Python Application for Analyzing Analog Filters' Transfer Function

Miljana Milić and Vančo Litovski
University of Niš, Faculty of Electronic Engineering,
Niš, Serbia
miljana.milic@elfak.ni.ac.rs, vanco.litovski@elfak.ni.ac.rs

***Abstract*— Analog filter represents an inevitable block in modern telecommunications and electronics. They suppress the undesired spectrum components, regardless of their nature. An interactive Python application, developed for analyzing the analog filters' transfer function is described in this paper. The application calculates and draws most of the filter's characteristics such as: step and impulse response of the filter in the time domain, and, module, phase and group delay in the frequency domain. The application accepts transfer function of the filter described both using poles and zeros of the function, or polynomials of the numerator and denominator as inputs. A Python programming language is used as a development platform, since it is free, and offers large available online support.**

Keywords- analog filters, circuit analysis, Python

I. INTRODUCTION

Almost every electronic device today has a filter. Filtering is an essential process in every data processing block. Its task is to suppress unwanted spectral components regardless of their origin.

The transfer function of the filter is often defined in domain of complex frequencies (s-domain). When designing and analyzing digital filters, calculations are performed in the z-domain.

The intention of this work is to develop an interactive software package that can analyze transfer functions of analog filters [1]. It involves the calculation and graphical representation of the module, phase and group delay of the filter in the frequency domain, as well as the response of the filter in the time domain to pulse and step stimuli. This can later be used for the extraction of the most important parameters of the function that's being analyzed. The application for these analyses accepts the analog filter transfer functions that can be described with its poles and zeros, or with the coefficients of the numerator's and denominator's polynomials. We are aware of the fact that there are many similar applications available on the internet. Nevertheless, in our experience, many of them do not show correct results of transfer function analysis.

Python is chosen as a programming platform for developing the application. It is a widely used programming language that supports object-oriented, imperative and functional programming styles. It is free and available for majority of operating systems and can be packed into stand-alone executable programs, allowing the distribution of Python-based software for use on those environments without requiring the installation of a Python interpreter. It has a large and wide-ranging standard library. As of October 2014, the Python Package Index [2], which is the official repository of the third-party software for Python, contains more than 49730 packages offering a variety of functionality, including:

- graphical user interfaces, web frameworks, multimedia, databases, networking and communications
- test frameworks, automation and web scraping, documentation tools, system administration
- scientific computing, data processing, image processing.

In our case there were two major reasons for choosing Python programming language. First, it is C/C++ compatible, and the second, it is also compatible with a platform for software radio development GNU [3].

The theoretical basics of analog filters analysis will be given first, as well as formulas for calculating their properties. After that, some most important features of the Python programming environment will be described, including all necessary packages. This is followed by description of the application development, results of filter analysis example and the conclusion.

II. FILTER'S TRANSFER FUNCTION AND ITS PROPERTIES

Analog filters that we intend to analyze here can be passive and active circuits. They meet the requirements of causality and stability. Their transfer functions are expressed as a rational function of two polynomials where the order of the numerator, *m*, is less than or equal to the order of the denominator *n*.

In this section, we will give basic definitions related to the transfer function of signal filters and formulas for calculating the characteristics that are extracted from them [4], [5], [6]. This analysis is performed in the frequency and in the time domain. Since the response of the filter in the time domain can be calculated only using the Residue Calculus, it is necessary to know poles and zeros of the filter transfer function. Similar requirements stand for the group delay and phase characteristics calculation. Namely, calculation of the filter's phase characteristic using only the coefficients of the

polynomials can cause erroneous results. Therefore, our application applies formulas that use only zeros and poles of the transfer function. It is assumed that polynomials of the transfer function are solved first, by the designer, or alternatively by this Python application.

The factorized form of the transfer function in the s-plane can be represented with:

$$T(s) = T_0 \cdot \frac{\prod_{k=1}^{m}(s - z_k)}{\prod_{i=1}^{n}(s - pi)}. \quad (1)$$

Here, s stands for the complex angular frequency: $s = \sigma + j\omega$. The imaginary part $\omega = \mathrm{Im}\{s\}$, is referred to as real angular frequency: $\omega = 2\pi f$, while f stands for the actual frequency, expressed in (Hz). Zeros of the transfer function are $z_k = \alpha_k + j\beta_k$, $k = 1, 2, ..., m$, while the poles are $p_i = \gamma_i + j\delta_i$, $i = 1, 2, ..., n$. where n represented the order of the filter and should satisfy that $n \geq m$. Using the formula:

$$T(0) = T_0 \cdot \frac{\prod_{k=1}^{m}(-z_k)}{\prod_{i=1}^{n}(-p_i)}, \quad (2)$$

one can calculate gain of the filter at zero angular frequency, while using:

$$T(j1) = T_0 \cdot \frac{\prod_{k=1}^{m}(j1 - z_k)}{\prod_{i=1}^{n}(j1 - p_i)}, \quad (3)$$

one can calculate gain of the filter at unit angular frequency. Since the angular frequency is usually normalized so that $\omega_{ctitical} = 1$, (for low pass filters this is usual bandwidth upper cutoff frequency, and for bandpass filters, this is usual bandwidth center frequency), value expressed by (3) is often of great importance.

On the axis of real frequencies, i.e. for s = jω, the amplitude characteristic is calculated as:

$$\left|T(s)\right|_{s=j\omega} = \left[T(s)T(-s)\big|_{s=j\omega}\right]^{1/2}, \quad (4)$$

or,

$$\left|T(s)\right|_{s=j\omega} = \left|T_0\right| \frac{\prod_{k=1}^{m}\left[\alpha_k^2 + (\omega - \beta_k)^2\right]^{1/2}}{\prod_{i=1}^{n}\left[\gamma_i^2 + (\omega - \delta_i)^2\right]^{1/2}}. \quad (5)$$

If presented in the *semilog* scale, an amplitude characteristic is calculated with (6):

$$A(\omega) = 20 \cdot \log\left\{\left|T(s)\right|_{s=j\omega}\right\} \text{ [dB]}, \quad (6)$$

while the attenuation can be calculated using:

$$a(\omega) = 20 \cdot \log\left\{\frac{1}{\left|T(s)\right|_{s=j\omega}}\right\} \text{ [dB]}. \quad (7)$$

It is extremely important that the phase characteristics are calculated using:

$$\varphi(\omega) = \arg\left\{T(s)_{|s=j\omega}\right\} = \frac{1}{2j}\ln\left[\frac{T(s)}{T(-s)}\right]_{s=j\omega} \quad (8)$$

or:

$$\varphi(\omega) = \sum_{k=1}^{m}\mathrm{arctg}\left[\frac{\beta_k - \omega}{\alpha_k}\right] - \sum_{i=1}^{n}\mathrm{arctg}\left[\frac{\delta_i - \omega}{\gamma_i}\right]. \quad (9)$$

During the *arctg* function calculation, for each value of the angular frequency, one should first determine the quadrant of the angle by observing the signs of the denominator and the numerator. After that it is possible to calculate the desired angle. For example: if signs of the numerator and the denominator are both negative, the observed angle lays in the third quadrant of the complex plane, so that *arctg* function can be calculated as:

$$\pi + \mathrm{arctg}\left(\left|numer./denom.\right|\right). \quad (10)$$

The group delay is calculated with the formula:

$$\tau(\omega) = -\frac{d\varphi(\omega)}{d\omega} = \sum_{k=1}^{m}\frac{\alpha_k}{\alpha_k^2 + (\omega - \beta_k)^2} - \sum_{i=1}^{n}\frac{\gamma_i}{\gamma_i^2 + (\omega - \delta_i)^2}. \quad (11)$$

The general form of the response in the time domain at the impulse function is shown in the following expression:

$$h(t) = \sum_{i=1}^{n}\lim_{s \to p_i}\frac{1}{(q-1)!}\left\{\frac{d^{q-1}}{ds^{q-1}}\left[(s - p_i)^q T(s) e^{st}\right]\right\} \quad (12a)$$

where q represents the order of the denominator's polynomial. If the transfer function has only simple poles, than we have:

$$h(t) = \sum_{i=1}^{n}\left(\lim_{s \to p_i}\left\{\left[(s - p_i)\cdot T(s)\cdot e^{st}\right]\right\}\right). \quad (12b)$$

Substituting (1) into (12b), we obtain:

$$h(t) = \sum_{i=1}^{n}\lim_{s \to p_i}\left\{T_0 \frac{\prod_{k=1}^{m}(s - z_k)}{\prod_{\substack{r=1 \\ r \neq i}}^{n}(s - p_r)} e^{st}\right\} =$$

$$= \sum_{i=1}^{n}\left\{T_0 \frac{\prod_{k=1}^{m}(p_i - z_k)}{\prod_{\substack{r=1 \\ r \neq i}}^{n}(p_i - p_r)}\left[\cos(\delta_i t) + j\cdot\sin(\delta_i t)\right] e^{\gamma_i t}\right\}, \quad (13)$$

where $p_i=\gamma_i+j\delta_i$ and $p_r=\gamma_r+j\delta_r$ represent vectors of poles (complex), while $z_k=\alpha_k+j\beta_k$ denominates vector of complex zeros. The order of the filter is n (equal to the number of poles), m is the order of the numerator's polynomial (the number of zeros); σ_i is the real part of the pole, ω_i is the imaginary part of the pole, α_k is a real part of the zero, β_k is the imaginary part of the zero.

The response to the step-function is obtained when the set of poles is extended with one pole at zero. The new function whose inverse Laplace transform is to be determined would be:

$$T_0 \frac{1}{s} \frac{\prod_{k=1}^{m}(s-z_k)}{\prod_{j=1}^{n}(s-p_j)}. \qquad (14)$$

In (13) the sum is increased by one, and set of poles is extended with: $p_{n+1}=0+j0$.

III. The Development of the Python Application

Python is a programming language that was developed in compliance with the OSI (Open Systems Interconnection) standards. It supports various programming styles. License for this programming language is opened (Open source) and it is free to use and distribute, even for commercial purposes. There are hundreds of software modules and packages for Python, available as standard libraries, or as those created by specialized groups and communities [7], [8].

For the development of our Python application, the Eclipse programming environment [9] was used. The main reasons for this choice are its availability and openness, and easy development and debugging of programs. For the development of GUI applications, we used the standard Python's Tkinter GUI package [10]. Other Python's GUIs are also available, like WX [11], or QT [12]. We have chosen Tkinter, because it is a Python's default GUI toolkit. It is also stabile, mature, well documented, and has strong binding mechanism.

For the development of application for the analysis of analog filters, it was necessary to use tree Python's package. The first allowed the use of certain functions and operations of the numerical mathematics - Numpy (Numerical Python) [13], second enables symbolical calculations - Sympy (Symbolical Python) [14], while the third allows displaying of calculation results using versatile graphs – matplotlib [15].

A brief description of the application and instructions for its use will be given next.

IV. Analog Filter Analysis with the Python Application

When the application **Transfer function properties** is launched, first the user is asked to enter two integers: the number of zeros and poles of the filter's transfer function. It is also possible to enter the transfer function of the filter in the alternative way, representing it with coefficients of the numerator's and denominators polynomials. This is shown in Fig. 1. After pressing the "Fetch" button, the application window is expanded, in order to enter simple zeros and poles of the function, as well as the initial gain of the filter. This is shown in Fig. 2.

Figure 1. Openning the application; input parameters

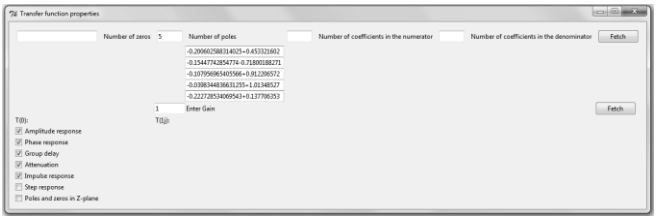

Figure 2. Extended application window for entering complex zeros and poles, and selecting commands

When represented with its polynomials' coefficients, the application automatically calculates their roots, considering them afterwards as sets of poles and zeros.

The coefficients for this filter example are taken from [16]. This is an example of one H-type critical-monotonic all-poles low-pass filter, of 10^{th} order. Table 1. shows those coefficients.

After pressing the "Fetch" button from the extended application window, the program first calculates typical gains of the filter. This is shown in Fig. 3.

Figure 3. Calculation of the filter unity gains

Figs. 4, 5, 6, and 7 give the results of the filter analysis in the frequency domain. Those characteristics are: the amplitude characteristic (in dB), phase characteristic (in rad), group delay (in sec), and the attenuation (in dB), respectively. At Fig. 7, a detailed view at the most important part of the attenuation characteristic is zoomed. The application offers that option as well as saving the graphs into corresponding .png files.

The responses of the filter in the time domain are shown in Fig. 8.

Finally, Fig. 9 shows the position of poles and zeros in the s-plane. Poles are marked with crosses, while zeros are marked with small circles. It can be noticed that this graph is symmetrical along the real axes.

V. Conclusion

This paper presents the development of application for the analysis of analog filters described with zeros, poles and gain, or with the coefficients of the numerator's and the denominator's polynomials of the transfer function. The developed applications successfully perform these analyzes. We are aware that many commercial tools support the same calculation and analysis. However, the important advantage of our solution is the availability of Python programming environment, and therefore the openness and availability of developed application. Further research and application development will be oriented towards the implementation of calculation of additional features of filters, as well as the additional controls in the application.

TABLE I. THE COEFFICIENTS OF THE DENOMINATOR POLYNOMIAL OF THE H FILTER

s^{10}	s^9	s^8	s^7	s^6	s^5	s^4	s^3	s^2	s^1	s^0
1.0	1.4512	3.5253	3.6016	4.1933	2.9624	1.9859	0.8937456	0.3222619	0.0712828	0.0078896

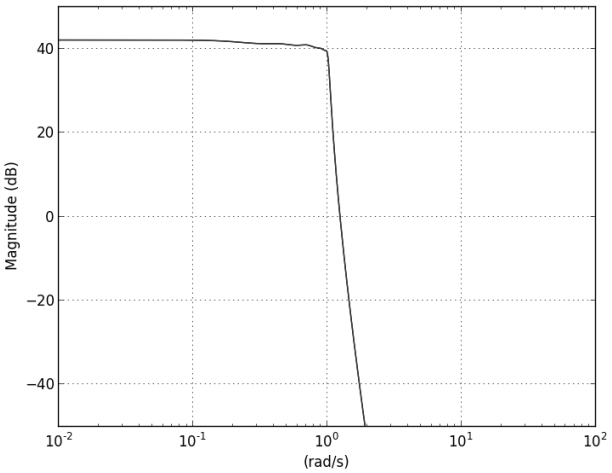

Figure 4. Amplitude characteristic of the filter

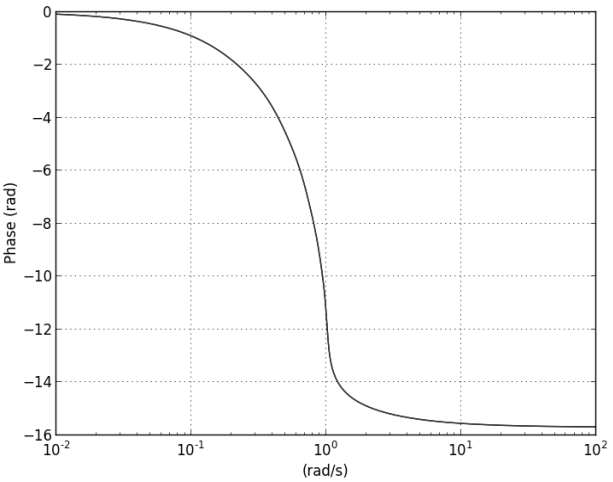

Figure 5. Phase characteristic of the filter

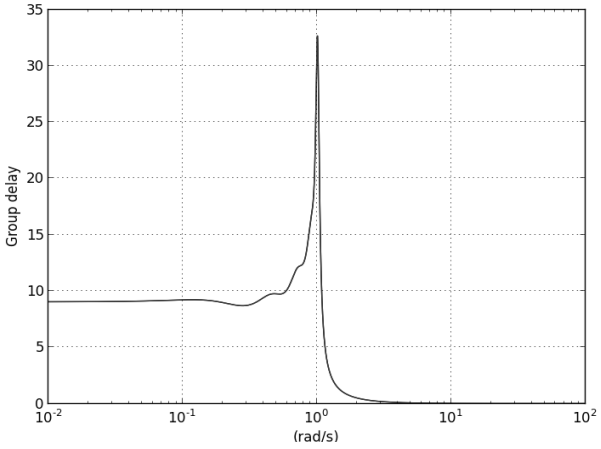

Figure 6. Group delay

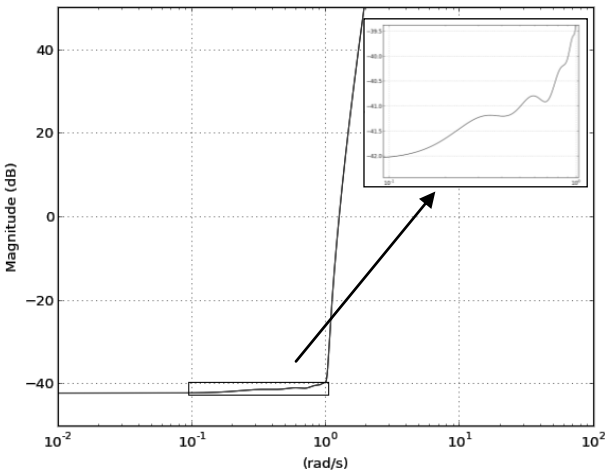

Figure 7. The attenuation

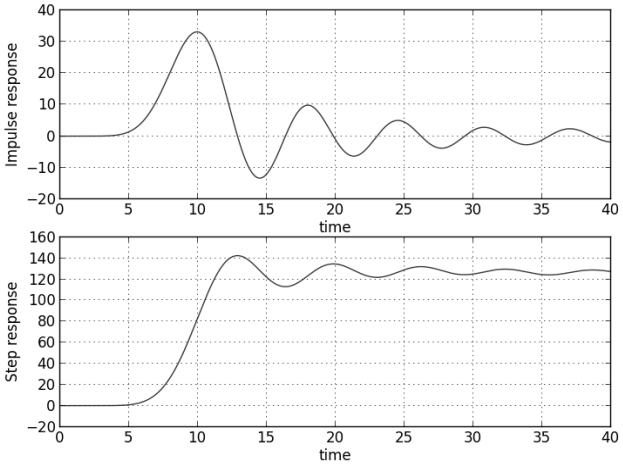

Figure 8. Impulse and Step response of the filter

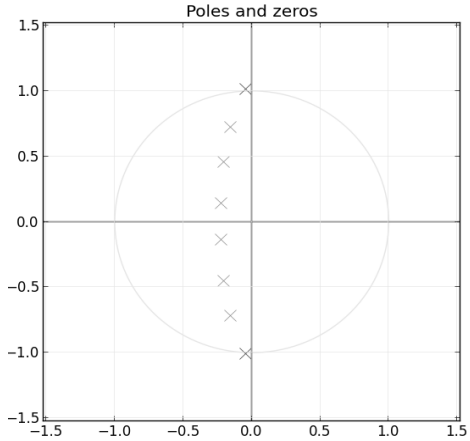

Figure 9. S-plane with all poles and zeros

ACKNOWLEDGMENT

This research was partly funded by The Ministry of Education, Science and and Technological Development of Republic of Serbia under the contract No. TR32004.

REFERENCES

[1] M. Milić and V. Litovski, "Analysis of analog filters' transfer function using Python programming language", Proc. of the LVIII ETRAN conf. Vrnjačka Banja, Serbia, 2014, EL.2.4.

[2] https://pypi.python.org/pypi

[3] http://gnuradio.org/

[4] M. Lutovac, D, Tošić, and B. Evans, Filter Design for Signal Processing Using MATLAB and Mathematica, Prentice Hall, NY, 2001.

[5] DeVerl Humpherys, The Analysis, design, and synthesis of electrical filters, Englewood Cliffs, Prentice-Hall, NY, 1970.

[6] V. Litovski and M. Zwolinski, VLSI Circuit Simulation and Optimization, Chapman and Hall, London, 1997.

[7] www.python.org

[8] www.pydev.org

[9] www.eclipse.org

[10] https://wikipython.org/moin/Tkinter

[11] http://wiki.wxpython.org/

[12] https://qt-project.org

[13] www.numpy.org

[14] www.sympy.org

[15] www.matplotlib.org

[16] D. Topisirović, V. Litovski, and M. A. Stošović, "Unified theory and state-variable implementation of critical-monotonic all-pole filters," Int. J. Circ. Theor. Appl. Wiley, 22 Oct, 2013.