FOMCON: a MATLAB Toolbox for Fractional-order System Identification and Control

Aleksei Tepljakov, Eduard Petlenkov, and Juri Belikov

***Abstract*—FOMCON is a new fractional-order modeling and control toolbox for MATLAB. It offers a set of tools for researchers in the field of fractional-order control. In this paper, we present an overview of the toolbox, motivation for its development and relation to other toolboxes devoted to fractional calculus. We discuss all of the major modules of the FOMCON toolbox as well as relevant mathematical concepts. Three modules are presented. The main module is used for fractional-order system analysis. The identification module allows identifying a fractional system from either time or frequency domain data. The control module focuses on fractional-order PID controller design, tuning and optimization, but also has basic support for design of fractional lead-lag compensators and TID controllers. Finally, a Simulink blockset is presented. It allows more sophisticated modeling tasks to be carried out.**

***Index Terms*—fractional calculus, matlab toolbox, automatic control, pid controller, identification, control system design**

I. Introduction

In recent years fractional-order calculus has gained a lot of attention, especially in the field of system theory and control systems design due to more accurate modeling and control enhancement possibilities [1], [2]. Several tools have been developed for fractional order system analysis, modeling and controller synthesis. Among these tools are *MATLAB* toolboxes *CRONE* [3], developed by the CRONE team, and *NINTEGER* [4], developed by Duarte Valério.

The *FOMCON* toolbox for MATLAB [5] is an extension to the mini toolbox introduced in [6], [7], [8], providing graphical user interfaces (GUIs), convenience functions, means of model identification in both time and frequency domains and fractional PID controller design and optimization and a Simulink block set. The goal of the toolbox is to provide an easy-to-use, convenient and useful toolset for a wide range of users. It is especially suitable for beginners in fractional order control because of the availability of GUIs, encompassing nearly every toolbox feature, applied workflow considerations and the ability to get practical results quickly.

This work was supported by the Estonian Doctoral School in Information and Communication Technology under interdisciplinary project FOMCON, the Governmental funding project no. SF0140113As08 and the Estonian Science Foundation Grant no. 8738.

A. Tepljakov is with Department of Computer Control, Tallinn University of Technology Ehitajate tee 5, 19086, Tallinn, Estonia (e-mail: aleksei.tepljakov@dcc.ttu.ee)

E. Petlenkov is with Department of Computer Control, Tallinn University of Technology Ehitajate tee 5, 19086, Tallinn, Estonia (e-mail: eduard.petlenkov@dcc.ttu.ee)

J. Belikov is with Institute of Cybernetics, Tallinn University of Technology, Akadeemia tee 21, 12618, Tallinn, Estonia, e-mail: (e-mail: jbelikov@cc.ioc.ee)

In this paper we present an overview of the FOMCON toolbox and its functions with a summary of used theoretical aspects as well as illustrative examples. The paper is organized as follows. In Section II the reader is introduced to some basic concepts of fractional-order calculus used in control. In Section III an overview of FOMCON toolbox and its features is presented. In Section IV the main module and main GUI facility used for fractional-order system analysis are introduced. Then, the fractional-order identification toolset is presented and discussed in Section V. An overview of the fractional controllers follows in Section VI with particular focus on the PI$^\lambda$D$^\mu$ control design and optimization. Section VII is devoted to an overview of the provided Simulink blockset which can be used for more sophisticated fractional-order system modeling. In Section VIII some of the current limitations of the toolbox are outlined. Finally, in Section IX conclusions are drawn.

II. An Introduction to Fractional Calculus

Fractional calculus is a generalization of integration and differentiation to non-integer order operator ${}_a\mathscr{D}_t^\alpha$, where a and t denote the limits of the operation and α denotes the fractional order such that

$$ {}_a\mathscr{D}_t^\alpha = \begin{cases} \frac{\mathrm{d}^\alpha}{\mathrm{d}t^\alpha} & \Re(\alpha) > 0, \\ 1 & \Re(\alpha) = 0, \\ \int_a^t (\mathrm{d}t)^{-\alpha} & \Re(\alpha) < 0, \end{cases} \tag{1} $$

where generally it is assumed, that $\alpha \in \mathbb{R}$, but it may also be a complex number [7]. There exist multiple definitions of the fractional differintegral. The Riemann-Liouville differintegral is a commonly used definition [8]

$$ {}_a\mathscr{D}_t^\alpha f(t) = \frac{1}{\Gamma(m-\alpha)} \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^m \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-m+1}}\, \mathrm{d}\tau \tag{2} $$

for $m-1 < \alpha < m$, $m \in \mathbb{N}$, where $\Gamma(\cdot)$ is Euler's gamma function. Consider also the Grünwald-Letnikov definition

$$ {}_a\mathscr{D}_t^\alpha f(t) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{\alpha}{j} f(t-jh), \tag{3} $$

where $[\cdot]$ denotes the integer part.

The Laplace transform of an α-th derivative with $\alpha \in \mathbb{R}_+$ of a signal $x(t)$ relaxed at $t = 0$ (assuming zero initial conditions) is given by

$$ \mathscr{L}\left\{\mathscr{D}^\alpha x(t)\right\} = s^\alpha X(s). \tag{4} $$

Copyright © 2011 by Department of Microelectronics & Computer Science, Technical University of Lodz

Thus, a fractional-order differential equation

$$a_n \mathscr{D}^{\alpha_n} y(t) + a_{n-1} \mathscr{D}^{\alpha_{n-1}} y(t) + \cdots + a_0 \mathscr{D}^{\alpha_0} y(t) = b_m \mathscr{D}^{\beta_m} u(t) + b_{m-1} \mathscr{D}^{\beta_{m-1}} u(t) + \cdots + b_0 \mathscr{D}^{\beta_0} u(t), \quad (5)$$

where a_k, $b_k \in \mathbb{R}$ can be expressed as a fractional-order transfer function in form

$$G(s) = \frac{b_m s^{\beta_m} + b_{m-1} s^{\beta_{m-1}} + \cdots + b_0 s^{\beta_0}}{a_n s^{\alpha_n} + a_{n-1} s^{\alpha_{n-1}} + \cdots + a_0 s^{\alpha_0}}. \quad (6)$$

A system given by (6) is said to be of commensurate order if all the orders of the fractional operator s are integer multiples of a base order q such that $\alpha_k, \beta_k = kq$, $q \in \mathbb{R}^+$, $0 < q < 1$. The continuous-time transfer function can be expressed as a pseudo-rational function $H(\lambda)$, where $\lambda = s^q$:

$$H(\lambda) = \frac{\sum_{k=0}^{m} b_k \lambda^k}{\sum_{k=0}^{n} a_k \lambda^k}. \quad (7)$$

Based on this concept, a fractional-order linear time-invariant system can also be represented by a state-space model

$$\begin{aligned} \mathscr{D}^q x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t). \end{aligned} \quad (8)$$

For more information on fractional-order calculus the interested reader is referred to the books [8], [9], [10], [11].

III. Overview of the FOMCON Toolbox

A. Motivation for Development

FOMCON stands for "Fractional-Order Modeling and Control". The basic motivation for developing it was the desire to obtain a set of useful and convenient tools to facilitate the research of fractional-order systems in application to control system design. This involved writing convenience functions, e.g. the polynomial string parser, and building graphical user interfaces to improve the general workflow. However, a full suite of tools was also desired due to certain limitations in existing toolboxes, which mostly focus on novel control strategies (such as the CRONE control). FOMCON presently aims at extending classical control schemes with concepts of fractional-order calculus. The relation of FOMCON to other MATLAB fractional calculus toolboxes is depicted in Fig. 1.

Further the relation is explained. FOMCON was built upon an existing mini toolbox FOTF. It also uses several functions from NINTEGER toolbox for system identification and if the CRONE toolbox is available, it is also possible to export objects into the CRONE format for further processing. FOMCON also incorporates the `optimize()` function [12]. The latter and the NINTEGER functions are included with respect to the two-clause BSD license.

With all previous considerations, the motivations for developing FOMCON are as follows:

- It is a product suitable for both beginners and more demanding users due to availability of graphical user interfaces and advanced functionality;
- It focuses on extending classical control schemes with concepts of fractional-order calculus;
- It can be viewed as a "missing link" between CRONE and NINTEGER;
- With the Simulink blockset the toolbox aims at a more sophisticated modeling approach;
- The toolbox can be ported to other platforms, such as *Scilab* or *Octave* (some limitations may apply).

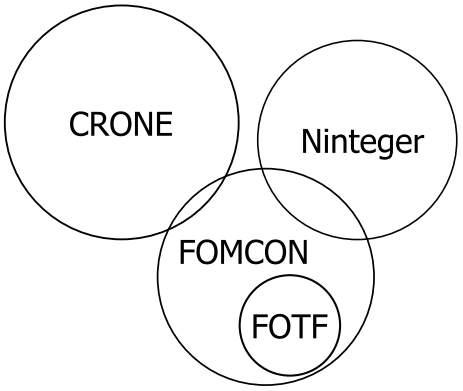


Fig. 1. MATLAB fractional-order calculus toolbox relations to FOMCON

Further we present an overview of the toolbox and its features.

B. Toolbox Features

In FOMCON the main object of analysis is the fractional-order transfer function given by (6). The toolbox focuses on the SISO (single input-single output), LTI (linear time-invariant) systems.

The toolbox is comprised of the following modules:

- Main module (fractional system analysis);
- Identification module (system identification in time and frequency domains);
- Control module (fractional PID controller design, tuning and optimization tools as well as some additional features).

All the modules are interconnected and can be accessed from the main module GUI as depicted in Fig. 2.

A Simulink blockset is also provided in the toolbox allowing complex modeling tasks to be carried out. General approach to block construction is used where applicable.

The FOMCON toolbox relies on the following MATLAB products:

- Control System toolbox, required for most features;
- Optimization toolbox, required for time-domain identification and integer-order PID tuning for common process model approximation.

It is also possible to export fractional-order systems to the CRONE toolbox format (this feature requires the object-oriented CRONE toolbox to be installed).

Further we present an overview of each FOMCON module, providing some theoretical background for the features as well as illustrative examples.

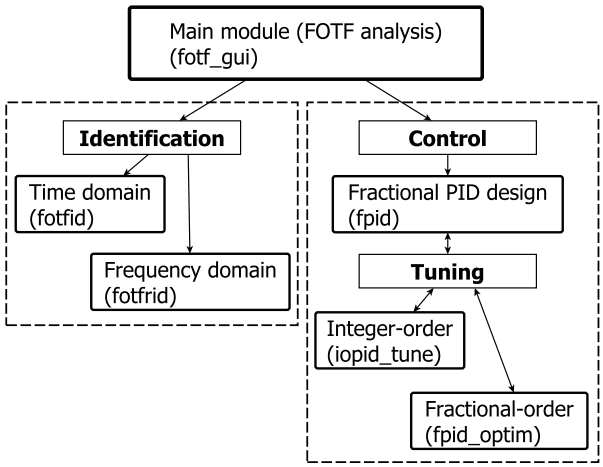


Fig. 2. FOMCON module relations (name of corresponding function to open the GUI is given in parentheses)

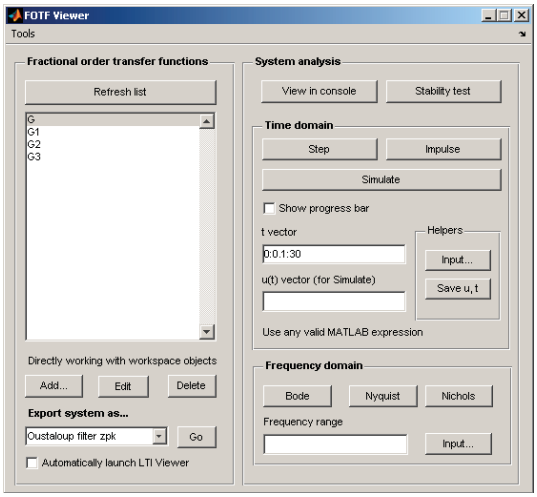


Fig. 3. Main GUI window

IV. Fractional-Order System Analysis

FOMCON provides time-domain and frequency-domain fractional-order system analysis, as well as verifying system stability. In the toolbox fractional-order systems are given by fractional-order transfer function (FOTF) objects in the form (6). These objects are generalizations of the rational transfer functions to the fractional order. To get started one could enter the following into the MATLAB command line

```
fotf_gui
```

The main toolbox GUI called *FOTF Viewer* is then displayed (see Fig. 3). It is divided into two panels:

- The left panel entitled *Fractional order transfer functions* is used to input, edit, delete and convert FOTF objects. The tool is directly working with MATLAB base workspace variables;
- The right panel entitled *System analysis* contains means for fractional-order system analysis in the time domain and in the frequency domain.

The *Tools* menu contains links to the time-domain and frequency-domain identification tools and the fractional PID design tool.

Fractional-order transfer functions may be created in the workspace by pressing the *Add...* button in the GUI. A dialog is shown allowing to enter the zero/pole fractional polynomials of the system (a simple string parser is provided). The system can then be analyzed using the tools in the right panel. Further we discuss the algorithms used to carry out the analysis.

Stability of a fractional-order LTI system (8) can be determined from the following relation

$$\left|\arg\left(\operatorname{eig}\left(A\right)\right)\right| > \gamma\frac{\pi}{2}, \tag{9}$$

where $0 < \gamma < 1$ is the commensurate order of a fractional state-space system and $\operatorname{eig}(A)$ represents the eigenvalues of the associated matrix A. If condition (9) is satisfied, then the system is stable [13]. During the stability test a figure is drawn and populated by the corresponding rational-order system (7) poles. This is an illustration to condition (9): if any of the poles are inside the shaded area of the figure the system is not stable.

Time-domain analysis of the fractional systems, i.e. simulation of the system response to an arbitrary input signal, is carried out using a revised Grünwald-Letnikov definition in (3). The closed-form numerical solution to the fractional-order differential equation is obtained in [7] as

$$y_t = \frac{1}{\sum\limits_{i=0}^{n}\frac{a_i}{h^{\alpha_i}}}\left[u_t - \sum_{i=0}^{n}\frac{a_i}{h^{\alpha_i}}\sum_{j=1}^{\frac{t-a}{h}} w_j^{(\alpha_j)} y_{t-jh}\right], \tag{10}$$

where h is the step-size in computation and $w_j^{(\alpha)}$ can be computed recursively from

$$w_0^{(\alpha)} = 1,\ w_j^{(\alpha)} = \left(1 - \frac{\alpha+1}{j}\right) w_{j-1}^{(\alpha)},\ j = 1,\ 2,\ \ldots. \tag{11}$$

The signal $\hat{u}\left(t\right)$ is calculated by using (3) substituting $(-1)^{\alpha}\binom{\alpha}{j} = w_j^{(\alpha)}$ and finally the time response under the signal $u\left(t\right)$ is obtained. Due to the fixed-step computation the accuracy of the simulation may depend on the chosen step-size h. Thus it is suggested to validate the results by gradually decreasing h until there is no variation in simulation results. Simulation of a large number of points may take a lot of time. A progress bar option is provided to allow keeping track of simulation progress in such cases.

The frequency-domain analysis is done by substituting $s = j\omega$. All the required system frequency characteristics are then obtained using Control System toolbox by supplying the complex frequency response of the plant to the frequency response data object `frd` and then using the standard toolbox frequency response analysis functions `bode()`, `nyquist()`, `nichols()`.

The export facility in the main GUI allows converting FOTF systems into objects of the following type

- Oustaloup filter `zpk`;
- Oustaloup refined filter `zpk`;
- Fractional-order state-space `foss`;
- CRONE `frac_tf`;

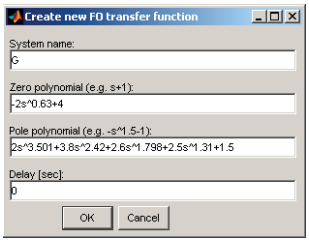


Fig. 4. FOTF entry dialog

- CRONE `frac_ss`.

The last two options are specific to the Object-Oriented CRONE toolbox and require it to be installed.

The Oustaloup filters give a very good approximation of the fractional operators [14] in a specified frequency range (ω_b, ω_h) and of order N. Oustaloup's recursive filter for s^γ for $0 < \gamma < 1$ is given by

$$G_f(s) = K \prod_{k=-N}^{N} \frac{s + \omega'_k}{s + \omega_k}, \tag{12}$$

where ω'_k, ω_k and K are obtained from

$$\begin{aligned} \omega'_k &= \omega_b \left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1-\gamma)}{2N+1}}, \\ \omega_k &= \omega_b \left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1+\gamma)}{2N+1}}, \quad K = \omega_h^\gamma. \end{aligned} \tag{13}$$

A refined Oustaloup filter has been proposed in [6]. It is given by

$$s^\alpha \approx \left(\frac{d\omega_h}{b}\right)^\alpha \left(\frac{ds^2 + b\omega_h s}{d(1-\alpha)s^2 + b\omega_h s + d\alpha}\right) G_p, \tag{14}$$

where G_p, ω_k and ω'_k can be computed from

$$\begin{aligned} G_p &= \prod_{k=-N}^{N} \frac{s + \omega'_k}{s + \omega_k}, \\ \omega_k &= \left(\frac{b\omega_h}{d}\right)^{\frac{\alpha+2k}{2N+1}}, \quad \omega'_k = \left(\frac{d\omega_b}{b}\right)^{\frac{\alpha-2k}{2N+1}}. \end{aligned} \tag{15}$$

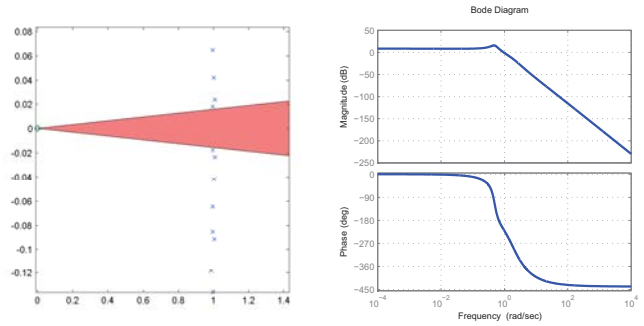
It is expected that a good approximation using (14) is obtained with $b = 10$, $d = 9$.

Fractional-order systems are converted to Oustaloup filter `zpk` objects by approximating fractional orders $\alpha \geq 1$ by $s^\alpha = s^n s^\gamma$, where n denotes the integer part of α and s^γ is obtained by the Oustaloup approximation. Objects exported this way can be analyzed using regular Control System toolbox means. There is also an option to automatically launch the *LTI Viewer* tool upon a successful export.

Example 1. Consider a system given in [7] by

$$G(s) = \frac{-2s^{0.63} + 4}{2s^{3.501} + 3.8s^{2.42} + 2.6s^{1.798} + 2.5s^{1.31} + 1.5}.$$

To supply this system as "G3" one would enter the following in the *Add...* dialog (see Fig. 4).



(a) Stability (b) Bode diagram

Fig. 5. G3 system analysis

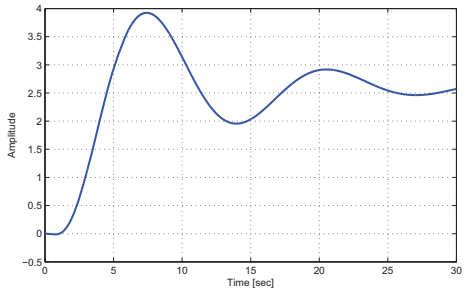


Fig. 6. G3 step response using different calculation methods

Suppose we need to check this system for stability, obtain a Bode diagram and a step response.

The stability analysis illustration is given in Fig. 5a. It can be seen from the zoomed plot that there are no poles inside the shaded region. Therefore, the condition (9) is satisfied and the system is stable. The Bode diagram is shown in Fig. 5b.

A step response in time range $t = [0; 30]$ with a step of $h = 0.01$ is obtained using the Grünwald-Letnikov method (10). The results are given in Fig. 6.

Example 2. Consider a dynamic model of a heating furnace discussed in [15], [16] given by a differential equation

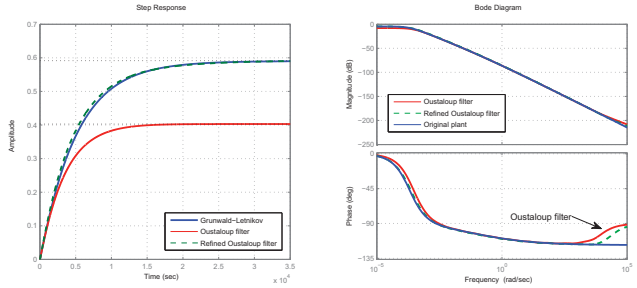
$$a_2 \mathscr{D}^\alpha y(t) + a_1 \mathscr{D}^\beta y(t) + a_0 y(t) = u(t), \tag{16}$$

with $\alpha = 1.31$, $\beta = 0.97$, $a_2 = 14994$, $a_1 = 6009.5$, $a_0 = 1.69$. In the Laplace domain, assuming zero initial conditions, the system is described by a fractional-order transfer function

$$G_1(s) = \frac{1}{14994s^{1.31} + 6009.5s^{0.97} + 1.69}.$$

We shall examine Oustaloup filter approximations of this fractional system. Let us create two filters, an Oustaloup filter `Z1` and a refined Oustaloup filter `Z2` with the default parameters ($\omega = [10^{-4}; 10^4]$, $N = 5$) and compare the resulting system step response (at $t = [0; 35000]$ with $\mathrm{d}t = 0.5$) and frequency response characteristics (Fig. 7a and 7b respectively).

From this example it can be clearly seen that only the refined Oustaloup filter proposed in [6] provides a valid approximation of the fractional-order system than the Oustaloup filter with the same approximation parameters. However, it is also possible to



(a) Step response comparison (b) Frequency response comparison

Fig. 7. Oustaloup and refined Oustaloup filter approximations of system `G1`

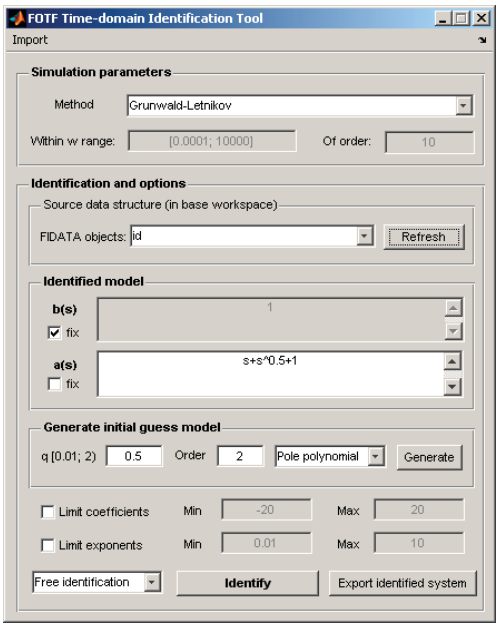


Fig. 8. FOTF Time-domain Identification Tool user interface

obtain a better approximation for this particular system with the Oustaloup filter by shifting the frequency range to $\omega = [10^{-6}; 10^{2}]$.

V. Identification by Fractional-order Model

A. Time-domain Identification

The time-domain identification tool can be accessed from the main GUI via menu *Tools→Identification→Time domain...* or by typing the following into the MATLAB command line:

```
fotfid
```

The corresponding GUI will be launched (depicted in Fig. 8). The tool allows to identify a system by a continuous-time fractional-order model in the form (6).

This is done by fitting an initial model using the least-squares approach minimizing the error norm $\left\|e\left(t\right)\right\|_2^2$, where

$$e\left(t\right) = y\left(t\right) - y_{id}\left(t\right), \tag{17}$$

by searching for a set of parameters θ of model (6), where

$$\theta = \left[\begin{array}{cccc} a_p & \alpha_p & b_z & \beta_z \end{array}\right] \tag{18}$$

and

$$\begin{aligned} a_p &= \left[\begin{array}{cccc} a_n & a_{n-1} & \cdots & a_0 \end{array}\right], \\ \alpha_p &= \left[\begin{array}{cccc} \alpha_n & \alpha_{n-1} & \cdots & \alpha_0 \end{array}\right], \\ b_z &= \left[\begin{array}{cccc} b_m & b_{m-1} & \cdots & b_0 \end{array}\right], \\ \beta_z &= \left[\begin{array}{cccc} \beta_n & \beta_{n-1} & \cdots & \beta_0 \end{array}\right]. \end{aligned} \tag{19}$$

The given parameter set can be further reduced allowing different identification strategies to be applied. The initial model can also be generated form a given commensurate order and the highest order of the model. This is useful when identification is carried out using methods discussed in [17]. The initial guess model can also be imported from the MATLAB workspace.

There is a number of possibilities to fix either fractional-order polynomials, polynomial term coefficients or exponents. Thus a generalized identification tool is obtained, capable of identifying fractional-order systems as well as integer-order systems. There is a possibility to limit the value ranges of the identified parameters. Since it is possible to reduce the number of identified parameters, the identification for complex systems is better conditioned for the underlying optimization task.

A special data structure is used to store the identification data. It can be constructed from the command-line in the following manner:

```
id1 = fidata(y, u, t);
```

where `id1` is the data structure used for identification, `y` is the experimental output signal, `u` is the experimental input signal and `t` is the time vector. The identification tool only works with this type of data structure.

Example 3. Suppose a fractional-order system is given by

$$G_2\left(s\right) = \frac{1}{0.8s^{2.2} + 0.5s^{0.9} + 1}.$$

In order to generate identification test data, the following MATLAB commands can be used:

```
t = (0:0.01:20)';
u = zeros(length(t),1);
u(1:200) = ones(200,1);
u(1000:1500) = ones(501,1);
y = lsim(G2,u,t)';
iddata1 = fidata(y,u,t);
```

If nothing but experimental data is known the only way to identify the system is by experimenting with different commensurate orders and orders of the initial model. Also the term coefficients and differentiation orders can be adjusted manually. In this case, if a fractional pole polynomial is selected such, that it is generated with commensurate order $q = 1.2$ and order $N = 2$, the zero polynomial is fixed at $z_p = 1$, free identification is used with term coefficients limited to $c_{lim} = [-20; 20]$ and exponents limited to $e_{lim} = [1 \cdot 10^{-8}; 3]$ then the system is identified as

$$\hat{G}(s) = \frac{1}{0.8s^{2.2} + 0.5s^{0.90001} + s^{9.7153 \cdot 10^{-7}}}.$$

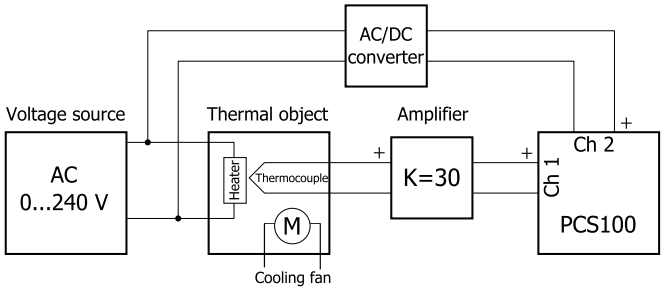


Fig. 9. Experiment schematic diagram

It can be seen that the last term has an order $\alpha = 9.7153 \cdot 10^{-7} \to 0$ and thus the original model is successfully recovered and can be obtained by typing `round(Gid, 1e-4, 1e-4)` in MATLAB. In order to validate the data, one could also type `validate(iddata1, Gid)`. A figure will be drawn showing the comparison of the sampled and identified system responses to the experimental input data as well as the output error.

Example 4. Consider now an example, where a real system is identified. Experimental data is collected from a real thermal object. The schematic diagram corresponding to the experiment is depicted in Fig. 9.

The temperature is measured using a type *K* thermocouple with a DC output of 0...10 mV, amplified with a gain of 30 and fed into a *Velleman PCS100* oscilloscope, which is used to register both the temperature obtained from the amplified thermocouple signal and the voltage source signal. Data was collected from three consecutive experiments. Different voltage set values were used. Due to some limitations of used software and hardware, a total of 1700 points were recorded with a sampling interval of $T_s = 2$ seconds. The obtained system output vector was then filtered ensuring zero phase distortion using a low-pass filter by means of MATLAB `filtfilt()` function, and a transformation was applied so that the output signal vector would contain real temperature values in °C. To account for zero initial conditions requirement the temperature output signal was also shifted such that $t = 0 \to y(t) = 0$. A transformation was applied such that $\hat{u}(t) = 0.01 \cdot u^2(t)$ — the obtained input signal is thus a rough approximation of the final temperature value.

The identification was then carried out using the `fotfid` tool. From previous experience it is known that in case of an integer-order model this system can be approximated by a second order model. Thus, it is possible to obtain the initial guess model by generating a fractional pole polynomial with $q = 1$, $n = 2$ and fixing the zero polynomial at "1" so that a classical, integer order model is initially obtained in the form

$$G_{init}(s) = \frac{1}{s^2 + s + 1}.$$

The free identification method was used with coefficient limits $c_{lim} = [0; 3000]$ and exponent limits $e_{lim} = [10^{-9}; 3]$. The following model was obtained:

$$\hat{G}(s) = \frac{1}{2012.409s^{1.8063} + 107.2882s^{0.93529} + 1.0305}.$$

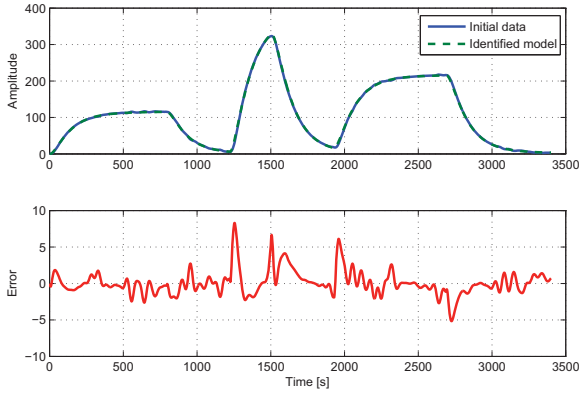


Fig. 10. Thermal system fractional model validation

TABLE I
PROCESS MODEL IDENTIFICATION COMPARISON

	IDENTIFIED MODEL	SQUARE ERROR NORM
$\hat{G}_1(s) =$	$\frac{1}{2012.409s^{1.8063}+107.2882s^{0.93529}+1.0305}$	71.7702
$\hat{G}_2(s) =$	$\frac{0.96039}{2655.4725s^2+151.626s+1}e^{-1.1844s}$	72.3396
$\hat{G}_3(s) =$	$\frac{0.96035}{2835.2438s^2+152.593s+1}$	81.8971

Validation is carried out with the same dataset as for identification. The corresponding plot is given in Fig. 10.

Two integer-order models were obtained from the same experimental dataset by using the MATLAB Identification toolbox for comparison. Results are provided in Table I.

Taking the square error norm as a measure of model precision, one could say that the fractional-order model $\hat{G}_1$ is more accurate than the integer-order models $\hat{G}_2$ and $\hat{G}_3$. This is to be expected due to the properties of fractional operators and the extra degrees of modeling freedom and allows for an improvement in control system characteristics. However, in order to claim this explicitly one would need to use hardware and software methods with a more strict precision requirement.

B. Frequency-domain Identification

The time-domain identification tool can be launched from *FOTF Viewer* via menu *Tools→Identification→Frequency domain...* or by typing the following into the MATLAB command line:

```
fotfrid
```

The frequency-domain identification tool GUI will then be displayed. The tool allows to identify a fractional-order model either in the form

$$G(s) = \frac{1}{c_n s^{n\gamma} + c_{n-1} s^{(n-1)\gamma} + \cdots + c_1 s^{\gamma} + c_0} \tag{20}$$

or in the form

$$G(s) = \frac{b_m s^{m\gamma} + b_{m-1} s^{(m-1)\gamma} + \cdots + b_1 s^{\gamma} + b_0}{a_n s^{n\gamma} + a_{n-1} s^{(n-1)\gamma} + \cdots + a_1 s^{\gamma} + 1}, \tag{21}$$

where γ is the fractional-order transfer function commensurate order and n, m are the corresponding polynomial orders. Further we discuss the identification methods used by the tool.

Three algorithms for system identification in the frequency domain are available [18], [19]. The Hartley method allows to obtain the model (20) parameters $c_0, c_1, \ldots, c_n$ from the experimentally collected complex frequency response by solving the following equation

$$\begin{bmatrix} \frac{1}{G(j\omega_1)} \\ \frac{1}{G(j\omega_2)} \\ \vdots \\ \frac{1}{G(j\omega_m)} \end{bmatrix} = \tag{22}$$

$$\begin{bmatrix} 1 & (j\omega_1)^{\gamma} & (j\omega_1)^{2\gamma} & \cdots & (j\omega_1)^{n\gamma} \\ 1 & (j\omega_2)^{\gamma} & (j\omega_2)^{2\gamma} & \cdots & (j\omega_2)^{n\gamma} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (j\omega_m)^{\gamma} & (j\omega_m)^{2\gamma} & \cdots & (j\omega_m)^{n\gamma} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix},$$

where $\omega_1, \omega_2, \ldots, \omega_n$ are the sampling frequencies. When using this algorithm, the user must supply the commensurate order γ as well as model order n.

The Levy and Vinagre identification methods allow to identify a fractional-order model in the form (21). The underlying algorithm for both methods is the same. It finds the parameters for an experimental frequency response given by $G(j\omega) = \Re(\omega) + j\Im(\omega)$ by solving the following equation

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_m \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} e \\ g \end{bmatrix}, \tag{23}$$

where $b_0, \ldots b_m$ and $a_1, \ldots a_n$ are the identified model parameters and A, B, C, D and e, g are constructed from the experimental data. Please see [19] for detailed explanation of these parameters.

During identification using Levy's method the following square norm is minimized:

$$\begin{aligned} \epsilon &= G(j\omega)\left[a_n(j\omega)^{n\gamma} + \cdots + a_1(j\omega)^{\gamma} + 1\right] \\ &- \left[b_m(j\omega)^{m\gamma} + \cdots + b_1(j\omega)^{\gamma} + b_0\right]. \end{aligned} \tag{24}$$

The Vinagre method adds weights to the norm in order to improve the approximation at low frequencies such that $\epsilon' = w \cdot \epsilon$, where weights w are frequency dependent and for frequencies ω_i, $i = 1$, ..., f they are given by

$$w = \begin{cases} \frac{\omega_2 - \omega_1}{2\omega_1^2} & , \quad i = 1, \\ \frac{\omega_{i+1} - \omega_{i-1}}{2\omega_i^2} & , \quad 1 < i < f, \\ \frac{\omega_f - \omega_{f-1}}{2\omega_f^2} & , \quad i = f. \end{cases}$$

In order to use these methods, the user needs to supply the commensurate order, as well as fractional polynomial orders n and m. This allows for an additional optimization problem to be stated for a set of parameters $\theta = \begin{bmatrix} \gamma & n & m \end{bmatrix}$. An objective function to minimize is given by a performance index in the form

$$J = \frac{1}{n_\omega} \sum_{i=1}^{n_\omega} \left| G(j\omega) - \hat{G}(j\omega) \right|^2,$$

where n_ω is the number of frequencies in ω, G is the original plant, from which the response was obtained, and $\hat{G}$ is the identified plant. The error index is also used to evaluate the identification result in general.

As with the time-domain identification, a special data structure `ffidata` is used to hold the experimental frequency response. It can be constructed from the MATLAB command-line as follows:

```
id1 = ffidata(mag, ph, w);
```

or

```
id1 = ffidata(r, w);
```

where `mag` is the observed frequency response magnitude in dB, `ph` is the observed frequency response phase angle in degrees and `w` is the vector containing frequencies in rad/s, where the response is known. Alternatively, it is possible to create the identification data structure using the complex response `r`.

Example 5. In this example we will illustrate the use of the frequency-domain identification tool. Consider a plant given by

$$G(s) = \frac{s^{0.32} + 5}{100s^{1.92} + 20s^{0.96} - 5s^{0.64} + 1}.$$

Let us generate an identification dataset `fid1` with 50 logarithmically spaced frequency sample points in the range $\omega = [10^{-4}; 10^{4}]$. This can be done by writing the following into the MATLAB command line:

```
G    = newfotf('s^0.32+5', ...
       '100s^1.92+20s^0.96-5s^0.64+1');
w    = logspace(-4, 4, 50);
fid1 = ffidata(freqresp(G, w), w);
```

We will now identify the fractional model by means of the `fotfrid` tool.

As with the time-domain identification, if no information about the model is given, the only way to identify it is by trial and error. The good news is that frequency-domain identification is fast. In this case, one could use the Vinagre method and choose a commensurate order $q = 0.2$, leaving the polynomial orders at their default value $n = m = 5$, and also apply the best fit identification by going to *Tools→Best fit* and setting the maximum model orders to $N = 6$ in the optimization settings dialog. With these settings applied the

following model is obtained:

$$\begin{aligned}\hat{G}(s) &= \frac{\hat{b}(s)}{\hat{a}(s)},\\ \hat{b}(s) &= 2.6322\cdot 10^{-15}s^{1.92} - 1.4416\cdot 10^{-13}s^{1.6}\\ &+ 3.2699\cdot 10^{-12}s^{1.28} - 3.7288\cdot 10^{-11}s^{0.96}\\ &+ 2.1969\cdot 10^{-10}s^{0.64} + s^{0.32} + 5,\\ \hat{a}(s) &= 100s^{1.92} + 1.6987\cdot 10^{-8}s^{1.6}\\ &- 1.0219\cdot 10^{-8}s^{1.28} + 20s^{0.96} - 5s^{0.64}\\ &- 1.5758\cdot 10^{-10}s^{0.32} + 1.\end{aligned}$$

The initial system can be obtained using the following:

```
G = trunc(Gid, 1e-5, 1e-5)
```

Obviously, the system used in this example was relatively easy to identify. In practical cases, one should carefully consider the choice of the commensurate order. The identified system polynomials may be of a very high order, in which case the *Best fit* tool will be less useful due to large computational efforts involved.

For a practical example of frequency-domain identification see the fractional lead-lag compensator realization in Example 7.

VI. Fractional-order Control

In this section we discuss the fractional-order controllers used in the FOMCON toolbox, namely the fractional PID controller, fractional lead-lag compensator and the TID controller. Our main focus will be on the fractional PID controller, due to its importance in the industry [7].

A. $PI^{\lambda}D^{\mu}$ Controller Design, Tuning and Optimization

The fractional-order PID controller was first introduced by Podlubny in [1]. This generalized controller is called the $PI^{\lambda}D^{\mu}$ controller (notation $PI^{\lambda}D^{\delta}$ is also used in literature) and has an integrator with an order λ and a differentiator of order μ. Recent researches show that the fractional-order PID outperforms the classical PID [20], [21].

The fractional PID controller transfer function has the following form

$$G_c(s) = K_p + \frac{K_i}{s^{\lambda}} + K_d s^{\mu}. \tag{25}$$

Obviously, when taking $\lambda = \mu = 1$ the result is the classical integer-order PID controller. With more freedom in tuning the controller, the four-point PID diagram can now be seen as a PID controller plane, which is conveyed in Fig. 11.

The fractional-order PID design tool can be accessed from the main GUI by *Tools→Fractional PID design* or by the `fpid` command. It allows to design a $PI^{\lambda}D^{\mu}$ controller for a typical negative feedback unity system shown in Fig. 12.

There are several approaches for fractional PID design which depend on the plant to be controlled. If the plant is given by an integer-order model, then classical tuning procedures could be employed to obtain integer-order PID parameters.

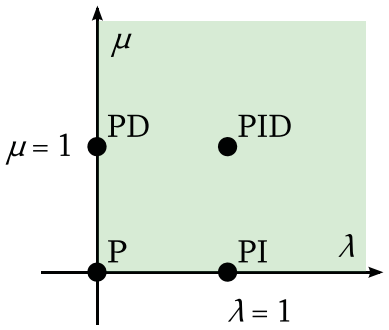


Fig. 11. The $PI^{\lambda}D^{\mu}$ controller plane

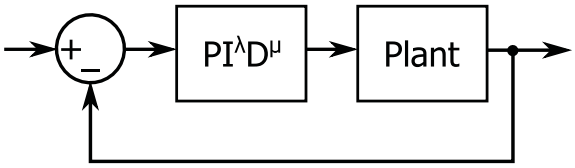


Fig. 12. Feedback control system with fractional PID controller

Fractional PID orders can then be tuned to achieve enhanced performance. A tool is provided which permits identifying the process (which could also be fractional-order) by well-known models (FOPDT, IPDT, FOIPDT) and computing integer-order PID gains using classical tuning strategies such as Ziegler-Nichols, Åström-Hägglund etc. [6]. This tool can be accessed from the menu *Tuning→Integer-order PID* or by typing `iopid_tune`. For fractional-order PID tuning consider methods proposed in [8], [22].

Another case is when the plant is of fractional-order. No special tuning method is currently provided. However, a tuning method for a class of plants can be found in [16].

The optimization tool, provided in FOMCON, can in practice be used for fractional PID tuning due to its flexibility. The tool can be accessed from the PID design tool menu *Tuning→Optimize* (or by typing `fpid_optim`). The tool is shown in Fig. 13. Here is a summary of the options provided:

- Plant model and fractional PID approximation type. Only Oustaloup filter type simulations are used mainly due to processing speed.
- Possibility to tune all parameters, fix gains or fix fractional exponents.
- Possibility to constrain every tuned parameter, except for the lower bound of the exponents which is fixed.
- Optimization to several performance metrics (ISE, IAE, ITSE, ITAE).
- Some control over control system performance specifications (gain and phase margin).
- User-defined number of optimization iterations.

The optimization tool uses the `optimize` function [12] in order to tune the fractional PID parameters by minimizing the function given by the corresponding performance index. These are as follows:

- Integral square error $ISE = \int_0^t e^2(t)\ \mathrm{d}t$,
- Integral absolute error $IAE = \int_0^t |e(t)|\ \mathrm{d}t$,
- Integral time-square error $ITSE = \int_0^t te(t)^2\ \mathrm{d}t$,

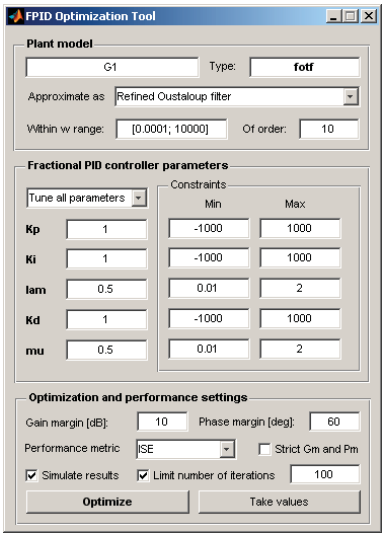


Fig. 13. Fractional PID optimize tool

- Integral time-absolute error $ITAE = \int_0^t t\,|e(t)|\,\mathrm{d}t$,

where $e(t) = 1 - y(t)$, $y(t)$ is the tuned fractional control system step response.

Example 6. Consider the model of a thermal object we obtained through identification in Example 4:

$$G = \frac{1}{2012.4087s^{1.8063} + 107.2882s^{0.93529} + 1.0305}.$$

We will now design a PI$^\lambda$D$^\mu$ controller for this plant using the optimization tool.

Initially the PID parameters are set to $K_p = K_i = K_d = 100$, $\lambda = \mu = 1$. The exponents are fixed so that an integer-order PID could be designed. Search limits are set to $K = [-500;\ 500]$ for gains and $\gamma = [0.01;\ 2]$. For simulation, the refined Oustaloup filter approximation is used with default parameters ($\omega = [0.0001;\ 10000]$, $N = 10$). Specifications are as follows. Gain margin is set to 10 dB, while phase margin to 45 degrees (non-strict). Performance metric is *IAE*.

Optimization with these settings leads to the following integer-order PID controller parameter set: $K_p = 457.8607$, $K_i = 0.97807$, $K_d = 408.3947$. Obtained open-loop phase margin is $\varphi_m = 45.01°$. Next the gains are fixed and integrator and differentiator orders are set to $\lambda = \mu = 0.5$. The strict option is enabled. The optimization is then continued. As a result, the orders are found such that $\lambda = 0.24726$ and $\mu = 0.7528$.

A comparison of simulation of the designed control systems with a set value $SV = 150$ is shown in Fig. 14. It can be seen, that by tuning only the orders of the controller a better result is achieved. It is important to note, however, that this result is obtained with an unconstrained control effort value, which is always limited in practical situations. Thus it may be required to review the controller settings according to these limitations for practical use.

B. Fractional Lead-Lag Compensator

Lead-lag compensators are a well-known type of feedback controller widely used in practice. Extending it with ideas from fractional calculus can lead to a more robust controller.

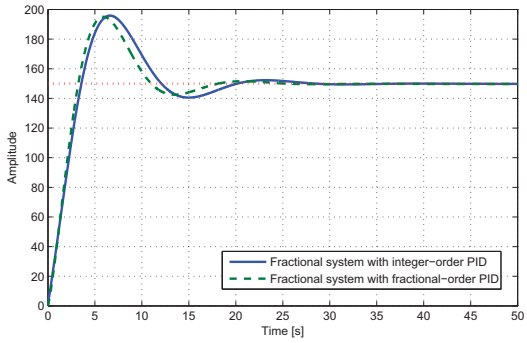


Fig. 14. Control system for thermal object comparison

A fractional-order lead-lag compensator has the following transfer function:

$$G_c(s) = k'\left(\frac{\lambda s + 1}{x\lambda s + 1}\right)^{\alpha}, \tag{26}$$

where α is the fractional order of the controller, λ and x are parameters such that $\frac{1}{\lambda} = \omega_z$ is the zero frequency and $\frac{1}{x\lambda} = \omega_p$ is the pole frequency and $k' = K_c x^{\alpha}$. When $\alpha > 0$ the controller (26) corresponds to a fractional-order lead compensator and when $\alpha < 0$ it corresponds to a fractional lag compensator.

The contribution of parameter α is such, that the lower its value, the longer the distance between the zero and pole and vice versa so that the contribution of phase at a certain frequency stands still. This makes the controller more flexible and allows a more robust approach to the design. Tuning and auto-tuning techniques are discussed in [8].

No specialized tool is yet available in FOMCON for fractional lead-lag controller tuning. However, tools are proposed for the analysis of this controller. Since the controller is given in implicit form, to obtain a transfer function the following can be done:

- Obtain a complex frequency response of the controller, a special function `frlc()` is available in FOMCON for this task;
- Identify the controller using an appropriate tool.

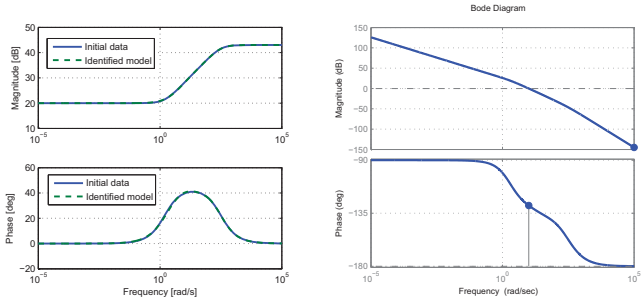
Further we illustrate this procedure.

Example 7. Consider an integer-order plant given by a model

$$G(s) = \frac{2}{s(0.5s + 1)}.$$

In this example, we will realize a fractional lead-lag compensator for this plant discussed in [8]. The gain crossover frequency is chosen such that $\omega_{cg} = 10\,rad/sec$. At this frequency the plant has a magnitude of $-28.1291\,\mathrm{dB}$ and a phase of $-168.69°$. To achieve a magnitude of 0 dB at the gain crossover frequency and a phase margin $\varphi_m = 50°$ the fractional lead compensator is designed with parameters $k' = 10$, $x = 0.005$, $\lambda = 0.6404$, $\alpha = 0.5$ and thus has the following implicit fractional-order transfer function

$$G_c(s) = 10\left(\frac{0.6404s + 1}{0.0032s + 1}\right)^{0.5}.$$



(a) Controller identification result (b) Control system open-loop frequency response

Fig. 15. Fractional lead compensator realization

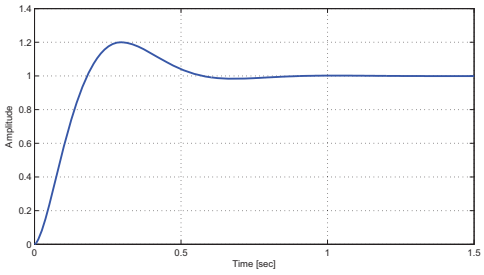


Fig. 16. Control system with fractional lead compensator step response

In order to implement this controller, the frequency response data is obtained using the `frlc()` function in the range $\omega = [10^{-5}; 10^{5}]$ and a frequency-domain identification dataset is created by typing the following in MATLAB:

```
w   = logspace(-5,5,1000);
r   = frlc(10,0.005,0.6404,0.5,w);
flc = ffidata(r,w);
```

Next, the `fotfrid` tool is used to obtain a fractional-order approximation of the compensator. With $q = 0.492$, setting both polynomial orders to 4 and using the Vinagre method, the following fractional-order transfer function is obtained with an error $J = 0.014867$:

$$\hat{G}_c(s) = \frac{\hat{b}(s)}{\hat{a}(s)},$$

$$\begin{aligned}\hat{b}(s) &= 0.031325s^{1.968} + 0.30643s^{1.476} \\ &+ 4.6284s^{0.984} + 4.0234s^{0.492} + 10.0005,\end{aligned}$$

$$\begin{aligned}\hat{a}(s) &= 0.0002215s^{1.968} + 0.0021625s^{1.476} \\ &+ 0.061928s^{0.984} + 0.41302s^{0.492} + 1.\end{aligned}$$

The frequency fitting result is also shown in Fig. 15a. The open-loop control system frequency response is given in Fig. 15b. It can be seen, that the desired crossover frequency $\omega_{cg} = 9.94$ and phase margin $\varphi_m = 51.6°$ are very close to specification.

Finally, the step response of the designed control system is given in Fig. 16.

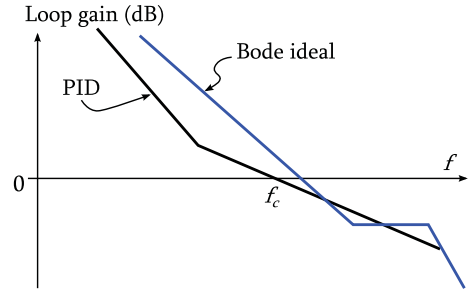


Fig. 17. Bode plots for PID controlled plant and the ideal loop response

C. TID Controller

The TID (tilt-integral-derivative) controller was first proposed in [23]. The structure of the TID controller is given by the following transfer function

$$G_c(s) = \frac{K_t}{s^{\frac{1}{n}}} + \frac{K_i}{s} + K_d s, \tag{27}$$

where $K_t/s^{\frac{1}{n}}$ is the *Tilt* type compensator and $n \in \mathbb{R}$, $n > 0$, preferably $n \in [2; 3]$. It can be seen, that the TID controller corresponds to a conventional PID controller with proportional gain replaced by the compensator component. The motivation for this type of controller is from the consideration of Bode's theoretically optimal loop response (see Fig. 17). A possible tuning strategy according to this consideration is given in [23], [24].

In order to obtain a fractional transfer function in FOMCON, one could use the following:

```
Gc = tid(Kt, n, Ki, Kd);
```

where parameters K_t, n, K_i, K_d correspond to those in (27).

VII. Modeling in Simulink

The FOMCON Simulink block library currently consists of eight blocks and is shown in Fig. 18.

The library is based on Oustaloup filter approximation by means of the `oustapp()` function. The discrete blocks use the Control System toolbox function `c2d()` to obtain the discrete model from the Oustaloup filter LTI system. General block structure is used where applicable.

The difference between the *Fractional operator* and *Fractional derivative* blocks is that the order α of the former is limited to $0 < \alpha < 1$.

In order to ensure efficient and accurate simulation, the model built with these blocks may be made up of stiff systems and an appropriate solver should be used in Simulink in such a case (`ode15s` or `ode23tb`).

Example 8. Consider a model of a dynamic system and the corresponding fractional-order controller discussed in [6] and given by the following transfer functions:

$$G(s) = \frac{1}{0.8s^{2.2} + 0.5s^{0.9} + 1},$$

$$G_c(s) = 233.4234 + \frac{22.3972}{s^{0.1}} + 18.5274 \cdot s^{1.15}.$$

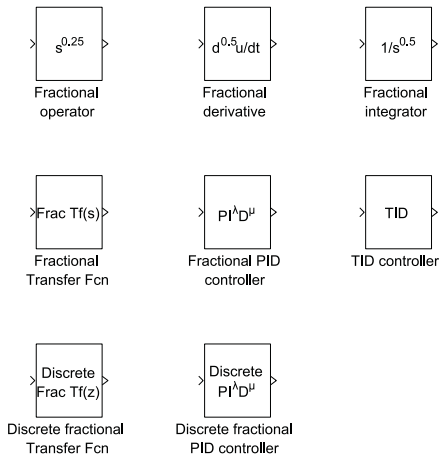


Fig. 18. FOMCON Simulink library

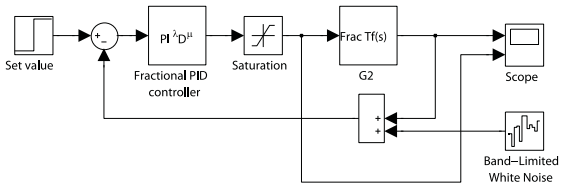


Fig. 19. System model in Simulink

Let us build the corresponding model in Simulink, using the above blockset. The resulting model is given in Fig. 19. A saturation block is added, limiting the control signal within an interval $U_{lim} = [-100; 100]$ and adding a band-limited white noise block for simulating disturbance in the system with power of $P = 10^{-9}$, sample time of $T = 0.01$ and seed value of 23341. System simulation result is given in Fig. 20.

VIII. Discussion

The FOMCON toolbox was developed and tested in MATLAB v. 7.7. However, most of the features are backwards-compatible and were tested with earlier releases of MATLAB (versions 7.4-7.6). FOMCON requires System Control toolbox for general functionality and Optimization toolbox for model identification in the time domain.

Further, we discuss some of the current limitations of the FOMCON toolbox.

- The PID optimization tool lacks complete control over control system gain and phase margins. The algorithm can only guarantee that the minimum given specifications are met by evaluating the open-loop control system at every optimization step when the *Strict* option is checked. However, the initial fractional PID parameters should strictly satisfy the minimum specifications or else optimization will not be carried out and an error will be issued.
- More design specifications settings are required for PID tuning, including minimum and maximum allowed value settings for the control effort.
- Both the identification and optimization tools work with numbers at a fixed accuracy of four decimal places.
- Time domain identification tool does not yet identify the system lag parameter.
- There are no automatic tuning algorithms implemented for the fractional lead-lag compensator and TID controller.
- While the order of the fractional derivative block in Simulink can have an order $\alpha > 1$, the accuracy of the simulation will be reduced with higher orders.

The current limitations of the FOMCON toolbox will be the subject of further development and will be gradually eliminated in future releases.

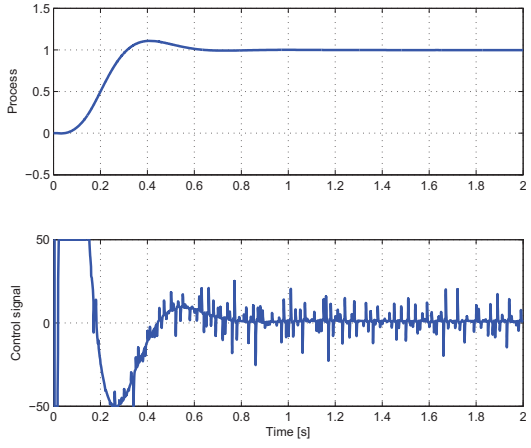


Fig. 20. Fractional-order control system simulation result

IX. Conclusions

In this paper, we presented a MATLAB toolbox containing the necessary tools to work with a class of fractional-order models in control. Theoretical aspects behind the tools were also covered with illustrative examples. A set of graphical user interfaces was introduced with relevant comments. We have discussed fractional-order system analysis, identification in both time and frequency domains and a set of fractional-order controllers, focusing on tuning and optimization of the fractional PID controller. The performance of the latter was found to be superior to an integer-order PID obtained during the same tuning procedure. A Simulink blockset was also presented in the paper.

References

[1] I. Podlubny, "Fractional-order systems and PI$^{\lambda}$D$^{\mu}$-controllers," vol. 44, no. 1, pp. 208–214, 1999.
[2] M. Yahyazadeh and M. Haeri, "Application of fractional derivative in control functions," in *Proc. Annual IEEE India Conf. INDICON 2008*, vol. 1, 2008, pp. 252–257.
[3] A. Oustaloup, P. Melchior, P. Lanusse, O. Cois, and F. Dancla, "The CRONE toolbox for Matlab," in *Proc. IEEE Int. Symp. Computer-Aided Control System Design CACSD 2000*, 2000, pp. 190–195.
[4] D. Valrio. (2005) Toolbox ninteger for MatLab, v. 2.3. [Online]. Available: http://web.ist.utl.pt/duarte.valerio/ninteger/ninteger.htm
[5] A. Tepljakov, E. Petlenkov, and J. Belikov. (2011) FOMCON toolbox. [Online]. Available: http://www.fomcon.net/

[6] D. Xue, Y. Chen, and D. P. Atherton, *Linear Feedback Control: Analysis and Design with MATLAB (Advances in Design and Control)*, 1st ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.
[7] Y. Q. Chen, I. Petras, and D. Xue, "Fractional order control - a tutorial," in *Proc. ACC '09. American Control Conference*, 2009, pp. 1397–1411.
[8] C. A. Monje, Y. Chen, B. Vinagre, D. Xue, and V. Feliu, *Fractional-order Systems and Controls: Fundamentals and Applications*, ser. Advances in Industrial Control. Springer Verlag, 2010.
[9] K. Miller and B. Ross, *An introduction to the fractional calculus and fractional differential equations*. Wiley, 1993.
[10] I. Podlubny, *Fractional differential equations*, ser. Mathematics in science and engineering. Academic Press, 1999.
[11] A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo, *Theory and Applications of Fractional Differential Equations, Volume 204 (North-Holland Mathematics Studies)*. New York, NY, USA: Elsevier Science Inc., 2006.
[12] R. Oldenhuis. (2009) Optimize. MathWorks File Exchange. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/24298-optimize
[13] D. Matignon, "Generalized fractional differential and difference equations: Stability properties and modeling issues," in *Proc. of Math. Theory of Networks and Systems Symposium*, 1998, pp. 503–506.
[14] A. Oustaloup, F. Levron, B. Mathieu, and F. M. Nanot, "Frequency-band complex noninteger differentiator: characterization and synthesis," vol. 47, no. 1, pp. 25–39, 2000.
[15] I. Podlubny, L. Dorcak, and I. Kostial, "On fractional derivatives, fractional-order dynamic systems and $PI^{\lambda}D^{\mu}$-controllers," in *Proc. 36th IEEE Conf. Decision and Control*, vol. 5, 1997, pp. 4985–4990.
[16] C. Zhao, D. Xue, and Y. Chen, "A fractional order pid tuning algorithm for a class of fractional order plants," in *Proc. IEEE Int Mechatronics and Automation Conf*, vol. 1, 2005, pp. 216–221.
[17] A. Oustaloup, R. Malti, M. Aoun, and J. Sabatier, "Tutorial on system identification using fractional differentiation models," in *14th IFAC Symposium on System Identification*, 2006, pp. 606–611.
[18] T. T. Hartley and C. F. Lorenzo, "Fractional-order system identification based on continuous order-distributions," *Signal Process.*, vol. 83, pp. 2287–2300, November 2003.
[19] D. Valrio and J. Costa, "Levy's identification method extended to commensurate fractional order transfer function," in *EUROMECH Nonlinear Dynamics conference*, Eindhoven, Netherlands, 2005.
[20] Y. Luo and Y. Chen, "Fractional-order [proportional derivative] controller for robust motion control: Tuning procedure and validation," in *Proc. ACC '09. American Control Conference*, 2009, pp. 1412–1417.
[21] M. Čech and M. Schlegel, "The fractional-order pid controller outperforms the classical one," in *Process control 2006*. Pardubice Technical University, 2006, pp. 1–6.
[22] C. Yeroglu, C. Onat, and N. Tan, "A new tuning method for $PI^{\lambda}D^{\mu}$ controller," in *Proc. Int. Conf. Electrical and Electronics Engineering ELECO 2009*, vol. II, 2009, pp. 312–316.
[23] B. J. Lurie, "Three-parameter tunable tilt-integral-derivative (TID) controller," US Patent US5 371 670, 1994.
[24] D. Xue and Y. Chen, "A comparative introduction of four fractional order controllers," in *Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002, pp. 3228–3235.

Aleksei Tepljakov was born in 1987 in Tallinn. He received his B.Sc and M.Sc in computer and systems engineering from Tallinn University of Technology. He is currently working at the Department of Comupter Control at Tallinn University of Technology. His main research interests include fractional-order control of complex systems and fractional filter based signal processing.

Eduard Petlenkov Eduard Petlenkov was born in 1979. He recieved his B.Sc, M.Sc and PhD degrees in computer and systems engineering from Tallinn University of Technology. He is an Associate Professor in the Department of Computer Control at Tallinn University of Technology. His main research interests lie in the domain of nonlinear control, system analysis and computational intelligence.

Juri Belikov was born in 1985. He recieved his B.Sc degree in mathematics from Tallinn University, and his M.Sc in computer and systems engineering from Tallinn University of Technology. He joined the the Institute of Cybernetics and Department of Computer Control at Tallinn University of Technology. His main research interests lie in the domain of nonlinear control theory.