

A Toolbox for Virtual Reference Feedback Tuning (VRFT)

Algo Carè, Fabrizio Torricelli, Marco C. Campi, Sergio M. Savaresi

Abstract—The Virtual Reference Feedback Tuning (VRFT) is a direct data-driven method for controller design. In this paper, we present a MATLAB toolbox that implements the most important procedures of VRFT. The Toolbox is freely available online. The main VRFT procedures are described in this paper from a user-oriented point of view, and are illustrated on some numerical examples with a linear and a nonlinear plant. The tuning of a proportional-integral-derivative (PID) controller will serve as a running example throughout the paper.

Index Terms—Computer aided control design, Control courses and labs, Adaptive control.

I. INTRODUCTION

Virtual Reference Feedback Tuning (VRFT) is a direct data-driven method for controller design. It prescribes to select a controller from a given class based on a batch of data that are collected from the plant. The controller-selection procedure is *direct* in the sense that data are not preliminarily used to identify a model of the plant; instead, the controller parameters are directly optimized as decision variables in a data-dependent minimization problem. In this way, VRFT reduces the design to a data-driven optimization problem.

VRFT was introduced in [1], building on a general idea first presented in [2]. It was firstly introduced for the design of one-degree-of-freedom controllers to shape the reference-to-output transfer function of a closed-loop linear time invariant system, and then extended in several directions, including the tuning of two-degree-of-freedom controllers [3], the study of procedures for nonlinear control [4], multivariable systems [5], unstable or non-minimum phase systems, [6], [7].

Other methods developed in the last two decades are also direct and data-driven. We mention here the Iterative Feedback Tuning (IFT) method [8], [9], [10], the data-driven loop-shaping [11], and the Correlation-based Tuning (CbT) method [12], [13]. See also [14], [15], [16], [17] and the references therein for a general presentation. We refer the reader to the bibliography for comparisons and discussions on the merits of one over another, while we refer in particular to [18] for a study that compares direct data-driven methods with model-based design approaches.

This work was supported by the Ministero dell’Istruzione, dell’Università e della Ricerca (MIUR).

A.C (algo.care@unibs.it), F.T. and M.C.C. are with Dipartimento di Ingegneria dell’Informazione, Università di Brescia, via Branze 38, 25123 Brescia, Italia. S.M.S. is with Dipartimento di Elettronica, Informazione e Biongegneria, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italia.

A. Aim and structure of this paper

This paper presents a Toolbox for the implementation of VRFT that has been realized and made available by the paper’s authors at the URL

<http://marco-campi.unibs.it/VRFTwebsite/index.html>

The Toolbox implements a selection of VRFT procedures that are also revisited here from a user-oriented point of view.

In Section II, the idea behind the VRFT method is recalled. To exemplify the method, we will in particular obtain some simple “one-shot” data-driven tuning rules for a classic proportional-integral-derivative (PID) controller design. Section III presents the main functions of the VRFT Toolbox for classic controller design problems. Although the linear set-up takes the largest part of the presentation, a more advanced control design tuning scheme for nonlinear plants is also presented at the end of Section III. The usage of the VRFT Toolbox is illustrated on two numerical examples (with a linear and a nonlinear plant) in Section IV.

II. THE VRFT IDEA

A. Basic facts in a noise-free set-up

Consider the closed-loop control scheme of a linear, time-invariant, single-input single-output discrete-time dynamical system with transfer function $P(z)$ in Figure 1. We will first focus on the noise-free case, i.e. $d_t = 0$.

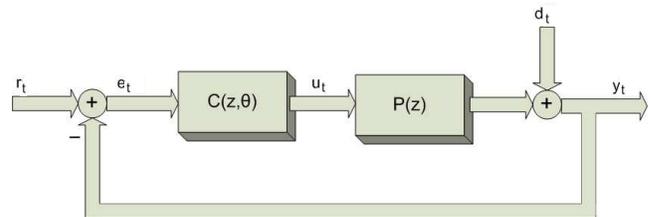


Fig. 1. Control loop.

Given a reference model $M_r(z)$ that describes the desired transfer function from r_t to y_t and a family of linear controllers $C = \{C(z, \theta) : \theta \in \mathbb{R}^n\}$, parametrized by a real vector θ , the control design problem is cast as the following optimization problem: choose the parameter θ that minimizes the 2-norm model-reference criterion

$$J_{MR}(\theta) = \left\| \left(\frac{P(z)C(z, \theta)}{1 + P(z)C(z, \theta)} - M_r(z) \right) W(z) \right\|_2^2, \quad (1)$$

where $W(z)$ is a user-chosen weighting function that modulates the importance of matching the reference-to-output transfer function with $M_r(z)$ at different frequencies. By

definition of 2-norm, (1) can be written more explicitly as $J_{\text{MR}}(\theta) =$

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{P(e^{j\omega})C(e^{j\omega}, \theta)}{1 + P(e^{j\omega})C(e^{j\omega}, \theta)} - M_r(e^{j\omega}) \right|^2 |W(e^{j\omega})|^2 d\omega. \quad (2)$$

Since the plant $P(z)$ appears in the cost function $J_{\text{MR}}(\theta)$, knowledge of $P(z)$ seems to be necessary to minimize $J_{\text{MR}}(\theta)$. The VRFT method offers a way to design the controller when $P(z)$ is unknown by using a collection of input/output data from the plant, $\mathcal{D}_N = \{(u_t, y_t), t = 1, \dots, N\}$. \mathcal{D}_N can be collected either in open loop or closed loop. The core of VRFT consists in the introduction of a cost function $J_{\text{VRFT}}^N(\theta)$ (in substitution of $J_{\text{MR}}(\theta)$) that depends on \mathcal{D}_N but not on $P(z)$.

The idea behind VRFT

The intuitive idea for building $J_{\text{VRFT}}^N(\theta)$ is as follows. Given the measured y_t , one computes the so-called *virtual reference* signal \bar{r}_t such that $y_t = M_r(z)\bar{r}_t$. This is the reference that would produce an output equal to the measured y_t if a perfect controller $C^*(z)$ (such that $\frac{P(z)C^*(z)}{1+P(z)C^*(z)} = M_r(z)$) were placed in the control loop. Considering that the relation $y_t = P(z)u_t$ holds true, it must also be true that $C^*(z)(\bar{r}_t - y_t) = u_t$, which provides us with a description of the input-output behavior of the perfect controller $C^*(z)$. The idea is then to choose θ in such a way that this behavior is mimicked at best by the chosen controller, that is, in such a way that $C(z, \theta)(\bar{r}_t - y_t)$ approximates u_t as much as possible. Using relation $M_r(z)\bar{r}_t = y_t$, we thus obtain the data-dependent cost function $J_{\text{VRFT}}^N(\theta) = \frac{1}{N} \sum_{t=1}^N (C(z, \theta)(M_r(z)^{-1} - 1)y_t - u_t)^2$, which depends only on the known signals u_t and y_t and the known transfer function $M_r(z)$. This idea has proven effective, and has been backed by a solid theoretical analysis and extended in several directions in the literature indicated in the introduction.

Theoretical results

In [1], it was shown that if \mathcal{C} includes the perfect controller $C^*(z) = C(z, \theta^*)$ so that $J_{\text{MR}}(\theta^*) = 0$, then θ^* minimizes also $\lim_{N \rightarrow \infty} J_{\text{VRFT}}^N(\theta)$. More in general, when $C^*(z) \notin \mathcal{C}$, the minimizer of $J_{\text{VRFT}}^N(\theta)$ can be proved to be a ‘‘good’’ approximation of $J_{\text{MR}}(\theta)$ if the measured u_t and y_t signals are suitably prefiltered leading to the following cost function:

$$J_{\text{VRFT}}^N(\theta) = \frac{1}{N} \sum_{i=1}^N (C(z, \theta)(M_r(z)^{-1} - 1)L(z)y_t - L(z)u_t)^2. \quad (3)$$

Indeed, in [1] it was proved that the choice

$$L(z) = \frac{W(z)(1 - M_r(z))M_r(z)}{\Phi_u(z)^{1/2}}, \quad (4)$$

where $\Phi_u(z)^{1/2}$ is a spectral factorization of (an estimate of) the spectral density $\Phi_u(z)$ of the time sequence u_t , i.e. $|\Phi_u(z)^{1/2}|^2 = \Phi_u(z)$, leads to a VRFT cost function $J_{\text{VRFT}}^N(\theta)$ which approximates a second-order expansion of $J_{\text{MR}}(\theta)$ around the perfect controller, see [1].

B. VRFT in practice

In practical usage, the class of controllers is often linearly parameterized, that is, a *basis* of controllers $C_1(z), \dots, C_n(z)$ is chosen by the user, and the generic controller in \mathcal{C} is defined as $C(z, \theta) = \theta_1 C_1(z) + \dots + \theta_n C_n(z)$, $\theta \in \mathbb{R}^n$. In this way, the minimization of $J_{\text{VRFT}}^N(\theta)$ becomes a standard least-squares problem that can be solved in closed form. This fact is shown in detail below in the notable case of proportional-integral-derivative (PID) controller tuning, where $n = 3$. The generalization to any n is immediate.

PID controllers

A continuous-time PID controller, [19], has transfer function $C_{\text{PID}}(s, (K_p, K_i, K_d)) = K_p + K_i \frac{1}{s} + K_d \frac{s}{1 + T_d s}$, where K_p, K_i, K_d are the tunable parameters and T_d is a fixed fast time constant (taming the derivative action). In order to adhere to the discrete-time framework of this paper¹, we use the Tustin transform with sampling time $T_s = T_d$, which yields the discrete-time PID

$$C_{\text{PID}}(z, (K_p, K_i, K_d)) = K_p C_1(z) + K_i C_2(z) + K_d C_3(z), \quad (5)$$

with $C_1(z) = 1$, $C_2(z) = \frac{T_s}{2} \frac{1+z^{-1}}{1-z^{-1}}$, and $C_3(z) = \frac{2}{T_s} \frac{1-z^{-1}}{3-z^{-1}}$. Equation (5) gives a class of controllers that are linearly parametrized by $\theta = (K_p, K_i, K_d)$. The VRFT procedure in this case becomes:

- Given the input/output data u_t, y_t , the target $M_r(z)$ and the weighting function $W(z)$, compute

– the prefiltering²

$$L(z) = (1 - M_r(z))M_r(z) \frac{W(z)}{\Phi_u(z)^{1/2}};$$

- the three sequences $x_{i,t}$, $i = 1, 2, 3$ defined as follows:

$$x_{i,t} = C_i(z)(M_r(z)^{-1} - 1)L(z)y_t;$$

- the prefiltered sequence $u_t^L = L(z)u_t$.

- Denote by \mathbf{x}_i (respectively, \mathbf{u}) the column vector that contains the sequence $x_{i,t}$ (respectively, u_t^L).
- It is immediate to show that $J_{\text{VRFT}}^N(\theta) = 1/N([\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]\theta - \mathbf{u})^T([\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]\theta - \mathbf{u})$ (so that the minimizer θ^* depends only on the scalar products $\mathbf{x}_i^T \mathbf{x}_j = \sum_{t=1}^N x_{i,t} x_{j,t}$ and $\mathbf{x}_i^T \mathbf{u} = \sum_{t=1}^N x_{i,t} u_t^L$), which yields the following rule:

PID tuning rule:

$$\begin{bmatrix} K_p^* \\ K_i^* \\ K_d^* \end{bmatrix} = \left(\begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} \mathbf{u}. \quad (6)$$

¹A study of a continuous-time version of VRFT can be found in [20].

² $\Phi_u(z)^{1/2}$ can be either estimated from u_t (which is done in the Toolbox when `preFilt`=[], see Section III) or $W(z)$ can be chosen as $F(z)\Phi_u(z)^{1/2}$, e.g. as a low pass filter $F(z)$ modulated by the unknown $\Phi_u(z)^{1/2}$, so that $\frac{W(z)}{\Phi_u(z)^{1/2}}$ simplifies to $F(z)$ in which case $\Phi_u(z)^{1/2}$ need not be estimated.

Dealing with noisy data

When the plant output y_t is affected by an additive noise d_t , such a noise generates bias in the controller parameters, and $J_{\text{VRFT}}^N(\theta)$ may no longer approximate $J_{\text{MR}}(\theta)$. To counteract this biasing effect, an instrumental variable method can be used. In the Toolbox, two methods are implemented for generating the instrumental variables when the plant is operated in open loop³:

- **Method I** (repeated experiment method): a second experiment on the plant is performed using the same input sequence u_t as in the first experiment; the corresponding output y'_t is recorded; the instrumental variables $\xi_{1,t}, \dots, \xi_{n,t}$ are obtained as $\xi_{i,t} = C_i(z)(M_r(z)^{-1} - 1)L(z)y'_t, i = 1, \dots, n$.
- **Method II** (system identification method): a model $\hat{P}(z)$ of the plant $P(z)$ is obtained from the available u_t, y_t ; an output is generated as $\tilde{y}_t = \hat{P}(z)u_t$, and the sequences $\xi_{i,t} = C_i(z)(M_r(z)^{-1} - 1)L(z)\tilde{y}_t, i = 1, \dots, n$, are obtained; these sequences are approximately uncorrelated with the additive noise d_t and therefore can be used as instrumental variables. Although identifying $P(z)$ makes the method less direct, it is worth remarking that $\hat{P}(z)$ is just a tool to generate the instrumental variables, and can be a very rough approximation of $P(z)$.

The instrumental variables are then used in place of the classic least-squares regressors: to exemplify this, we note that in the PID case the method yields the following rule:

PID tuning rule with Instrumental Variables:

$$\begin{bmatrix} K_p^* \\ K_i^* \\ K_d^* \end{bmatrix} = \left(\begin{bmatrix} \xi_1^T \\ \xi_2^T \\ \xi_3^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} \right)^{-1} \begin{bmatrix} \xi_1^T \\ \xi_2^T \\ \xi_3^T \end{bmatrix} \mathbf{u}. \quad (7)$$

C. Generalizations

The ideas illustrated above for shaping the reference-to-output transfer function carry over to many other control design problems and the reader is referred to the papers cited in the introduction for a presentation. In the following section, we focus on the VRFT Toolbox and show how it can be used in various set-ups.

III. THE VRFT TOOLBOX

The core of the VRFT Toolbox is a set of six MATLAB functions. An online help can be read by typing `help VRFT` in the MATLAB console. A Graphical User Interface (GUI) is also provided (see Figure 2 for a screenshot). The GUI can be run by typing `VRFT_GUI` and comes with an online help and a “demo mode”, which allows the user to test the main VRFT functions. The package requires MATLAB version 6, or above, the Control System Toolbox and the System Identification Toolbox.

³When the plant is operated in closed loop, the instrumental variables method has to be applied as discussed in [1], but it is not yet implemented in the present version of the Toolbox.

A. Reference-to-output

The VRFT algorithm illustrated in Section II-A is implemented by the function

$$C = \text{VRFT1_ry}(u, y, Mr, B, W, k, \text{preFilt})$$

In the function call, u and y are the column-vectors that contain the measured input and output, which must have the same length; Mr is the desired transfer function, e.g., a low pass filter $M_r(z) = \frac{0.025+0.025z^{-1}}{1-0.95z^{-1}}$ is obtained by (T_s is the sampling time by which $M_r(z)$ is obtained as a discrete-time counterpart of a continuous-time reference model; if the whole study is conducted in discrete-time, set $T_s=1$) `Mr=tf(0.025*[1, 1], [1, -0.95], Ts, 'variable', 'z^-1')`; B is the vector that contains the basis of controllers $C_1(z), \dots, C_n(z)$, e.g., in order to tune a PID controller as in (5) one can define `B=[tf([1], [1], Ts, 'variable', 'z^-1'); tf(Ts*[1 1], 2*[1 -1], Ts, 'variable', 'z^-1'); tf([2 -2], Ts*[3 -1], Ts, 'variable', 'z^-1')]`. W is the weighting transfer function $W(z)$; the variable k is set to `k=[]` in the basic set-up, another usage will be explained later; the variable `preFilt` contains the user-specified function $L(z)$, and, when it is set to `[]`, the filter in (4) is used. The function output C is the designed controller. The values of the optimal θ^* can be extracted by calling the same function with an additional output `theta`, i.e. `[C, theta] = VRFT1_ry(u, y, Mr, B, W, k, preFilt)`. For instance, in the case of PID tuning K_p^*, K_i^*, K_d^* are obtained respectively as `theta(1), theta(2), theta(3)`.

We also remark that the PID tuning rules (6), (7) are preset in the GUI, where the user can select to tune the PID controller (5) by clicking on the “use PID” option in the “Controller structure” drop-down menu (see Figure 2).

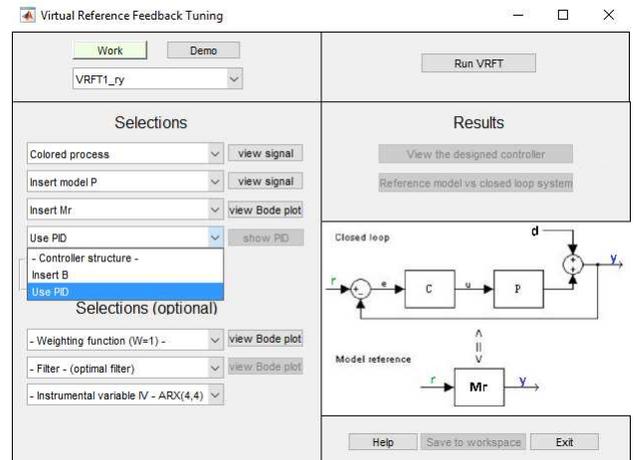


Fig. 2. The Graphical User Interface with the preset “use PID” option.

In the presence of noisy output, the user can feed the algorithm with a second output y'_t by adding a second column to the vector y , which is used to construct the instrumental variables according to Method I in Section II-B.

Another option is generating artificial instrumental variables according to Method II in Section II-B. In this case, the plant is estimated using an ARX(k,k) model. This method is used when the user specifies a numeric value for the model order k , e.g. $k=4$, and calls the function

$$C = \text{VRFT1_ry}(u, y, Mr, B, W, k, \text{preFilt})$$

For a numerical example, see Section IV.

B. Output-sensitivity

As shown in [21], VRFT can be used to shape the output-sensitivity function so as to make it resemble a user-chosen $M_d(z)$, i.e., to minimize

$$J_{\text{MR}}(\theta) = \left\| \left(\frac{1}{1 + P(z)C(z, \theta)} - M_d(z) \right) W_d(z) \right\|_2^2. \quad (8)$$

As in the reference-to-output case, the starting point is a set of input-output data u_t, y_t that were collected in open or closed loop. The syntax is the same as in the reference-to-output case:

$$C = \text{VRFT1_dy}(u, y, Md, B, W, k, \text{preFilt}).$$

C. Dealing with input constraints

A VRFT approach is possible also when the cost functions (1) and (8) are complemented with a term that penalizes the deviation from a desired r_t -to- u_t or d_t -to- u_t transfer function, see [21]. This option is implemented in the Toolbox by the functions `VRFT1_ry_ru` and `VRFT1_dy_du`.

D. Two-degree-of-freedom controller

The reference-to-output transfer function and the output-sensitivity transfer function can be simultaneously shaped by resorting to a two-degree-of-freedom (2 d.o.f.) controller as in Figure 3, where one wants to design the controllers

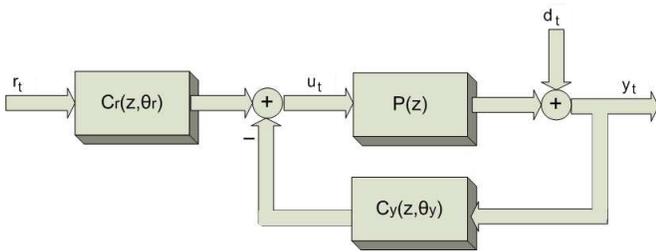


Fig. 3. Controller with two degrees of freedom.

$C_r(z, \theta_r)$ and $C_y(z, \theta_y)$ so as to minimize the cost function

$$J_{\text{MR}}(\theta) = \left\| \left(\frac{P(z)C_r(z, \theta_r)}{1 + P(z)C_y(z, \theta_y)} - M_r(z) \right) W_r(z) \right\|_2^2 + \left\| \left(\frac{1}{1 + P(z)C_y(z, \theta_y)} - M_d(z) \right) W_d(z) \right\|_2^2. \quad (9)$$

In [3], VRFT procedures were designed in this context, for which it is required to avail of a couple u_t, y_t , and possibly a further output realization y'_t to be used to obtain the instrumental variables. The command to synthesize C_r and C_y is

$$[Cr, Cy] = \text{VRFT2_ry_dy}(u, y, Mr, Md, Br, By, \dots, Wr, Wd, \text{fIA}, k, \text{preFilt})$$

where the only input argument that requires some more explanation is `fIA`. `fIA` has no effect when it is empty (`fIA=[]`). If instead `fIA='y'` and both `Br` and `By` contain integral actions, then a constraint is introduced in the VRFT optimization procedure so that the static gain of the transfer function from r_t to y_t is set to 1. In the implementation of the controller, the integrator will be placed in the loop.

E. Nonlinear VRFT

The VRFT method and its theory can be generalized to nonlinear control design, see [4].

Nonlinear plant - nonlinear controller design

The Toolbox includes the function `NL_VRFT1_ry` that computes a nonlinear controller given by a linear combination (according to a vector of parameters θ) of nonlinear ARX blocks. This is not further described here and the reader is referred to the related `help` for more information.

Nonlinear plant - linear controller design

Alternatively, when the plant is nicely nonlinear, the same functions, e.g. `VRFT1_ry`, that we have presented in previous sections for synthesizing a linear controller can be used. In the nonlinear case, one is interested in the closed loop behaviour in response to a given signal r_t^o . The usual VRFT procedure can be applied to u_t, y_t as usual, by selecting $M_r(z)$ so that a desired output is obtained when $M_r(z)$ is fed with r_t^o . However, the nonlinear effects are likely to cause a deterioration in the performance as compared to the linear case. Better performances can often be achieved by iterating a few times the standard VRFT procedure as follows.

Iterative VRFT for nonlinear plants

- 1 Given u_t, y_t and $M_r(z)$, compute the controller $C(z, \theta^*)$ that minimizes (3).
- 2 Run a new experiment on the control loop with the controller $C(z, \theta^*)$ and with r_t^o as a reference signal. Measure the resulting control input signal u_t^* and the plant output signal y_t^* .
- 3 Compare y_t^* with the desired output $y_t^o = M_r(z)r_t^o$. If the performance is unsatisfactory, then
 - rename the last measured signals u_t^*, y_t^* to u_t, y_t ,
 - go back to step 1.

It might be worth remarking that the selection of $L(z)$ in this context may involve subtleties as discussed in [4].

IV. NUMERICAL ILLUSTRATION OF THE TOOLBOX

We provide two examples where the Toolbox is used to shape the r_t -to- y_t transfer function of a control system as in Figure 1. For continuous-time signals, we will use the notation $x(t)$, while x_t will denote the sampled versions.

A. PID tuning example

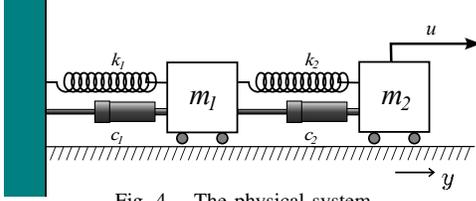


Fig. 4. The physical system

1) *VRFT workflow on a simple example*: Referring to Figure 4, our aim is to design a PID controller that applies a force u to the second mass in order to regulate y , the deviation of the second mass position from its equilibrium point. The desired closed-loop behavior is given by the reference model $M_r(s) = \frac{1}{3s+1}$, which ensures unit gain and a settling time of 15s. A low-pass filter $W(s)$ can be used to penalize the mismatch between $M_r(s)$ and the actual control system at frequencies below 2 rad/s: we choose $W(s) = \frac{1}{0.3s+1}$.

A set of input-output data was recorded in open loop for 500 seconds, with sampling time 1/10s ($T_s = 0.1$). Measurements of $y(t)$ were affected by noise. Recorded data are shown in Figure 5.

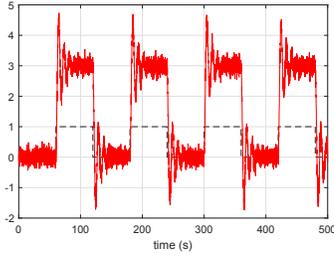


Fig. 5. The available sequences of input-output data. The input $u(t)$ is the unit square wave with period 120s (dashed line). The (noisy) measurements of $y(t)$ are represented by the solid line.

In order to use the VRFT Toolbox in MATLAB, we have to load the sampled data into two column-vectors u and y , and to transform $M_r(s)$ and $W(s)$ into discrete-time transfer functions by the command `c2d`. For PID tuning, we set `B=tf([1],[1],Ts,'variable','z^-1');` `tf(Ts*[1 1],2*[1 -1],Ts,'variable','z^-1');` `tf([2 -2],Ts*[3 -1],Ts,'variable','z^-1');`, as explained in Section II-B. Finally, since data are noisy and we have only one recorded experiment, we set `k=4` to generate the instrumental variables according to Method II (see again Section II-B).

By calling `[C,theta]=VRFT1_ry(u,y,Mr,B,W,4,[])` we obtain the PID parameters `theta(1)`, `theta(2)`, `theta(3)`, which are respectively $K_p^* = 0.0683$, $K_i^* = 0.1105$, $K_d^* = 0.2307$.

2) *Behind the curtains*: For the sake of reproducibility, data in Figure 5 were generated from the following plant

$$P(s) = \frac{m_1 s^2 + (c_1 + c_2)s + (k_1 + k_2)}{(m_1 s^2 + (c_1 + c_2)s + k_1 + k_2)(m_2 s^2 + c_2 s + k_2) - (k_2 + c_2 s)^2}$$

(with $m_1 = 1$, $m_2 = 0.5$, $c_1 = 0.2$, $c_2 = 0.5$, $k_1 = 1$ and $k_2 = 0.5$) which is an ideal descriptor of the u -to- y relationship of the physical system of Figure 4. Moreover, a Gaussian white noise with zero mean and 0.025 variance was added to the noiseless output $y(t) = P(s)u(t)$.

Since $P(s)$ is known in this example (it is not in real applications), we can compare in Figure 6 $M_r(s)$ with the closed-loop system with the designed PID.

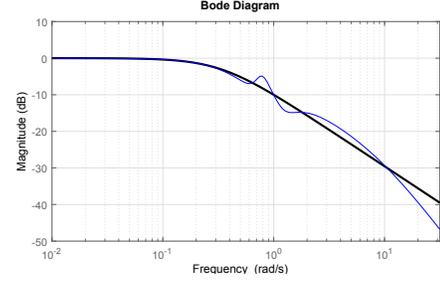


Fig. 6. The Bode magnitude plot of $M_r(s)$ (black thick line); the magnitude plot of the control loop obtained by VRFT (thin blue line).

Finally, we also show in Figure 7 the (noiseless) response of the PID control loop to two cycles of a square wave input signal (the same signal as in Figure 5), compared with the desired one as given by $M_r(z)$.

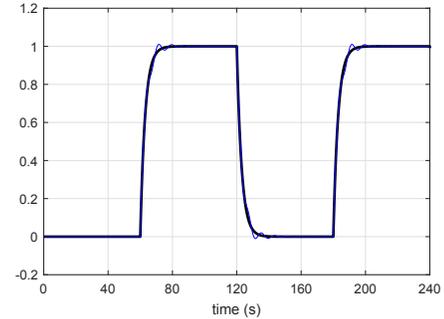


Fig. 7. Black thick line: desired response to the input square wave of Figure 5 (note that the scale of the vertical axis has changed). Blue thin line: the designed control loop response.

B. An example with a nonlinear plant

Consider the nonlinear (Hammerstein) system:

$$\frac{dy(t)}{dt} = 200 \cdot y(t) + 2000 \cdot (u(t)^3 + \frac{2}{10} u(t)) + \frac{10}{19} \cdot \frac{d(u(t)^3 + \frac{2}{10} u(t))}{dt} \quad (10)$$

We want to design a simple PI controller to shape the $r(t)$ -to- $y(t)$ step response like one of a first-order linear system $M_r(s) = \frac{1}{0.005s+1}$, with a settling time of 0.025s.

To this purpose, a set of input $u(t)$ and output $y(t)$ data is recorded for 0.1s, with sampling time 1/1900s, see Figure 8.

