European Union
European Social Fund

Investing in your future

tiigriülikool

Embedded System Implementation of Digital Fractional Filter Approximations for Control Applications

Aleksei Tepljakov, Eduard Petlenkov, Juri Belikov

June 21, 2014

TALLINN UNIVERSITY OF TECHNOLOGY

Motivation, contribution, and outline

- Our long-term goal is to design and build a fractional-order PID-type controller capable of efficient and reliable self-tuning for robust industrial process control.
- Here we treat the problem of fractional-order digital filter synthesis for embedded control applications. Summary of the contribution is as follows:
    - We use the well-established Oustaloup filter method for obtaining continuous-time zeros and poles and apply a discrete-time transformation yielding the digital filter.
    - Further, the zero-pole representation is converted to the second-order section form to ensure computational stability of the resulting filter.
    - We then implement the fractional-order PID controller software for the Atmel AVR ATmega8A microcontroller, and test it using a MATLAB/Simulink based real-time simulation platform.
- We consider a particular plant as an example—a laboratory system comprising a set of coupled tanks.

Fractional Calculus: The generalized operator

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

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

where generally it is assumed that $\alpha \in \mathbb{R}$, but it may also be a complex number. We restrict our attention to the former case.

Fractional Calculus: Laplace transform

Assuming zero initial conditions, the Laplace transform of the fractional derivative with $\alpha \in \mathbb{R}^+$ is given by

$$\int_0^\infty \mathrm{e}^{-st} {}_0\mathscr{D}_t^\alpha f(t)\mathrm{d}t = s^\alpha F(s), \tag{2}$$

where $s = \sigma + i\omega$ is the Laplace transform variable and $F(s)$ is the Laplace image of $f(t)$. Thus a fractional-order differential equation can be expressed in transfer function form in the Laplace domain as follows

$$G(s) = \frac{b_m s^{\beta_m} + b_{m-1} s^{\beta_{m-1}} + \cdots + b_0 s^{\beta_0}}{a_n s^{\alpha_n} + a_{n-1} s^{\alpha_{n-1}} + \cdots + a_0 s^{\alpha_0}}. \tag{3}$$

Fractional-order Controllers

The fractional PI$^\lambda$D$^\mu$ controller, where λ and μ denote the orders of the integral and differential components, respectively, is given by

$$C(s) = K_p + \frac{K_i}{s^\lambda} + K_d \cdot s^\mu. \tag{4}$$

The transfer function, corresponding to the fractional lead-lag compensator of order α, has the following form:

$$C_L(s) = K \left(\frac{1+bs}{1+as} \right)^\alpha . \tag{5}$$

When $\alpha > 0$ we have the fractional zero and pole frequencies $\omega_z = 1/b$, $\omega_h = 1/a$ and the transfer function in (5) corresponds to a fractional lead compensator. For $\alpha < 0$, a fractional lag compensator is obtained.

Original Oustaloup method: Brief summary

Given the approximation frequency range $[\omega_b, \omega_h]$ rad/s, order of approximation $\nu \in \mathbb{Z}^+$ and fractional power $\alpha \in [-1, 1] \subset \mathbb{R}$, we proceed to compute $(2\nu + 1)$ zeros and $(2\nu + 1)$ poles of the filter as

$$\omega'_k = \omega_b \theta^{\frac{(k+\nu+0.5-0.5\alpha)}{2\nu+1}}, \quad \omega_k = \omega_b \theta^{\frac{(k+\nu+0.5+0.5\alpha)}{2\nu+1}}, \tag{6}$$

where $k = \{-\nu, -\nu + 1, \ldots, 0, \ldots, \nu - 1, \nu\}$ and $\theta = \omega_h/\omega_b$. Thus the continuous recursive Oustaloup filter transfer function, approximating an operator s^α, is obtained in the form

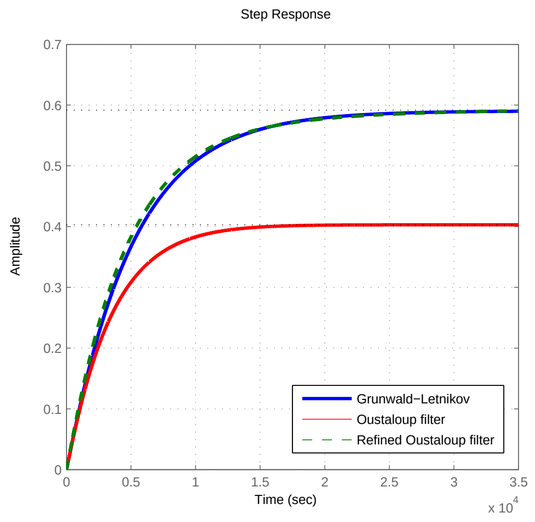
$$\hat{G}(s) = \omega_h^\alpha \frac{(s - \omega'_{-\nu})(s - \omega'_{-\nu+1}) \cdots (s - \omega'_\nu)}{(s - \omega_{-\nu})(s - \omega_{-\nu+1}) \cdots (s - \omega_\nu)}. \tag{7}$$

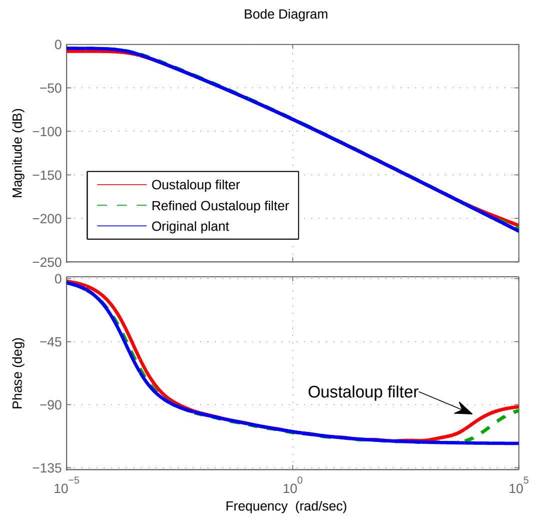
Oustaloup filter approximation example

Consider a fractional-order transfer function

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

and approximation parameters $\omega = [10^{-4}; 10^{4}]$, $N = 5$.





Discrete-time Oustaloup approximation

Suppose that we are given a sampling interval $T_s \in \mathbb{R}^+$. Then we may set the higher frequency bound of approximation (6) to

$$\omega_h = 2/T_s.$$

Consider the zero-pole matching equivalents method for obtaining a discrete-time equivalent of a continuous time transfer function. The following mapping is used:

$$z = \mathrm{e}^{sT_s}. \tag{8}$$

Therefore, for each k in (6) we take

$$\sigma_k' = \mathrm{e}^{-T_s \omega_k'}, \quad \sigma_k = \mathrm{e}^{-T_s \omega_k}. \tag{9}$$

Discrete-time Oustaloup approximation (continued)

We now need to compute the gain K_u of the resulting discrete-time system at the central frequency $\omega_u = \sqrt{\omega_b \omega_h}$ by using

$$K_u = \left\| H(\mathrm{e}^{j\omega_u T_s}) \right\| . \tag{10}$$

We also know the correct gain at this frequency

$$K_s = \omega_u^\alpha . \tag{11}$$

So finally we obtain the gain of the system as

$$K_c = K_s / K_u . \tag{12}$$

The discrete-time system is thus described by a transfer function of the form

$$H(z) = K_c \frac{(z - \sigma'_{-\nu})(z - \sigma'_{-\nu+1}) \cdots (z - \sigma'_{\nu})}{(z - \sigma_{-\nu})(z - \sigma_{-\nu+1}) \cdots (z - \sigma_{\nu})} . \tag{13}$$

Considerations for embedded device implementation

We would like to address the problems associated with implementing the generation scheme described above on an embedded device, such as a microcontroller. We have to take the following into consideration:

- Performance limitations;
- Limited computational abilities;
- Potential memory size limitations.

The first item completely depends on the type of microprocessor (and potentially additional hardware computational units) used in the implementation. In what follows, we shall discuss issues related to computational stability and memory management.

Finding the gain of the zero-pole matching equivalents discrete-time approximation

We notice, that (10) involves computations with complex numbers. However, we can compute the absolute value of a particular factor $(z - \sigma)$ in (13) at the frequency ω_u as follows

$$\left\| \mathrm{e}^{j\omega_u T_s} - \sigma \right\| = \left\| \cos(\omega_u T_s) + j\sin(\omega_u T_s) - \sigma \right\| = \\ = \sqrt{1 - 2\sigma\cos(\omega_u T_s) - \sigma^2} \quad (14)$$

due to Euler's formula and basic trigonometric transformations. The gain of the system given by discrete-time zeros and poles in (9) may be computed by means of

$$K_u = \frac{\prod\limits_{k=-\nu}^{\nu} \left(1 - \sigma'_k\theta - (\sigma'_k)^2\right)^{0.5}}{\prod\limits_{k=-\nu}^{\nu} \left(1 - \sigma_k\theta - \sigma_k^2\right)^{0.5}}, \quad (15)$$

where $\theta = 2 \cdot \cos(\omega_u T_s)$ is constant and needs to be computed only once.

IIR filter implementation: Second-order sections

Consider the set of discrete-time zeros (poles), that we have obtained earlier

$$z = \{\sigma_{-\nu}, \sigma_{-\nu+1}, \ldots, \sigma_0, \ldots \sigma_\nu, \sigma_{\nu-1}, \sigma_\nu\}\,. \tag{16}$$

Due to the generation method (6) the set in (16) is an ordered set. We have $2\nu + 1$ zeros (poles), so there are $\nu + 1$ second-order sections (including a single first-order section). Therefore, we have the polynomial

$$h(z) \quad = \quad (1 - \sigma_\nu z^{-1}) \cdot \prod_{k=0}^{\nu-1} \zeta(z^{-1}) \tag{17}$$

in the variable z, where $\zeta(z) = 1 + (c_k + d_k)\, z^{-1} + (c_k \cdot d_k)\, z^{-2}$, $c_k = -\sigma_{-\nu+2k}$ and $d_k = -\sigma_{-\nu+2k+1}$. So finally we arrive at the form

$$H(z) = K_c \prod_{k=1}^{\nu} \frac{1 + b_{0k} z^{-1} + b_{1k} z^{-2}}{1 + a_{0k} z^{-1} + a_{1k} z^{-2}}, \tag{18}$$

which can be effectively used as an IIR filter in control applications.

FOMCON toolbox for MATLAB: `doustafod(·)` function

Calling sequence [upcoming toolbox release]:

```
Z = doustafod(r,N,wb,wh,Ts)
```

Output arguments:

- Z — a Control System toolbox zero-pole-gain model.

Input arguments:

- r — order of the operator in s^r;
- N — order of approximation;
- ω_b, ω_h — lower and higher frequencies, respectively, which determine the approximation range;
- T_s — desired sampling interval.

Realization: Static memory allocation size requirements

We choose a sufficient maximal approximation order ν_{max}. Suppose that a floating-point datatype has a size of ψ bytes. First of all, to store arrays of values for discrete zero/pole calculation:

$$\text{Memory for zero/pole arrays} = 2\psi(2\nu_{max} + 1) \text{ bytes.} \tag{19}$$

Now we provide the memory requirements for second-order section coefficient storage:

$$\text{Memory for SOS arrays} = 4\psi\chi(\nu_{max} + 1) \text{ bytes,} \tag{20}$$

where χ is the number of approximated operators. Finally, for the digital signal processing application:

$$\text{Memory for DSP} = 2\psi\chi(\nu_{\max} + 1) \text{ bytes.}$$

The total amount of memory required for the arrays is thus

$$\text{Total memory} = 2\psi\left((3\chi + 2)\nu_{max} + 3\chi + 1\right) \text{ bytes.} \tag{21}$$

Digital realization of a fractional-order PID controller

We may digitally implement the fractional-order PID controller as

$$H_{PI^\lambda D^\mu}(z) = K_p + K_i H_I^{-\lambda}(z) + K_d H_D^{\mu}(z), \tag{22}$$

where K_p, K_i, and K_d are gains of the parallel form of the controller, $H_I^{\lambda}(z)$ corresponds to a discrete-time approximation of a fractional-order integrator of order λ and $H_D^{\mu}(z)$ corresponds to a discrete-time approximation of a fractional-order differentiator of order μ. In addition, we have

$$0 \leqslant \lambda, \mu \leqslant 1. \tag{23}$$

Note, that by choosing the appropriate frequencies of approximation we may also implement a fractional lead or lag compensator using this method.

Fractional-order integrator: Implementation considerations

We now address the issue of implementing the fractional-order integrator component. A continuous-time integrator of order λ has to be implemented as

$$G_I(s) = \frac{1}{s^\lambda} = \frac{s^{1-\lambda}}{s}$$

to ensure a nice control effect at lower frequencies. Its discrete-time equivalent is given by

$$H_I(z) = H^{1-\lambda}(z) \cdot H_I(z), \tag{24}$$

where $H^{1-\lambda}(z)$ is computed using the method presented above, and

$$H_I(z) = \frac{T_s}{(1 - z^{-1})} \tag{25}$$

is a simple discrete-time integrator.

PI$^\mu$D$^\mu$ controller operation: IIR filter reset logic

Finally, we address the state reset logic for the IIR filters and the integrator in (25). Denote by $e(k)$ the kth sample of the error signal $e(\cdot)$. We propose the following filter memory reset logic based on the notion of a maximal error change rate margin ρ. The reset condition is expressed as follows

$$\big|e(k) - e(k-1)\big| > \rho. \tag{26}$$

Thus, if the controller detects a sudden change in the error signal, IIR filter and integer-order integrator memory will be cleared, yielding zero initial conditions for the whole fractional-order PID controller. It is important to select the value of the margin ρ well above measurement noise or potential disturbance level.

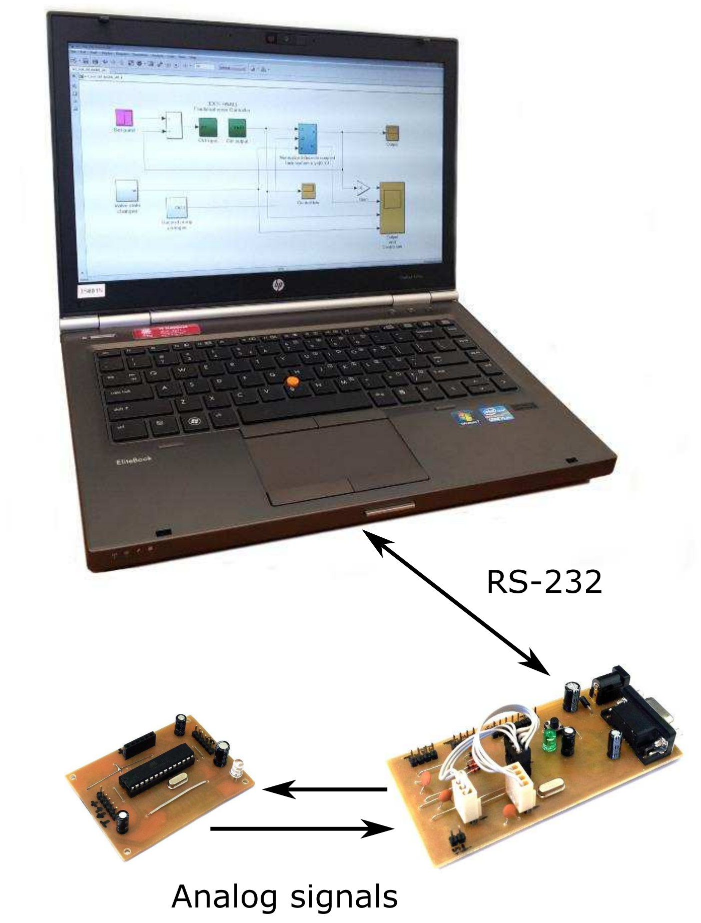
Choosing a microcontroller for the implementation: Atmel AVR 8-bit family

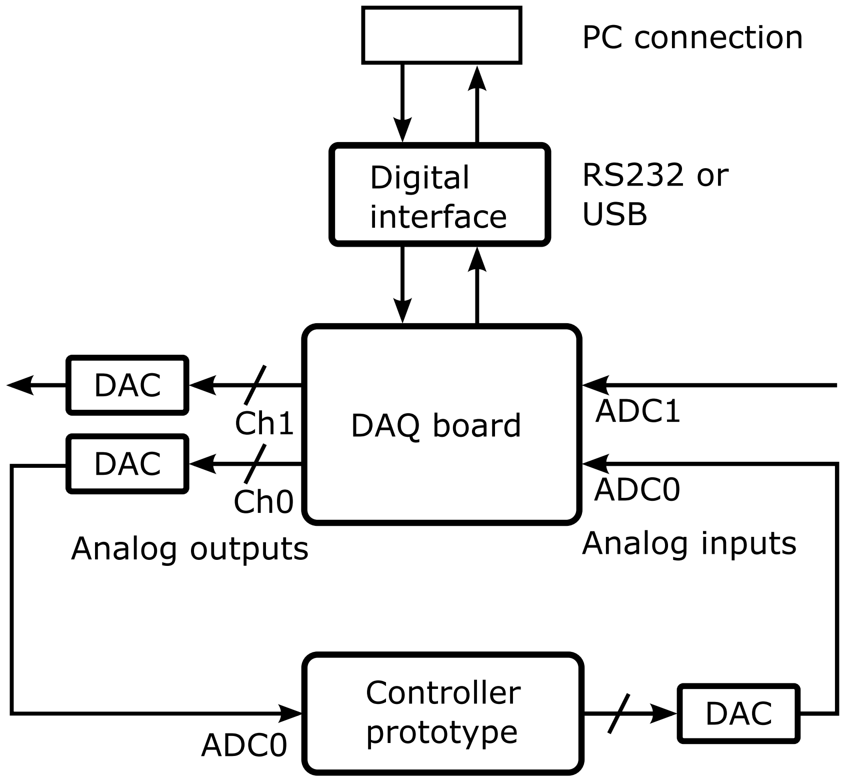
Our choice for the implementation of the fractional-order PID controller prototype in this work is the Atmel AVR ATmega8A microcontroller. This is due to several reasons:

- The microcontroller technology in question is stable, reliable and thoroughly tested since its introduction.
- This microcontroller family is very popular, and has been used in numerous applications in the industry.
- The chip itself is inexpensive and widespread.

We study the performance of this microcontroller for the fractional-order PID controller generation as well as DSP functions. Additionally, we employ external A/D and D/A converters in our FOPID controller prototype to bypass some DAQ hardware limitaions of the microcontroller yielding a system with 12 bit sample resolution.

Experiments with controller implementation: Hardware platform





Control experiment: Coupled tanks experiment

The system is modeled in continuous time in the following way:

$$\dot{x}_1 = \frac{1}{A}u_1 - d_{12} - w_1 c_1 \sqrt{x_1}, \quad (27)$$

$$\dot{x}_2 = \frac{1}{A}u_2 + d_{12} - w_2 c_2 \sqrt{x_2},$$

where x_1 and x_2 are levels of fluid, A is the cross section of both tanks; c_1, c_2, and c_{12} are flow coefficients, u_1 and u_2 are pump powers; valves are denoted by $w_i : w_i \in \{0, 1\}$ and

$$d_{12} = w_{12} \cdot c_{12} \cdot \mathrm{sign}(x_1 - x_2)\sqrt{|x_1 - x_2|}.$$

Control experiment: Coupled tanks experiment (continued)

In our experiments a model of a laboratory plant—coupled fluid tanks—is running in Simulink and it is our task to control this plant by means of our external controller prototype using a DAQ board.

Further, we provide performance evaluation figures based on the time it takes for the microcontroller to compute a particular FOPID approximation and to do a single sample computation. The microcontroller is clocked at 16MHz and is running firmware written in C-language and compiled with AVR-GCC with optimization level "O1". In this experiment, we generate a FOPID with the following parameters:

$$K_p = 6.9514, \quad K_i = 0.13522, \quad K_d = -0.99874, \\ \lambda = 0.93187, \quad \mu = 0.29915. \tag{28}$$

Controller synthesis: Results

The suitable frequency range for an Oustaloup filter of order $\nu = 5$ is $\omega = [0.0001, 10]\,\mathrm{rad/s}$ with $\nu_{max} = 10$. The sampling interval is $T_s = 0.2\mathrm{s}$. Denote by τ_g and τ_s the time interval that is required for controller generation and sample computation, respectively, under the conditions above. We have the following per the report of AVR Simulator:

$$\tau_g = 28.629\,\mathrm{ms}, \quad \tau_s = 1.670\,\mathrm{ms}.$$

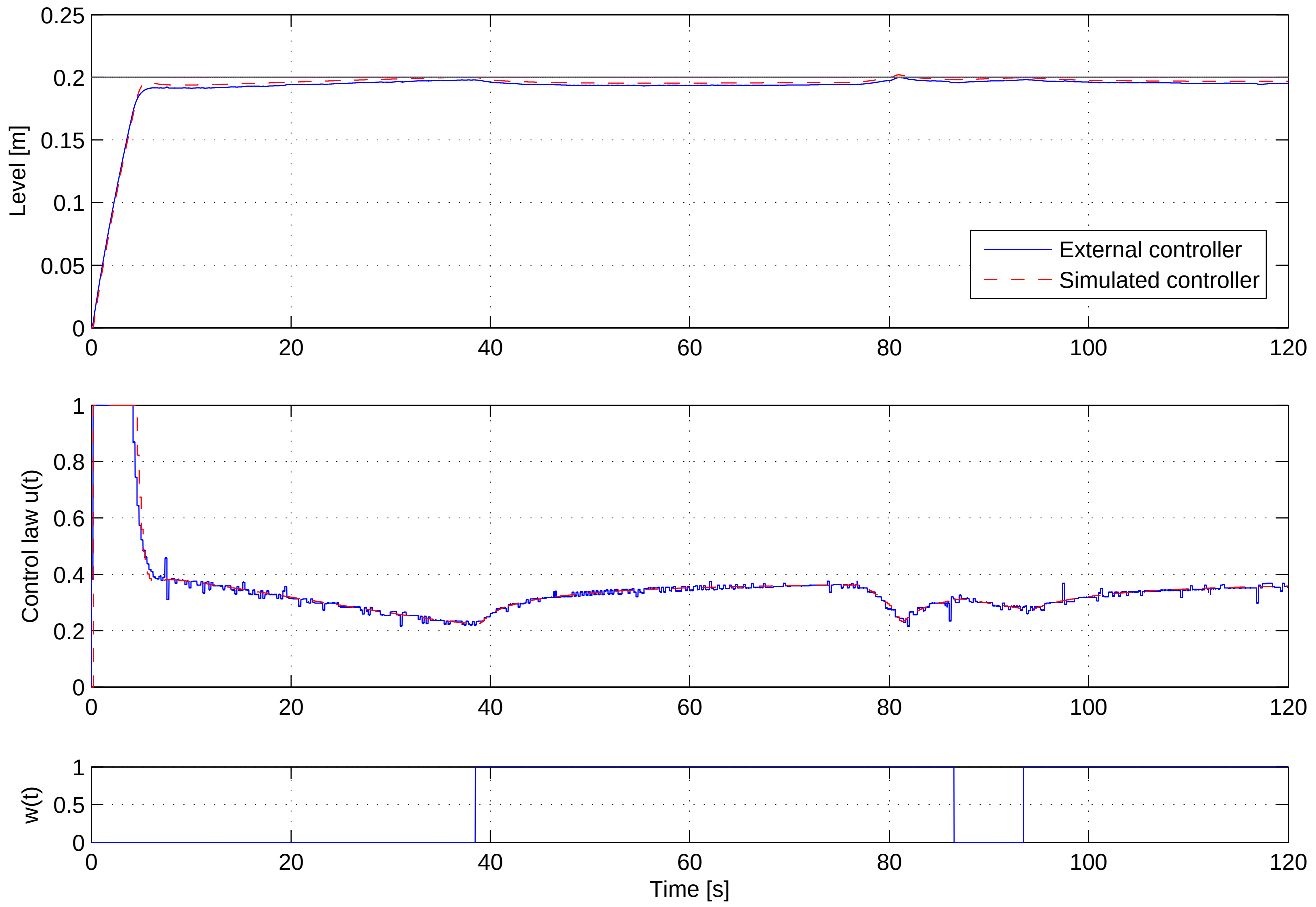
Thus, sampling rates up to $f_s = 500$ Hz are possible. Note, however, that it takes much longer to compute the controller. This should be considered when the controller, running in a closed loop, needs to be recomputed.

Controller synthesis: Performance

Table 1: Time requirements for controller generation and sample computation for different Oustaloup filter orders

ν	τ_g [ms]	τ_s [ms]	Max. applicable f_s [Hz]
6	32.2914	1.9868	480
7	37.1326	2.0832	450
8	42.2011	2.1796	425
9	46.8712	2.2759	400
10	51.5617	2.3723	400

Control experiment: External FOPID vs. Simulated FOPID



Conclusions

- In this paper we have touched upon the important topic of digital approximations of fractional-order differential and integral operators.
- A discrete-time implementation method, based on the Oustaloup recursive filter, was introduced.
- The implementation of a fractional-order PID controller was discussed.
- The method was successfully tested on an Atmel AVR ATmega8A microcontroller.
- Further research should cover automatic tuning opportunities and implementations thereof on other microcontroller families.

Acknowledgement

Supported by the Tiger University Program of the Information Technology Foundation for Education.

More information about HITSA can be found on the website:

`http://www.hitsa.ee/en/`

FOMCON project: Fractional-order Modeling and Control

FOMCON

- Official website: http://www.fomcon.net/
- Toolbox for MATLAB available;
- An interdisciplinary project supported by the Estonian Doctoral School in ICT and Estonian Science Foundation grant nr. 8738.

Discussion

Thank you for listening!

Aleksei Tepljakov

Engineer/PhD student at Alpha Control Lab, TUT

http://www.a-lab.ee/, http://www.ttu.ee/

aleksei.tepljakov@ttu.ee