Application of Newton's Method to Analog and Digital Realization of Fractional-order Controllers

Aleksei Tepljakov, Eduard Petlenkov, and Juri Belikov

***Abstract*—In this paper, a method for approximating a first-order implicit fractional transfer function, that corresponds to a frequency-bounded fractional differentiator or integrator, is presented. The proposed method is based on a modification of the well-known Newton's method for iterative root approximation. First-order implicit fractional transfer functions have several applications in modeling and control. This type of transfer function is the basis for the fractional lead-lag compensator. In the following, we provide the description of our algorithm, that enhances the existing technique, and illustrate its use in analog and digital implementations of fractional-order systems and controllers with relevant examples and comments.**

***Index Terms*—fractional calculus, Newton's method, Carlson's method, Matlab, implicit fractional transfer function, fractional power zero-pole**

I. Introduction

NOWADAYS, fractional-order calculus is a rapidly evolving scientific field. It allows for more accurate modeling of complex systems, such as those that possess memory and hereditary properties [1]. The benefits of using fractional calculus in control are also evident. New types of controllers have been developed [2], [3], [4] based on the added flexibility of the fractional-order models.

However, many problems arise in the process of implementation of fractional-order controllers. Since fractional models are inherently complex, which follows from the fact that they describe infinite-dimensional systems [5], deducing an effective direct realization method is a difficult task. Therefore, methods for approximating the fractional operators have been developed, including both continuous and discrete approximations.

In [6] we focused on one particular continuous integer-order approximation, derived from Newton's method. In this paper, we provide further information about the generalization of this method for higher order iteration formulas as well as describe the use of the updated second order method in analog and digital implementation of fractional-order systems and controllers. In Section II the underlying method together with its modification are summarized and applications to fractional-order modeling are presented. In Section III our method used for implicit first-order fractional transfer function approximation is proposed and discussed. Applications of this method to controller implementation are also presented and a MATLAB realization is described. Analog and digital realizations of the approximated fractional-order models are discussed in Section IV. Two illustrative examples follow in Section V. Some issues and limitations of the proposed method are discussed in Section VI. Finally, conclusions are drawn in Section VII.

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

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)

II. Application of Newton's Method to Fractional Calculus

Newton's method, also known as Newton-Rhapson method [7], is a numerical algorithm for finding a real root of a function $f(x)$. It suggests that in order to solve a general nonlinear equation $f(x) = 0$, the following iterative formula can be used, given an initial estimate x_0:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{1}$$

A modified algorithm is proposed in [8], [9] such that the convergence of the sequence $\{x_k\}$ is more rapid than that resulting from using formula (1). The corresponding formula is called Halley's formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k) - \frac{f(x_k)f''(x_k)}{2f'(x_k)}}. \tag{2}$$

Consider now a problem of finding an nth root of a real number. The corresponding function is $f(x) = x^n - A$ and using (2) the following particular iteration formula is obtained:

$$x_{k+1} = x_k \cdot \frac{(n-1)x_k^n + (n+1)A}{(n+1)x_k^n + (n-1)A}. \tag{3}$$

For instance, to approximate $\sqrt{23}$ we may set $x_0 = 1$ and use (3) to obtain the solution. After three iterations, the results of which are depicted in Fig. 1, we arrive at the result $x = 4.795831523312707$, which is accurate to 13 decimal places.

The formula (3) is considered by Carlson [10] and more recently in [5], [11], [12]. In his paper, Carlson has shown, that this formula holds for both even $n = 2m$ and odd $n = 2m+1$ roots. The method can be applied to approximate fractional

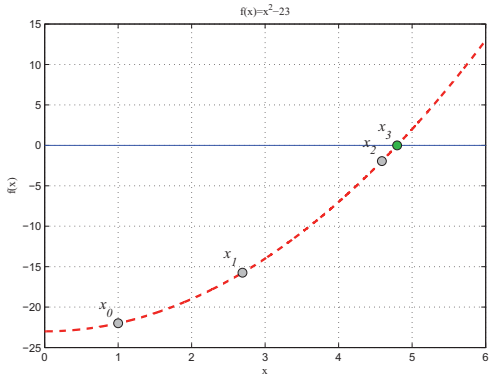

Fig. 1. Using the modified Newton method to approximate $x = \sqrt{23}$

capacitors of the form $(1/s)^{1/n}$ in the following way:

$$G_{k+1}(s) = G_k(s)\frac{(n-1)G_k^n(s) + (n+1)H(s)}{(n+1)G_k^n(s) + (n-1)H(s)}, \quad (4)$$
$$H(s) = \frac{1}{s}, \quad G_0(s) = 1.$$

Since in this case the real variable A is replaced by the transfer function $H(s)$, convergence and rate of convergence cannot be evaluated in the same way as in the case of a real-valued function.

Consider an example. Using equation (4) we shall obtain an approximation of a fractional capacitor $\sqrt[5]{1/s}$. After two iterations the following transfer functions are obtained:

$$\begin{aligned} G_0(s) &= 1, \\ G_1(s) &= \frac{0.66667s(s+1.5)}{s(s+0.6667)}, \\ G_2(s) &= \frac{G_{21}(s)}{G_{22}(s)}, \end{aligned}$$

where

$$\begin{aligned} G_{21}(s) &= 0.4444s^7 + 9.062s^6 + 39.47s^5 + 77.81s^4 \\ &\quad + 82.5s^3 + 47.75s^2 + 13.23s + 1, \\ G_{22}(s) &= s^7 + 13.23s^6 + 47.75s^5 + 82.5s^4 \\ &\quad + 77.81s^3 + 39.47s^2 + 9.063s + 0.4444. \end{aligned}$$

In Fig. 2 the frequency response of the obtained approximation is shown. The response of the corresponding ideal fractional capacitor is also given for comparison. It can be seen, that the frequency range, where the approximation is valid, is quite narrow. It is possible to improve this result by increasing the number of formula iterations. However, in this case the order of the obtained rational transfer function may be very high.

It is interesting to note the relation of Newton's method to the fractal theory. This relation is illustrated in e.g. [13]. Further connections between fractional-order calculus and the fractal theory as well as the arising applications are studied in [14], [15], [16].

Next, we provide further comments about the generalization of this method due to Householder [17]. The proposed iterative

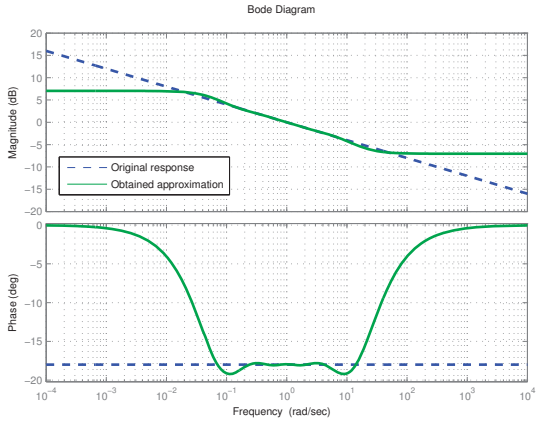

Fig. 2. Frequency response of Carlson's approximation of the fractional capacitor $\sqrt[5]{1/s}$

process formula is given by

$$x_{n+1} = x_n + d \cdot \frac{[1/f(x)]^{(d-1)}(x_n)}{[1/f(x)]^{(d)}(x_n)}, \quad (5)$$

where d is the order of the resulting method. It can be easily seen, that the original Newton's method in (1) corresponds to the case when $d = 1$ and Halley's method in (3) is obtained with $d = 2$.

This equation can easily be implemented in a CAS, such as Maxima[1], in the following way

```
hh(f,x,d) := x + d * diff(1/f,x,d-1) /
                    diff(1/f,x,d);
```

For example, one could generate a third-order iterative formula for $f(G) = G^n - H$ by using

```
hh(G^n-H,G,3);
```

After simplification the following equation is obtained:

$$G_{n+1} = G_n \cdot \frac{G_{num}}{G_{den}},$$

where

$$\begin{aligned} G_{num} &= (n^2-1)\cdot H^2 + (4n^2+2)\cdot G^n H \\ &\quad + (n^2-1)\cdot G^{2n}, \\ G_{den} &= (n^2-3n+2)\cdot H^2 + (4n^2-4)\cdot G^n H \\ &\quad + (n^2+3n+2)\cdot G^{2n}. \end{aligned}$$

In this work we shall consider the second-order method, which is Carlson's original method, since it provides a compromise between accuracy and efficiency, although using higher order methods could be justified in some cases. We will use this method for frequency-bounded implicit fractional transfer function approximation and treat the case of a first-order transfer function.

[1]Maxima is a cross-platform GPL-licensed computer algebra system and can be obtained for free at http://maxima.sourceforge.net/

III. Approximation Method for First-order Implicit Fractional Transfer Functions

A. First-order Implicit Fractional Transfer Function in Modeling and Control

In general, a frequency-bounded non-integer differentiator/integrator may be represented by a first-order implicit fractional transfer function in the form

$$G(s) = K \cdot \left(\frac{bs+1}{as+1}\right)^{\alpha}, \tag{6}$$

where K is the static gain of the system and $0 < \alpha < 1$. The frequency of the zero is in this case $\omega_z = 1/b$ and the frequency of the pole is $\omega_p = 1/a$, when $\alpha > 0$. Following the terminology in [14] and since in this case the transfer function has a single fractional power zero and a single fractional power pole, we also refer to this form as a Fractional Power Zero-Pole (FPZP) pair.

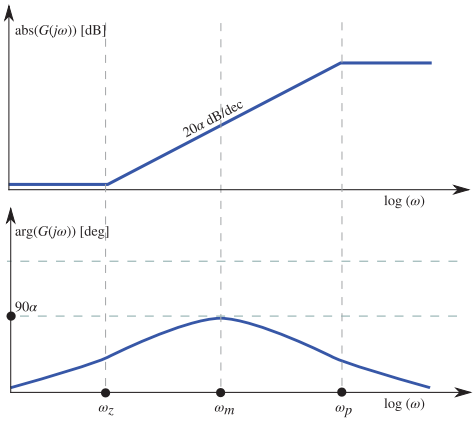

Fig. 3. Bode diagram corresponding to a fractional power pole-zero pair transfer function frequency response with $\alpha > 0$ and $b > a$

The benefits of using fractional calculus in modeling are most evident when analyzing the frequency behavior of the resulting models. A bode diagram corresponding to (6) with $\alpha > 0$ and $b > a$ is given in Fig. 3. It can be seen, that by varying α a magnitude slope of 20α dB/dec and a phase of $90\alpha^{\circ}$ can be achieved, which allows for more intricate modeling possibilities. This additional freedom is also very important in control design. In particular, the transfer function in (6) corresponds to the fractional lead-lag compensator — a generalization of the conventional controller used in many industrial applications [18], [19]. The fractional lead-lag compensator has the following form [20]:

$$C(s) = K_c x^{\alpha} \left(\frac{\lambda s+1}{x\lambda s+1}\right)^{\alpha}, \quad 0 < x < 1, \tag{7}$$

where $\lambda = b$ and $x\lambda = a$ in (6) and $K_c x^{\alpha}$ is the controller gain.

We now describe the algorithm, which can be used to obtain accurate approximations in form of zero-pole distributions for the fractional transfer function in (6).

B. Approximation Algorithm

Based on the previous discussion, several problems of the original algorithm in [10] may be outlined:

- the initial estimate for approximation problem is not addressed,
- the method only allows to obtain approximations for transfer functions of order $1/n$,
- resulting approximations can be of a very high order,
- the limited frequency range where the approximation is valid.

The specific implementation of Carlson's method could be different. In fact, when applied to the problem of approximating the transfer function in (6) for a limited frequency range, the algorithm provides very accurate results. Further, we describe the refined algorithm, which aims to solve the aforementioned problems.

First, we consider the initial estimate problem. Using the iteration formula (4) results in a recursive distribution of zeros and poles around a central frequency. In case of the fractional power zero-pole pair transfer function, this frequency is the geometric mean computed from the zero and pole frequencies such that

$$\omega_m = \sqrt{\omega_z \omega_p} = \frac{1}{\sqrt{ab}}. \tag{8}$$

It relates to the initial estimate choice through the magnitude of the fractional transfer function obtained at this frequency:

$$G_0(\omega) = |G(j\omega_m)| = \left|\frac{jb\omega_m + 1}{ja\omega_m + 1}\right|^{\alpha}. \tag{9}$$

When selecting the initial estimate according to (9) the resulting zero-pole distribution is then centered around ω_m ensuring that way the validity of the approximation around this frequency. When the ratio a/b is small, only two iterations are usually required to achieve a good result in the full frequency range. Note, that this choice of the initial estimate is useful for such formulas in (5) that $d = 2k$, $k \in \mathbb{Z}^+$.

The problem of approximating transfer functions of arbitrary real order using this method is much more difficult to solve. Here, we must choose a balance between accuracy and efficiency, since in case of order $1/n$ each iteration step involves computing the nth power of a transfer function obtained in the previous step. The order of the approximation grows rapidly. Thus, until a different, more efficient iteration formula is developed, we limit the resolution to $1/10$. This allows to obtain approximations of orders accurate to at least one decimal place. However, there is no reason why a class of arbitrary orders could not be considered as well.

The problem of using the method to obtain an approximation for an arbitrary real α falls under the Egyptian fraction decomposition class of problems, i.e. an order α is decomposed into k simple fractions $1/m_k$:

$$\alpha = \frac{1}{m_1} + \frac{1}{m_2} + \cdots + \frac{1}{m_k}, \tag{10}$$

where $m_k \in \mathbb{Z}^+$. The order decomposition algorithm is depicted in Fig. 4 and is discussed below.

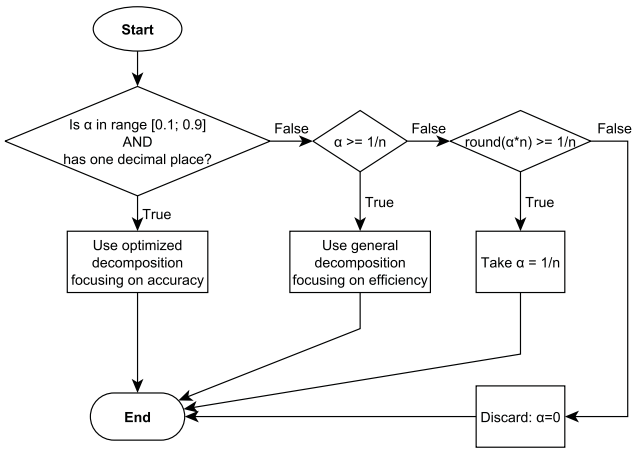

Fig. 4. Order α decomposition algorithm

for $P = 2$ to M **do**
  if $\alpha \geq (1/P)$ **then**
    $G \leftarrow G \cdot G^{1/P}$
    $\alpha \leftarrow \alpha - (1/P)$
  end if
end for

Fig. 5. General decomposition algorithm

The optimized decomposition is conducted using fractions $1/2$ (most efficient), $1/5$ and $1/10$ (accuracy consideration). The decimal fractions are then decomposed as follows:

$$0.1 = \tfrac{1}{10}, \quad 0.2 = \tfrac{1}{5}, \quad 0.3 = \tfrac{1}{5} + \tfrac{1}{10},$$
$$0.4 = 2 \cdot \tfrac{1}{5}, \quad 0.5 = \tfrac{1}{2}, \quad 0.6 = 3 \cdot \tfrac{1}{5},$$
$$0.7 = \tfrac{1}{2} + \tfrac{1}{5}, \quad 0.8 = 4 \cdot \tfrac{1}{5}, \quad 0.9 = 4 \cdot \tfrac{1}{5} + \tfrac{1}{10}.$$

The fractional transfer function is then approximated as

$$G^{\alpha}(s) = \prod_{j=1}^{k} G_{base}^{\frac{1}{m_j}}(s), \tag{11}$$

where

$$G_{base}(s) = \left(\frac{bs+1}{as+1}\right). \tag{12}$$

Note, that the initial estimate is computed for every approximation of $G_{base}^{1/m_j}(s)$.

The general decomposition algorithm is given in Fig. 5. In our case $M = 10$ and thus for $0 < \alpha < 1$ a decomposition will always be found, since

$$\sum_{k=2}^{10} \left(\frac{1}{k}\right) > 1.$$

For $\alpha > 1$ the general commutative property of a fractional operator is considered, so the approximation is found such that

$$G^{\alpha}(s) = G^{n}(s) \cdot G^{\gamma}(s), \tag{13}$$

where $n = \alpha - \gamma$ denotes the integer part of α and $G^{\gamma}(s)$ is obtained using (11). For the case when $\alpha < 0$, the approximation is

$$G^{-\alpha}(s) = \left(\frac{1}{G(s)}\right)^{\alpha}. \tag{14}$$

Finally, we address the problem of approximation order. We propose two possibilities for order reduction:

1) Reduction of matching zeros and poles;
2) Applying a balancing reduction technique, e.g. [21].

The first method may be invoked on each step of iteration when the order α is small to improve performance. The second method can be applied to the resulting approximation.

We conclude this section by noting the similarities in the approaches to realization of the fractional transfer function in (6) found in this work and in [14], [22]. Also, a similar implementation can be found in [12].

Therefore, it is possible to obtain the fractional differentiator/integrator approximations in the desired frequency range $\omega = [\omega_z; \omega_p]$ by selecting $b = \frac{1}{\omega_z}$, $a = \frac{1}{\omega_p}$ and using the following equation:

$$s^{\alpha} \approx \omega_z^{\alpha} \hat{G}(s), \tag{15}$$

where $\alpha > 0$ corresponds to a fractional-order differentiator, $\alpha < 0$ corresponds to a fractional-order integrator and $\hat{G}(s)$ is the approximation obtained using the above algorithm.

C. Realization in MATLAB

Hereafter, we provide a description of the function, in which the algorithm is implemented. The function is part of the FOMCON toolbox [23]. The calling sequence is the following:

```
[G,J,err]=fpzp_new(b,a,alpha,N,w,retol)
```

Input arguments:

- b, a, and α — parameters from (6);
- N — desired approximation order (default $N = 2$);
- ω — frequency range in rad/s, used for approximation validation (default $\omega = [0.0001; 10000]$ rad/s);
- $retol$ — matching pole-zero pair reduction tolerance (empty by default).

Output arguments:

- G — the resulting integer-order approximation;
- J — error index, used to assess approximation quality, computed in the following way

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

  where n_{ω} is the number of frequencies in ω, $G(j\omega_i)$ is the response of the original fractional transfer function at frequency ω_i and $\hat{G}(j\omega_i)$ is the response of the obtained approximation at the same frequency;
- err — order error, the fraction decomposition residue of the fractional order α.

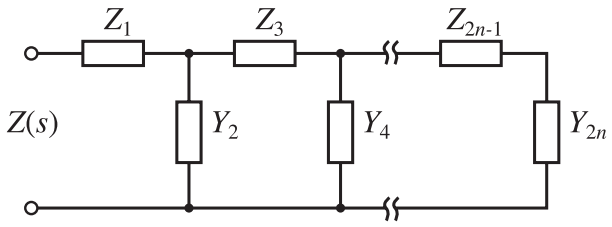

Fig. 6. Implementation of a fractance element by a finite ladder circuit

IV. Analog and Digital Realization of Fractional-order Systems

Once a continuous-time integer-order approximation of the implicit fractional transfer function is obtained using the method described above, one needs to use a suitable implementation technique. In this section, we investigate two possibilities.

A. Analog Implementation

An electronic circuit that exhibits fractional behavior is called a fractance [1], [24]. Some applications of fractance elements are summarized in [25], [26].

One way of implementing fractances is by means of finite ladder circuits (Fig. 6). To obtain the values of the components comprising the circuit one may consider the following relationship between the circuit impedance $Z(s)$ and the impedances $Z_k(s)$ and admittances $Y_{k+1}(s)$ found in the continued fraction expansion (CFE) of the corresponding transfer function [11].

$$\begin{aligned} Z(s) &= Z_1(s) + \cfrac{1}{Y_2(s) + \cfrac{1}{Z_3(s) + \cfrac{1}{Y_4(s) + \cfrac{1}{\ddots}}}} \\ &= Z_1(s) + \frac{1}{Y_2(s)+}\,\frac{1}{Z_3(s)+}\,\frac{1}{Y_4(s)+}\cdots . \end{aligned} \tag{16}$$

Remark 1. The continued fraction expansion of $Z(s)$ may lead to negative impedances to appear in the expansion. It is possible to realize such negative impedances using a method described in [11], i.e. by using negative impedance converters.

A function for obtaining the continued fraction representation of a transfer function is implemented in FOMCON and has the following calling sequence:

```
[c] = polycfe(b,a)
```

Input arguments:

- b, a — vectors of polynomial coefficients such that

$$G(s) = \frac{b_m s^m + \cdots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0}.$$

Output arguments:

- c — cell array of strings containing expressions for $Z_1(s)$, $Y_2(s)$, $Z_3(s)$, ... in (16).

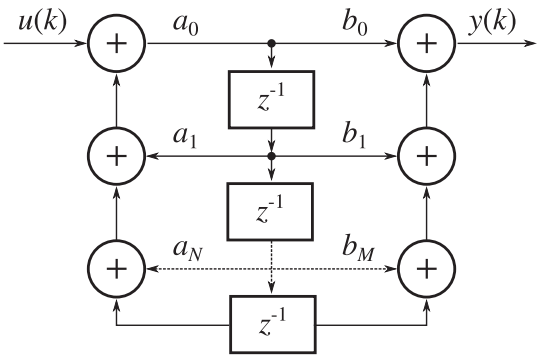

Fig. 7. Canonical form of the IIR filter

B. Digital Implementation

For the digital implementation it is possible to use a discrete approximation of the continuous transfer function in form of a IIR filter described by the difference equation:

$$y[n] = \sum_{j=0}^{N} b_j \cdot x[n-j] - \sum_{i=1}^{M} a_i \cdot y[n-i], \tag{17}$$

which corresponds to a discrete transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{j=0}^{M} b_j \cdot z^{-j}}{1 + \sum_{i=1}^{N} a_i \cdot z^{-i}}. \tag{18}$$

The IIR filter may be implemented in a suitable way, e.g. in canonical form [27] shown in Fig. 7.

Thus, the discrete implementation comprises the following steps:

- Find the approximation of the fractional transfer function proposed in Section III;
- Use a suitable discretization method to arrive at the representation in (18);
- If necessary, consider the limitations of the target hardware used for digital implementation of the fractional-order system.

V. Examples

A. Example 1

We shall obtain an approximation for the following implicit transfer function:

$$G_1(s) = \left(\frac{0.137s + 1}{15.294s + 1}\right)^{-0.25}.$$

In order to do this, the following MATLAB command could be used:

```
[G1,J,err] = fpzp_new(0.137, 15.294, ...
                      -0.25,  2)
```

The resulting performance index is $J = 1.4887 \cdot 10^{-4}$ and order error is $\epsilon = 0.0$ since we have $-0.25 = -1/4$. The comparison of the ideal response and the response of the obtained approximation is given in Fig. 8.

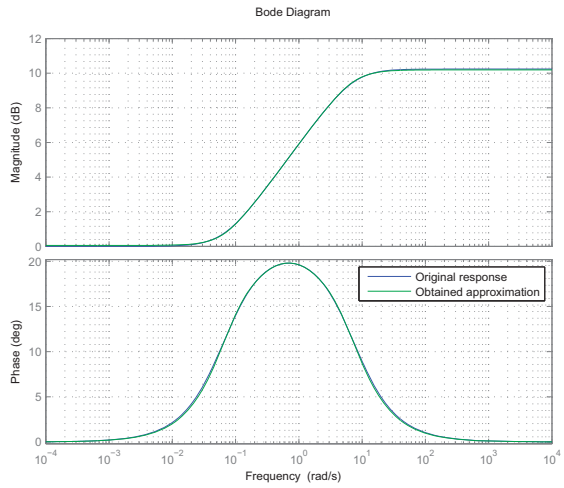

Fig. 8. $\hat{G}_1(s)$ approximation frequency response vs. $G_1(s)$ ideal frequency response

Next, we derive an analog implementation of the resulting transfer function. The following commands are executed in MATLAB:

```
G1s   = tf(balred(G1, 4));
[b,a] = tfdata(G1s, 'v');
c     = polycfe(a, b);
```

The order of the system is reduced to 4. The following continued fraction expansion is obtained:

$$Z(s) = \cfrac{1}{0.30934+\cfrac{1}{1.9067s+\cfrac{1}{0.22832+\cfrac{1}{5.3469s+\cfrac{1}{0.20264+\cfrac{1}{14.8389s+\cfrac{1}{0.16035+\cfrac{1}{62.7838s+\cfrac{1}{0.093498}}}}}}}}}.$$

Based upon this expansion it is possible to construct a RL-type network. We have assembled this network in LTspice IV. The resulting schematic is given in Fig. 9.

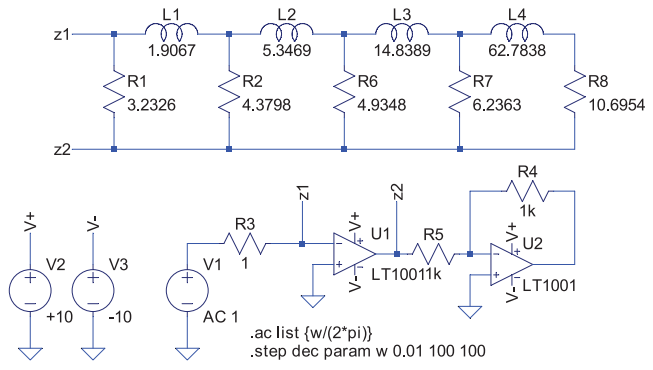

Fig. 9. LTspice IV diagram of the RL network

The results of a frequency sweep simulation conducted to determine the obtained network frequency characteristics are given in Fig. 10. It can be seen, that the frequency

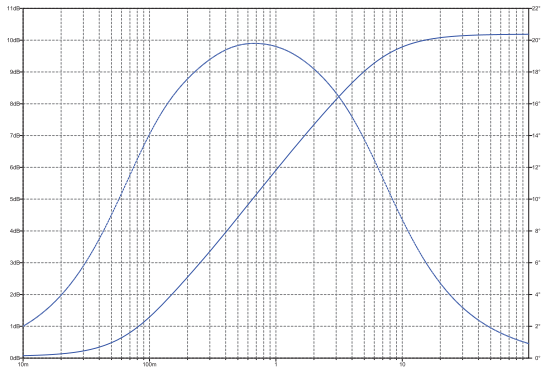

Fig. 10. RL network frequency characteristics

characteristics correspond to the ones obtained in MATLAB within reasonable tolerance.

Note, that we regard this example as a proof of concept. It may be difficult to implement some of the components of the analog network. A heuristic method for determination of suitable equivalent circuits may be employed. This is a subject for further study.

B. Example 2

In this example we will implement a digital fractional lead compensator, discussed in [20]. Consider a transfer function that describes a position servo:

$$G_2(s) = \frac{1.4}{s(0.7s+1)}\mathrm{e}^{-0.05s}.$$

Based on some performance specifications (phase margin $\varphi_m = 80°$ and gain crossover frequency $\omega_{cg} = 2.2$ rad/s), the controller was proposed such that

$$C_1(s) = \left(\frac{2.0161s+1}{0.0015s+1}\right)^{0.702}.$$

In order to obtain a rational approximation of this controller, the following MATLAB commands can be employed:

```
C1 = fpzp_new(2.0161,0.0015,0.702,3);
```

The resulting performance index $J = 0.8115$ and error is $\epsilon = 0.002$. However, the order of the approximated model is 328. Applying the `minreal()` function results in a system of 56th-order. One may use the balancing reduction technique with the MATLAB function `balred()` of the Control System toolbox in the following manner:

```
C1 = balred(C1, 5);
```

The performance index is now $J = 0.8289$, and the reduced model of order 5 is still a very good approximation of the fractional transfer function. The resulting control system open-loop frequency response $C_1(j\omega)G_2(j\omega)$ is shown in Fig. 11. It can be seen, that the design specifications are correctly fulfilled.

In [28] we have successfully realized this controller using a prototype board working in real time on a prototyping platform using the method described in Section IV. In this

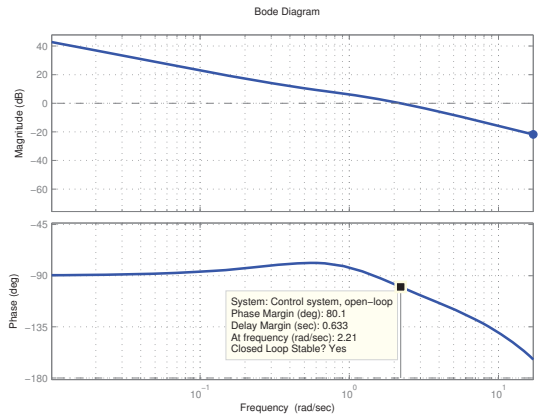

Fig. 11. Control system open-loop frequency response

work, however, we consider the implementation of the IIR filter in second-order sections, that is

$$C(z) = b_0 \prod_{k=1}^{N} \frac{1 + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} + a_2 z^{-2}}.$$

In MATLAB we issue the following set of commands to obtain the discretization of the approximated controller and arrive at the second-order section form:

```
Z1 = c2d(C1, 0.01, ...
     c2dOptions( ...
     'Method','tustin', ...
     'PrewarpFrequency',2.2));
[z,p,k] = zpkdata(Z1,'v');
[sos,g] = zp2sos(z,p,k);
```

In Fig. 12 the comparison of simulations of the control system is shown. The set point is set to $SP = 5$ and it changes on the 20th second of the simulation to 2.5. The solid line represents the results of software simulation while the dashed line shows the hardware-in-loop simulation, when the plant model was implemented in Simulink and was connected to the digital controller by means of a DAQ board. The result obtained is comparable to that in [20], where a real plant was used, while the controller was implemented in MATLAB.

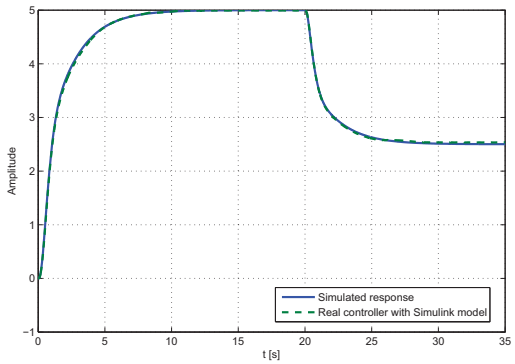

Fig. 12. Simulation in MATLAB vs. hardware-in-loop simulation of the position servo controller

VI. Discussion

In the following, we list the limitations of the proposed approximation method.

- Limited resolution, currently fixed at $1/10$;
- Unequal distribution of computational complexity with different orders;
- High order of the resulting approximation, which may not be practical in certain situations.

It is also important to consider other existing methods, that can be applied to the same approximation problem, for example [14], [22].

As for the realization methods, the following should be considered:

- An analog implementation with feasible component values should be sought; a heuristic algorithm may be employed to construct a suitable circuit that possesses the given impedance $Z(s)$.
- Further study of efficient digital implementation should be conducted. For example, we have had some trouble due to the limited sample resolution of the used hardware.

VII. Conclusions

In this paper, we presented a method, derived from the Newton root-finding method, for approximating a first-order implicit fractional-order transfer function. We have shown, that although the method has limitations, it can be applied to solving a class of modeling and control problems. Further research should be devoted to derivation of an alternative iteration formula so that the expensive operations of taking an nth power for fractional orders $1/n$ could be avoided, thus enhancing the resolution of the approximated order. It is also natural to expect, that this method, given correct treatment, could be applied to approximation of more complex systems.

We have also presented a general technique to derive an nth order Householder method fit for solving the same task which could be useful in some cases.

Finally, we have illustrated the use of the second-order method in analog and digital implementations of fractional-order systems and controllers.

References

[1] I. Podlubny, *Fractional differential equations*, ser. Mathematics in science and engineering. Academic Press, 1999.

[2] I. Podlubny, L. Dorčák, 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.

[3] I. Podlubny, "Fractional-order systems and PI$^{\lambda}$D$^{\mu}$-controllers," *IEEE Trans. Autom. Control*, vol. 44, no. 1, pp. 208–214, 1999.

[4] 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.

[5] B. M. Vinagre, I. Podlubny, A. Hernández, and V. Feliu, "Some approximations of fractional order operators used in control theory and applications," *Fractional Calculus & Applied Analysis*, vol. 3, pp. 945–950, 2000.

[6] A. Tepljakov, E. Petlenkov, and J. Belikov, "Application of the Newton method to first-order implicit fractional transfer function approximation," in *Proc. 19th Int Mixed Design of Integrated Circuits and Systems (MIXDES) Conference*, A. Napieralski, Ed., 2012, pp. 473–477.

[7] F. Hildebrand, *Introduction to numerical analysis*, ser. International series in pure and applied mathematics. McGraw-Hill, 1956.

[8] H. S. Wall, "A modification of Newton's method," *The American Mathematical Monthly*, vol. 55, no. 2, pp. 90–94, 1948.
[9] J. F. Traub, "Comparison of iterative methods for the calculation of nth roots," *Commun. ACM*, vol. 4, pp. 143–145, March 1961.
[10] G. Carlson and C. Halijak, "Approximation of fractional capacitors $(1/s)^{1/n}$ by a regular Newton process," *IEEE Trans. Circuit Theory*, vol. 11, no. 2, pp. 210–213, 1964.
[11] I. Podlubny, I. Petráš, B. M. Vinagre, P. O'Leary, and L. Dorčák, "Analogue realizations of fractional-order controllers," *Nonlinear Dynamics*, vol. 29, pp. 281–296, 2002.
[12] D. Valério. (2005) Toolbox ninteger for MatLab, v. 2.3. [Online]. Available: http://web.ist.utl.pt/duarte.valerio/ninteger/ninteger.htm
[13] H. Peitgen, H. Jürgens, and D. Saupe, *Chaos and fractals: new frontiers of science*. Springer-Verlag, 1992.
[14] A. Charef, H. H. Sun, Y. Y. Tsao, and B. Onaral, "Fractal system as represented by singularity function," *IEEE Trans. Autom. Control*, vol. 37, no. 9, pp. 1465–1470, 1992.
[15] A. Oustaloup, J. Sabatier, and P. Lanusse, "From fractal robustness to the CRONE control," *Fractionnal Calculus and Applied Analysis*, vol. 2, no. 1, pp. 1–30, 1999.
[16] H. Bensoudane, C. Gentil, and M. Neveu, "The local fractional derivative of fractal curves," in *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, ser. SITIS'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 422–429.
[17] A. Householder, *The numerical treatment of a single nonlinear equation*, ser. International series in pure and applied mathematics. McGraw-Hill, 1970.
[18] C. A. Monje, B. M. Vinagre, A. J. Calderon, V. Feliu, and Y. Q. Chen, "Auto-tuning of fractional lead-lag compensators," in *Proceedings of the 16th IFAC World Congress*, 2005.
[19] C. Monje, B. Vinagre, V. Feliu, and Y. Chen, "Tuning and auto-tuning of fractional order controllers for industry applications," *Control Engineering Practice*, vol. 16, no. 7, pp. 798–812, 2008.
[20] 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.
[21] A. Varga, "Balancing free square-root algorithm for computing singular perturbation approximations," in *Proc. 30th IEEE Conf. Decision and Control*, 1991, pp. 1062–1065.
[22] A. Oustaloup, F. Levron, B. Mathieu, and F. M. Nanot, "Frequency-band complex noninteger differentiator: characterization and synthesis," *IEEE Trans. Circuits Syst. I*, vol. 47, no. 1, pp. 25–39, 2000.
[23] A. Tepljakov, E. Petlenkov, and J. Belikov, "FOMCON: Fractional-order modeling and control toolbox for MATLAB," in *Proc. 18th Int Mixed Design of Integrated Circuits and Systems (MIXDES) Conference*, A. Napieralski, Ed., 2011, pp. 684–689.
[24] I. Podlubny, "Geometric and physical interpretation of fractional integration and fractional differentiation," *Fractional Calculus & Applied Analysis*, vol. 5, no. 4, pp. 367–386, 2002.
[25] S. Roy, "On the realization of a constant-argument immittance or fractional operator," *Circuit Theory, IEEE Transactions on*, vol. 14, no. 3, pp. 264–274, september 1967.
[26] B. Maundy, A. Elwakil, and S. Gift, "On a multivibrator that employs a fractional capacitor," *Analog Integr. Circuits Signal Process.*, vol. 62, no. 1, pp. 99–103, 2010.
[27] Y. Q. Chen, I. Petráš, and D. Xue, "Fractional order control - a tutorial," in *Proc. ACC '09. American Control Conference*, 2009, pp. 1397–1411.
[28] A. Tepljakov, E. Petlenkov, and J. Belikov, "Implementation and real-time simulation of a fractional-order controller using a MATLAB based prototyping platform," in *Proc. 13th Biennial Baltic Electronics Conference*, 2012, pp. 145–148.

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 Computer 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 was born in 1979. He received 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 received his B.Sc degree in mathematics from Tallinn University, and his M.Sc and PhD degrees in computer and systems engineering from Tallinn University of Technology. At present he holds the positions of researcher in the Institute of Cybernetics and Associate Professor in the Department at Computer Control of Tallinn University of Technology. His main research interests lie in the domain of nonlinear control theory.