SoftwareX 11 (2020) 100383

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

pyvrft: A Python package for the Virtual Reference Feedback Tuning, a direct data-driven control method

Emerson Boeira *, Diego Eckhard

Universidade Federal do Rio Grande do Sul, Av. Osvaldo Aranha, 103 - Porto Alegre-RS - CEP: 90035-190, Brazil

ARTICLE INFO

Article history:
Received 17 September 2019
Received in revised form 19 November 2019
Accepted 9 December 2019

Keywords:
Control systems
Data-driven control
VRFT
Python

ABSTRACT

In this paper, the *pyvrft*, a Python package for the data-driven control method known as Virtual Reference Feedback Tuning (VRFT), is presented. Virtual Reference Feedback Tuning is a control design technique that does not use a mathematical model from the process to be controlled. Instead, it uses input and output data from an experiment to compute the controller's parameters, aiming to minimize an H_2 Model Reference criterion. The package implements an unbiased estimate of the controller for MIMO (Multiple-Input Multiple-Output) processes using both least-squares and instrumental variable techniques. The package also provides accessory functions to import data and to perform MIMO systems simulations, together with some examples.

Code metadata

Current code version	v1.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_285
Legal Code License	MIT
Code versioning system used	Git
Software code languages, tools, and services used	Python 3
Compilation requirements, operating environments & dependencies	numpy, scipy, matplotlib
If available Link to developer documentation/manual	
Support email for questions	emerson.boeira@ufrgs.br

1. Motivation and significance

Data-driven control methods emerged in the literature on the early 40's, with the work of Ziegler and Nichols [1], which provided practical formulas, based on simple experiments, to tune PID controllers. However, most of the data-driven control methods appeared and attained more visibility after the 90's, and they still are being researched and developed on the control systems community until the present day. The main objective of these techniques is to tune a predefined and fixed order controller for a dynamic process, using batches of input and output data, without the necessity of the process' mathematical model [2]. Therefore, since the task to obtain a good and reliable process' model can be very expensive and time-consuming, data-driven control became a very attractive matter for a wide range of practical and industrial applications [3].

It is usual, among data-driven control literature, to classify the methods on two distinct groups: the group of iterative methods, which use several experiments to update the controller's parameters iteratively, and the group of direct methods, that require only one or two batches of data to tune the controller. The most popular iterative methods are the Iterative Feedback Tuning (IFT) [4], the Frequency Domain Tuning (FDT) [5] and the Iterative Correlation-based Tuning (ICbT) [6] and the most popular direct methods are the Virtual Reference Feedback Tuning (VRFT) [7], the Non-iterative Correlation-based Tuning (CbT) [8] and the Optimal Controller Identification (OCI) [9]. Within the group of direct methods, the VRFT is the most researched and disseminated one, and it possesses several extensions, applications, and analysis around its properties on the literature. So, to show this high popularity and the significance of the method, some theoretical contributions and some recent papers of VRFT applications are described in the sequence.

* Corresponding author.
E-mail addresses: emerson.boeira@ufrgs.br (E. Boeira), diegoeck@ufrgs.br (D. Eckhard).

https://doi.org/10.1016/j.softx.2019.100383

One of the first extensions for the VRFT was proposed on [10], which introduced the possibility to design two degree of freedom controllers. Another relevant contribution to the method was its extension for multivariable (or MIMO) control systems, demonstrated on [11]. The MIMO scenario was also investigated on [12] and on [13], where the latter introduced an unbiased formulation. The VRFT approach for nonlinear control systems was presented on [14], and on [15] the method was applied to design adaptive PID controllers. As more recent contributions, the work [16] addressed the problem to deal with non-minimum phase plants, and on [17] a solution with asymptotically guaranteed stability was formulated. Also, the work [18] introduced the use of regularization to enhance the method's statistical properties, and the paper [19] formulated a new setup of the VRFT to perform load disturbance rejection.

Concerning the applications of the VRFT, there is also an extensive literature on the subject. For example, on [20], the VRFT was employed to a classic benchmark problem, where it was used to tune high order controllers for an active suspension system. On [21], the method was used to design feedback controllers for knee joint movement using functional electrical stimulation. The VRFT was also used to tune multivariable controllers in a simulation of a wastewater treatment plant on [22], to design the active braking control of vehicles on [23], to control a non-minimum phase level plant on [24], to control the attitude of a quadcopter on [25], to control MIMO tank systems on [26] and on [27], and to control cavity tuners in particle accelerators on [28].

Accordingly, given the importance of data-driven control in the literature and on practical and industrial applications, the main idea of this work is to introduce a free and open-source package, developed in Python, that implements the unbiased VRFT method for MIMO processes [13], which is, as stated before, the most popular direct data-driven method. It is important to emphasize that there is also a Matlab toolbox for the VRFT, described on [29]. Both the Matlab toolbox and *pyvrft* implement the standard VRFT, with the option to use the least-squares or the instrumental variable to tune the controller. Besides, there are some differences between both software. The Matlab toolbox gives additional options, as to tune nonlinear and two degree of freedom controllers. Those options are not available on *pyvrft*. On the other hand, *pyvrft* implements the unbiased MIMO version of the VRFT, while the Matlab Toolbox was developed only for the case of SISO (Single-Input Single-Output) systems.

1.1. Theoretical background

The core idea of the unbiased MIMO VRFT is to tune a controller for a linear time-invariant (LTI) discrete-time process, which can be represented by the following equation:

$$y(t) = G(q)u(t) + v(t), \tag{1}$$

where q is the forward-shift operator, $G(q)$ is an $n \times n$ rational transfer function matrix, $u(t)$ and $y(t)$ represent, respectively, the control input and output of the process, both described by a n-dimension column vector. The noise vector $v(t)$ can be written as

$$v(t) = H(q)w(t), \tag{2}$$

with $w(t)$ being a white noise n-dimension column vector, where each of its elements has variance denoted by $\sigma^2_{w_i}$, $i = 1, \ldots, n$ and $H(q)$ is an $n \times n$ stable transfer function matrix.

The controller to be tuned is also an $n \times n$ linear time-invariant system, which belongs to a predefined (user-specified) class of transfer function matrices. The controller is parameterized by a parameter vector $\rho \in \mathbb{R}^p$ so that the control action $u(t)$ can be written as

$$u(t, \rho) = C(q, \rho)(r(t) - y(t)), \tag{3}$$

where $r(t)$ is an n-dimensional column vector that represents the reference signal. The MIMO structure of the controller is given by

$$C(q, \rho) = \begin{bmatrix} C_{11}(q, \rho_{11}) & C_{12}(q, \rho_{12}) & \cdots & C_{1n}(q, \rho_{1n}) \\ \vdots & \vdots & & \vdots \\ C_{n1}(q, \rho_{n1}) & C_{n2}(q, \rho_{n2}) & \cdots & C_{nn}(q, \rho_{nn}) \end{bmatrix}, \tag{4}$$

where $\rho = [\rho_{11} \;\; \rho_{12} \;\; \ldots \;\; \rho_{n1} \;\; \ldots \;\; \rho_{nn}]^T$ and it is assumed that each subcontroller has a linear parametrization, i.e. they can be written as

$$C_{ij}(q, \rho_{ij}) = \rho_{ij}^T \bar{C}_{ij}(q), \;\; \rho_{ij} \in \mathbb{R}^m, \tag{5}$$

with $\bar{C}_{ij}(q)$ being an m-dimension column vector of fixed causal rational functions. Each subcontroller can have a different structure, provided that they are linear in the parameters. Finally, the equations that describe the system (1)–(3) under closed-loop control, are

$$y(t, \rho) = T(q, \rho)r(t) + S(q, \rho)v(t) \tag{6}$$

$$S(q, \rho) = (G(q)C(q, \rho) + I)^{-1} \tag{7}$$

$$T(q, \rho) = S(q, \rho)G(q)C(q, \rho) = G(q)C(q, \rho)S(q, \rho), \tag{8}$$

where the dependence of the output on the controller's parameters was made explicit.

With the VRFT method, the objective is to tune the parameter vector ρ to achieve a desired closed-loop performance for $y(t, \rho)$, which is specified through a transfer function matrix, denoted by $T_d(q)$ and also known as the *reference model*. The reference model defines the relationship between the reference signal $r(t)$ and the desired output of the closed-loop system, denoted by $y_d(t)$:

$$y_d(t) = T_d(q)r(t). \tag{9}$$

Then, the VRFT designs the controller in a Model Reference (MR) framework, where the goal is to minimize the $\mathcal{L}_2$ norm of the difference between $y(t, \rho)$ and $y_d(t)$. This problem can be expressed by

$$\hat{\rho}_{MR} = \arg\min_{\rho} J_{MR}(\rho) \tag{10}$$

$$J_{MR}(\rho) \triangleq \sum_{t=1}^{N} \|(T_d(q) - T(q, \rho))r(t)\|_2^2. \tag{11}$$

The optimization problem described on (10)–(11) is non-convex, and from (8), observe that $T(q, \rho)$ depends explicitly on the process' transfer function $G(q)$ that is usually unknown by the user.

On the other hand, the VRFT proposes a convex optimization problem that does not depend on the process' model but uses only input and output data to tune ρ. It assumes that a batch of input/output data is collected from the process:

$$\mathcal{Z} = \{u(1), y(1), u(2), y(2), \ldots, u(N), y(N)\} \tag{12}$$

and, then, the controller's parameters are computed by solving the following optimization problem:

$$\hat{\rho}_{VR} = \arg\min_{\rho} J_{VR}(\rho) \tag{13}$$

$$J_{VR}(\rho) \triangleq \sum_{t=1}^{N} \|L(q)u(t) - L(q)C(q, \rho)(T_d^{-1}(q) - I)y(t)\|_2^2. \tag{14}$$

The filter $L(q)$ is inserted on the method as an extra degree of freedom that can be used to improve the properties of $\hat{\rho}_{VR}$ under non-ideal conditions. The choice of the filter is discussed with more depth and formality at [7,13], where the authors demonstrate that an approximation for the optimal solution can be achieved by

$$|L(e^{j\omega})|^2 = |T_d(e^{j\omega})(I - T_d(e^{j\omega}))|^2 \Phi_r(e^{j\omega})\Phi_u(e^{j\omega})^{-1}, \tag{15}$$

for both SISO and MIMO cases, with $\Phi_r(e^{j\omega})$ and $\Phi_u(e^{j\omega})$ being the power spectrum of $r(t)$ and $u(t)$ respectively.

The convex optimization problem introduced on (13)–(14) has a closed-form solution, given by the least-squares equation. Moreover, under ideal conditions, both $J_{MR}(\rho)$ and $J_{VR}(\rho)$ have the same global minimum, such that the unbiased MIMO VRFT can be successfully used to solve the MR problem [13]. Under non-ideal conditions, when the complexity of the controller is restricted, and the desired performance cannot be achieved, the VRFT proposes the use of the filter $L(q)$ to shape the method's criterion, aiming to approximate the minimum of both cost functions. Also, when the signals are corrupted by noise, an instrumental variable can be used to provide unbiased estimates for ρ. In this case, a second batch of data that is uncorrelated with the first one must be collected, either by running a second experiment or by simulating the process with an identified model [7,13].

1.2. Practical aspects and preprocessing the data

When the input/output data are collected from a real process, which possibly is nonlinear, presents high frequency measurement noises, drifts or offsets, missing data, periodic disturbances, and other undesirable effects corrupting them, they are not likely to be directly used on system identification [30] or on data-driven control methods, such as the VRFT. Hence, to improve the efficiency of *pyvrft* on real applications, a preprocessing step on the data is highly recommended and it can done using basic operations that are also known and employed by the system identification community and described on its classic books [30, 31]:

- **Remove sample means** [30]: if the data presents offsets or drifts, the user can subtract the mean value of the samples of $y(t)$ and $u(t)$, assuming that the system is around an equilibrium point and reducing the impact of this undesirable effect;
- **Prefiltering the data** [30,31]: prefiltering $y(t)$ and $u(t)$ with the same filter will not change the shape of the VRFT criterion. So, the user can prefilter the signals with a low-pass filter (to reduce high frequency noise/disturbances), a high-pass filter (to reduce low frequency noise/disturbances, or even drifts and offsets), or a band-stop filter, like a notch filter (to reduce the effects of a disturbance with a specific frequency on the data).

It is important to reiterate that those are just basic and simple recommendations to preprocess the data in order to reduce the most typical undesirable effects that appears on the data. However, the user can also apply more sophisticated and complex preprocessing procedures, as those discussed with more depth at [30,31].

1.3. How to use the software

The first step to use the software is to install the package, which can be made via pip, running the command: `pip install pyvrft`. The pip command also installs the package's prerequisites if they are not present in the Python environment: `numpy` for numerical computations, `scipy` for signal processing and `matplotlib` to create graphics.

To run the main function of *pyvrft*, namely `vrft.design()`, to tune the parameters of an MIMO LTI controller, it is necessary to collect input and output data from the process (in an open-loop or closed-loop experiment) and organize them as a matrix (*numpy.ndarray*) with dimension (N, n):

$$U = [u(1)\, u(2)\, \cdots\, u(N)]^T, \tag{16}$$

$$Y = [y(1)\, y(2)\, \cdots\, y(N)]^T. \tag{17}$$

Besides, if the signals are corrupted by noise, the user may collect output data from a second experiment to use the instrumental variable technique:

$$\bar{Y} = [\bar{y}(1)\, \bar{y}(2)\, \cdots\, \bar{y}(N)]^T. \tag{18}$$

After collecting and organizing the data, the user must specify the desired closed-loop performance through $T_d(q)$, the controller structure that will be tuned, and the VRFT filter $L(q)$. All these quantities are transfer matrices (or MIMO transfer functions) and, since there is no structure of such type on the traditional Python packages for signal and numerical processing (`numpy` and `scipy`), in *pyvrft* it was decided to organize them as nested Python lists, as described below

- $T_d(q)$: nested Python list with two levels. The first one represents a line of the transfer matrix $T_d(q)$ and the second one represents a column of $T_d(q)$. Each element of the list has a specific variable type, which is the *signal.ltisys.TransferFunctionDiscrete* (the *scipy* variable type for discrete-time transfer functions);
- $C(q, \rho)$: nested Python list with three levels. The first level represents a line of the transfer matrix $C(q, \rho)$ and the second represents a column of $C(q, \rho)$. Yet, the third level represents each element of the m-vector $\bar{C}_{ij}(q)$. Each element of this list is also a *signal.ltisys.TransferFunctionDiscrete* variable;
- $L(q)$: has the same structure of $T_d(q)$, which is a nested Python list of two levels and each element is a *signal.ltisys.TransferFunctionDiscrete*.

The following code exhibits an example for the design of an SISO controller:

```
Td = signal.TransferFunction([0.2], [1, -0.8], dt=1)

L = signal.TransferFunction([0.25], [1, -0.75], dt=1)

C = [
   [signal.TransferFunction([1, 0], [1, -1], dt=1)],
   [signal.TransferFunction([1], [1, -1], dt=1)],
]

p = vrft.design(u, y, y, Td, C, L)
```

To assist the user with the definitions of these quantities on Python and how to use them on the software, there are a few examples of controllers design within the package. Also, a detailed description of each function is available and can be accessed with the `help()` command on the Python environment.

2. Software description

The *pyvrft* is a Python package that implements the unbiased MIMO VRFT method, to tune an MIMO (or SISO) LTI controller, with the possibility to use the least-squares or the instrumental variable to solve the problem. To familiarize the user with the software's architecture and its particularities, they are described on the following subsections.

2.1. Software architecture

The package contains 3 main modules, namely *control.py*, *csvfunc.py* and *invfunc.py*. Each one of these modules has its purpose inside the package and its functions that are described in the sequence.

2.1.1. control.py

This module implements the main algorithms of the package, that are used to compute the controller's parameters. It possesses three functions:

- **design():** this is the main function of the *pyvrft* package, the one that implements the Unbiased MIMO VRFT method. It calls several functions from different modules of the package as *filter()*, *colfilter()*, *mtf2ss()* and *stbinv()*;
- **filter():** implements a multivariable filtering operation, which is essential for the main function of the package. This function can also be used for simulations and validation of the controllers;
- **colfilter():** implements a column filtering operation, which filters every column of a matrix of signals with the same filter.

2.1.2. csvfunc.py

This module is used to import csv data into Python. It possesses just one function, that is

- **datafromcsv():** provide the option to read the data from a csv file. The csv file must have a strict organization of its columns, such as $[y_1, y_2, \ldots, y_n, u_1, u_2, \ldots, u_n]$.

2.1.3. invfunc.py

This module contains the functions that are related to the algorithm that performs a stable inversion of a linear system, which is used on the VRFT to calculate the virtual error, i.e., the signal $\bar{e}(t) = (Td(q)^{-1} - 1)y(t)$. The algorithm was firstly presented on [32], and it was implemented on *pyvrft* through the following functions

- **invredc():** implements one step of the inversion algorithm, which is the system reduction step. This function is called at each iteration of the stable inversion algorithm;
- **stbinv():** implements the inversion of the system, calling *invredc()* to perform the system reduction step. It also calculates the conditions for the stable algorithm to be successful. Otherwise, it produces a warning to the user;
- **mtf2ss():** transforms a MIMO transfer function on a state–space representation. The algorithm that was employed is a simple one, that does not intend to produce a minimal realization. This procedure is important because the VRFT method was implemented with a transfer function notation (as it is usually done in the literature of this theme). Although, the stable inversion algorithm is implemented using a state–space representation, as presented on [32].

2.2. Software functionalities

The main functionality of the *pyvrft* package is to tune an MIMO LTI controller for a dynamic process with the unbiased MIMO VRFT method, using batches of input and output data. This can be achieved with the function `design()`, described above. Besides, as some secondary functionalities, it is possible to use other functions of the package for different kinds of applications. For example, `filter()` can be used to simulate MIMO LTI systems, or equivalently, to filter a vector of signals with an MIMO filter. The `stbinv()` function can be applied to calculate the inverse of an MIMO system, i.e, the signal $u(t) = G(q)^{-1}y(t)$, which can be useful on some circumstances, and `mtf2ss()` can be used to transform a MIMO transfer function representation (done with the nested Python list structure) to a state–space model.

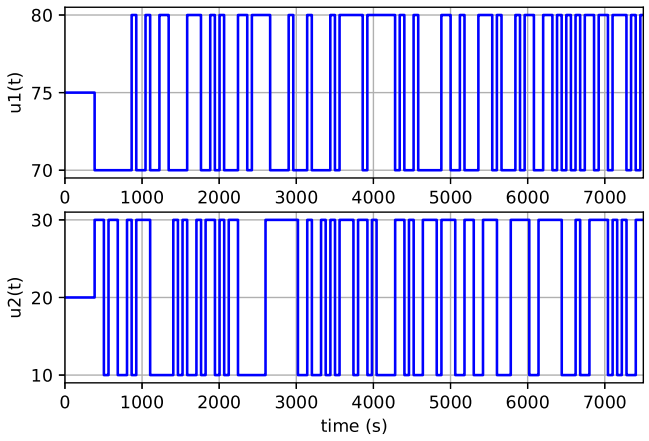

Fig. 1. PRBS signals applied in the pilot plant's input on the open-loop experiment.

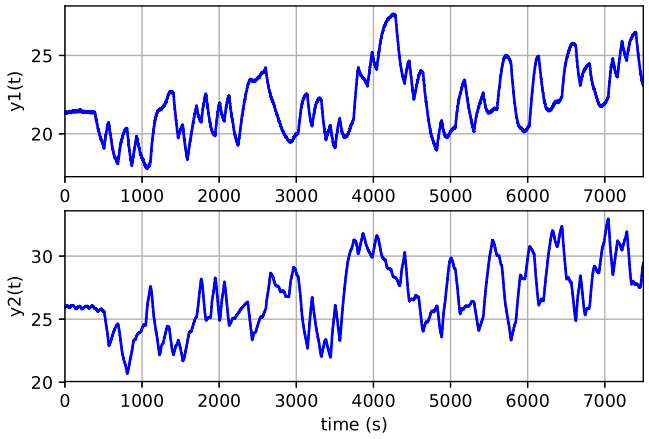

Fig. 2. Output signals collected in the pilot plant on the open-loop experiment.

3. Illustrative examples

To motivate the application of *pyvrft* on real control systems, this section demonstrates an experimental example, where the main objective is to tune a full PI controller for an MIMO level control process. This example, along with two others, are also provided on the package. The process that is considered in this example belongs to a pilot plant, which was fully described at [26] and used to compare different control strategies for multivariable processes. The plant possesses a Two-Input Two-Output (TITO) process, where the outputs $y_1(t)$ and $y_2(t)$ are the levels of water on two distinct tanks and the inputs $u_1(t)$ and $u_2(t)$ are the percentage opening of two globe valves that are used to regulate the water flow of the system.

To collect the data from the process, two distinct pseudorandom binary sequences (PRBS) were applied as input signals on the process in an open-loop experiment. Fig. 1 exhibits the input signals, and Fig. 2 exhibits the output signals obtained. These data were stored on a csv file, that was read with the *pyvrft* `datafromcsv()` function.

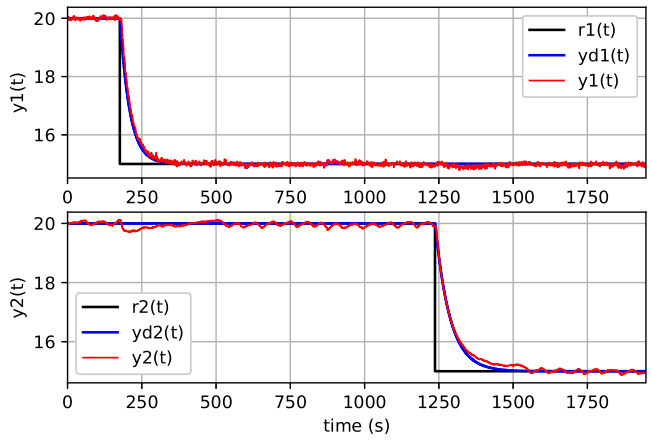

Fig. 3. Comparison between outputs, desired outputs and reference signals on the closed-loop experiment with the controller (21).

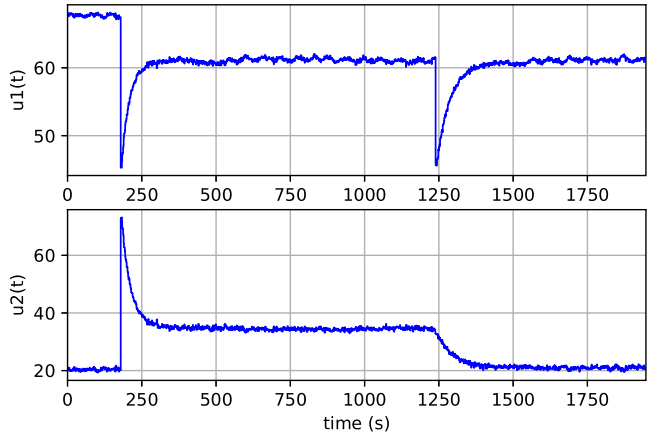

Fig. 4. Control signal on the closed-loop experiment with the controller (21).

The reference model was defined as

$$T_d(q) = \begin{bmatrix} \dfrac{0.03}{q-0.97} & 0 \\ 0 & \dfrac{0.02}{q-0.98} \end{bmatrix}, \tag{19}$$

and the controller structure to be tuned was a full PI:

$$C(q,\rho) = \begin{bmatrix} \dfrac{\rho_{11}^0 q + \rho_{11}^1}{q-1} & \dfrac{\rho_{12}^0 q + \rho_{12}^1}{q-1} \\ \dfrac{\rho_{21}^0 q + \rho_{21}^1}{q-1} & \dfrac{\rho_{22}^0 q + \rho_{22}^1}{q-1} \end{bmatrix}. \tag{20}$$

The filter of the VRFT was chosen as $L(q) = T_d(q)(I - T_d(q))$, and the *pyvrft* was used to estimate the controller's parameters. The controller designed with the software was

$$C(q,\rho) = \begin{bmatrix} \dfrac{4.39(q-0.992)}{q-1} & 3.12 \\ \dfrac{-10.49(q-0.993)}{q-1} & \dfrac{0.26(q-0.828)}{q-1} \end{bmatrix}, \tag{21}$$

which is practically the same obtained on [13,26]. This controller was implemented on the pilot plant, and the achieved closed-loop performance is exhibited on Figs. 3 and 4. Fig. 3 shows a comparison between the output signals, the reference signals, and the desired outputs on the closed-loop experiment and Fig. 4 exhibits the control signals in this experiment.

Notice that the process' outputs, obtained on the closed-loop experiment, were very close to the desired ones. Also, the control system almost achieved a perfect decoupling between the loops, where one reference/output signal does not disturb the other, which is a significant result for a multivariable control system.

4. Impact

The *pyvrft* package intends to impact the industrial users, researchers, and developers of data-driven control methods, providing for these groups, a free and open-source software that allows them to use the VRFT method on practical situations and to develop and test further theoretical contributions. As far as the authors know, this is the first implementation of the unbiased MIMO VRFT that is free and open for the community, as well as peer-reviewed. From now on, new users of VRFT do not need to implement the algorithms from zero and by themselves. Moreover, the package was developed in Python, which is the largest growing programming language used by scientific communities, and that provides free and open-source algorithms that can be verified, corrected, and expanded by other users.

5. Conclusions

The *pyvrft* is a Python package for designing feedback controllers using the Virtual Reference Feedback Tuning method. The package computes unbiased estimates for MIMO and SISO controllers. Both classical least-squares and instrumental variables are available to be used. The package simplifies the task of calculating feedback controllers since it implements a data-driven solution that does not depend on a mathematical process' model. The architecture of the package makes it simple to use, and it is a timesaving tool for control designers. The *pyvrft* package is open-source, and it is distributed under an MIT license.

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.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Diego Eckhard is partially supported by CNPq/Brasil.

References

[1] Ziegler JG, Nichols NB. Optimum settings for automatic controllers. Trans ASME 1942;64(11):759–68.

[2] Bazanella AS, Campestrini L, Eckhard D. Data-driven controller design: the H_2 approach. Amsterdam: Springer Science & Business Media; 2012, http://dx.doi.org/10.1007/978-94-007-2300-9.

[3] Hou Z-S, Wang Z. From model-based control to data-driven control: survey, classification and perspective. Inform Sci 2013;235:3–35. http://dx.doi.org/10.1016/j.ins.2012.07.014.

[4] Hjalmarsson H, Gevers M, Gunnarsson S, Lequin O. Iterative feedback tuning: theory and applications. IEEE Control Syst 1998;18(4):26–41. http://dx.doi.org/10.1109/37.710876.

[5] Kammer LC, Bitmead RR, Bartlett PL. Direct iterative tuning via spectral analysis. Automatica 2000;36(9):1301–7. http://dx.doi.org/10.1016/S0005-1098(00)00040-6.

[6] Karimi A, Mišković L, Bonvin D. Iterative correlation-based controller tuning. Internat J Adapt Control Signal Process 2004;18(8):645–64. http://dx.doi.org/10.1016/S0967-0661(02)00191-0.

[7] Campi M, Lecchini A, Savaresi S. Virtual reference feedback tuning: a direct method for the design of feedback controllers. Automatica 2002;38(8):1337–46. http://dx.doi.org/10.1016/S0005-1098(02)00032-8.

[8] Karimi A, Van Heusden K, Bonvin D. Non-iterative data-driven controller tuning using the correlation approach. In: Proceedings of the European control conference. Kos, 2007, New York: IEEE; 2007, p. 5189–95. http://dx.doi.org/10.23919/ECC.2007.7068802.
[9] Campestrini L, Eckhard D, Bazanella AS, Gevers M. Data-driven model reference control design by prediction error identification. J Franklin Inst B 2017;354(6):2628–47. http://dx.doi.org/10.1016/j.jfranklin.2016.08.006.
[10] Lecchini A, Campi M, Savaresi S. Virtual reference feedback tuning for two degree of freedom controllers. Internat J Adapt Control Signal Process 2002;16(5):355–71. http://dx.doi.org/10.1002/acs.711.
[11] Nakamoto M. An application of the virtual reference feedback tuning for an MIMO process. In: SICE 2004 annual conference, vol. 3; 2004, p. 2208–13.
[12] Formentin S, Savaresi SM. Noniterative data-driven design of multivariable controllers. In: 2011 50th IEEE conference on decision and control and european control conference; 2011. p. 5106–11. http://dx.doi.org/10.1109/CDC.2011.6160388.
[13] Campestrini L, Eckhard D, Chía LA, Boeira E. Unbiased MIMO VRFT with application to process control. J Process Control 2016;39:35–49. http://dx.doi.org/10.1016/j.jprocont.2015.12.010.
[14] Campi MC, Savaresi SM. Direct nonlinear control design: the virtual reference feedback tuning (VRFT) approach. IEEE Trans Automat Control 2006;51(1):14–27. http://dx.doi.org/10.1109/TAC.2005.861689.
[15] Kansha Y, Hashimoto Y, Chiu M-S. New results on VRFT design of PID controller. Chem Eng Res Des 2008;86(8):925–31. http://dx.doi.org/10.1016/j.cherd.2008.02.018.
[16] Campestrini L, Eckhard D, Gevers M, Bazanella A. Virtual reference feedback tuning for non-minimum phase plants. Automatica 2011;47(8):1778–84. http://dx.doi.org/10.1016/j.automatica.2011.04.002.
[17] Van Heusden K, Karimi A, Bonvin D. Data-driven model reference control with asymptotically guaranteed stability. Internat J Adapt Control Signal Process 2011;25(4):331–51. http://dx.doi.org/10.1002/acs.1212.
[18] Formentin S, Karimi A. Enhancing statistical performance of data-driven controller tuning via L2-regularization. Automatica 2014;50(5):1514–20. http://dx.doi.org/10.1016/j.automatica.2014.04.001.
[19] Eckhard D, Campestrini L, Boeira EC. Virtual disturbance feedback tuning. IFAC J Syst Control 2018;3:23–9. http://dx.doi.org/10.1016/j.ifacsc.2018.01.003.
[20] Campi MC, Lecchini A, Savaresi SM. An application of the virtual reference feedback tuning method to a benchmark problem. Eur J Control 2003;9(1):66–76. http://dx.doi.org/10.3166/ejc.9.66-76.
[21] Previdi F, Schauer T, Savaresi SM, Hunt KJ. Data-driven control design for neuroprotheses: a virtual reference feedback tuning (VRFT) approach. IEEE Trans Control Syst Technol 2004;12(1):176–82. http://dx.doi.org/10.1109/TCST.2003.821967.
[22] Rojas JD, Flores-Alsina X, Jeppsson U, Vilanova R. Application of multivariate virtual reference feedback tuning for wastewater treatment plant control. Control Eng Pract 2012;20(5):499–510. http://dx.doi.org/10.1016/j.conengprac.2012.01.004.
[23] Formentin S, De Filippi P, Corno M, Tanelli M, Savaresi SM. Data-driven design of braking control systems. IEEE Trans Control Syst Technol 2013;21(1):186–93. http://dx.doi.org/10.1109/TCST.2011.2171965.
[24] Scheid Filho R, Eckhard D, Gonçalves da Silva GR, Campestrini L. Application of virtual reference feedback tuning to a non-minimum phase pilot plant. In: IEEE conference on control applications, buenos aires; 2016. p. 1318–23. http://dx.doi.org/10.1109/CCA.2016.7587989.
[25] Invernizzi D, Panizza P, Riccardi F, Formentin S, Lovera M. Data-driven attitude control law of a variable-pitch quadrotor: a comparison study. In: IFAC symposium on automatic control in aerospace - ACA 2016. IFAC-PapersOnLine 2016;49(17):236–41. http://dx.doi.org/10.1016/j.ifacol.2016.09.041.
[26] Boeira E, Bordignon V, Eckhard D, Campestrini L. Comparing MIMO process control methods on a pilot plant. J Control Autom Electr Syst 2018;29(4):411–25. http://dx.doi.org/10.1007/s40313-018-0387-6.
[27] Radac M-B, Precup R-E, Roman R-C. Data-driven model reference control of mimo vertical tank systems with model-free vrft and q-learning. ISA Trans 2018;73:227–38. http://dx.doi.org/10.1016/j.isatra.2018.01.014.
[28] Zanchettin AM, Formentin S, Loddo R, Paparella R. Direct data-driven control of cavity tuners in particle accelerators. In: 18th IFAC symposium on system identification - SYSID 2018. IFAC-PapersOnLine 2018;51(15):138–43. http://dx.doi.org/10.1016/j.ifacol.2018.09.104.
[29] Carè A, Torricelli F, Campi MC, Savaresi SM. A toolbox for virtual reference feedback tuning (VRFT). In: 2019 18th European control conference (ECC); 2019. p. 4252–7. http://dx.doi.org/10.23919/ECC.2019.8795811.
[30] Ljung L. System identification: theory for the user. second ed.. Englewoods Cliff: Prentice Hall; 1999.
[31] Söderström T, Stoica P. System identification. Englewoods Cliff: Prentice Hall; 1989.
[32] Moylan P. Stable inversion of linear systems. IEEE Trans Automat Control 1977;22(1):74–8. http://dx.doi.org/10.1109/TAC.1977.1101430.