & editing. Yales Rômulo
de Novais: Conceptualization, Writing – original draft, Supervision,
Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial
interests or personal relationships that could have appeared to
influence the work reported in this paper.

Data availability

The data used for the obtention of results is the outcome of simu-
lations, which can be replicated by following the equations and proce-
dures mentioned and described in the paper.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoama-
tamento de Pessoal de Nível Superior (CAPES) - Brazil - Finance Code

001, partly by the Fundação de Amparo à Pesquisa e Inovação do Es-
tado de Santa Catarina (FAPESC) - Grant number 288/2021, and partly
by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Brazil.

Appendix A

Algorithm 1: Particle swarm optimization pseudo-code

\begin{algorithm}
\caption{Particle swarm optimization pseudo-code}
\begin{algorithmic}
\STATE \textbf{input} : $f, l_\ell, u_\ell, \ell, \text{max}_i, \ell, \ell, C_1, C_2$
\STATE \textbf{output} : $\bar{G}$
\STATE 1 \textbf{initialize} $\ell$ particles with random position $X_i(0)$ within $l_\ell, u_\ell$,
\STATE \quad \text{and velocity $\bar{V}_i(0)$};
\STATE 2 \textbf{define} (initial) best local solution $\bar{P}_j$ as $f[\bar{X}_i(0)]$ for each
\STATE \quad particle;
\STATE 3 \textbf{define} (initial) best global solution $\bar{G}$ as the position of the
\STATE \quad particle with the best fitness;
\STATE 4 \textbf{for} $n \in \{1, ..., \text{max}_i\}$ \textbf{do}
\STATE \quad \textbf{for} $i \in \{1, ..., \ell\}$ \textbf{do}
\STATE \quad \quad \text{define new velocities $\bar{V}_i(n)$ for each particle with $w_1,$}
\STATE \quad \quad \quad $C_1, C_2$;
\STATE \quad \quad \text{define new positions $\bar{X}_i(n)$ for each particle;}
\STATE \quad \quad \text{update best local solution for all particles;}
\STATE \quad \text{if} \quad \text{abandonment counter exceeds} \ L \text{then}
\STATE \quad \quad \text{increase abandonment counter;}
\STATE \quad \text{else}
\STATE \quad \quad \text{select a food source by roulette wheel selection based}
\STATE \quad \quad \quad \text{on the probability of such position to have more}
\STATE \quad \quad \quad \text{nectar in comparison with others;}
\STATE \quad \quad \text{define a position for the $i$-th onlooker bee based on}
\STATE \quad \quad \quad \text{the selected food source;}
\STATE \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \text{if the new position has higher fitness than the current then}
\STATE \quad \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \quad \text{else}
\STATE \quad \quad \quad \text{increase abandonment counter;}
\STATE \quad \text{else}
\STATE \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \text{if the new position has higher fitness than the current then}
\STATE \quad \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \quad \text{else}
\STATE \quad \quad \quad \text{increase abandonment counter;}
\STATE \quad \quad \text{end if}
\STATE \quad \text{end if}
\STATE \text{end for}
\STATE \text{end for}
\STATE \textbf{end for}
\STATE \text{position of the particle with higher fitness}
\end{algorithmic}
\end{algorithm}

Appendix B

Algorithm 2: Artificial bee colony pseudo-code

\begin{algorithm}
\caption{Artificial bee colony pseudo-code}
\begin{algorithmic}
\STATE \textbf{input} : $f, l_\ell, u_\ell, \ell, \text{max}_i, \ell, L$
\STATE \textbf{output} : food source with more nectar (higher fitness)
\STATE 1 \textbf{initialize} $i$ food sources with position $X_i(0)$ within boundaries
\STATE \quad $l_\ell, u_\ell$;
\STATE 2 \textbf{define} (initial) best solution $\bar{G}$ as the food source with higher
\STATE \quad fitness;
\STATE 3 \textbf{for} $n \in \{1, ..., \text{max}_i\}$ \textbf{do}
\STATE \quad \textbf{for} $i \in \{1, ..., \ell\}$ \textbf{do}
\STATE \quad \quad \text{define a new position for the $i$-th employed bee based}
\STATE \quad \quad \quad \text{on the current food sources;}
\STATE \quad \quad \quad \text{if the new position has higher fitness than the current then}
\STATE \quad \quad \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \quad \text{else}
\STATE \quad \quad \quad \quad \text{increase abandonment counter;}
\STATE \quad \quad \text{end if}
\STATE \quad \text{end for}
\STATE \quad \textbf{for} $i \in \{1, ..., \ell\}$ \textbf{do}
\STATE \quad \quad \text{select a food source by roulette wheel selection based}
\STATE \quad \quad \quad \text{on the probability of such position to have more}
\STATE \quad \quad \quad \text{nectar in comparison with others;}
\STATE \quad \quad \text{define a position for the $i$-th onlooker bee based on}
\STATE \quad \quad \quad \text{the selected food source;}
\STATE \quad \quad \text{if the new position has higher fitness than the current then}
\STATE \quad \quad \quad \quad \text{set as the new position for the employed bee;}
\STATE \quad \quad \quad \text{else}
\STATE \quad \quad \quad \quad \text{increase abandonment counter;}
\STATE \quad \text{end if}
\STATE \quad \text{end for}
\STATE \textbf{end for}
\STATE \text{position of the particle with higher fitness}
\end{algorithmic}
\end{algorithm}
Appendix C

Algorithm 3: Grey wolf optimizer pseudo-code

\[
\begin{align*}
\text{input:} & \quad f, \mathbf{l}_p, \mathbf{u}_p, \epsilon, \max_{it} \\
\text{output:} & \quad \text{position of the } \alpha \text{ wolf} \\
\text{initialize } \epsilon & \text{ wolves with random position within boundaries } \mathbf{l}_p, \\
& \text{define the three higher fitness wolves as } a, \beta, \text{ and } \delta; \\
& \text{for } n \in \{1, \ldots, \max_{it}\} \\
& \text{for } i \in \{1, \ldots, \ell\} \\
& \quad \text{position is updated following the mean of the } a, \beta, \text{ and } \delta \text{ wolves;} \\
& \quad \text{the three wolves with higher fitness are the new } a, \beta, \text{ and } \delta; \\
\end{align*}
\]

Appendix D

Algorithm 4: Improved grey wolf optimizer pseudo-code

\[
\begin{align*}
\text{input:} & \quad f, \mathbf{l}_p, \mathbf{u}_p, \epsilon, \max_{it} \\
\text{output:} & \quad \text{position of the } \alpha \text{ wolf} \\
\text{initialize } \epsilon & \text{ wolves with random position within boundaries } \mathbf{l}_p, \\
& \text{define the three higher fitness wolves as } a, \beta, \text{ and } \delta; \\
& \text{for } n \in \{1, \ldots, \max_{it}\} \\
& \text{for } i \in \{1, \ldots, \ell\} \\
& \quad \text{the candidate position is defined following the mean of the } a, \beta, \text{ and } \delta \text{ wolves as in GWO;} \\
& \quad \text{a neighborhood is constructed based on the current position and candidate position;} \\
& \quad \text{multi-neighbor learning is performed (DLH solution);} \\
& \quad \text{choose best position between candidate and DLH for the update;} \\
& \quad \text{the three wolves with higher fitness are the new } a, \beta, \text{ and } \delta; \\
\end{align*}
\]

References


