COURSE SUBJECT: Computational Techniques (2023/2024)

STUDY COURSE: Electronics and Telecommunications (first level)

UNIVERSITY: AGH University of Science and Technology, Kraków, Poland

TOPIC: Evaluation and illustration of Butterworth lowpass filter's characteristics for orders from 1 to 5	
AUTHOR: Konrad Włodarczyk	**DATE OF SUBMISSION:** 03.01.2024
	SUPERVISED BY: Przemysław Korohoda, Dr.Eng.

ABSTRACT:

This project's aim is to delve into the realm of butterworth lowpass filter. This venture explores the obtaining, illustraiting and evaluating the filter using Matlab software environment (ver: MATLAB R2023a), emphasizing the key characteristics like finite impulse response, transfer function or frequency response of the filter, as well as illustrating the influence of the filter on a cosine signal 'fed' to it. Leveraging Matlab capabilites in sumulations and plotting tools I aim to provide a clear and comprehensive illustration of previously mentioned characteristics. The principal goals include:

- **Understanding the theoretical background behind the project,**
- **Implementation and visualization the theory in Matlab software environment.**

To reach my goals, I will make use of the Inverse Laplace Transform by partial fraction expansion in order to obtain the finite impulse response from the transfer function, as well as Continuous Time Fourier transform to acquire the frequency response of the filter. I will also apply the convolution of the input cosine signal and the finite impulse response of the filter in order to obtain the output signal going out of the filter. The expected results are that the characteristics of the transfer function obtained using the formula should be approximately (due to the approximation of the Fourier Transform) the same as the frequency response of the filter.

SECTION 1: THEORETICAL INTRODUCTION TO THE TOPIC

A Butterworth (named after engineer Stephen Butterworth) lowpass filter is an electronic filter designed in order to allow signals with frequencies below specified cutoff frequency (ω_c) to pass through, while attenuating the frequencies exceeding the cutoff frequency. It exhibits a flat frequency response in the passband, making it ideal for applications where a smooth and gradual transition between the passband and stopband is crucial.

TRANSFER FUNCTION OF THE FILTER

In the realm of signal processing the butterworth lowpass filter is characterized by its transfer function (H(s)) typically expressed in the Laplace domain. Formulas for transfer functions of the Butterworth lowpass filter for orders from 1 to 5:

ORDER:	FORMULA:
1	$\frac{1}{s+1}$
2	$\frac{1}{s^2+1.414214s+1}$
3	$\frac{1}{(s+1)(s^2+s+1)}$
4	$\frac{1}{(s^2+0.765367s+1)(s^2+1.847759s+1)}$
5	$\frac{1}{(s+1)(s^2+0.618034s+1)(s^2+1.847759s+1)}$

Generally, the normalized cutoff frequency of the Butterworth lowpass filter is 1, hence it is not mentioned in the formulas. In order to incorporate the cutoff frequency in case we want to change its value is to divide every 's' by the cutoff frequency: $\frac{s}{\omega c}$

The characteristics of the transfer functions:

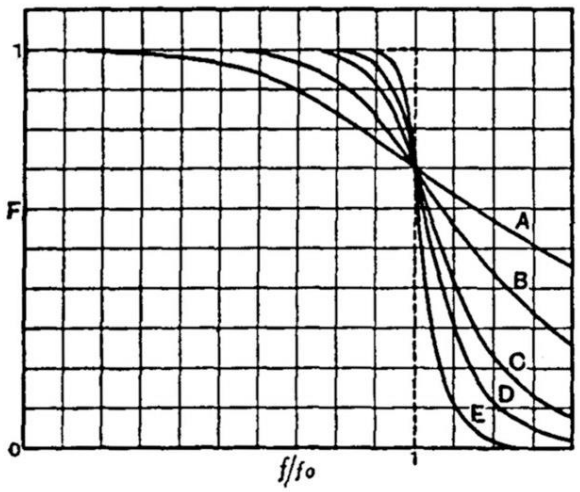


Fig. 3.

FINITE IMPULSE RESPONSE OF THE FILTER

In order to obtain the finite impulse response (h(t)) of the Butterworth lowpass filter we can make use of the Inverse Laplace Transform by partial fraction expansion, but due to the fact I am not able to provide the theoretical introduction for it in a way that is comprehensive enough to fit in a few lines, here is a link to an article exploring that part of the material:

https://lpsa.swarthmore.edu/LaplaceXform/InvLaplace/InvLaplaceXformPFE.html 02.01.2024

After obtaining the residues (r(n)) and poles (p(n)) of the transfer function we can calculate the Inverse Laplace Transform using following formula:

$$h(t) = r(n) \cdot e^{(p(n) \cdot t)} + r(n+1) \cdot e^{(p(n+1) \cdot t)} + \cdots \quad (1)$$

FREQUENCY RESPONSE OF THE FILTER

Next, as a means to obtain the Frequency Response of the filter (H(ω)) we can leverage the Continuous Time Fourier Transform. The formula for that in case of the Butterworth lowpass filter is:

$$H(\omega) = \int_{-\infty}^{\infty} h(t) \cdot e^{-j\omega t}\, dt \quad (2)$$

The formula assumes the integral to be indefinite, but in order to use it in Matlab software environment, we have to approximate the transform in a specified interval.

INFLUENCE OF THE FILTER ON GIVEN INPUT SIGNAL

If we introduce an input signal going into the filter (in our case: $x(t) = \cos(\omega c \cdot t)$), having the finite impulse response of the filter, we can use the Fourier Transform Convolution Property in order to obtain the signal going out of the filter (y(t)):

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau) d\tau \quad (3)$$

Or:

$$Y(\omega) = X(\omega) \cdot H(\omega)$$

And then performing the Inverse Fourier Transform, which I did not leverage.

For more in-depth description or in order to obtain the theory exceeding the one presented here You can visit:

https://en.wikipedia.org/wiki/Butterworth_filter 02.01.2024

SECTION 2: DESCRIPTION OF THE STRUCTURE OF PERFORMED COMPUTING

A) INPUT DATA FOR EXPERIMENTS

The input data for beginning my experiments is implementing the formulas for transfer functions (H_s(1:5)) provided in previous section, as well as the cutoff frequency (w_c) into the MATLAB script

B) OBTAINING THE FINITE IMPULSE RESPONSE OF THE FILTER

In the first step of computations i made use of the Inverse Laplace Transform by partial fraction expansion in order to obtain the Finite Impulse Response (h_t(1:5)) out of the Transfer Functions implemented before. To do that I made two custom functions: CustomResidue.m (which calculates the residues and poles of the transfer functions) and InverseLaplaceTransform.m (which uses obtained residues and poles and calculates the inverse laplace transform using the formula (1) from section 1).

C) EVALUATING THE INFLUENCE OF THE FILTER ON AN INPUT SIGNAL

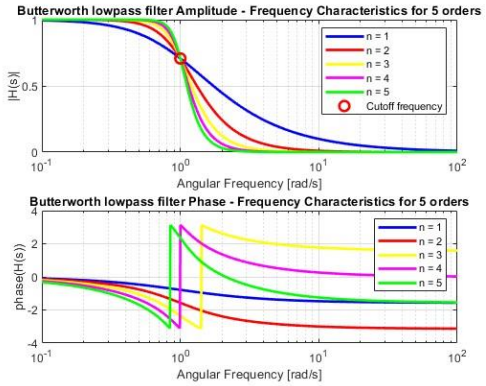
In the next step of my computations, I introduced an input cosine signal (x_t) with the amplitude A and the argument of ($\omega c \cdot t$). Then, I used the 'conv' (I did not leverage the formula (3) from section 1 since i found that not necessary) matlab function in order to get a convolution of the x_t input signal and h_t FIR of the filter to get the output signal (y_t).

D) OBTAINING THE FREQUENCY RESPONSE OF THE FILTER

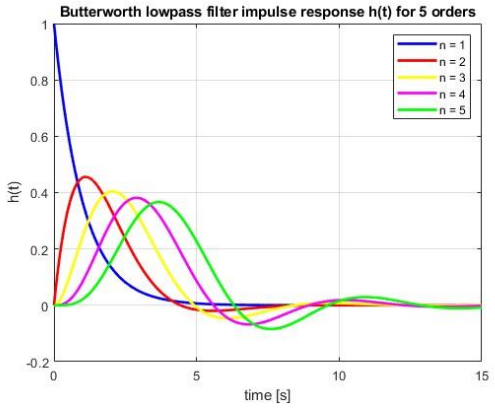
In the last stage of the computations, I determined the Frequency Response (H_w(1:5)) of the filter calculating the Continuous Time Fourier Transform (using formula(2) from section 1) of the FIR (h_t(1:5)) and then compared it with the characteristics of the Transfer Function. In order to do that I created another function: FourierTransform.m which makes use of the formula provided in section 1.

E) PRESENTATION OF INTERMEDIATE AND FINAL RESULTS

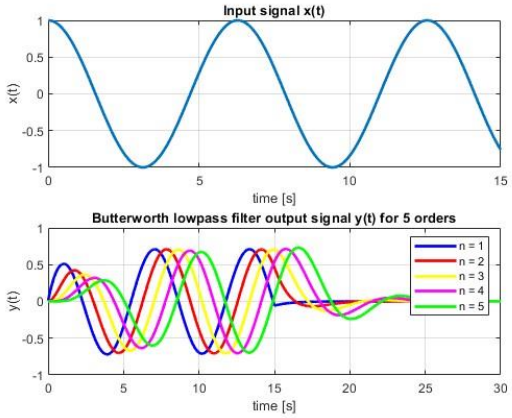
- **AMPLITUDE AND PHASE – FREQUENCY CHARACTERISTICS**



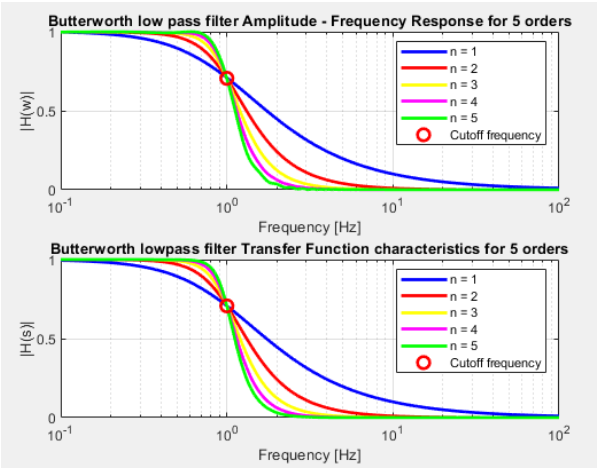
- **FINITE IMPULSE RESPONSE CHARACTERISTICS**



- **INFLUENCE OF THE FILTER ON AN INPUT COSINE SIGNAL**



- **COMPARISON OF OBTAINED FREQUENCY RESPONSE AND TRANSFER FUNCTION CHARACTERISTICS**

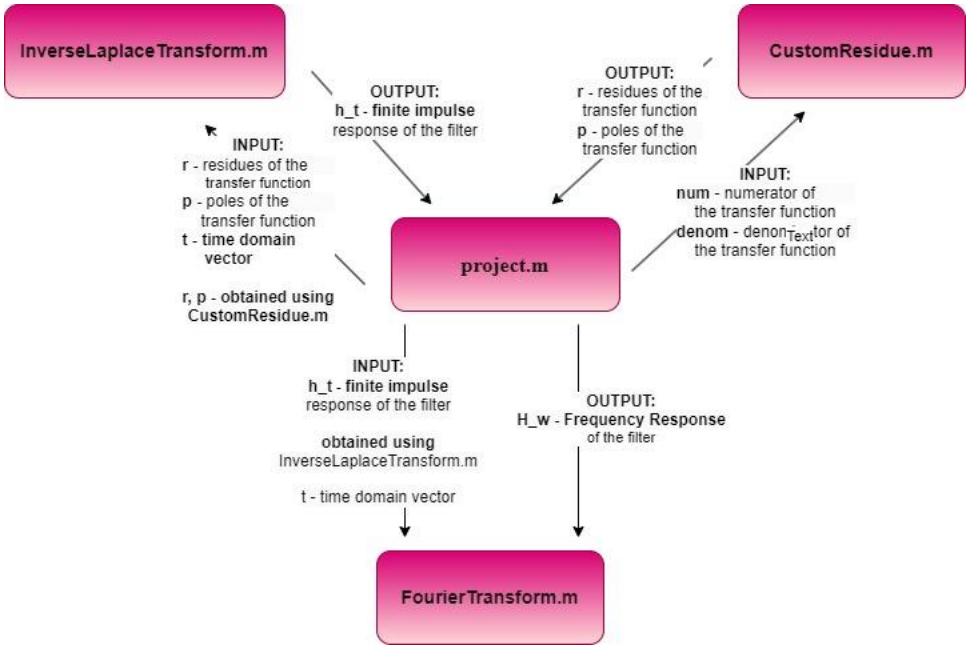


As I can conclude from obtained characteristics, the computations have been done correctly since the results look as they were previously expected to. The code provides a conclusive and comprehensive analysis of the methods used for obtaining different characteristics of the Butterworth lowpass filter

SECTION 3: MUTUAL RELATIONSHIPS BETWEEN THE PROJECT FILES

LIST OF ALL FILES USED IN THE PROJECT

- Project.m (main script)
- CustomResidue.m (Function used to calculate residues and poles of the transfer functions)
- InverseLaplaceTransform.m (Function used to calculate the Inverse Laplace Transform having the residues and poles)
- FourierTransform.m (Functioned used to calculate Continuous Time Fourier Transform)



SECTION 4: ALPHABETICAL LIST OF USED MATLAB FUNCTIONS:

- **Abs –** calculates the absolute value of a variable
- **Clc –** clears the command window
- **Clear –** clears the variables from the workspace
- **Close –** closes all figure windows
- **Conv -** convolution and polynomial multiplication
- **Cos –** creates a cosine of argument in radians
- **End –** indicates last array index
- **Exp -** exponential
- **Figure –** creates a figure window
- **For –** for loop to repeat specified number of times
- **Grid –** displays grid lines on plot
- **Hold –** retains current plot when adding new plots
- **Legend –** adds a legend to the plot
- **Length –** determines the length of largest array dimension
- **Logspace –** generates logarithmically spaced vector
- **Plot –** plots the data on the graph
- **Polyval –** evaluates the polynomial
- **Roots –** calculates polynomial roots
- **Semilogx –** semilog plot (x – axis has log scale)
- **Size –** size of an array
- **Subplot –** creates multiple figures in one window
- **Title –** adds title to the plot

- **Xlabel –** adds the title to the x – axis
- **Ylabel –** adds the title to the y – axis
- **Zeros –** creates an array consisting of zeros

SECTION 5: CONCLUSIONS AND PERSONAL THOUGHTS RELATED TO THE PROJECT

This project was a very great opportunity for me not only to experiment and become more 'fluent' with the Matlab environment, but also to study into the Butterworth lowpass filter dimensions, since the topic was totally new for me. With few troubles along the way, I have successfully managed to meet my previously set goals. The output of the project is very satisfactory, however that does not mean that there are no impurities and that there is no way to further optimize the code and delve deeper into the topic. The code I have prepared could play a strong part for example in another project comparing many different filter approximations (like: Butterworth, Chebyshev or Bessel) with each other. To sum everything up, the project was in my opinion a great experience, allowing me to further my acquaintance with Matlab software environment, as well as with the chosen topic. The topic may not be interesting for everyone, since it has not come up in our academic excursion yet, but I hope it would provide a very clear and coherent explanation for people looking for answers regarding the Butterworth lowpass filter.

SECTION 6: LIST OF REFERENCES AND SOURCES

- **Butterworth lowpass filter basics:**

https://en.wikipedia.org/wiki/Butterworth_filter 02.01.2024

https://www.electronics-tutorials.ws/filter/filter_8.html 02.01.2024

- **Inverse Laplace Transform by partial fraction expansion:**

https://lpsa.swarthmore.edu/LaplaceXform/InvLaplace/InvLaplaceXformPFE.html 02.01.2024

- **Inverse Laplace Transform calculator (used in order to check the FIR waveforms):**

https://www.wolframalpha.com/input?i=inverse+Laplace+transform+1%2F%28s%2B1%29 02.01.2024

- **Continuous Time Fourier Transform:**

https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al.)/08%3A_Continuous_Time_Fourier_Transform_(CTFT)/8.02%3A_Continuous_Time_Fourier_Transform_(CTFT) 02.01.2024

https://www.mathworks.com/matlabcentral/answers/15770-scaling-the-fft-and-the-ifft 02.01.2024

SECTION 7: APPENDIX

• Project.m (MAIN SCRIPT):

```
%Script m - file: project.m
%
%Matlab script used for evaluating the butterworth low - pass filter for 5
%orders from 1 through 5
%
%Konrad Włodarczyk
%Date: 18.12.2023
%Computational Techniques Laboratory Project
%AGH - University of science

clc; clear; close all; %Cleaning the workspace with each launch of the script

w_tf = logspace(-1, 2, 1000); %Defining the frequency range
w_c = 1; %Cutoff frequency of the filter (normalized frequency = 1, but can change to desired)
s = 1i.*w_tf;

%Transfer functions of different orders of filter:
H_s1 = 1./(s./w_c + 1); %1st order
H_s2 = 1./((s./w_c).^2 + 1.414.*(s./w_c) + 1); %2nd order
H_s3 = 1./((s./w_c).^3 + 2.*(s./w_c).^2 + 2.*(s./w_c) + 1); %3rd order
H_s4 = 1./(((s./w_c).^4) + 2.6132.*((s./w_c).^3) + 3.4143.*((s./w_c).^2) + 2.6132.*(s./w_c) + 1); %4th order
H_s5 = 1./(((s./w_c).^5) + 3.236.*((s./w_c).^4) + 5.235924.*((s./w_c).^3) + 5.235924.*((s./w_c).^2) +
3.236.*(s./w_c) + 1); %5th order

%Obtaining the h(t) by the partial fraction decomposition inverse laplace transform of the H(s)
dt = 0.01; %Obtaining the sampling frequency
t = 0:dt:15; %Defining the time range
num = 1; %Common numerator of all the transfer functions

%1st order of the lowpass butterworth filter
denom1 = [1./w_c 1]; %Denominator of the 1st order lowpass butterworth filter
[r1, p1] = CustomResidue(num, denom1); %Calculating the residues and poles
h_t1 = InverseLaplaceTransform(r1, p1, t); %Calculating the inverse Laplace Transform

%2nd order of the lowpass butterworth filter
denom2 = [1./(w_c.^2) 1.414./w_c 1]; %Denominator of the 2nd order lowpass butterworth filter
[r2, p2] = CustomResidue(num, denom2);
h_t2 = InverseLaplaceTransform(r2, p2, t);

%3rd order of the lowpass butterworth filter
denom3 = [1./(w_c.^3) 2./(w_c.^2) 2./w_c 1]; %Denominator of the 3rd order lowpass butterworth filter
[r3, p3] = CustomResidue(num, denom3);
h_t3 = InverseLaplaceTransform(r3, p3, t);

%4th order of the lowpass butterworth filter
denom4 = [1./(w_c.^4) 2.6132./(w_c.^3) 3.4143./(w_c.^2) 2.6132./w_c 1]; %Denominator of the 4th order
lowpass butterworth filter
[r4, p4] = CustomResidue(num, denom4);
h_t4 = InverseLaplaceTransform(r4, p4, t);
```

```
48.
49.  %5th order of the lowpass butterworth filter
50.  denom5 = [1./(w_c.^5) 3.236./(w_c.^4) 5.235924./(w_c.^3) 5.235924./(w_c.^2) 3.236./w_c 1]; %Denominator of
     the 5th order lowpass butterworth filter
51.  [r5, p5] = CustomResidue(num, denom5);
52.  h_t5 = InverseLaplaceTransform(r5, p5, t);
53.
54.  %Introducing the input x(t) cosine signal
55.  A = 1; %Amplitude of the cosine signal
56.  x_t = A.*cos(w_c.*t); %Defining the input cosine signal x(t)
57.
58.  %Calculating the convolution for 5 orders of the filter
59.  %We multiply the result by 'dt' in order to get rid of the scaling issue
60.  y_t1 = conv(x_t, h_t1, 'full').*dt; %1st order
61.  y_t2 = conv(x_t, h_t2, 'full').*dt; %2nd order
62.  y_t3 = conv(x_t, h_t3, 'full').*dt; %3rd order
63.  y_t4 = conv(x_t, h_t4, 'full').*dt; %4th order
64.  y_t5 = conv(x_t, h_t5, 'full').*dt; %5th order
65.
66.  %Creating seperate time vector so that the plots look accurate
67.  t_conv = 0:dt:(length(y_t1)-1)*0.01;
68.
69.  %Creating a seperate frequency vector so that the plots to look accurate
70.  %and for the frequency reponse to have the same length as the transfer
71.  %function
72.  w_fft = w_tf;
73.
74.  %Calculating the fourier transforms for 5 different orders
75.  %We multiply the result by 'dt' in order to get rid of the scaling issue
76.  H_w1 = FourierTransform(h_t1, w_fft).*dt; %1st order
77.  H_w2 = FourierTransform(h_t2, w_fft).*dt; %2nd order
78.  H_w3 = FourierTransform(h_t3, w_fft).*dt; %3rd order
79.  H_w4 = FourierTransform(h_t4, w_fft).*dt; %4th order
80.  H_w5 = FourierTransform(h_t5, w_fft).*dt; %5th order
81.
82.  %Plotting the obtained characteristics
83.
84.  %Amplitude - frequency characteristics
85.
86.  figure(1);
87.  subplot(2, 1, 1)
88.  semilogx(w_tf, abs(H_s1), "LineWidth", 2, "Color", [0 0 1]); grid on; hold on;
89.  semilogx(w_tf, abs(H_s2), "LineWidth", 2, "Color", [1 0 0]);
90.  semilogx(w_tf, abs(H_s3), "LineWidth", 2, "Color", [1 1 0]);
91.  semilogx(w_tf, abs(H_s4), "LineWidth", 2, "Color", [1 0 1]);
92.  semilogx(w_tf, abs(H_s5), "LineWidth", 2, "Color", [0 1 0]);
93.  plot(w_c, abs(w_c./(w_c + 1i*w_c)), 'ro', "MarkerSize", 8, "LineWidth", 2); title('Butterworth lowpass filter
     Transfer Function Amplitude - Frequency Characteristics for 5 orders'); xlabel('Angular Frequency [rad/s]');
     ylabel('|H(s)|'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5", "Cutoff frequency");
94.
95.  %Phase - frequency characteristics
96.
97.  subplot(2, 1, 2)
98.  semilogx(w_tf, angle(H_s1), "LineWidth", 2, "Color", [0 0 1]); grid on; hold on;
99.  semilogx(w_tf, angle(H_s2), "LineWidth", 2, "Color", [1 0 0]);
100. semilogx(w_tf, angle(H_s3), "LineWidth", 2, "Color", [1 1 0]);
101. semilogx(w_tf, angle(H_s4), "LineWidth", 2, "Color", [1 0 1]);
```

```
semilogx(w_tf, angle(H_s5), "LineWidth", 2, "Color", [0 1 0]); title('Butterworth lowpass filter Transfer Function Phase - Frequency Characteristics for 5 orders'); xlabel('Angular Frequency [rad/s]'); ylabel('phase(H(s))'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5");

%Impulse responses

figure(2);
plot(t, h_t1, "LineWidth", 2, "Color", [0 0 1]); grid on; hold on;
plot(t, h_t2, "LineWidth", 2, "Color", [1 0 0]);
plot(t, h_t3, "LineWidth", 2, "Color", [1 1 0]);
plot(t, h_t4, "LineWidth", 2, "Color", [1 0 1]);
plot(t, h_t5, "LineWidth", 2, "Color", [0 1 0]); title("Butterworth lowpass filter impulse response h(t) for 5 orders"); xlabel('time [s]'); ylabel('h(t)'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5");

%x(t) input signal and y(t) output signals

figure(3);
subplot(2, 1, 1)
plot(t, x_t, "LineWidth", 2); grid on; title("Input signal x(t)"); xlabel('time [s]'); ylabel('x(t)');
subplot(2, 1, 2)
plot(t_conv, y_t1, "LineWidth" ,2, "Color", [0 0 1]); grid on; hold on;
plot(t_conv, y_t2, "LineWidth", 2, "Color", [1 0 0]);
plot(t_conv, y_t3, "LineWidth", 2, "Color", [1 1 0]);
plot(t_conv, y_t4, "LineWidth", 2, "Color", [1 0 1]);
plot(t_conv, y_t5, "LineWidth", 2, "Color", [0 1 0]); title("Butterworth lowpass filter output signal y(t) for 5 orders"); xlabel('time [s]'); ylabel('y(t)'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5");

%Amplitude - Frequency characteristics obtained from manually calculating them

figure(4);
subplot(2, 1, 1)
semilogx(w_fft, abs(H_w1), "LineWidth", 2, "Color", [0 0 1]); grid on; hold on;
semilogx(w_fft, abs(H_w2), "LineWidth", 2, "Color", [1 0 0]);
semilogx(w_fft, abs(H_w3), "LineWidth", 2, "Color", [1 1 0]);
semilogx(w_fft, abs(H_w4), "LineWidth", 2, "Color", [1 0 1]);
semilogx(w_fft, abs(H_w5), "LineWidth", 2, "Color", [0 1 0]);
plot(w_c, abs(w_c./(w_c + 1i*w_c)), 'ro', "MarkerSize", 8, "LineWidth", 2); title("Butterworth low pass filter Amplitude - Frequency Response for 5 orders"); xlabel('Frequency [Hz]'); ylabel('|H(w)|'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5", "Cutoff frequency");

%Transfer function characteristics obtained from given formulas

subplot(2, 1, 2)
semilogx(w_tf, abs(H_s1), "LineWidth", 2, "Color", [0 0 1]); grid on; hold on;
semilogx(w_tf, abs(H_s2), "LineWidth", 2, "Color", [1 0 0]);
semilogx(w_tf, abs(H_s3), "LineWidth", 2, "Color", [1 1 0]);
semilogx(w_tf, abs(H_s4), "LineWidth", 2, "Color", [1 0 1]);
semilogx(w_tf, abs(H_s5), "LineWidth", 2, "Color", [0 1 0]);
plot(w_c, abs(w_c./(w_c + 1i*w_c)), 'ro', "MarkerSize", 8, "LineWidth", 2); title('Butterworth lowpass filter Transfer Function characteristics for 5 orders'); xlabel('Frequency [Hz]'); ylabel('|H(s)|'); legend("n = 1", "n = 2", "n = 3", "n = 4", "n = 5", "Cutoff frequency");

%End of plots
%EOF
```

- **InverseLaplaceTransform.m (FUNCTION)**

```
function [h_t] = InverseLaplaceTransform(r, p, t)

%Function m - file: InverseLaplaceTransform.m
%
%Function that calculates the inverse Laplace transform in order
%to obtain the impulse response h(t) having the residues and poles
%of transfer function H(s).
%
%Konrad Włodarczyk
%Date: 19.12.2023
%Computational Techniques Laboratory Project
%AGH - University of science

h_t = zeros(size(t)); %Initializing the impulse response as vector of zeros

for i = 1:length(r)
h_t = h_t + r(i).*exp(p(i).*t); %Calculating the ILT from formula
end
end
```

- **CustomResidue.m (FUNCTION)**

```
function [r, p] = CustomResidue(num, denom)

%Function m - file: CustomResidue.m
%
%Function that obtains the residues and poles of the transfer
%function H(s).
%
%Konrad Włodarczyk
%Date: 19.12.2023
%Computational Techniques Laboratory Project
%AGH - University of science

p = roots(denom); %We obtain the poles of the denominator

num_val = polyval(num, p); %Evaluates the value of the numerator at the poles of the denominator
denom_der = (length(denom)-1:-1:1) .* denom(1:end-1); %Calculates the derivative of the denominator
denom_val = polyval(denom_der, p); %Evaluates the value of the derivative of the denominator at the poles of the
denominator
r = num_val ./ denom_val; %Calculates the residues
end
```

- **FourierTransform.m (FUNCTION)**

```
function H_w = FourierTransform(h_t, t)
%Function m-file: FourierTransform.m
%
%Function that approximates the Continuous Fourier Transform
 %of a given signal (in our case, the impulse responses h(t)).
%
%Konrad Włodarczyk
%Date: 29.12.2023
%Computational Techniques Laboratory Project
```

```
%AGH - University of Science

N = length(h_t); % Obtaining the length of the input signal
H_w = zeros(size(t)); % Initializing the Continuous Fourier Transform as a vector of zeros

for k = 1:N
H_w = H_w + h_t(k) * exp(-1i * 2 * pi * (k-1) * (t * 2.39 / N); %Calculating the Fourier Transform from the formula
end
end
```