**Digital filters**

* 1. Introduction

In Digital Signal processing, a filter is basically a “device”, that removes unwanted frequencies from signal. For an example, often we have noisy signal (let it be audio signal – while we are talking on cellphone), that contains lots of noise (cars, sound from outside, river, etc.). We want to remove that noise from our signal, so people at the other side can hear us well. That is possible to accomplish using filters (digital filters). But, first things first. Let us explain basics of filters (analog filters), their types, their spectres and how to plot them in Matlab.

* 1. Filter types

There are many different bases of classifying filters and these overlap in many different ways; there is no simple hierarchical classification. Filters may be:

* Non-linear or linear
* Time variant or time invariant
* Causal
* Analog or digital
* Discrete-time or continuous time
* Passive or active
* Infinite pulse response (IIR) or Finite pulse response (FIR) filters.

We will work with LTI (linear-time-invariant) FIR (finite pulse response) filters. It is possible to synthesize them so let’s show how.

* 1. Analog and digital filters

The frequency response can be classified into different bandforms describing which frequency [bands](https://en.wikipedia.org/wiki/Band_(radio)) the filter passes (the [passband](https://en.wikipedia.org/wiki/Passband)) and which it rejects (the [stopband](https://en.wikipedia.org/wiki/Stopband)) and common used are:

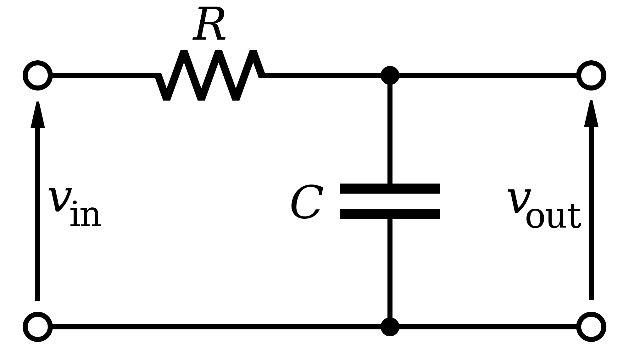
* Low-pass filter
* High-pass filter
* Band-pass filter
* Band-stop filter

When we want to synthesize filter, we are basically working in {S} (complex domain) or domain. Transfer function gives better view of situation. Let’s imagine that we have two sines, as given in formula:

,

If we represent that signal in domain (via Fourier transform), we would have only two delta functions at 50Hz and 150Hz frequencies. According to that, it is much easier to project filter, calculate and synthesize them.

In this article, we well talk more about low pass filters, how do they work, how to calculate them, what responses they have and etc. Scheme for analog low pass filter is given below:



1. Low pass RC filter

It consists of a resistor in series with capacitor. Differential equation for this low pass filter is:

Where . Laplace transform of this equation is:

G(s) is transfer function of given system, X(s) and U(s) are output and input of system.

In Matlab, this transfer function is defined as

s = tf (‘s’);

H = 1 / (1 + RCs)

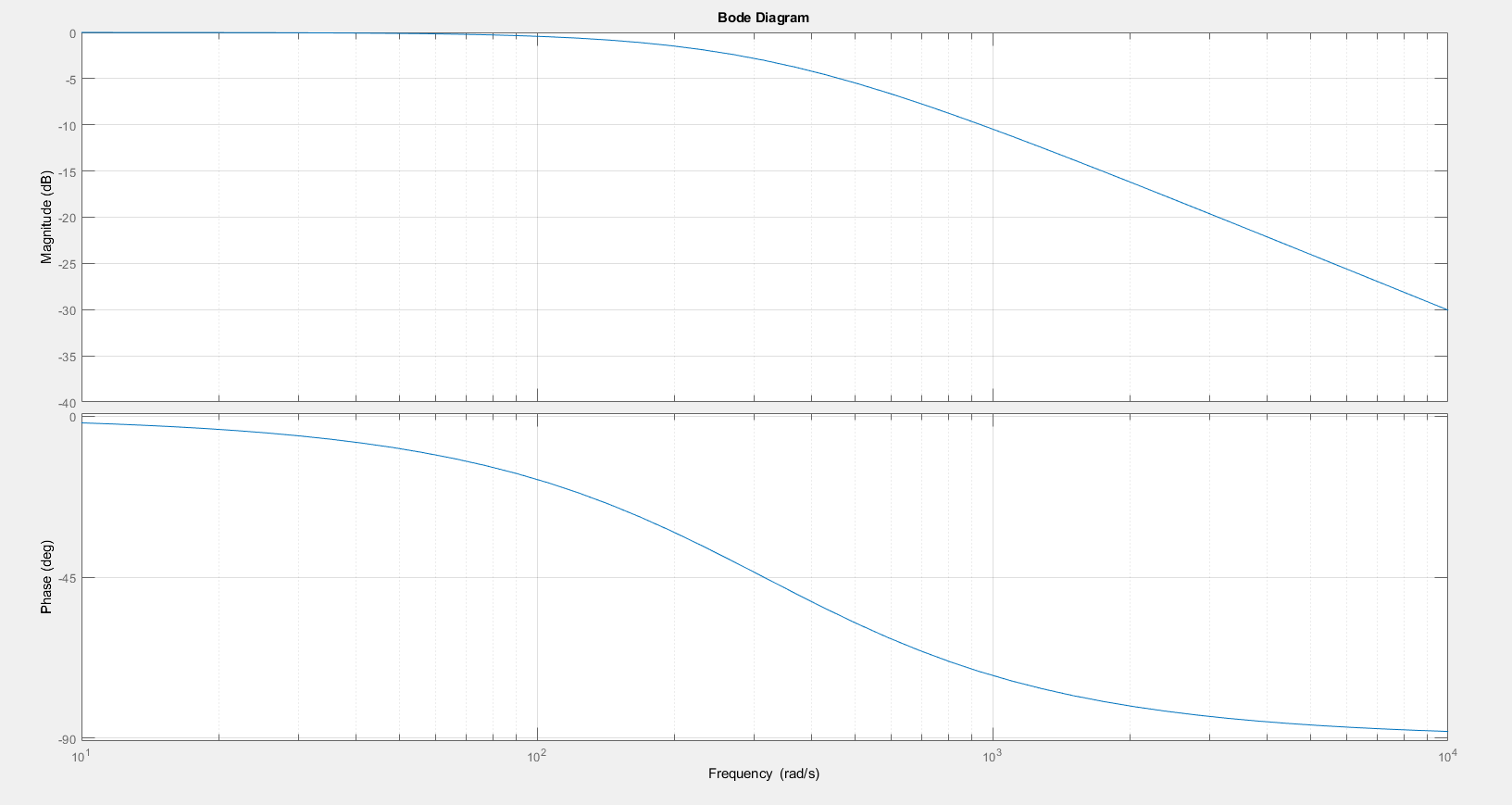
Cut off frequency for this filter is given as , that means combining different values of R and C, we can shift cut off frequency as we want.   
  
Most used method to preview frequency response of system is a Bode plot. It is usually in combination of Magnitude and Phase plot. It is for LTI systems with transfer function G(s) or H(S). Magnitude plot represents graph of the function of frequency . Phase plot is graph of phase, commonly expressed in degrees, .   
The axis of the magnitude and phase plot is logarithmic and the plot is given in decibels or degrees.

To manually calculate Bode plot, we use formula:

For system like this (first order system), that is not to complicate to calculate, but for systems higher order, things can be really messy. Great thing, in Matlab, this can be done with one single line of code, calling bode function

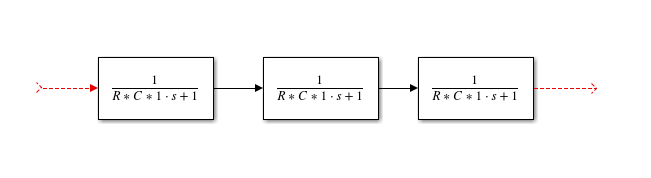
* Bode(H);

Now, let’s create filter that has cut off frequency of 50Hz or 314 . Values for R and C are and 3.2 respectively. Bode plot is given in picture below:



1. Bode plot of low pass filter

As we can see, filter has weakening of the amplitude for 3 [dB] at , and 20 dB per decade. To increase weakening of filter, we can bind more filters and get higher order filter. To accomplish weakening of 60 dB per decade, we need cascade of three filters. Block diagram (Simulink diagram) was given below:



1. Block diagram

Analytic form of equation is . That means, if we let sine wave of 100Hz through this filter, it will be better filtered than through filter of first order. Response become steeper for every block added to structure.

Interesting case is when we have filter of second order. Differential equation for this sort of system is:

The general solution of the differential equation can be obtained as the sum of the solutions of the particular and homogeneous part of the differential equation.

Where

As for the general integral of the homogeneous part of the equation, its character is determined by the nature of the root of the characteristic equation:

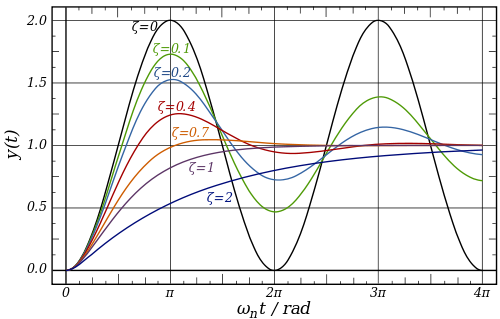
Assume that all coefficients are positive, its roots may be negative or in the form of a conjugated complex pair.

And can be:

1. Real, negative and different. In this case, , and the general solution of the homogeneous equation is of the form
2. Real, negative and equal. In that case, , and the general solution of the homogeneous equation is of the form
3. Form of a conjugated complex pair, and , where and , and the general solution of the homogeneous equation is of the form:

Where are integration constants. Often, differential equation is written in form

Where Now we can introduce the notion of natural frequency, as . That frequency is very important to know, because in case of input signal with frequency of , our system would start oscillating and became unstable. In image below, we will show second order system response, depending of coefficients:



1. Second order system responses

Now, in digital age, we often use digital filters, since almost every system runs on some kind of microprocessor. To use filters in that type of environment, we need discrete form of filters. There are couple of ways to transform from system from G(s) to G(z). Most used method is Tustin’s Bilinear transform. The bilinear transform essentially uses this first order approximation and substitutes into the continuous-time transfer function

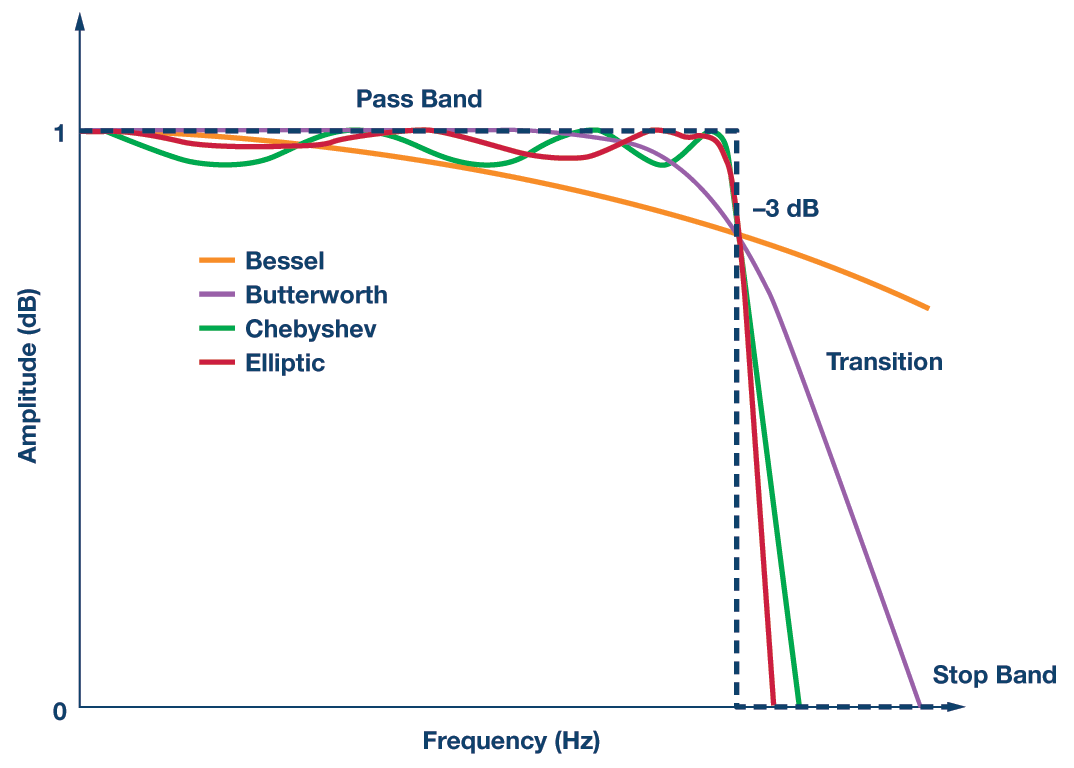
To transform system from G(s) to G(z), in Matlab we call function c2d (continuous to discrete)

G(z) = c2d(H, Ts, ‘tustin’);

H is H(s) – transfer function and Ts is sample period. Sample period represents period of sampling signal. That is important because Nyquist theorem that says: A sufficient sample-rate is therefore anything larger than 2B {\displaystyle 2B}samples per second. Equivalently, for a given sample rate {\displaystyle f\_{s}}, perfect reconstruction is guaranteed possible for a bandlimit{\displaystyle B<f\_{s}/2}.

For this particular example, G(z) is equal to . We also got first order equation.

Digital filters usually have better performance than analog filters. Response is much steeper, as we can see from next image. Dashed line represents ideal digital filter, and that is Fourier transform of . One interesting thing about signals, is that, finite signals in time domain have infinite frequency response and vice versa. That means, to have perfectly designed digital filter, time domain representation must have infinite length. Since, that is not possible, unfortunately dashed response ins impossible to achieve.   
  
Best approximations are given also in image. Those are Butterworth, Chebyshev, Elliptic filters.



1. Digital filters response

Here we can also find two sides. One, we have much better filters, and signal could be just like we want it. Other side is, that, we pay with performance. Lower order filters are not much different from analog filters. To achieve characteristic like in example image, filters must be higher order.

In Matlab, filters are easy to synthetize. Code is given as:

Fs = 10000;

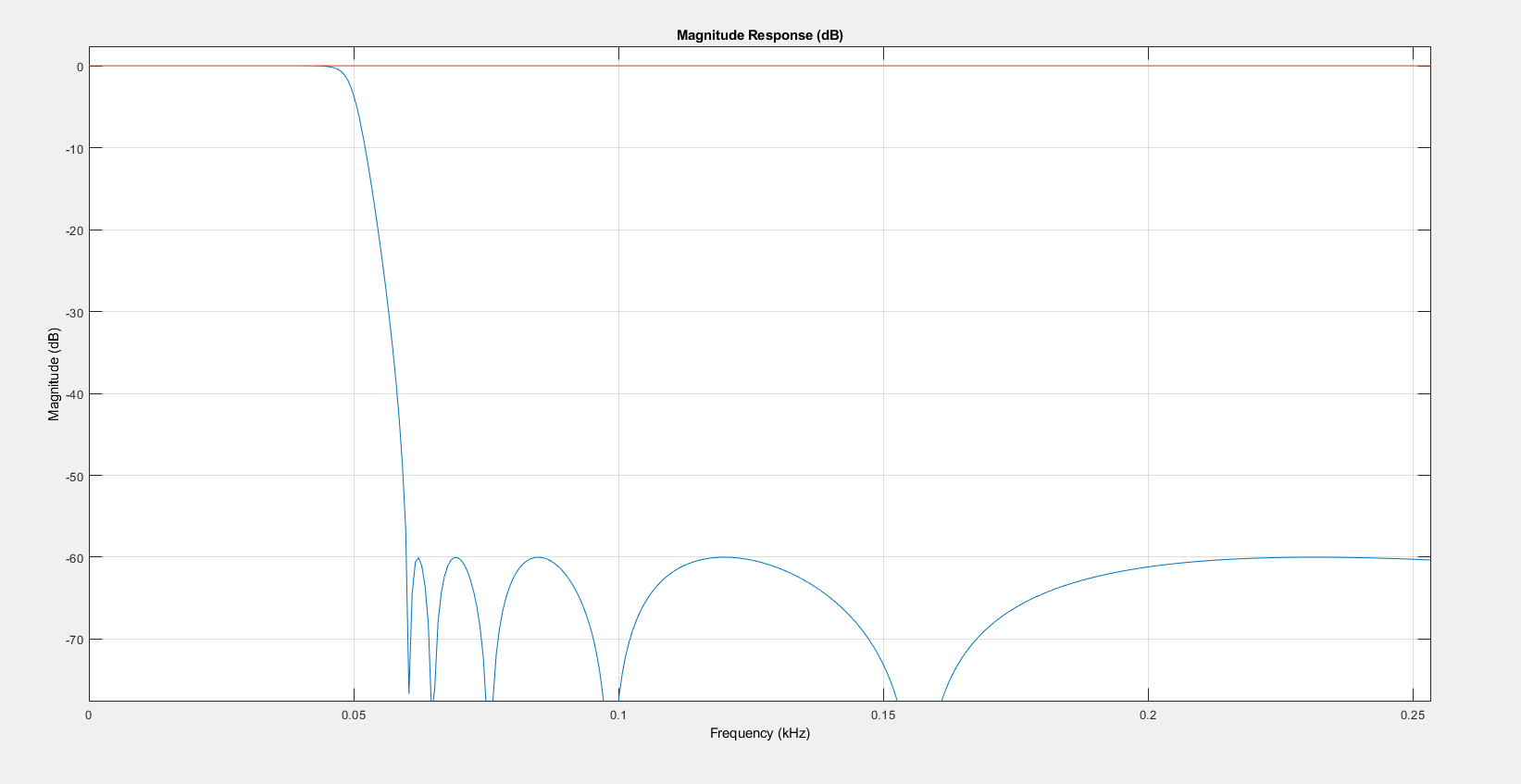
Ts = 1/Fs;

[z,p,k] = cheby2(12,60,6/500, 'low'); %call cheby2 function to create filter

sos = zp2sos(z,p,k);

fvtool(sos,1,'Fs',Fs); %plot it

Cheby2 creates discrete-time zero-pole-gain representation of filter. To convert it to an equivalent second-order section representation in z domain, we need to use zp2sos function.



1. Response of 12th order Chebyshev filter

Implementation of filters on microprocessors based system require discrete time equation. That means, first we need transfer function. Transfer function we can get as

[b,a] = sos2tf(sos);