Università degli Studi di Trento

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Fondamenti di Elettronica Analogica

Titolare del Corso: Prof. Gian Franco Dalla Betta

"Analog Time Domain Step-to-Gaussian Shaping Filter"

Studente: Simone Tollardo

Matricola: 209002

Anno Accademico 2021/2022

Index

What do we mean by “time domain filter”? 2
What is a “shaping filter”? 2
How can we obtain a gaussian shape from a step shape? 2
Types of filters 3
Approximation comparison 6
  Taylor approximation 6
  Raised-Cosine Approximation 7
  Bessel approximation 8
  Transitional Filters 11
  Observation: 15
    Fractional order filters 15
    Example 15
Implementation topologies 16
  Sallen-Key 17
  MFB 19
  Passive Ladder 20
A “real” simulation 21
Breadboard implementation 23
Final Conclusions 26
Bibliography 27

What do we mean by "time domain filter"?

In signal processing, a filter is a device or process that removes some unwanted components or features from a signal. We know that every linear time-invariant filter has a Frequency Response and so it can remove or modify some parts of the frequency spectrum of the input signal.
We also know that every linear time-invariant filter has an IRF (Impulse Response Function) that is the inverse LT (Laplace Transform) of the frequency response.
Although frequency response is very useful in a lot of application, sometimes we are interested in the effect that the filter produces in the time domain (or in the spatial domain for the case of Image Processing filters). One example can be image processing filter: we would want to blur an image, so a typical solution is a moving average filter, and its purpose is to replace the pixel level by the average of the level of the near pixels (the number of "near" pixels depends on the chosen window dimension).
This filter is simply a low pass in the frequency domain but in this case we are interested in the spatial domain response. Farther, for digital filters we often have to implement the filter in the time (or spatial) domain as a kernel and then we simply convolute the impulse response of the filter with the input signal. Same rules apply for analog filters, the only "problem" is that we typically think in terms of the frequency response, especially for electronic filters, and making a bidirectional correspondence between frequency domain and time domain isn't always trivial (without calculators and other mathematical tools).

What is a "shaping filter"?

In electronics and telecommunications, shaping is the process of changing the waveform of transmitted signal. A typical purpose is to make the transmitted signal better suited to its purpose or the communication channel, typically by limiting the effective bandwidth of the transmission (pulse shaping filter).

The purpose of this project is to find the best way to design a filter that shape a relatively long input step-like signal to a very short gaussian output signal.

So, for this purpose we need a Step Shaping Filter that transform an input step signal to a Gaussian shape output signal.

How can we obtain a gaussian shape from a step shape?

A Gaussian filter is a filter whose impulse response is a Gaussian function. Gaussian filters are designed to give no overshoot to a step function input while minimizing the rise and fall time**.** This behaviour is closely connected to the fact that the Gaussian filter has the minimum possible group delay. Mathematically, a Gaussian filter modifies the input signal by convolution with a Gaussian function; this transformation is also known as the Weierstrass transform.

Considering that we want the step response to be a gaussian, and not the impulse response as in the case of a gaussian filter, we can derive the wanted impulse response of our system in this way:
We know that for a linear time-invariant (LTI) system the step response $a(t)$ can be obtained by the integral of the impulse response $h(t)$ of the system itself:

$$a(t) = \int_0^t h(t)\ dt$$

Moving into frequencies domain:
$A(j\omega) = \frac{G(j\omega)}{j\omega}$, where $G(j\omega)$ is the frequency response of the system for real frequencies.

If we want $a(t)$ to be Gaussian, it can be proven that also $A(j\omega)$ must be Gaussian. (Fourier transform of a Gaussian is a Gaussian: if for example, we apply the Fourier Transform to a gaussian $g(t) = \sigma e^{-\frac{1}{2}\sigma^2 t^2}$, we get $G(j\omega) = \sigma e^{-\left(\frac{\omega^2}{2\sigma^2}\right)}$.)

Hence, our frequency response $G(j\omega)$ is a Gaussian multiplied by $j\omega$:
$G(j\omega) = (j\omega)\sigma e^{-\left(\frac{\omega^2}{2\sigma^2}\right)}$ and its corresponding transfer function is $G(s) = s\sigma e^{-\left(\frac{s^2}{2\sigma^2}\right)}$.

How can we obtain a transfer function that has $G(s) = s\sigma e^{-\left(\frac{s^2}{2\sigma^2}\right)}$ response?
We know that the transfer function of a LTI system can be always written as:

$$\mathrm{H(s)} = \frac{(\mathrm{s}-\mathrm{z}_1)(\mathrm{s}-\mathrm{z}_2)\cdots(\mathrm{s}-\mathrm{z}_\mathrm{n})}{(\mathrm{s}-\mathrm{p}_1)(\mathrm{s}-\mathrm{p}_2)\cdots(\mathrm{s}-\mathrm{p}_\mathrm{m})} = \frac{\prod_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{s}-\mathrm{z}_\mathrm{i})}{\prod_{\mathrm{i}=1}^{\mathrm{m}}(\mathrm{s}-\mathrm{p}_\mathrm{i})}$$

where z_i are Zeroes and p_i are Poles of the function.
Since $G(s)$ is a transcendental function, it can not be written as a rational function, so we need to approximate it as a polynomial.

Note the relation between the variance in the time domain Gaussian and the frequency response: as the Gaussian shrinks in time, the shape in the frequency domain gets larger!

Types of filters

We know that we can classify a filter for his frequency response in 5 categories:

- Low-Pass filters – high frequencies passed, low frequencies are attenuated
- High-Pass filter – low frequencies passed, high frequencies are attenuated
- Band-Pass filters – only frequencies in a specific band are passed
- Band-Stop filters – only frequencies in a specific band are attenuated
- All-Pass filters – all frequencies are passed, but the phase of the output signal is modified

We know also that it's impossible to make an ideal filter with an infinitely steep frequency response because it would be infinitely long in the time domain.

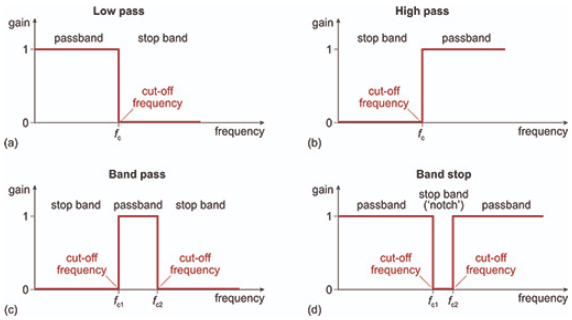


Fig. 1 – Ideal Filters response.

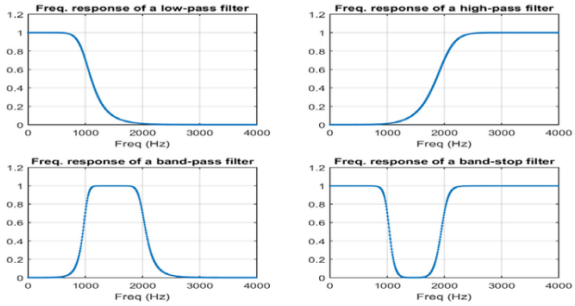


Fig. 2 – Real Filters response.

Another different classification can be made in terms of the approximation used and the principal classes and characteristics are:

- Butterworth filter – no gain ripple in pass band and stop band, slow cut-off
- Chebyshev filter (Type I and Type II) – no gain ripple in stop band, moderate cut-off
- Bessel filter – no group delay ripple, no gain ripple in both bands, slow gain cut-off

- Elliptic filter – gain ripple in pass and stop band, fast cut-off
- Gaussian filter – no ripple in response to step function
- Raised-cosine filter

Each family of filters can be specified to a particular order. The higher the order, the more the filter will approach the "ideal" filter; but also the longer the impulse response is and the longer the latency will be. An ideal filter has full transmission in the pass band, complete attenuation in the stop band, and an abrupt transition between the two bands, but this filter has infinite order (i.e., the response cannot be expressed as a linear differential equation with a finite sum) and infinite latency.

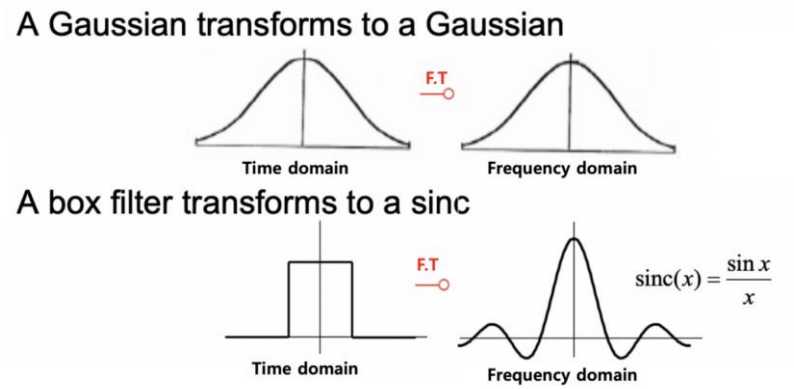


Fig 3 - Ideal Gaussian filter and ideal Box Filter impulse and magnitude of frequency responses, all infinitely long. (apart from Box).

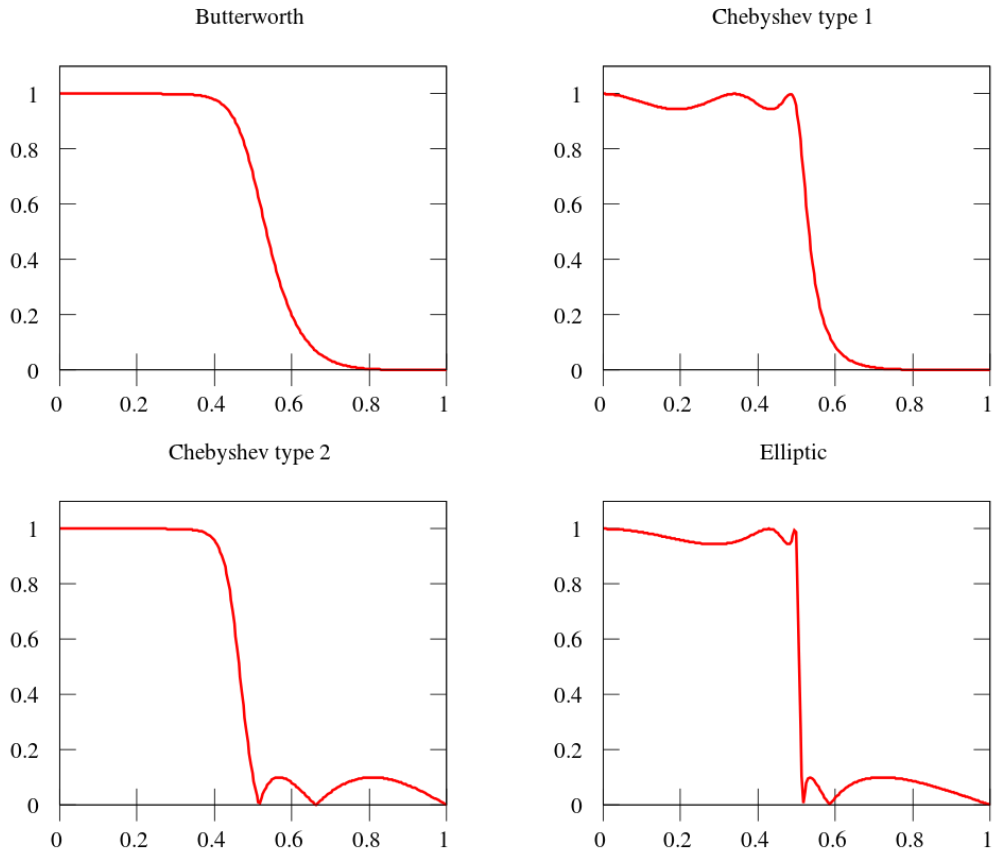


Fig 4 - Fifth order real low-pass filters comparison (Magnitude of Frequency response).

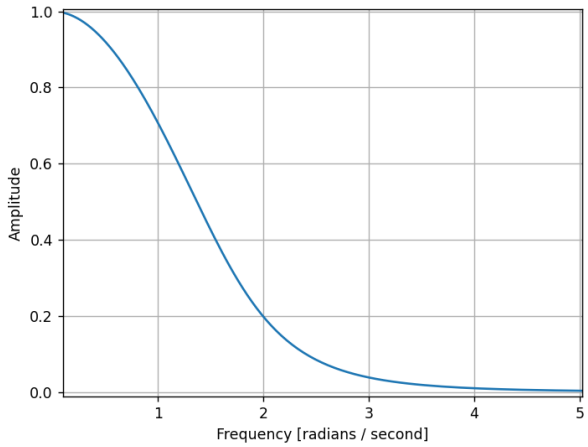


Fig 5 - Frequency response of real fifth order low-pass Bessel filter ($\omega_c = 1\ rad/s$).

NOTE: **a Butterworth filter** can always be designed to have a frequency response gain like this:

$$G(\omega) = \frac{1}{\sqrt{1 + (\frac{\omega}{\omega_c})^{2n}}}$$

where ω is the angular frequency in radians per second and n is the number of poles in the filter.
If $\omega = \omega_c$, the amplitude response of this type of filter in the passband is $1/\sqrt{2} \approx 0.707$, which is half power or −3 dB.
ω_c is commonly called "cut-off frequency".
For clarity, in this document, when we refer to cut-off frequency, for any type of filter, we refer to the frequency at which the gain is $1/\sqrt{2}$ before dropping.* ("normalized cut-off frequency")

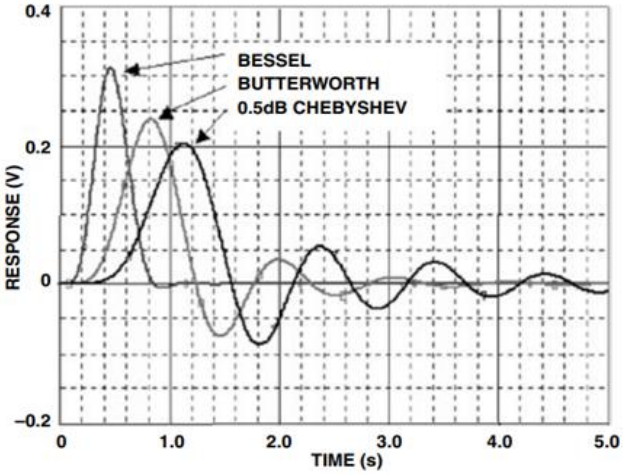


Fig 6 - Comparison between real low-pass filters Impulse response.

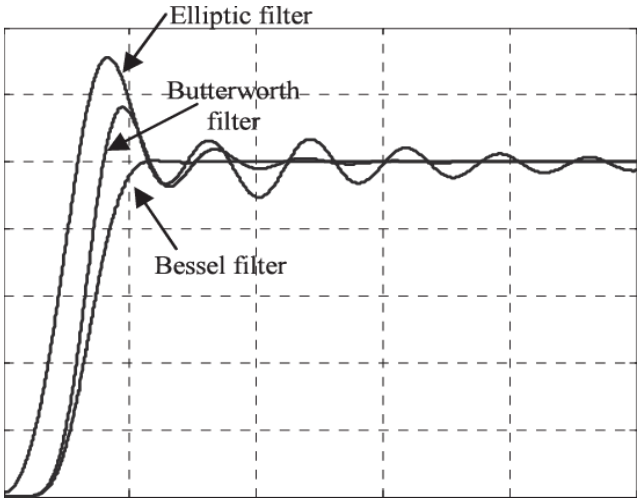


Fig. 7 – Comparison between real low-pass filters Step response.

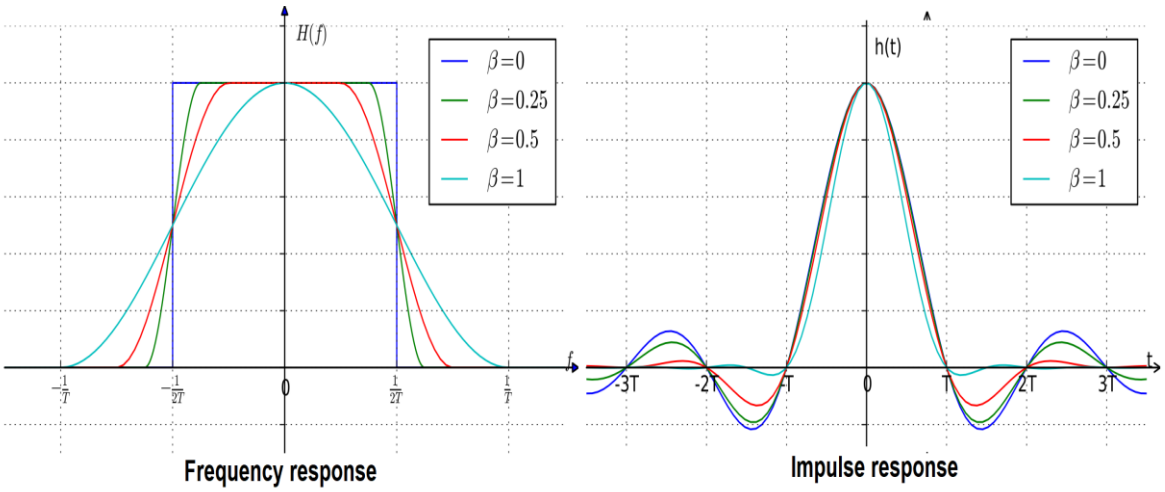


Fig 8 - Frequency response and impulse response of a real (beta>0) and ideal (beta=0) low-pass Raised-Cosine filter.

* usually, Chebyshev and Elliptic filters' cut-off frequency is defined as the frequency at which the response falls below the ripple band.

Approximation comparison

I did a comparison between **the magnitude** of some types of approximation here:
https://www.desmos.com/calculator/et3dns7k9l

NOTE: I immediately discarded Elliptic, Chebyshev and Butterworth filter for the comparison because of their evident overshoot in their step responses.

Let's discuss the results:

For simplicity we rewrite the two gaussian as:

$g(t) = \sigma e^{-\frac{x^2}{2\sigma_2^2}}$ time domain gaussian and $G(s) = \sigma x e^{-\frac{x^2}{2\sigma^2}}$ frequency response where $\sigma = \frac{1}{\sigma_2}$. **(Note the inverse proportionality between σ_1 and σ_2)**

These are the time domain gaussian centred in $\boldsymbol{t = 0}$ and with standard deviation $\boldsymbol{\sigma_2 = \frac{1}{\sigma} \cong 0.78}$, in blue, and the corresponding frequency domain response modulus (the one that we are trying to approximate), in black:

gaussianwidth.mp4

errata:(not scaled by 1/sigma)

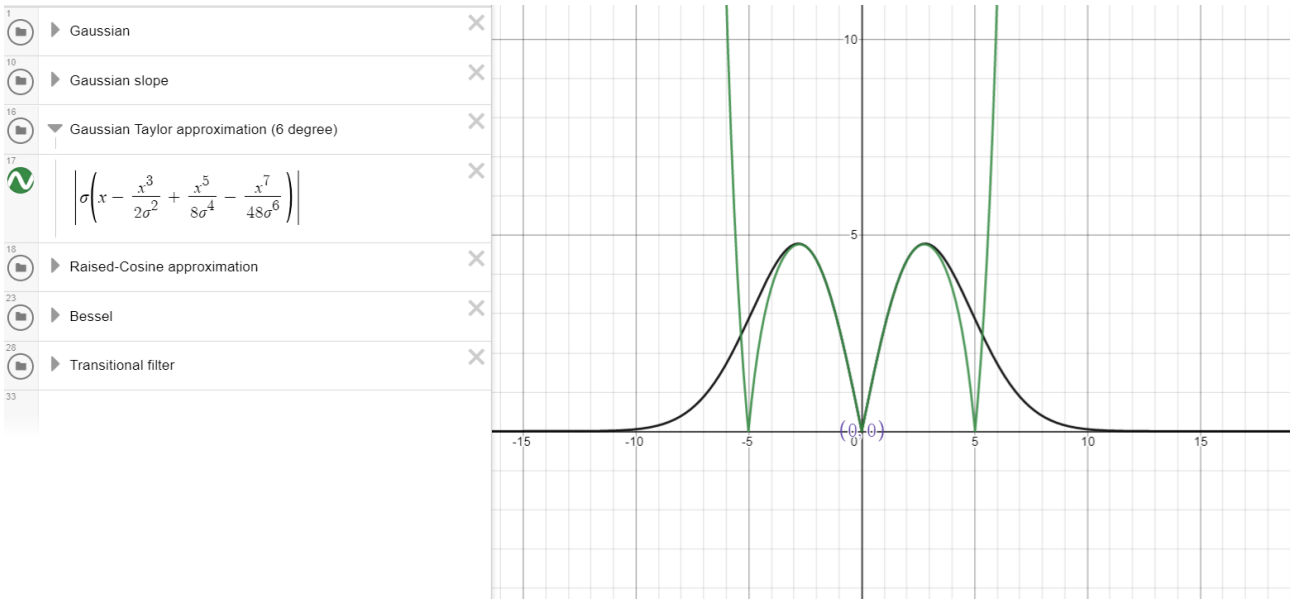
Taylor approximation

We can approximate $G(s)$ transfer function developing it into a Taylor power series:

Considering only first 7 terms we get:

$$T(s) = \left(x - \frac{x^3}{2\sigma^2} + \frac{x^5}{8\sigma^4} - \frac{x^7}{48\sigma^6}\right)$$

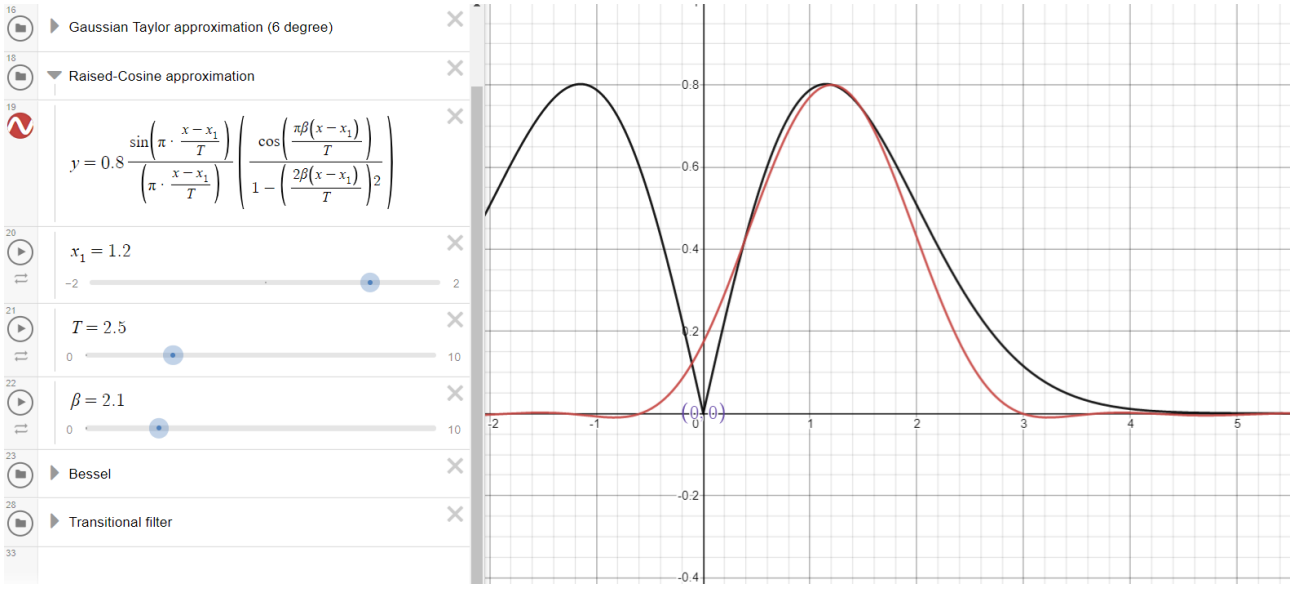
This is the comparison between Taylor approximation of the response with the response itself:



Raised-Cosine Approximation

Raised-cosine filter (Fig. 8) is another type of pulse shaping filter, commonly used in digital filters.
The raised-cosine filter is a frequently used pulse-shaping filter, often in digital modulation, due to its ability to minimise intersymbol interference (ISI). Its name stems from the fact that the non-zero portion of the frequency spectrum of its simplest form ($\beta = 1$) is a cosine function, 'raised' up to sit above the f (horizontal) axis.

This is the comparison between our transfer function and a raised-cosine function centred in $x_1 = 1.2$ and with maximum value $y(1.2) = 0.8$:



raised-cosine.mp4

Although Raised-Cosine function could be a good approximation of our frequency response, it's not a rational function so we can not directly implement it as a transfer function.

Bessel approximation

Another way to approximate our $G(s)$ is using Bessel approximation.
The Bessel filter is very similar to the ideal Gaussian filter (as said before it's a filter whose impulse response is a gaussian function) and tends towards the same shape as filter order increases (again, remember that Fourier transform of a Gaussian is a Gaussian). While the time-domain step response of the ideal Gaussian filter has zero overshoot, the Bessel filter has a small amount of overshoot but, as we can clearly see in the Fig. 6 and Fig. 7, still much less than common frequency domain filters.

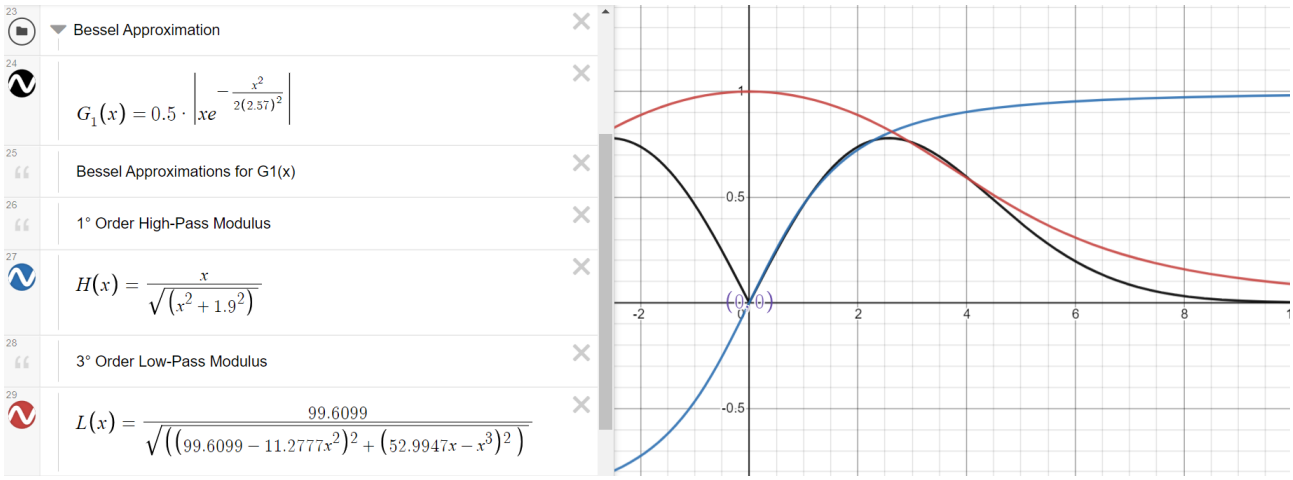
Compared to finite-order approximations of the Gaussian filter, the Bessel filter has better shaping factor, flatter phase delay, and flatter group delay than a Gaussian of the same order, though the Gaussian has lower time delay and zero overshoot.
Note that the transition from the pass band to the stop band is much slower than for other filters, but the group delay is practically constant in the passband.
(Group delay $D(w) \equiv -\frac{d}{dt}\theta(w)$; where $\theta(w)$ is bode phase response.
For linear phase responses, i.e., $\Theta(\omega) = -\alpha\omega$ for some constant α, the group delay and the phase delay are identical, and each may be interpreted as time delay. If the phase response is nonlinear, then the relative phases of the sinusoidal signal components are generally altered by the filter. A nonlinear phase response normally causes a "smearing" of attack transients.)

I computed the absolute value of Bessel functions (both Low Pass and High Pass, one for the first half of the shape and the second for the second half), and after some parametrical test, I found out that the best approximation is when our $G(s)$ is scaled by a factor $\beta = \frac{1.285}{\sigma}$ (in order to keep peak at a constant value of about 0.78). For example, if $\sigma = 2.57$ we should have $G_1(s) = 0.5 * G(s)$.

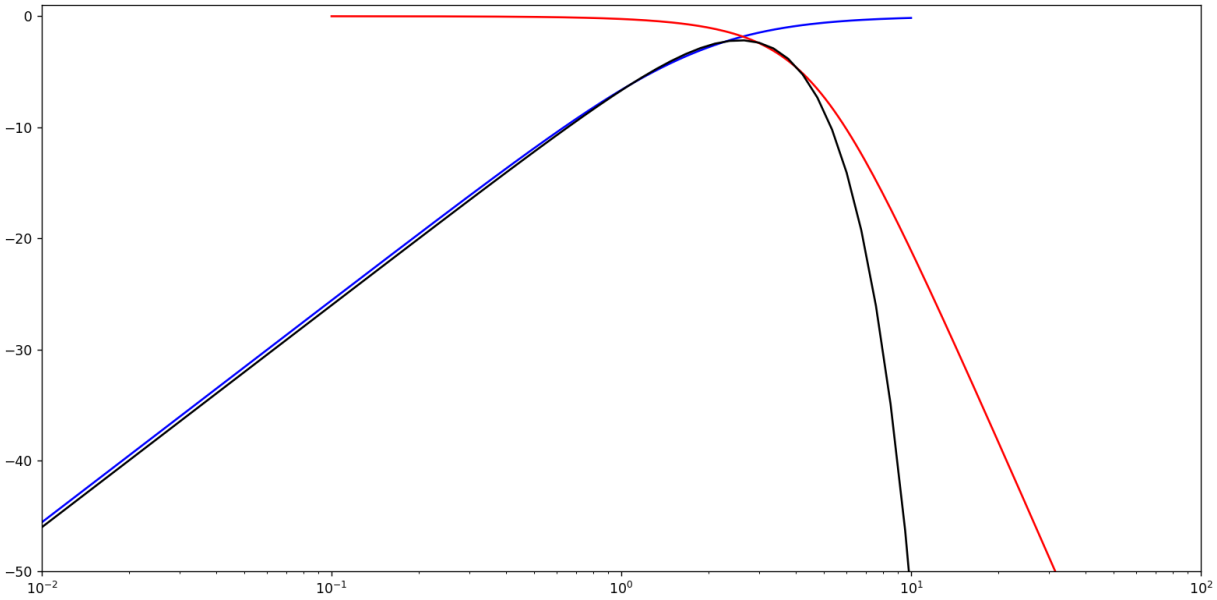
N	Transfer Function
2	$\frac{1.6221}{s^2+2.206s+1.6221}$
3	$\frac{2.7992}{(s+1.3270)(s^2+2.1018s+2.1094)}$
4	$\frac{5.1002}{(s^2+2.7192s+2.0142)(s^2+1.9754s+2.5321)}$
5	$\frac{11.3845}{(s+1.5069)(s^2+2.7702s+2.4370)(s^2+1.9212s+3.1001)}$
6	$\frac{26.8328}{(s^2+3.1470s+2.5791)(s^2+2.7672s+2.8605)(s^2+1.8636s+3.6371)}$
7	$\frac{69.5099}{(s+1.6853)(s^2+3.2262s+2.9497)(s^2+2.7594s+3.3251)(s^2+1.8208s+4.2052)}$
8	$\frac{198.7746}{(s^2+3.5254s+3.1820)(s^2+1.7910s+4.1895)(s^2+2.7560s+3.8382)(s^2+3.2838s+3.3770)}$

Bessel functions for a LP filter.

This, for example is the comparison between our $G_1(s)$ for $\sigma = 2.57$ (in black) and Bessel approximations.
I used an 3° order LP Bessel filter response (in red) for the second half of the shape and a 1° order HP filter response (in blue) (first order response are all the same, doesn't matter what approximation you use) for the first part of the shape:
(cut-off frequency are respectively $\omega_{ch} = 3.3\ rad/s$ and $\omega_{cl} = 1.9\ rad/s$)



This is the same result on a Bode plot (magnitude only):



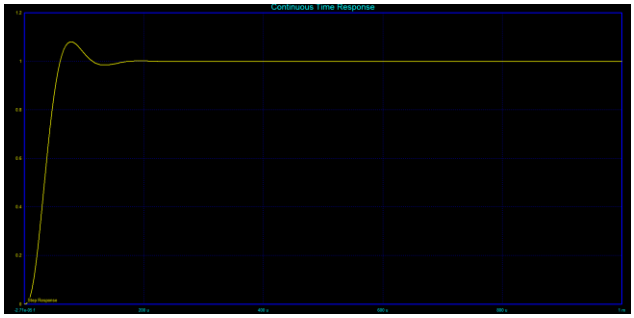
As we can see Bessel approximation is really good and don't add much complexity, in fact even 3° order + 1° order filters should satisfy our requirements.

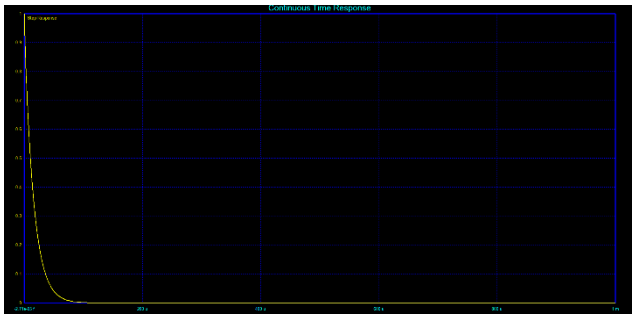
This is the python code used for the Bode plot:

```
import numpy as np
import matplotlib.pyplot as plot
from scipy import signal

fig1, ax1 = plot.subplots()
b, a = signal.bessel(1,1.9,"high", analog=True,norm="mag")
w, h = signal.freqs(b,a)
ax1.semilogx(w,20*np.log10(abs(h)),color="blue")
b, a = signal.bessel(3,3.3,"low", analog=True,norm="mag")
w, h = signal.freqs(b,a)
ax1.semilogx(w,20*np.log10(abs(h)),color="red")
expr="x*(0.5*np.e**((-x**(2))/(2*(2.57**(2)))))"
x=np.logspace(-2,3,100)
```

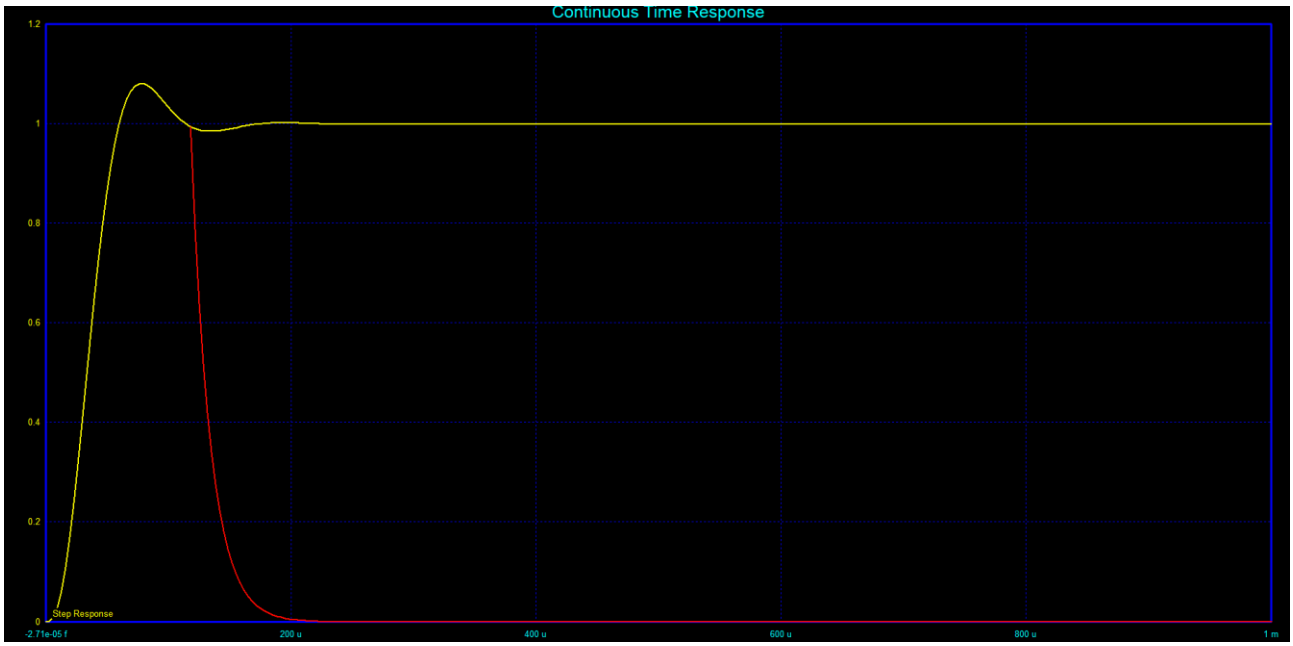
```
y=(20*np.log10(eval(expr)))
ax1.semilogx(x,y,color="black")
ax1.set_xlim(0.01,100)
ax1.set_ylim(-50,1)

plot.show()
```

Here we can see from the time domain point of view what's happening: these are the step responses of respectively a 3° Order Bessel LP filter and a 1° order Bessel HP filter with both $f_c = 10KHz$:





If we imagine cascading these two filter in a resulting BP filter, we can somehow imagine the two step responses combined in a sort of gaussian shape in this way: (Please note: this image is a photoshop and don't respect reality, it's only for a better understanding)



Transitional Filters

Another approximation can be made using so called “Transitional Filters”.

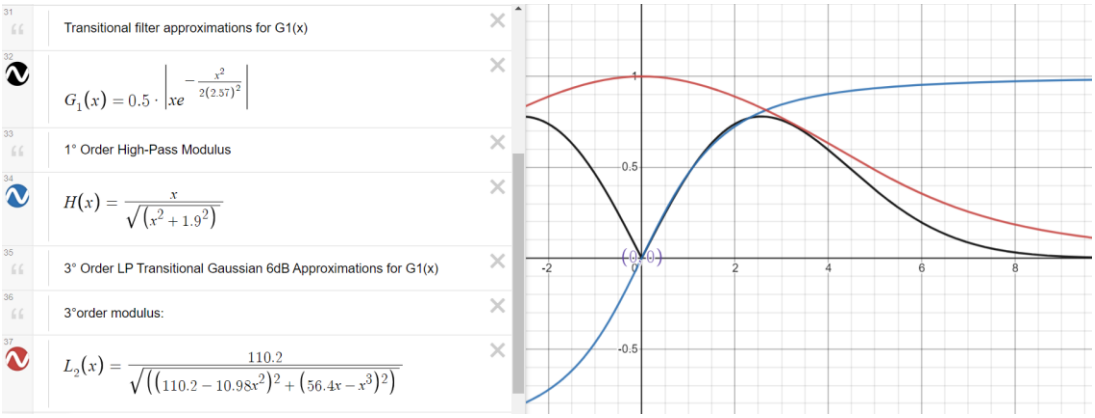
A transitional filter is a compromise between an ideal Gaussian filter and the Chebyshev. A transitional filter has nearly linear phase shift and smooth, monotonic roll-off in the pass band. Above the pass band there is a break point beyond which the attenuation increases dramatically compared to the Bessel, and especially at higher values of n (filter order). We can commonly find two tables of transition filters: these are the Gaussian to 6 dB and Gaussian to 12 dB.

The Transitional Gaussian to 6 dB filter has better transient response than the Butterworth in the passband. Beyond the break point, which occurs at $\omega_r = 2\ rad/s$ (roll-off frequency/pulsation) for a $\omega_c = 1\ rad/s$ filter, the roll-off is similar to the Butterworth.

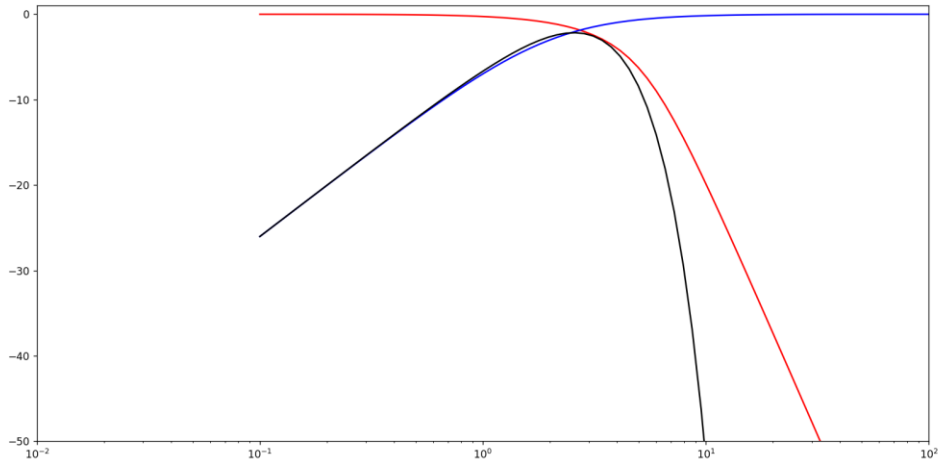
The Transitional Gaussian to 12 dB filter's transient response is much better than Butterworth in the passband. Beyond the 12 dB break point, which occurs at $\omega_r = 2\ rad/s$ for a $\omega_c = 1\ rad/s$ filter, the attenuation is less than the Butterworth.

For Transitional filters, pole locations do not have a closed form method for computation.

Using poles tables (you can find them easily on the web) I calculated the frequency response of a 3° order Transitional 6dB Gaussian filter:
(original gaussian in black, 3° Order Gaussian 6dB filter in red, 1° Order filter in blue)



Same results on a Bode plot (magnitude only):



As we can see, Transitional filter Gaussian 6dB and Bessel filter have almost same response.

After all these comparisons we can say that for approximating the first half of our wanted frequency response we simply need a 1° order HP filter, while for the second half we need a higher order filter like Bessel or Transitional filters (and the higher the order the better the approximation is).
In fact, overshoot for of step response decreases as filter order increases for a LP filter, while decreases as filter order decreases for an HP filter.
For convenience reasons (because it's better documented and easily computable) we will chose Bessel filter for our purpose.

Example of frequency response (magnitude) of increasing order low-pass Bessel Filters (from 1 to 10) with $\omega_c = 3.3\ rad/s$ vs the previous discussed gaussian (in black) and the corresponding impulse responses.

increasingorderbesselLP.mp4

This is the python code used for these plots:

```
import functools
import numpy as np
import matplotlib.pyplot as plot
from scipy import signal

def plotlive(func):
    plot.ion()
    @functools.wraps(func)
    def new_func(*args, **kwargs):
        result = func(*args, **kwargs)
        plot.draw()
        plot.pause(1)
        return result
    return new_func

@plotlive
def plot_something_live(ax, x, y):
    ax.plot(x, y)

#setting plots
plot.ion()
fig1, (ax1,ax2) = plot.subplots(1,2)
mng = plot.get_current_fig_manager()
mng.resize(1200,600)
ax1.set_title("Frequency response")
ax1.set_xlabel("Angular frequency [rad/s]")
ax1.set_ylabel("Gain [dB]")
ax2.set_title("Impulse response")
ax2.set_xlabel("Time [S]")
ax1.set_xlim(0.1,100)
ax1.set_ylim(-120,1)

#gaussian with variance 1.15
```

```
expr="x*(0.5*np.e**((-x**(2))/(2*(2.57**(2)))))"
x=np.logspace(-1,3,1000)
y=20*np.log10(eval(expr))
ax1.semilogx(x,y,color="black")

for i in range(10):
    b, a = signal.bessel(i+1,3.3,"low", analog=True,norm="mag")
    w, h = signal.freqs(b,a)
    plot_something_live(ax1, w, 20*np.log10(abs(h)))
    t2, v2 = signal.impulse((b,a))
    plot_something_live(ax2, t2, v2)
```

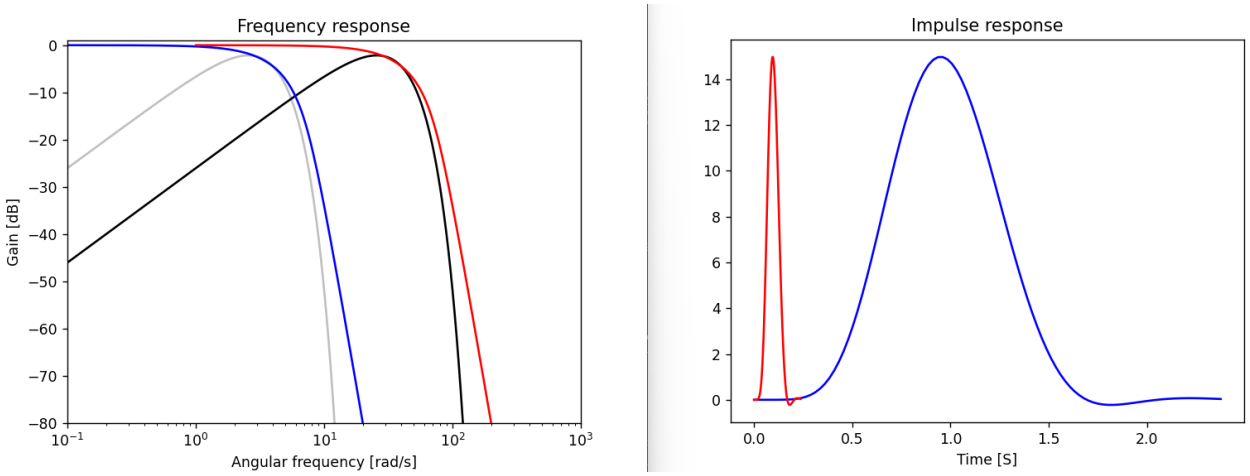
NOTE: How can we obtain the frequency response of a HP filter from a LP filter with the same cut-off frequency?

We can transform a LP filter to a HP filter simply considering that the transfer function of a LP filter is the transfer function of a HP filter with s replace by $\frac{1}{s}$. (Intuitively, remember that LP perform a time domain integration, that is a multiplication by s in the Laplace domain)

So, if for example our LP filter transfer function is $G_{LP}(s) = \frac{1}{s^3+2.433s^2+2.467s+1}$, our HP filter transfer function will be $G_{HP}(s) = \frac{s^3}{1+2.433s+2.467s^2+s^3}$.

Obviously, if our gaussian become larger in time domain (σ_2 increases) the frequency domain's gaussian shape become shorter (σ decreases) and vice-versa, so for best approximation we need a lower/higher cut-off frequency ω_c to follow the curve.

Here is an example: in grey we can see our frequency response $G(s)$ with standard deviation $\sigma_2 = 2.57$ while in black another with $\sigma_2 = 25.7$. In order to approximate the gaussian I plotted the frequency and impulse responses of two 8° order Bessel filters LP filters (one impulse response is scaled in amplitude in order to properly see it), one with $\omega_c = 3.3Hz$ (in blue) and one with $\omega_c = 33Hz$ (in red):



Obviously, same consideration applies for the HP filter.

Again, this is the python script used:

```
import functools
import numpy as np
import matplotlib.pyplot as plot
from scipy import signal

def plotlive(func):
    plot.ion()
    @functools.wraps(func)
    def new_func(*args, **kwargs):
        result = func(*args, **kwargs)
        plot.draw()
        plot.pause(1)
        return result
    return new_func

@plotlive
def plot_something_live(ax, x, y):
    ax.plot(x, y)

#setting plots
plot.ion()
fig1, (ax1,ax2) = plot.subplots(1,2)
mng = plot.get_current_fig_manager()
mng.resize(1200,600)
ax1.set_title("Frequency response")
ax1.set_xlabel("Angular frequency [rad/s]")
ax1.set_ylabel("Gain [dB]")
ax2.set_title("Impulse response")
ax2.set_xlabel("Time [S]")
ax1.set_xlim(0.1,100)
ax1.set_ylim(-120,1)

#gaussian with variance 1.15
expr="x*(0.5*np.e**((-x**(2))/(2*(2.57**(2)))))"
x=np.logspace(-1,3,1000)
y=20*np.log10(eval(expr))
ax1.semilogx(x,y,color="black")

for i in range(10):
    b, a = signal.bessel(i+1,3.3,"low", analog=True,norm="mag")
    w, h = signal.freqs(b,a)
    plot_something_live(ax1, w, 20*np.log10(abs(h)))
    t2, v2 = signal.impulse((b,a))
    plot_something_live(ax2, t2, v2)
```

Observation:

What happens if we want to approximate a shape whose slope is even less steep than the first order filter slope? How can we implement an analog filter that has less than the asymptotic -20dB/dec slope?

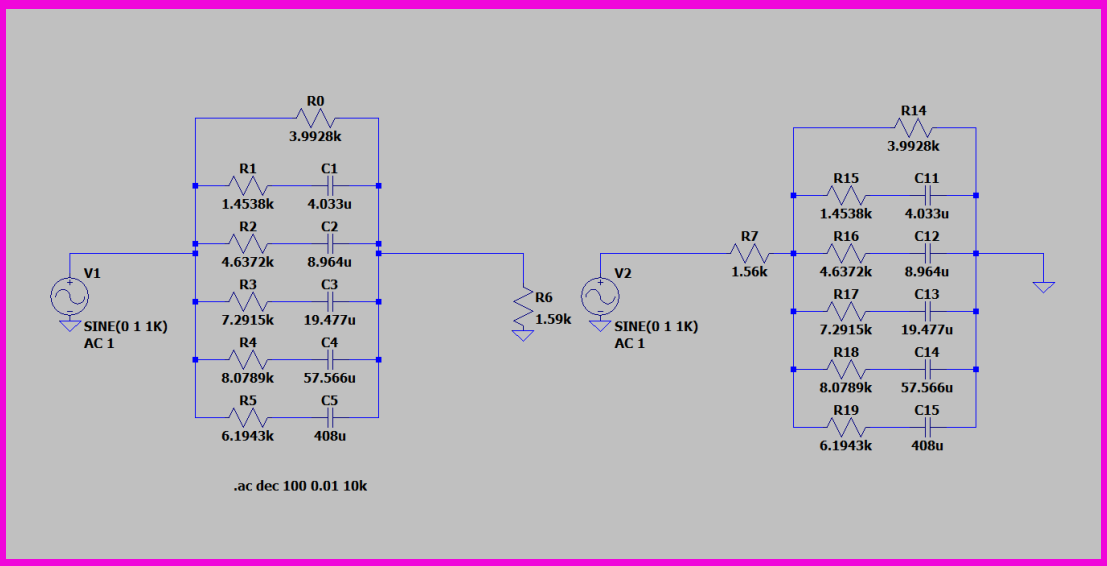
Fractional order filters

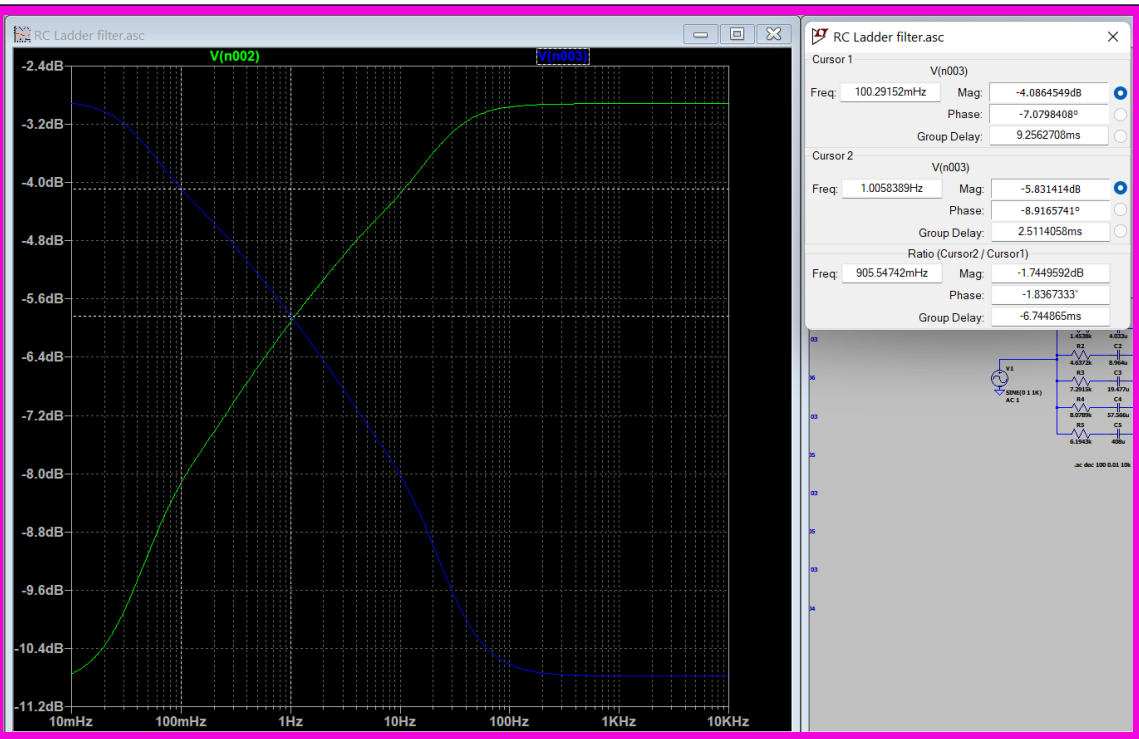
The answer is "**Fractional Filters**": these filters are modelled by fractional-order systems and they are based on fractional order elements, which are usually approximated with conventional, lumped-element circuits.

In the fields of dynamical systems and control theory, a fractional-order system is a dynamical system that can be modelled by a fractional differential equation containing derivatives of non-integer order. Such systems are said to have fractional dynamics. Derivatives and integrals of fractional orders are used to describe objects that can be characterized by power-law nonlocality, power-law long-range dependence or fractal properties. Fractional-order systems are useful in studying the anomalous behaviour of dynamical systems in physics, electrochemistry, biology, viscoelasticity and chaotic systems.

Example

The simplest example of a fractional-order filter is the LC ladder filter (in this case in Foster I topology). Here is a spice simulation of a LP and a HP passive LC Ladder fractional filters, all R and C values are calculated using the MatLab script provided in [15].





As we can see the slope of these HP and LP filters is more or less -2dB/dec!

Another simple example of (active) fractional filter is the fractional-order integrator.
The fractional-order integrator is a modification of the standard integrator circuit, where a capacitor is used as the feedback impedance on an op. amp. By replacing the capacitor with an RC Ladder circuit, we can obtain a fractional-order integrator.

Implementation topologies

While transfer function defines the characteristics of the filter, the topology defines the electronic circuit of the filter. Filter topologies may be divided into passive and active types. Passive topologies are composed exclusively of passive components: resistors, capacitors, and inductors. Active topologies also include active components (such as transistors, op amps, and other integrated circuits) that require power.

There are a lot of different types of analog filter topologies, the principals are:

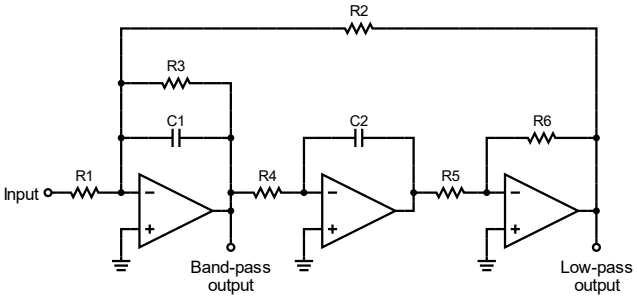
Passive topologies:

- Ladder topologies
- Modified ladder topologies (Usually the design applies some transform to a simple ladder topology)
- Bridged-T and Zobel Networks topologies

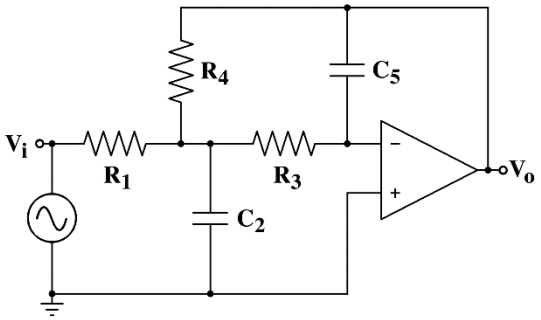
Active topologies:

- Multiple feedback topology (MFB)
- Biquad filter topology:
    1. Tow-Thomas filter
    2. Akerberg-Mossberg filter

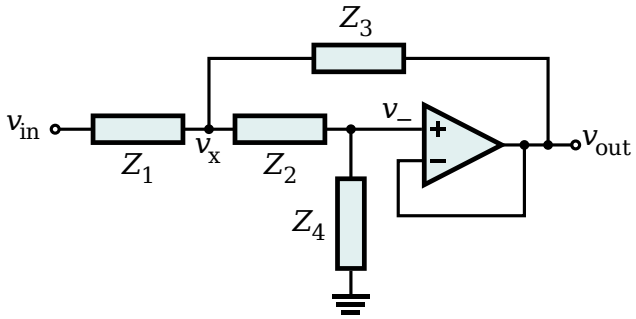
- Sallen–Key topology
- Switched capacitors topology



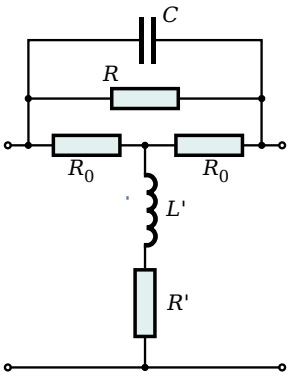
Example of Multiple Feedback topology using op. amp.



Example of Tow-Thomas biquad filter topology using op. amp.



Generic Sallen-Key filter topology using op. amp.



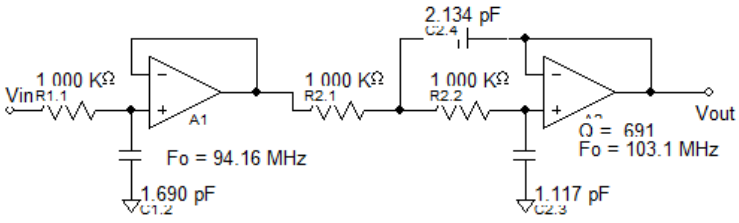
Example of Bridget-T Zobel Network.

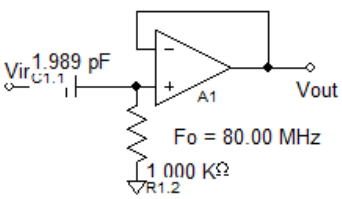
Because of the complexity of some of these topologies (and in our case probably doesn't makes any difference) we will try to implement only a few.

Sallen-Key

The first topology that we can implement is the Sallen-Key: for the LP filter we can start (as described in [24] 3.1) from a single pole section Sallen-Key and add a complex pole Sallen-Key section to obtain the 3° order filter. We then use a calculator to help us solve the equation to get our wanted poles and zeroes in order to have the wanted frequency response.
Same consideration applies for the 1° order HP filter.

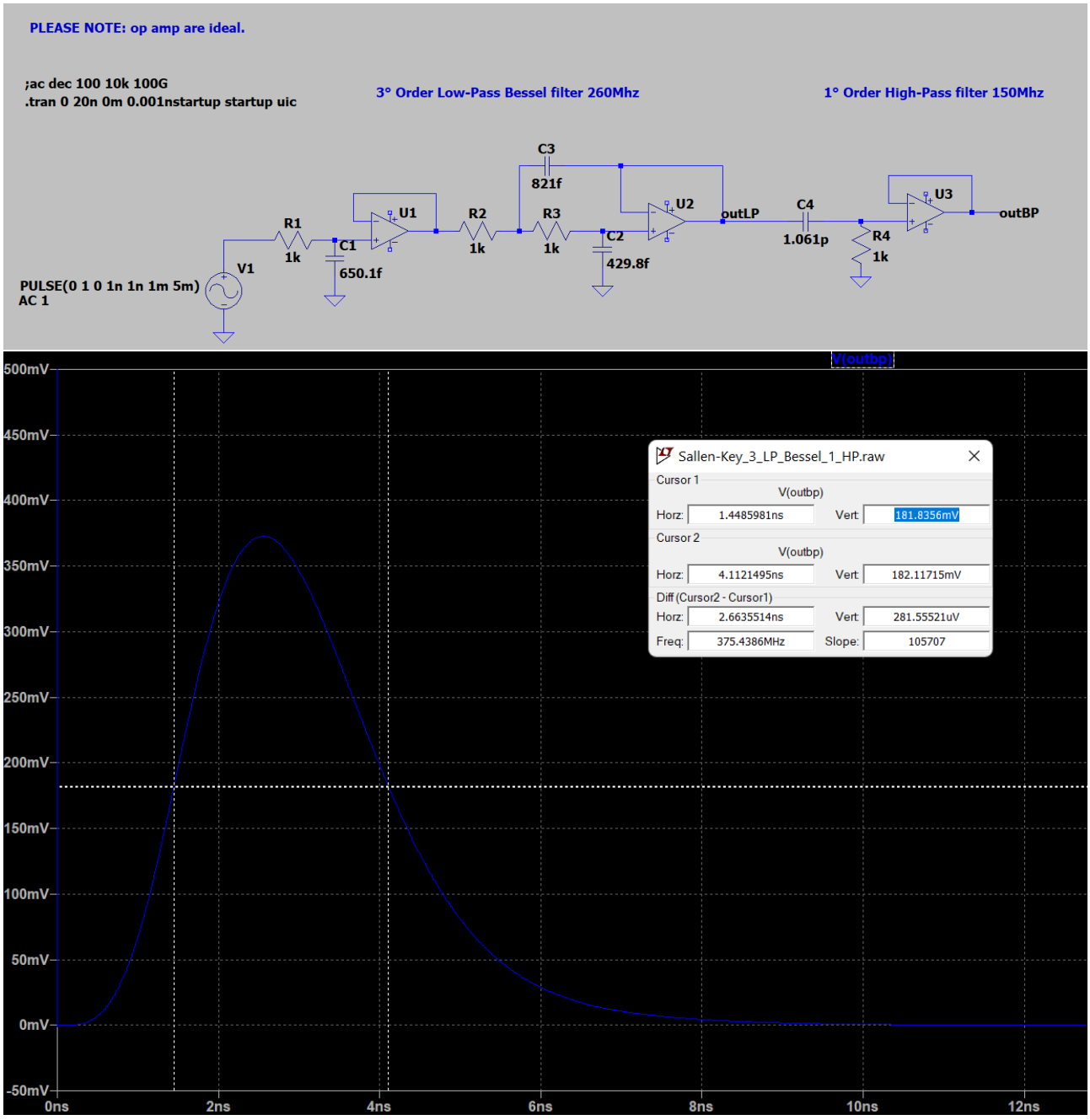
The result for the LP with $f_{cl} = 100MHz$ (left) and for the HP with $f_{ch} = 80MHz$ (right) is the following:





After some tests, as seen before, I observed that the HP filter is responsible for falling edge of the gaussian shape in the time domain, while the LP is responsible for the rising edge.
Obviously, we must maintain the band pass width proportion, otherwise the gaussian shape will be distorted. So, as seen before the optimal proportion was $1.9 - 3.3\frac{rad}{s}$ for a frequency response centred in $\omega_o = 2.57\frac{rad}{s}$. **(HP cut-off is about $0.739 * \omega_o$ and LP cut-off is about $1.284 * \omega_o$)**
If we want our gaussian to be 5nS long (about 2nS of FWHM) we can centre our frequency response in about $f_0 = \frac{1}{5*10^{-9}} = 200MHz$ and so have a $f_{cHP} \cong 150MHz$ and a $f_{cLP} \cong 260MHz$.

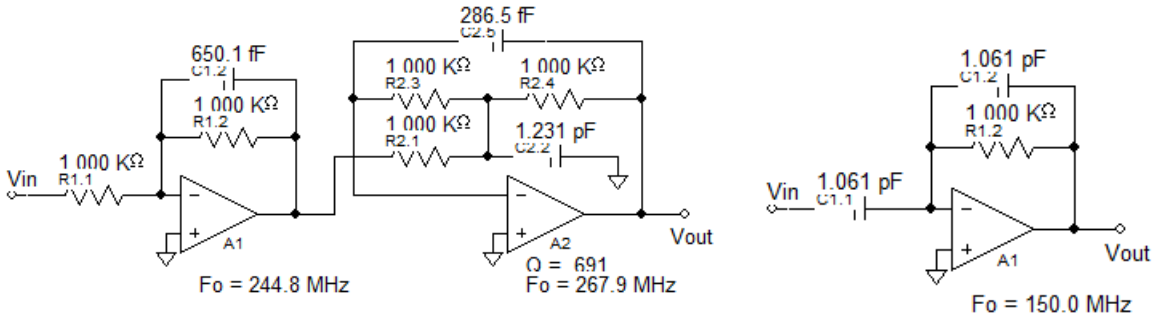
Here is the Spice simulation result with a pulse (step) of $1nS$ rising/falling edge and $1mS$ duration



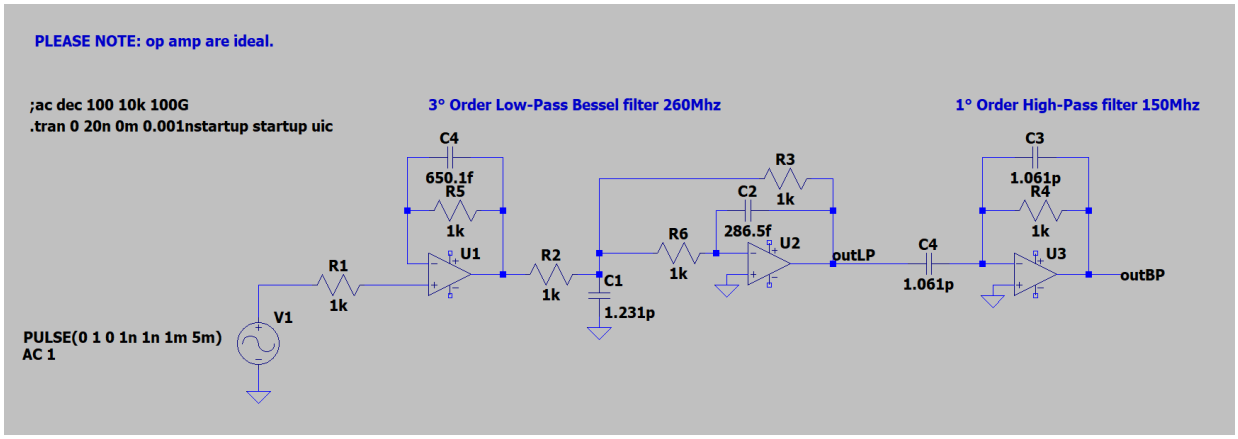
As we can see result are really good, we have no overshoot at all! Note that with some easy tuning we can almost immediately reach 2nS of FWHM.

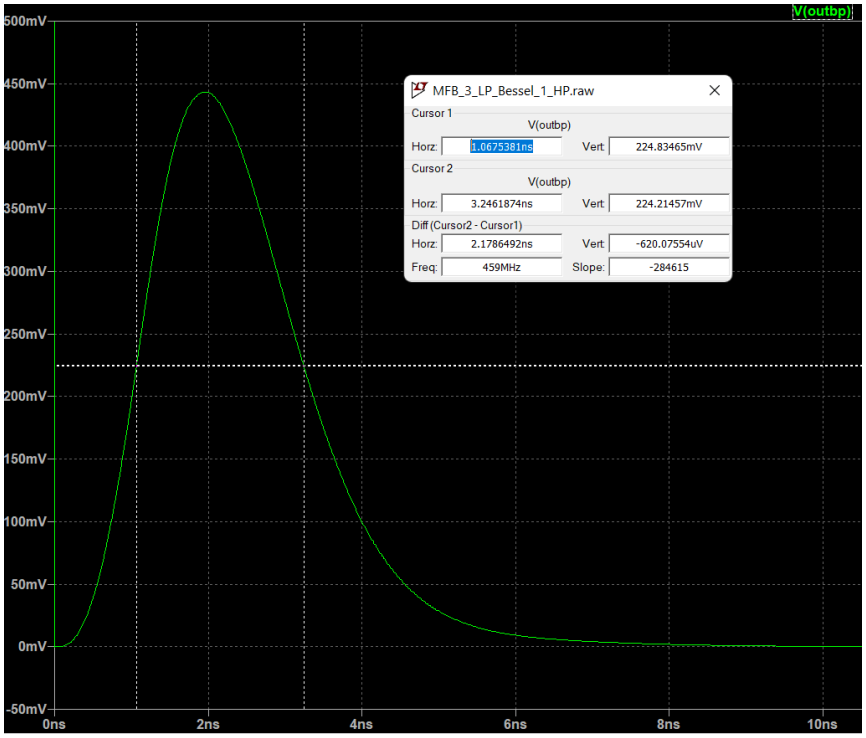
MFB

Let's try also MFB with same cut-off frequencies and filter orders and again we can use a calculator to help us finding wanted poles and zeroes.
The result for the LP with $f_{cl} = 260MHz$ (left) and for the HP with $f_{ch} = 150MHz$ (right) is the following:



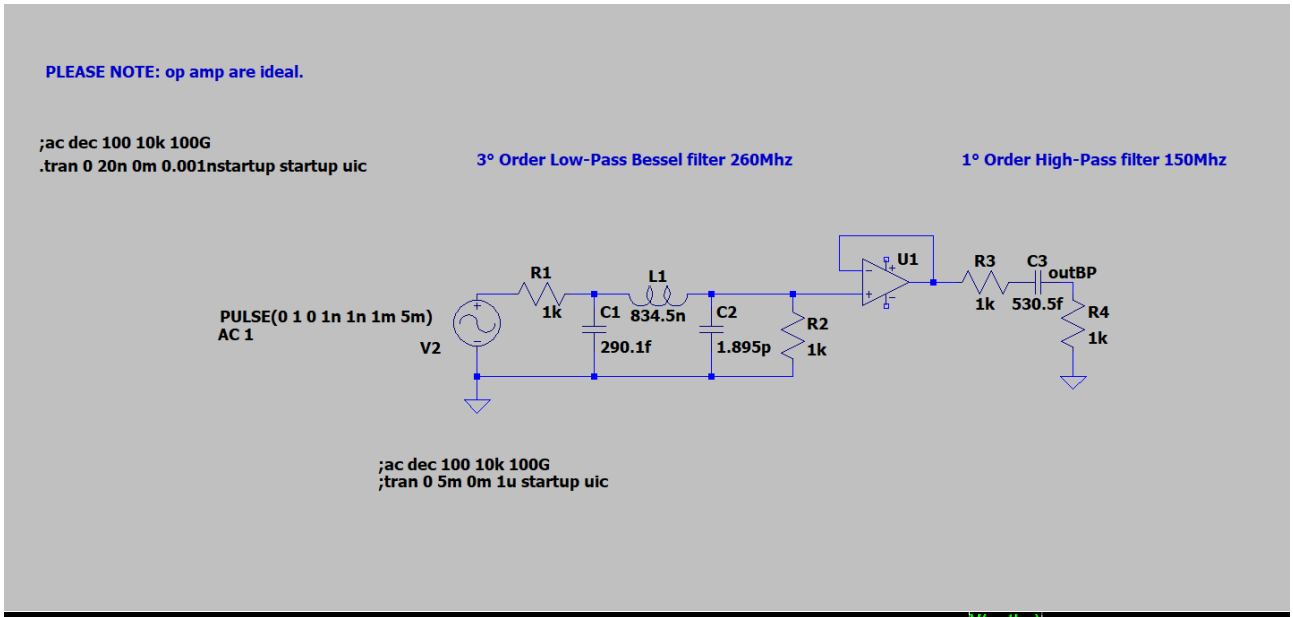
This is the Spice simulation result with a pulse (step) of $1nS$ rising/falling edge and $1mS$ duration:

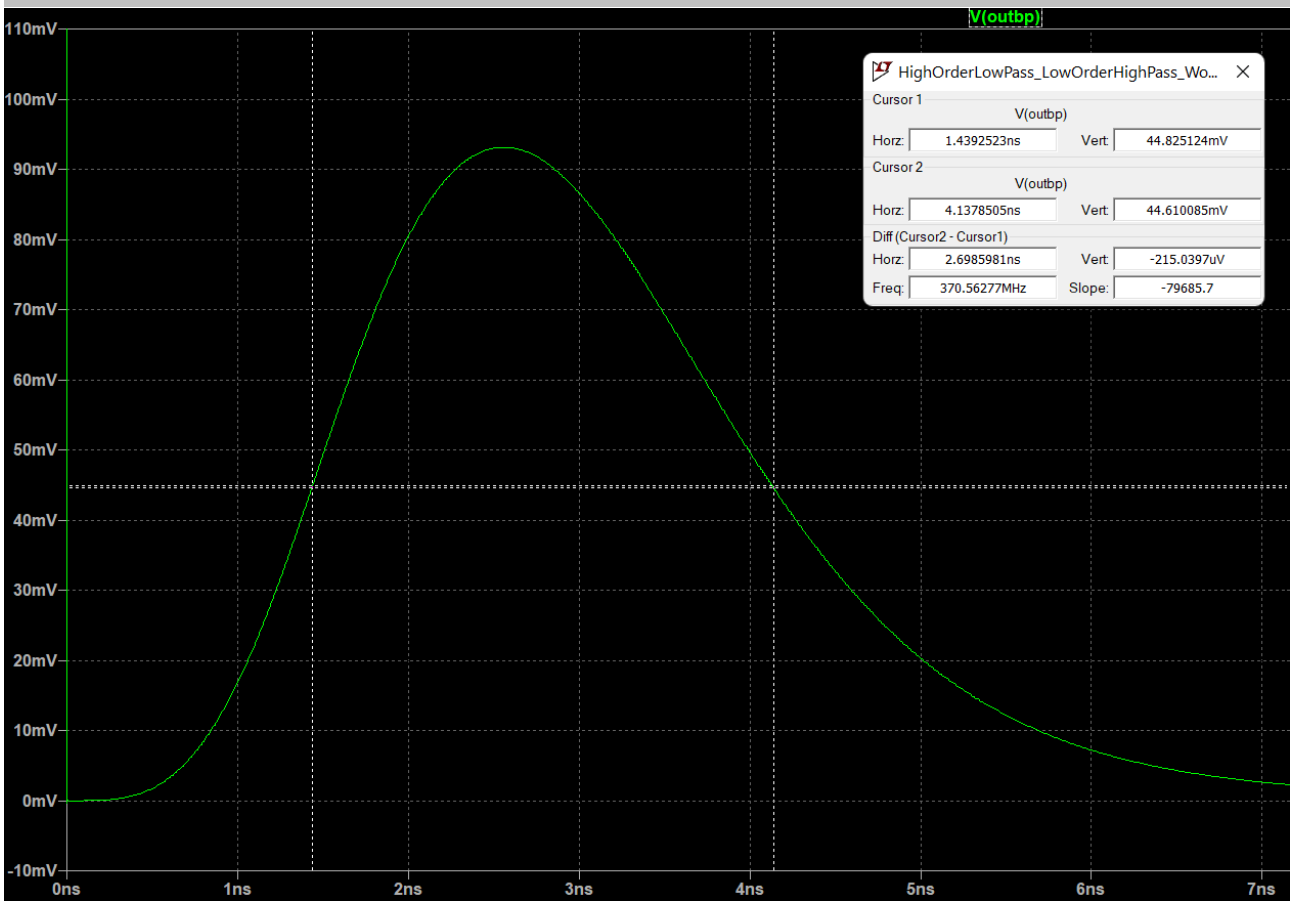




Passive Ladder

Of course, same result can be obtained with a passive filter, here is the Spice demonstration:
(3° Order LP Bessel in the left, 1° Order HP Bessel in the right, cascaded with the help of a buffer)





PLEASE NOTE: all the previous circuit implementations were calculated with all $R = 1Kohm$, so the value of capacitors is probably too small to implement it in reality (parasitic capacitance can easily be in the order of pF or tens of pF), anyway, reducing the value of R can partially solve this problem.

PLEASE NOTE 2: all the previous capacitors and resistors values are calculated values and they need to be rounded to the nearest available commercial value (see E6, E12, E24 etc... series) in order to implement a real circuit.

PLEASE NOTE 3: all the op amps used in previous simulation were ideal op amps (they can work without power supplies, they don't saturate, they have infinite GBW, infinite Slew Rate, infinite Open Loop Gain and no parasitic capacitances).
Thus, in a real circuit we must consider non-idealities and it can require expensive high speed operational amplifier as well as low tolerance passive components.

A "real" simulation

Let's see how we can implement our circuit in a simulation, with real components, that reflects reality. First we need a high-speed Operational Amplifier like AD8009 by Analog Devices, this is a screenshot from the datasheet:

FEATURES
Ultrahigh Speed
- 5,500 V/μs Slew Rate, 4 V Step, G = +2
- 545 ps Rise Time, 2 V Step, G = +2
- Large Signal Bandwidth
  - 440 MHz, G = +2
  - 320 MHz, G = +10
- Small Signal Bandwidth (–3 dB)
  - 1 GHz, G = +1
  - 700 MHz, G = +2
- Settling Time 10 ns to 0.1%, 2 V Step, G = +2

Low Distortion over Wide Bandwidth
- SFDR
  - –66 dBc @ 20 MHz, Second Harmonic
  - –75 dBc @ 20 MHz, Third Harmonic
- Third Order Intercept (3IP)
  - 26 dBm @ 70 MHz, G = +10

Good Video Specifications
- Gain Flatness 0.1 dB to 75 MHz
- 0.01% Differential Gain Error, R_L = 150 Ω
- 0.01° Differential Phase Error, R_L = 150 Ω

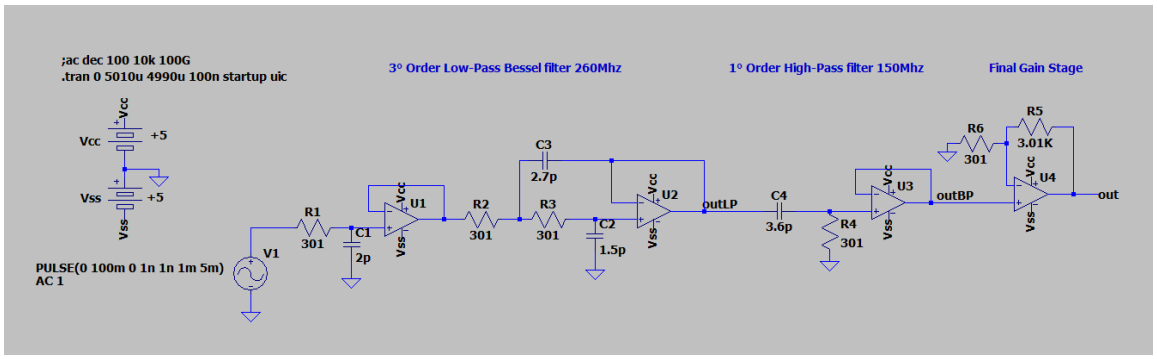
High Output Drive
- 175 mA Output Load Drive
- 10 dBm with –38 dBc SFDR @ 70 MHz, G = +10

Supply Operation
- +5 V to ±5 V Voltage Supply
- 14 mA (Typ) Supply Current

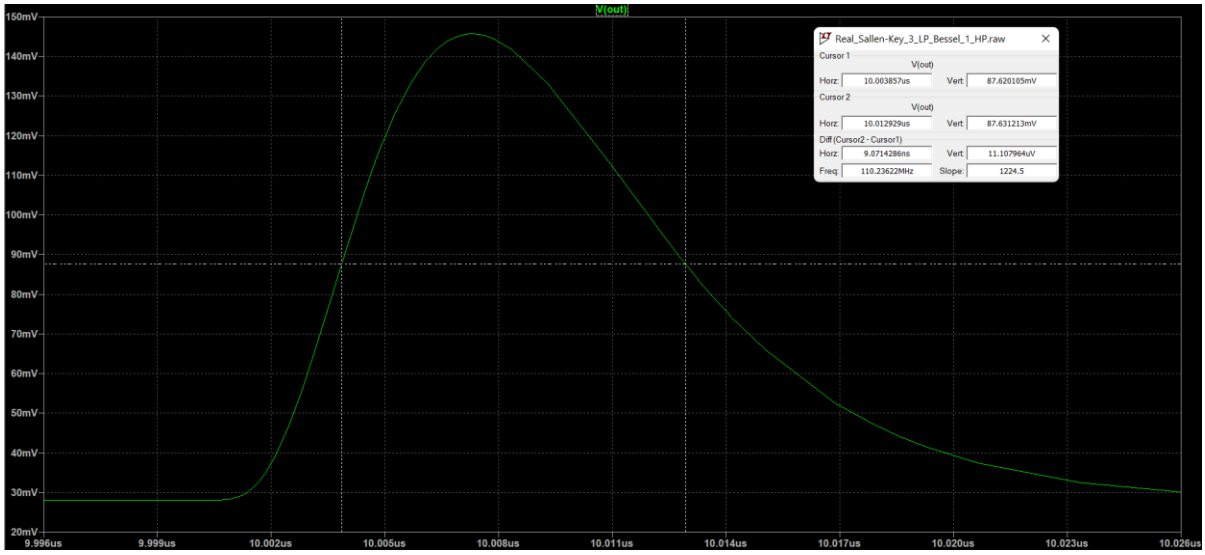
APPLICATIONS
Pulse Amplifier
IF/RF Gain Stage/Amplifiers
High Resolution Video Graphics
High Speed Instrumentations
CCD Imaging Amplifier

Next, we must round resistors and capacitors values to the nearest preferred values (we will use E96 series with 1% tolerance). Considering the parasitic capacitance, we better lower the resistors value and increase capacitors value.

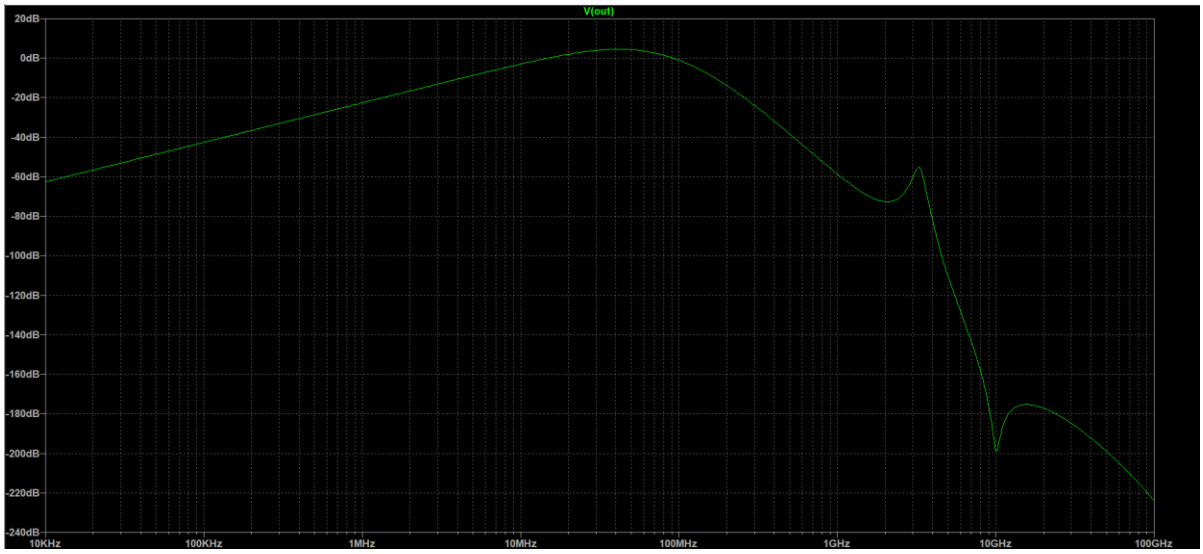
For the simulation I will use the previous discussed Sallen-Key topology.
This is the resulting circuit (note I added a final gain stage because this time input step signal is 100mV):



The result is the following:



As we can see, the gaussian shape is a bit distorted because of real Op Amp frequency response, but anyway it's pretty decent. Note that the FWHM is increased and it's about $9\sim10\ nS$ and that's because the GBW of the Op Amp is not sufficient to amplify frequencies above $200MHz$.
Note also the offset of about $30mV$ due to offset voltage and bias currents (it can be easily removed with some precautions like equalling equivalent resistance viewed from input nodes and/or adding trimmers to resistors or just filtering low frequencies at the output of last stage).
This is the frequency response of the circuit above:

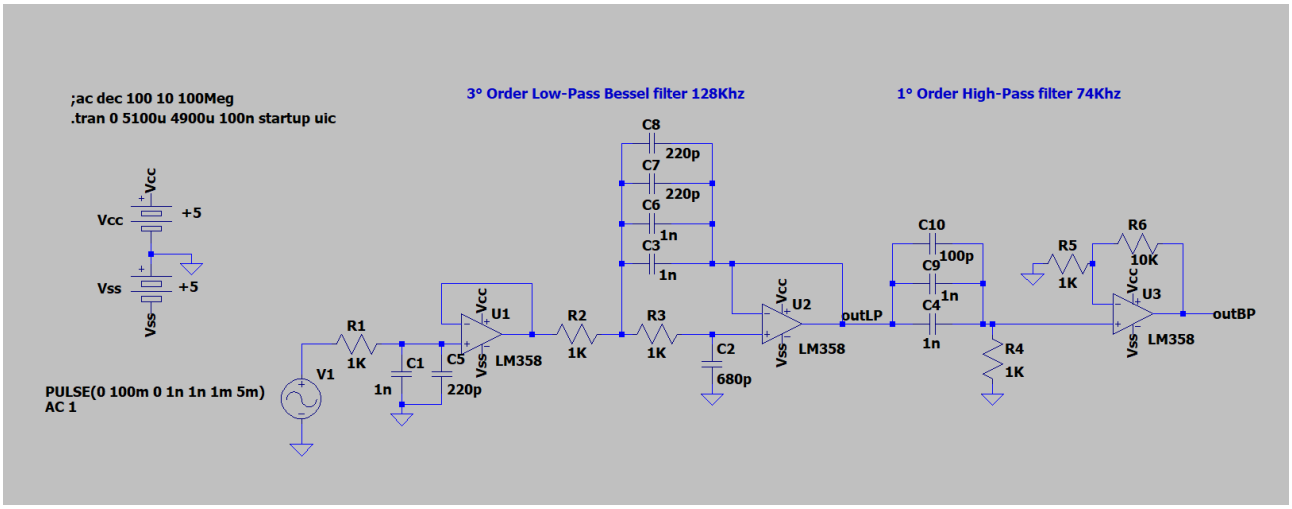


As we can see this is not our ideal wanted frequency response, even if it's really close to it, and this is the cause of our time domain gaussian distortion. This is caused by the rounded values of the passive components and by the Op Amp non-idealities.

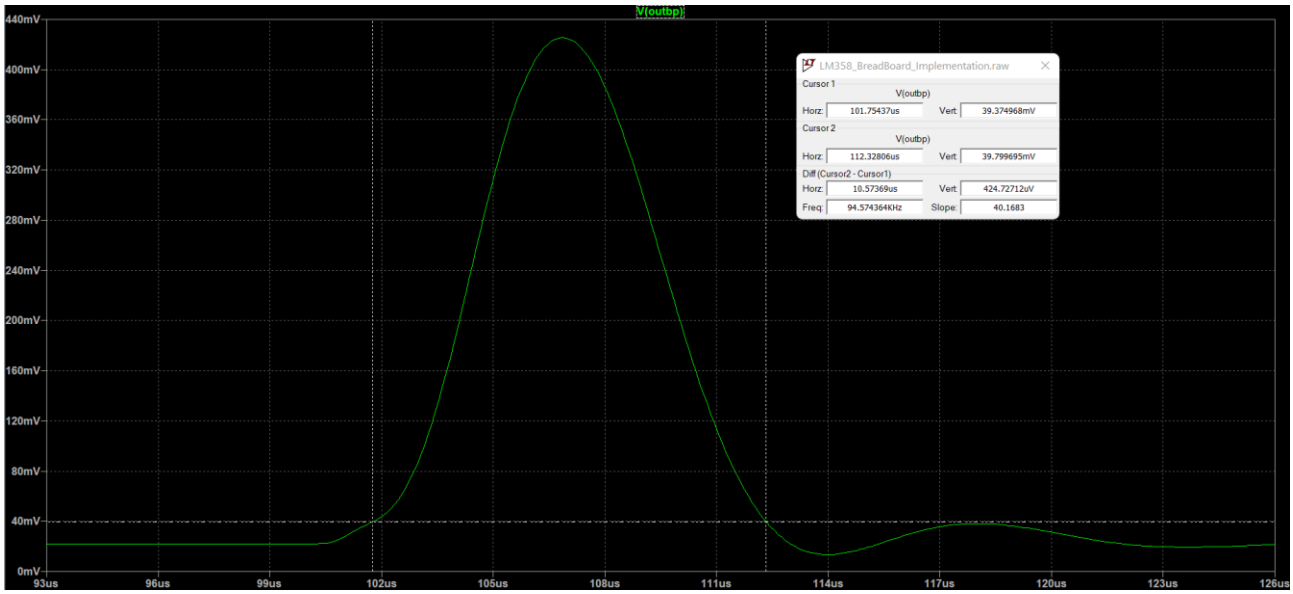
Breadboard implementation

We are now ready to set up a test on a breadboard with some real components. Since I haven't got any high-speed Operational Amplifier or capacitor with value below 10pF I decided to do the test with a longer gaussian shape: about $10\mu S$.

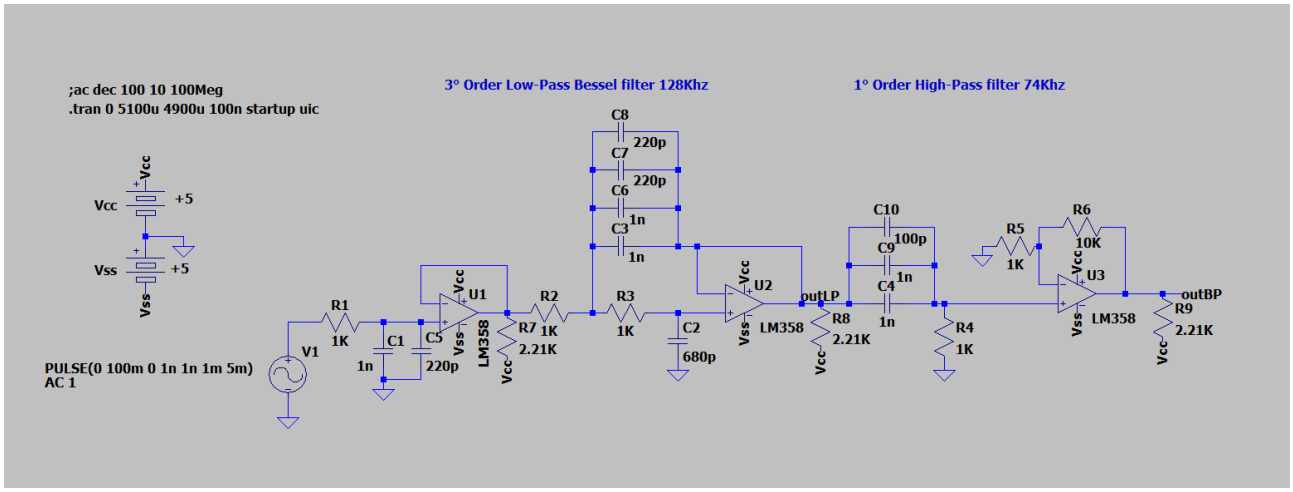
For the operation amplifiers I decided to use a rather old but also very cheap good friend: LM358P.
Following our previous calculations, $f_0 = \frac{1}{10*10^{-6}} = 100KHz$, $f_{cHP} = 0.739 * f_0 \cong 74KHz$ and $f_{cLP} = 1.284 * f_0 \cong 128KHz$.
I used parallel capacitor in order to approximate the calculated capacitance value with my available capacitors. I then modified last Op Amp stage in order to increase Gain.
The resulting schematic is the following:



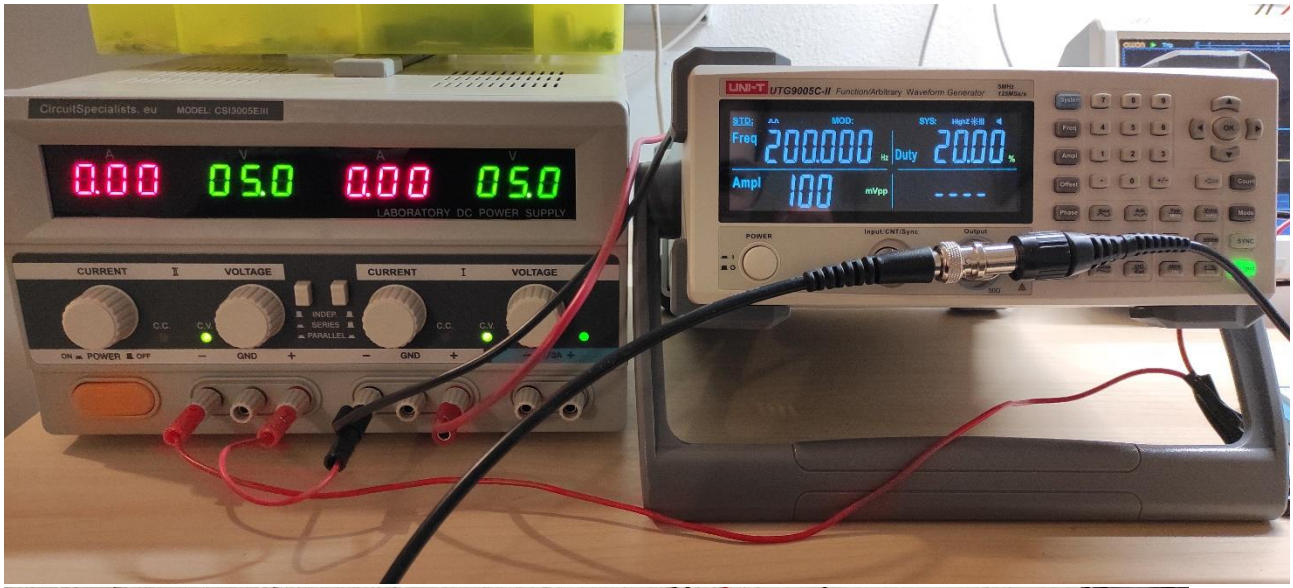
This should be this:

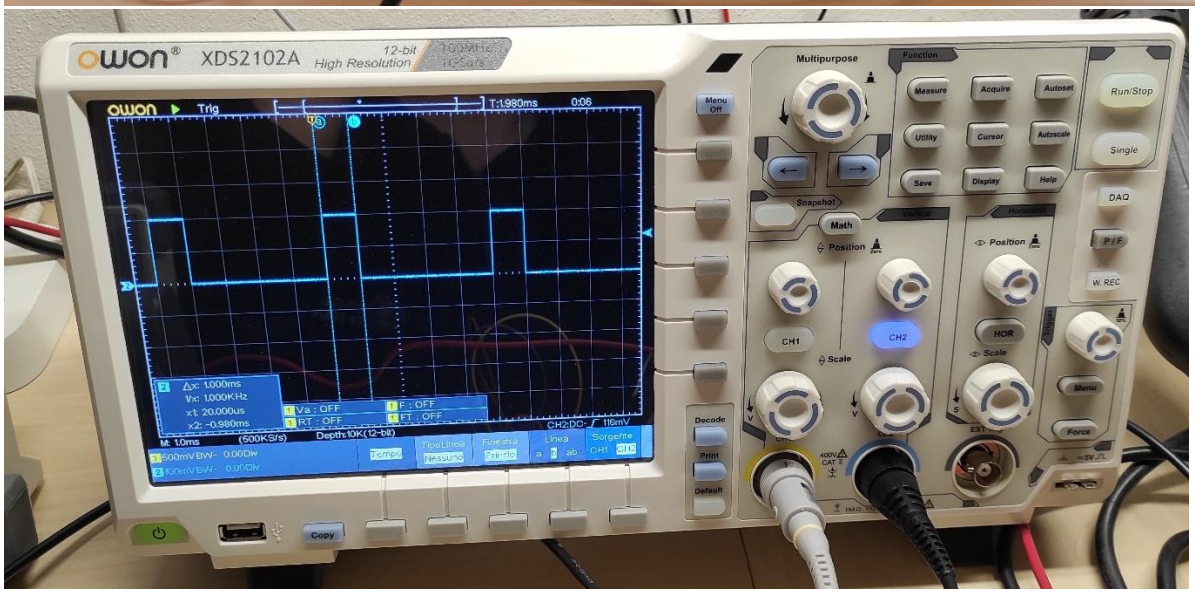


After some tests, I found out that every Op Amp require a resistor from output to one of the power rail in order to maintain stability (otherwise it starts oscillating).
So, the final circuit is the following:

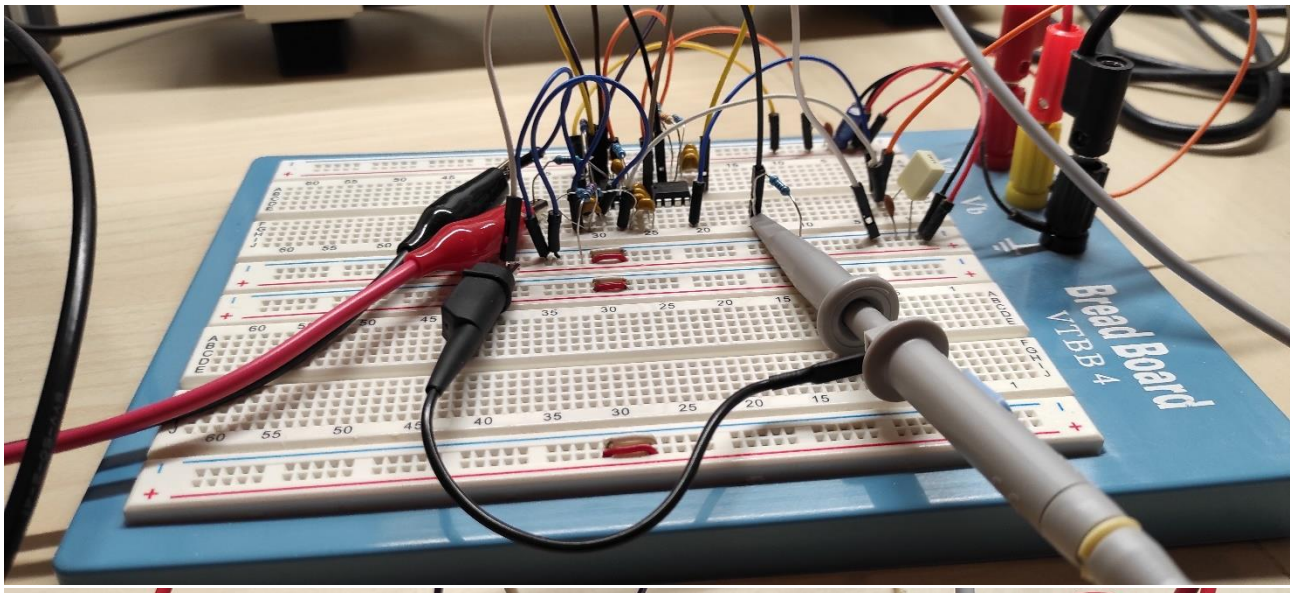


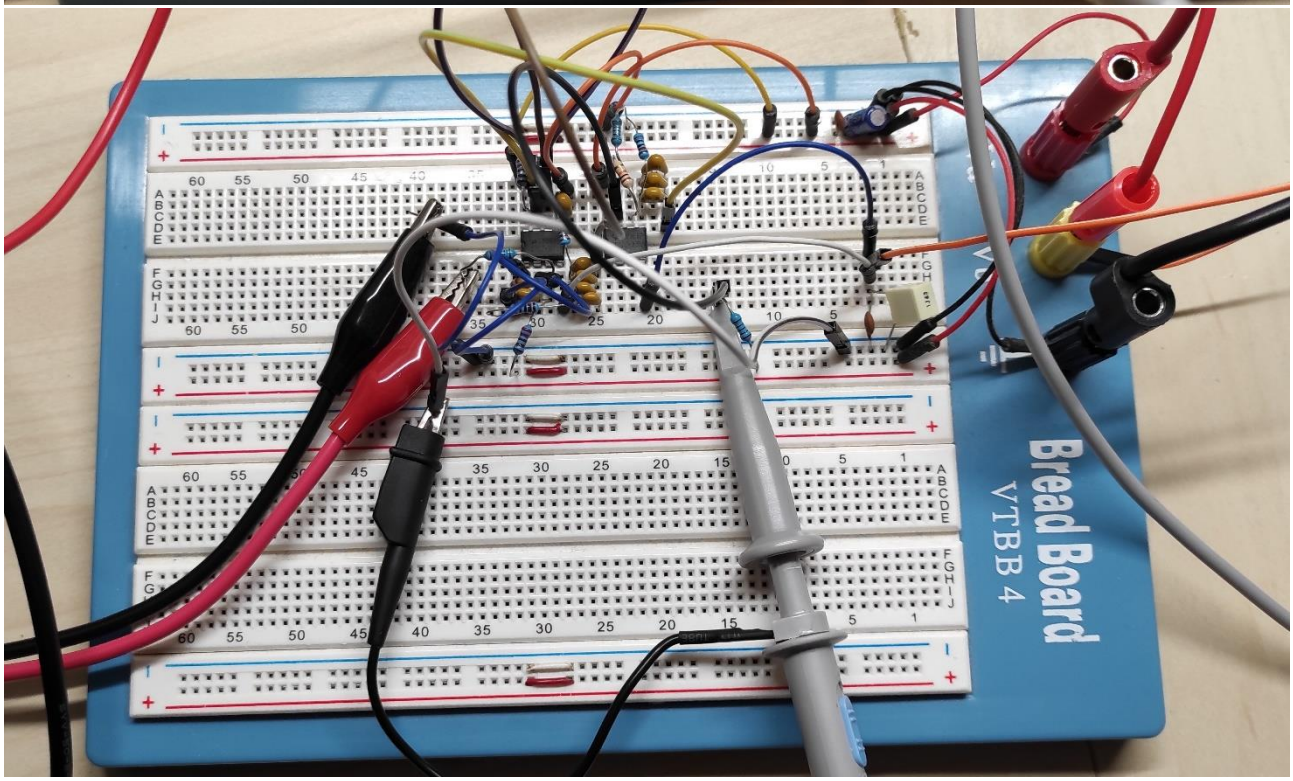
The setup used for the test was composed by a linear power supply with $Vcc = +5V$ and $Vss = -5V$ for the Op Amp power rails, a signal generator with a Pulse signal with $f = 200Hz$ ($5mS$ period) and Duty Cycle 20% ($1mS$ ON time) with amplitude $100mV$ and offset $50mV$ and a oscilloscope:



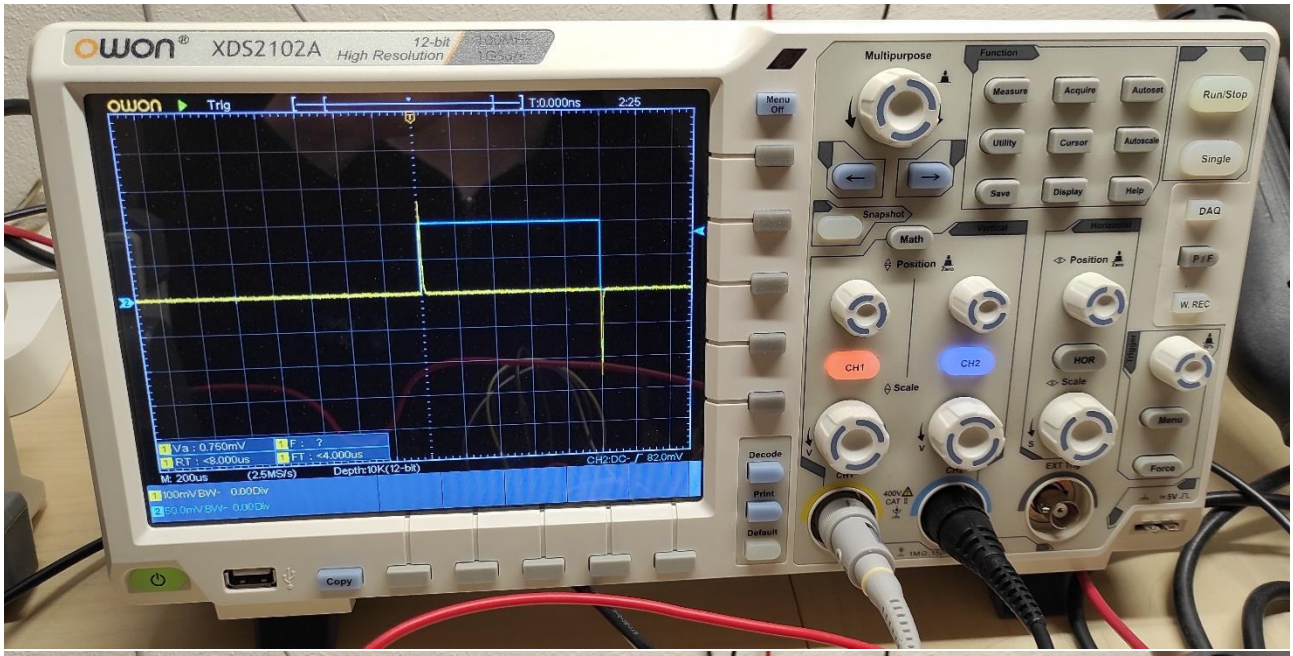


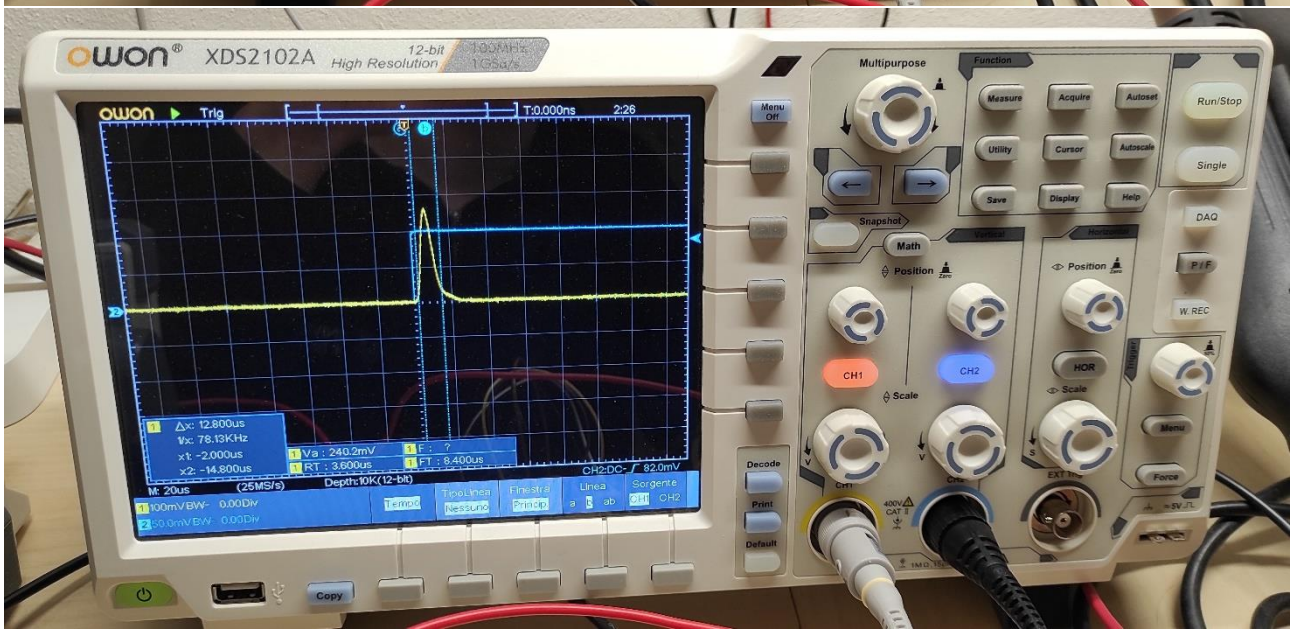
This is the sketchy circuit mounted on a breadboard:





And finally, this is the output waveform (output signal is in yellow, input signal in blue):





As we can see, although the low-end components and the sketchy circuit, the result is pretty decent.
The length of the "gaussian" waveform is about $12{\sim}13\mu S$ and with some simple adjustments we can shrink it to $10\mu S$.

Final Conclusions

We can finally assert that the discussed method is fully working, and with appropriate components (ASICs high speed Op Amps and low tolerance components) and approximations (5° or 8° order) we can reach gaussian shapes in the order of nS or maybe even below.

Bibliography

[1] https://en.wikipedia.org/wiki/Pulse_shaping
[2] https://kh6htv.files.wordpress.com/2015/11/an-07a-risetime-filters.pdf
[3] https://en.wikipedia.org/wiki/Gaussian_filter
[4] https://it.wikipedia.org/wiki/Filtro_Sallen-Key
[5] https://en.wikipedia.org/wiki/Filter_(signal_processing)
[6] https://en.wikipedia.org/wiki/Electronic_filter_topology
[7] https://en.wikipedia.org/wiki/Bessel_filter
[8] https://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter8.pdf
[9] https://en.wikipedia.org/wiki/Impulse_response
[10] https://en.wikipedia.org/wiki/Step_response
[11] https://it.wikipedia.org/wiki/Funzione_di_trasferimento
[12] https://www.dsprelated.com/freebooks/filters/Phase_Group_Delay.html
[13] https://www.rfcafe.com/references/electrical/gaussian-proto-values.htm
[14] https://www.rfcafe.com/references/electrical/gaussian-poles.htm
[15] https://www.sciencedirect.com/science/article/pii/S1434841117305800?casa_token=10mGfYFJJ-kAAAAA:e1gKW8Sm9SVIPuhTfh3u77-6jHKNlFvKn1dgnAA3qLp8CYrT0AuGOsJ05o0CQtB9IgdY9Ize_t4k
[16] https://www.sciencedirect.com/topics/engineering/bessel-filter
[17] https://en.wikipedia.org/wiki/Electronic_filter_topology
[18] https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Microwave_and_RF_Design_IV%3A_Modules_(Steer)/02%3A_Filters/2.09%3A_Filter_Transformations#:~:text=The%20corner%20frequency%20of%20the,%CF%892%E2%88%92%CF%891).
[19] https://en.wikipedia.org/wiki/Fractional-order_system
[20] https://en.wikipedia.org/wiki/Fractional-order_integrator
[21] https://www.researchgate.net/publication/326275945_Noise_analysis_of_fractional-order_two-integrator_CCII_low-pass_filter_using_Pspice
[22] https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Microwave_and_RF_Design_IV%3A_Modules_(Steer)/02%3A_Filters/2.09%3A_Filter_Transformations
[23] https://electronics.stackexchange.com/questions/245766/step-response-of-2nd-order-hpf
[24] https://www.ti.com/lit/an/sbfa001c/sbfa001c.pdf?ts=1645096937965&ref_url=https%253A%252F%252Fwww.google.sk%252F
[25] https://www.mouser.it/datasheet/2/609/AD8009-1502068.pdf
[26] https://pdf1.alldatasheetit.com/datasheet-pdf/view/22771/STMICROELECTRONICS/LM358P.html