FPGA Realization of Approximate Fractional-Order Derivative Functions and Its Applications

Dr. Murat Köseoğlu
Department of Electrical-Electronics Engineering
Inonu University
Malatya, Turkey
murat.koseoglu@inonu.edu.tr

Muhammed Tarık Yıldız
Department of Electrical-Electronics Engineering
Inonu University
Malatya,Turkey
muhammettarikyildiz@gmail.com

Elvan Çelik
Department of Electrical-Electronics Engineering
Inonu University
Malatya,Turkey
elvanncelik.49@gmail.com

***Abstract*—Exploring the use of fractional calculus is essential for it to be used properly in various applications. Implementing the fractional operator D^{α} in FPGA is an important research topic in fractional calculus; in the literature, only a few FPGA implementations have been proposed due to the memory dependence of the fractional order systems. The study presents FPGA realization of a modified version of SBL fitting method. This method is used for the first time to implement on FPGA. Two example of fractional order low pass filters are demonstrated for possibility of realization of approximate fractional-order derivative functions.**

Keywords— Fractional order approximation methods, FOPID filter, fractional order transfer function modeling, approximate FPGA implementation, FPGA realization of FOPID

I. INTRODUCTION

Fractional order derivative and integrator can be considered as an extension of integer order derivative and integrator operators to the case of non-integer orders and it is defined in general form as the following [11],

$$ {}_{a}\mathrm{D}_{t}^{\alpha} = \begin{cases} \dfrac{d^{\alpha}}{dt^{\alpha}} & \alpha > 0 \\ 1 & \alpha = 0 \\ \displaystyle\int_{a}^{t} (d\tau)^{(-\alpha)} & \alpha < 0 \end{cases} $$

where, ${}_{a}D_{t}^{\alpha}$ represents fundamental non-integer order operator of fractional calculus. Parameters a and t are the lower and upper bounds of integration, and $\alpha \in R$ denotes the fractional-order[7].

This subsection introduces an improved version of the SBL fit- ting approximation method that was previously proposed by [8] and [9] . SBL fitting approach is based on the fitting of the stability boundary locus of the fractional derivative operator s αand its approximation model [8,9]. It has characteristic properties of curve fitting methods such as selecting fitting points, finding model parameters that provide the best fitting to the sampled points. In former SBL fitting method[9], approximate model parameters were solved according to two linear equation systems: The first equation system is written for real part of characteristic equations and some parameters of the approximate model are solved. Then, to solve the unsolved parameters of the approximate model, imaginary part of characteristic equations is used. This two-stage solution strategy increases complexity of ap- proximate model calculations. In the current study, the solution scheme is modified to solve real and imaginary part equations all together, and the M-SBL fitting method is introduced[1]. Moreover, former SBL fitting method uses very narrow frequency ranges, and the frequency sampling points to fit SBL curve are trivial. In the M-SBL method, trivial frequency point selection problem is relieved by applying logarithmically equal spacing in frequency range. Some remarks on M-SBL method[1];

• Real and imaginary parts equations are solved together as a linear equation system[1].

• A general formulation of SBL fitting scheme is derived for approximate modeling of fractional-order derivative elements (sα)[1].

• Selection of frequency sampling points was carried out more systematically by using logarithmically spaced sampling point selection strategy[1].

In order to obtain stability boundary locus of fractional order derivative element s α, a closed loop control system, which consists of a plant function and a controller, should be considered. For this closed loop control system, the characteristic equation is written for plant function G (s) and controller function C (s) as follows[1]:

$$\Delta(s) = 1 + C(s)G(s) \tag{1}$$

To obtain SBL curves for s^{α}, fractional order derivate operatör is considered as the plant function $\mathrm{G}(\mathrm{s}) = s^{\alpha}$. A PI controller, $C(s) = k_p + \frac{k_i}{s}$, is commonly selected for SBL drawing in $k_p(\omega)$ and $k_i(\omega)$ parameter space[1]. By substituting PI controller and plant transfer function into Eq. (1), the characteristic equation is written by

$$\Delta(s) = 1 + \left(k_p + \frac{k_i}{s}\right) s^{\alpha} = 0. \tag{2}$$

This equation is solved to obtain parametric SBL curve in space of $(k_p(\omega), k_i(\omega))$. To this end, one can use $s = j\omega$ and $s^{\alpha} = (j\omega)^{\alpha} = \omega^{\alpha}[\cos\left(\frac{\pi}{2}\alpha\right) + jsin(\frac{\pi}{2}\alpha)]$ in Eq. (2) and obtain a continuous frequency domain expression of characteristic equaion as follows:

$$\Delta(j\omega) = 1 + \left(k_p + \frac{k_i}{j\omega}\right)\omega^{\alpha}\left[\cos\left(\frac{\pi}{2}\alpha\right) + jsin\left(\frac{\pi}{2}\alpha\right)\right] = 0 \tag{3}$$

$$k_p(\omega) = -\frac{\cos\left(\frac{\pi}{2}\alpha\right)}{\omega^{\alpha}} \text{ and } k_{pi}(\omega) = -\frac{\sin\left(\frac{\pi}{2}\alpha\right)}{\omega^{\alpha-1}} \text{ , for } \omega \neq 0 \tag{4}$$

For fitting of this characteristic equation to the SBL curve of s^{α}, $k_p(\omega)$ and $k_i(i)$ given in Eq. (4) are used in the Eq. (3) as follows

$$\Delta_n(j\omega) = \sum_{r=0}^{n} \left[(j\omega)^{r+1} - (j\omega)^{n-r+1} \frac{\cos(\pi\alpha/2)}{\omega^{\alpha}} - (j\omega)^{n-r} \frac{\sin(\pi\alpha/2)}{\omega^{\alpha-1}} \right] a_r = 0. \quad (5)$$

Let us denote $x_r = (j\omega)^{r+1} - (j\omega)^{n-r+1} \cdot \frac{\cos(\pi\alpha/2)}{\omega^{\alpha}} - (j\omega)^{n-r} \frac{\sin(\pi\alpha/2)}{\omega^{\alpha-1}}$ and Eq. (5) in a more compact form:

$$k_p(\omega) = \frac{\omega^{\alpha} \cos\left(\frac{\pi}{2}\alpha\right)}{\omega^{\alpha} \sin\left(\frac{\pi}{2}(\alpha+1)\right) \omega^{\alpha} \cos\left(\frac{\pi}{2}\alpha\right) - \omega^{\alpha-1} \sin\left(\frac{\pi}{2}\alpha\right) \omega^{\alpha+1} \cos\left(\frac{\pi}{2}(\alpha+1)\right)}$$

$$k_i(\omega) = \frac{\omega^{\alpha+1} \cos\left(\frac{\pi}{2}(\alpha+1)\right)}{\omega^{\alpha} \sin\left(\frac{\pi}{2}(\alpha+1)\right) \omega^{\alpha} \cos\left(\frac{\pi}{2}\alpha\right) - \omega^{\alpha-1} \sin\left(\frac{\pi}{2}\alpha\right) \omega^{\alpha+1} \cos\left(\frac{\pi}{2}(\alpha+1)\right)}$$

II. FPGA Implementation of M-SBL Method

A. Discretization Methods

In applied mathematics, discretization is the process of transferring continuous functions, models, variables, and equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation and implementation on digital computers. Dichotomization is the special case of discretization in which the number of discrete classes is 2, which can approximate a continuous variable as a binary variable (creating a dichotomy for modeling purposes, as in binary classification)[6].

Whenever continuous data is discretized, there is always some amount of discretization error. The goal is to reduce the amount to a level considered negligible for the modeling purposes at hand[6].

Discretization is also concerned with the transformation of continuous differential equations into discrete difference equations, suitable for numerical computing[6].

MATLAB Control System Toolbox™ offers several discretization and interpolation methods for converting dynamic system models between continuous time and discrete time and for resampling discrete-time models. Some methods tend to provide a better frequency-domain match between the original and converted systems, while others provide a better match in the time domain[3]. Some methods is shown below.

Bilinear

This method for converting a continuous system into a discrete system is called Bilinear, or Tustin transformation. It maps the left-hand complex plan in the s-domain into the unit circle in the z-domain. That is, for every stable point in the continuous system, the bilinear transformation will return a stable point in the discrete system[4].

This method relates the s-domain transfer function to the z-domain transfer function using the following approximation[4]:

$$s = \frac{2}{T_s} \cdot \frac{z-1}{z+1} \quad (8)$$

$$s = \frac{z-1}{zT_s} \quad (9)$$

Zero-order hold

The zero-order hold method takes into account the discretization delay of a continuous system. This method of discretization relates the s-domain system and the z-domain system by the following approximation[4]:

$$X(z) = (1 - z^{-1}) . Z \left\{ \mathcal{L}^{-1} \left\{ \frac{X(s)}{s} \right\} \right\} \quad (10)$$

B. Converting Discrete Transfer Function To Difference Equation

The Going from a transfer function to a single nth order differential equation is equally straightforward; the procedure is simply reversed. Starting with a third order transfer function with x(t) as input and y(t) as output[2].

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^2 + b_1 s + b_2}{a_0 s^3 + a_1 s^2 + a_2 s + a_3} \quad (11)$$

To find the transfer function, first write an equation for X(s) and Y(s), and then take the inverse Laplace Transform. Recall that multiplication by "s" in the Laplace domain is equivalent to differentiation in the time domain[2].

$$(a_0 s^3 + a_1 s^2 + a_2 s + a_3) Y(s) = (b_0 s^2 + b_1 s + b_2) X(s) \quad (12)$$

$$a_0 s^3 Y(s) + a_1 s^2 Y(s) + a_2 s Y(s) + a_3 Y(s) = b_0 s^2 X(s) + b_1 s X(s) + b_2 X(s) \quad (13)$$

$$a_0 \dddot{y}(t) + a_1 \ddot{y}(t) + a_2 \dot{y}(t) + a_3 y(t) = b_0 \ddot{x}(t) + b_1 \dot{x}(t) + b_2 x(t) \quad (14)$$

For the general case of an nth order transfer function:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \quad (15)$$

$$a_0 s^n Y(s) + a_1 s^{n-1} Y(s) + \cdots + a_{n-1} s Y(s) + a_n Y(s) = b_0 s^m X(s) + b_1 s^{m-1} X(s) + \cdots + b_{m-1} s X(s) + b_m X(s) \quad (16)$$

$$a_0 y^{(n)}(t) + a_1 y^{(n-1)}(t) + \cdots + a_{n-1} y(t) + a_n y(t) = b_0 x^{(m)}(t) + b_1 x^{(m-1)}(t) + \cdots + b_{m-1} x(t) + b_m x(t) \quad (17)$$

This can be written in even more compact notation:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\sum_{i=0}^{n} a_i s^{n-i}}{\sum_{i=0}^{m} b_i s^{m-i}} = \frac{\mathcal{L}(output)}{\mathcal{L}(input)} \quad (18)$$

$$Y(s) \sum_{i=0}^{n} b_i s^{n-i} = X(s) \sum_{i=0}^{m} a_i s^{m-i} \quad (19)$$

$$\sum_{i=0}^{n} b_i y^{(n-i)} = \sum_{i=0}^{m} a_i x^{(m-i)} \quad (20)$$

III. Examples Of Fractional-Order Low Pass Filter Implementation On A Xilinx FPGA

Two example of fractional-order low pass filter implementation on a Xilinx FPGA device.

A. Example-1

An fractional-order discrete IIR filter implementation of the fractional-order low pass filter given as,

$$F(s) = \frac{1}{s^{0.5}+1} \quad (21)$$

Firstly, we need to use M_SBL() MATLAB function to get M-SBL transfer function of Equation 21.

```
Alpha=0.5;%Fractional derivative operator
s^0.5
n=1;%degree of integer order approximate
model
wl=0.01;%Lower bound of frequency range
wh=1000;%Upper bound of frequency range
```

```
[Gs]=M_SBL(Alpha,n,wl,wh)
s=tf('s');
Gapp = 1/(1+Gs);
bode(Gapp);
```

Result will be $G_{s_{0.5}} = \frac{1000 \cdot s+1}{s+1000}$.

Then, obtained result is converted to discrete transfer function by c2d MATLAB function.

```
hdiscrete = c2d(Gapp,1,'zoh');
```

Result will be $H(z) = \frac{0.000999 \cdot z+0.6305}{z-0.3679}$.

Then, obtained discrete transfer function is converted difference equation.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-1}}{z^{-1}} \cdot \frac{0.0000999 \cdot z + 0.6305}{z - 0.3679} = \frac{0.000999 + 0.6305 \cdot z^{-1}}{1 - 0.3679 \cdot z^{-1}}$$

$$Y(z) \cdot (1 - 0.3679 \cdot z^{-1}) = X(z) \cdot (0.000999 + 0.6305 \cdot z^{-1})$$

$$y[k] - 0.3679 \cdot y[k-1] = 0.000999 \cdot x[k] + 0.6305 \cdot x[k-1]$$

$$y[k] = 0.3679 \cdot y[k-1] + 0.000999 \cdot x[k] + 0.6305 \cdot x[k-1]$$

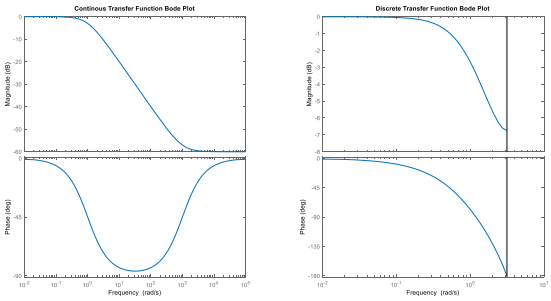

Figure 1. Bode plot of fractional-order transfer functions

Using Xilinx System Generator, we create the circuit as below.

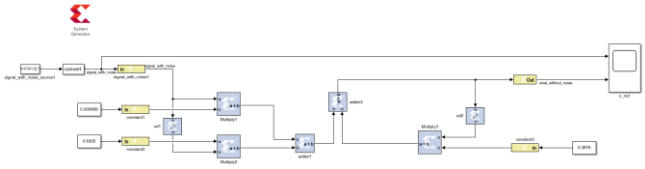

Figure 2. Implemented Circuit using System Generator

Simulation result will be as below.

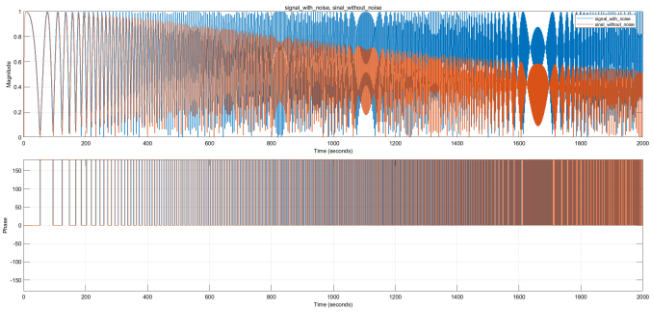

Figure 3. Simulation result of the circuit

Xilinx System Generator creates HDL files to use on a Xilinx FPGA device. The generated system is given below.

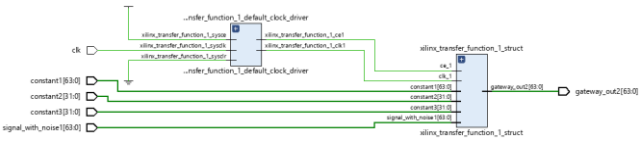

Figure 4. RTL implementation of HDL

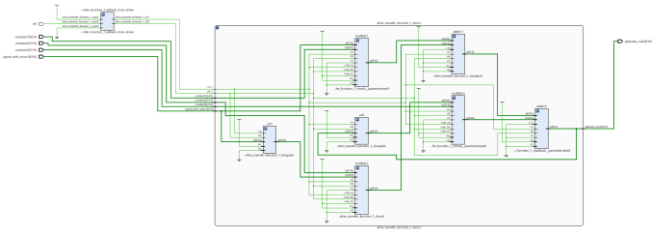

Figure 5. Detailed view RTL implementation of HDL

Table 1. Used FPGA sources

Resource	Estimation	Available	Utilization %
FF	128	41600	0.31

B. *Example-2*

Another fractional-order discrete IIR filter implementation of the fractional-order low pass filter given as,

$$F(s) = \frac{1}{1+5 \cdot s^{0.5}+0.3 \cdot s^{0.3}} \quad (22)$$

Firstly, we need to use M_SBL() MATLAB function to get M-SBL transfer function of Equation 22.

```
Alpha=0.5;%Fractional derivative operator
s^0.5
n=1;%degree of integer order approximate
model
wl=0.01;%Lower bound of frequency range
wh=1000;%Upper bound of frequency range
[Gs_0_5] = M_SBL(Alpha,n,wl,wh)
Alpha=0.3;%Fractional derivative operator
s^0.5
n=1;%degree of integer order approximate
model
wl=0.01;%Lower bound of frequency range
wh=1000;%Upper bound of frequency range
[Gs_0_3] = M_SBL(Alpha,n,wl,wh)
s=tf('s');
Gapp=1/(1+5*Gs_0_5+0.3*Gs_0_3);
bode(Gapp);
```

Results will be

$$G_{s_{0.5}} = \frac{1000 \cdot s + 1}{s + 1000}$$

$$G_{s_{0.3}}(s) = \frac{17.85 \cdot s + 1}{s + 17.85}$$

Then, obtained result is converted to discrete transfer function by c2d MATLAB function.

```
hdiscrete = c2d(Gapp,1,'zoh');
```

Result will be $H(z) = \frac{0.0001997 \cdot z^2 + 0.1718 \cdot z - 0.0004834}{z^2 - 0.8248 \cdot z + 5.038 \cdot 10^{-9}}$.

Then, obtained discrete transfer function is converted difference equation.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-2}}{z^{-2}} \cdot \frac{0.0001997 \cdot z^2 + 0.1718 \cdot z - 0.0004834}{z^2 - 0.8248 \cdot z + 5.038 \cdot 10^{-9}}$$

$$\frac{Y(z)}{X(z)} = \frac{0.0001997 + 0.1718 \cdot z^{-1} - 0.0004834 \cdot z^{-2}}{1 - 0.8248 \cdot z^{-1} + 5.038 \cdot 10^{-9} \cdot z^{-2}}$$

$$Y(z) \cdot (1 - 0.8248 \cdot z^{-1} + 5.038 \cdot 10^{-9} \cdot z^{-2}) = X(z) \cdot (0.0001997 + 0.1718 \cdot z^{-1} - 0.0004834 \cdot z^{-2})$$

$$y[k] - 0.8248 \cdot y[k-1] + 5.038 \cdot 10^{-9} \cdot y[k-2] = 0.0001997 \cdot x[k] + 0.1718 \cdot x[k-1] - 0.0004834 \cdot x[k-2]$$

$$y[k] = 0.8248 \cdot y[k-1] - 5.038 \cdot 10^{-9} \cdot y[k-2] + 0.0001997 \cdot x[k] + 0.1718 \cdot x[k-1] - 0.0004834 \cdot x[k-2]$$

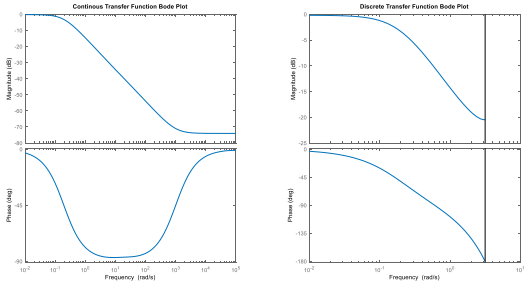

Figure 6. Bode plot of fractional-order transfer functions

Using Xilinx System Generator, we create the circuit as below.

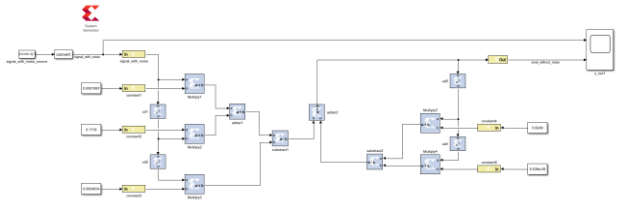

Figure 7. Implemented Circuit using System Generator

Simulation result will be as below.

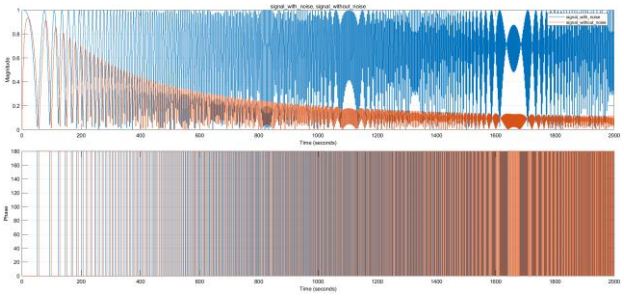

Figure 8. Simulation result of the circuit

Xilinx System Generator creates HDL files to use on a Xilinx FPGA device. The generated system is given below.

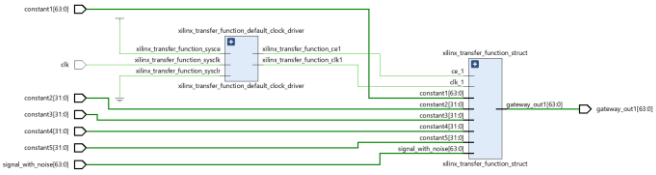

Figure 9. RTL implementation of HDL

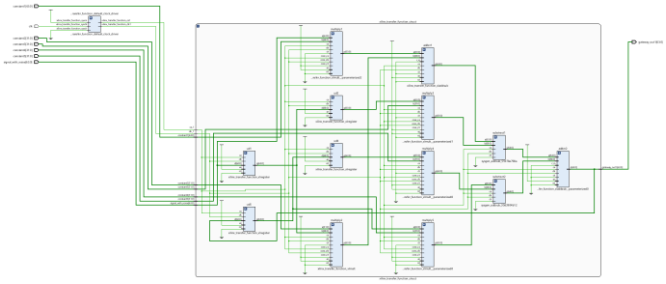

Figure 10. Detailed view RTL implementation of HDL

Table 2. Used FPGA sources

Resource	Estimation	Available	Utization %
LUT	121	20800	0.58
FF	256	41600	0.62

REFERENCES

[1] Deniz, Furkan & Alagoz, Baris & Tan, Nusret & Koseoglu, Murat. (2020). Revisiting four approximation methods for fractional order transfer function implementations: Stability preservation, time and frequency response matching analyses. Annual Reviews in Control. 49. 10.1016/j.arcontrol.2020.03.003.

[2] Erik Cheever, Swarthmore College. (n.d.). *Single Diff Eq → Transfer Function*. Transformation: Single Diff Eq ↔ Transfer Function. Retrieved June 30, 2021, from https://lpsa.swarthmore.edu/Representations/SysRepTransformations/TF2SDE.html

[3] *Continuous-Discrete Conversion Methods - MATLAB & Simulink*. (n.d.). MATLAB. Retrieved June 30, 2021, from https://www.mathworks.com/help/control/ug/continuous-discrete-conversion-methods.html

[4] *Discrete Transfer Function*. (n.d.). Typhoon HIL Documentation. Retrieved June 30, 2021, from https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/discrete_transfer_function.html

[5] Tan, N., Yüce, A., & Deniz, F. N. (2016). *TEACHING FRACTIONAL ORDER CONTROL SYSTEMS USING INTERACTIVE TOOLS*. ICEMST 2016: International Conference on Education in

Mathematics, Science & Technology, Bodrum, Turkey. https://dergipark.org.tr/tr/download/article-file/334405

[6] Wikipedia contributors. (2021, June 23). *Discretization*. Wikipedia. https://en.wikipedia.org/wiki/Discretization

[7] Oustaloup, A., Levron, F., Mathieu, B., & Nanot, F. M. (2000). Frequency-band com- plex noninteger differentiator: Characterization and synthesis. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47 (1), 25–39. doi: 10.1109/81.817385.

[8] Deniz, F. N. (2017). Modelling ang Control Applications in Fractional Order Systems PhD Thesis . Inonu University.

[9] Deniz, F. N., Alagoz, B. B., Tan, N., & Atherton, D. P. (2016). An integer order approx- imation method based on stability boundary locus for fractional order deriva- tive/integrator operators. ISA Transactions, 62 , 154–163. doi: 10.1016/j.isatra.2016. 01.020 .

[10] Deniz, F. N., Keles, C., Alagoz, B. B., & Tan, N. (2014). Design of fractional-order PI controllers for disturbance rejection using RDR measure. 2014 International Con- ference on Fractional Differentiation and Its Applications. ICFDA 2014. doi: 10.1109/ ICFDA.2014.6967446.

[11] Chen, Y. Q., Petras, I., & Xue, D. (2009). Fractional order control-a tutorial. 2009 American Control Conference , 1397–1411. Retrieved from http://ieeexplore.ieee. org/xpls/abs _ all.jsp?arnumber=5160719 .