Digital Filters For Music Synthesis

Kasper Nielsen
karmafx@karmafx.net

KarmaFX, January 2000[*]

ABSTRACT

The goal of this paper is to explore the use of digital filters for music synthesis, in particular subtractive synthesis. An introduction to filters and their use in synthesizers is presented. The related signal processing theory for low order digital IIR filters is described, and simple filters are designed directly in the digital domain. A set of guidelines related to the control and implementation of a filter is discussed, and possible design method for extending and transforming filters are explored. Several digital IIR filters based on analog prototype filters are explained, and the filters advantages and disadvantages are discussed. The covered analog filters are the Sallen & Key filter, the Butterworth filter and the State Variable Filter. Issues related to the implementation of digital filtering is described, and the filters' behavior when implemented, is tested and compared. The State Variable Filter is found to have properties, with regard to both functionality and speed, that in relation to computer music synthesis makes this filter very advantageous.

1. INTRODUCTION

In the last 30 years it has become widespread to use electronic-instruments or so called synthesizers to make music. Some of the first synthesizers were built using components from analog electronics, but today the majority of synthesizers use digital components. However most of the original design ideas used in the early synthesizers are still used today.
A synthesizer is internally built up of components, each performing different tasks related to the creation and shaping of sounds. At least this is the case for *subtractive synthesis* which we will concentrate on in this paper.
A simple subtractive synthesizer can roughly be said to be built up of three components: An oscillator, a filter and an amplifier [2]. The oscillators job is to create a signal, for instance a square wave, at a specified frequency. This is sent into the filter which removes or subtracts (hence the name: subtractive synthesis) certain frequency components from the signal. This signal is then sent into the amplifier which scales the signal according to a specified amplification factor. See Figure 1.

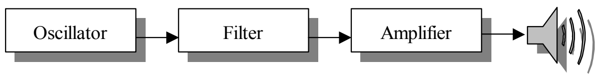


Figure 1

Furthermore the parameters for the different components can be independently changed or modulated by other components. For instance, if the used filter is a lowpass filter, the cutoff frequency can be swept from one frequency to another during a certain time-span. This type of filter is called a Voltage Controlled Filter (VCF), since it can be controlled and modulated by a voltage in an analog system. Similarly the amplifiers amplification factor can be changed over time to shape the volume of the resulting sound. The resulting signal can finally be sent to a hooked up sound system.
Although this scheme is a simplified version of the actual setup in a modern synthesizer, it can be used to create a wide variety of lifelike sounds as well as unnatural sound effects.

2. OVERVIEW

Today standard desktop computers have become so fast that it is possible to simulate most of the work a synthesizer does. As computers become more powerful in the future it will be advantageous to be able to implement synthesizer components in software for simulating a synthesizers behavior and thus use the computer to create sounds in real-time. To do this we need to implement each of the described synthesizer components in software. Fortunately most of these components can be easily implemented in software. The oscillator is a simple

[*] Written at Department of Digital Signal Processing, IMM, Technical University of Denmark, January 2000.

wave generator, preferably band limited. The amplifier is a matter of multiplying the signal by a scale-factor. The most complicated component is the filter.

Filter analysis and construction is a well studied area in digital signal processing. Thus we are interested in using the filter knowledge from digital signal processing to construct digital filters that have the same properties as filters in standard synthesizers.

In this paper we will examine different methods for constructing and implementing such filters. To get a better idea of what filters must be able to do, we can take a look at the specifications for some of the most common filters used in well-known synthesizers. The filters used are standard lowpass and highpass filters and sometimes also bandpass and notch filters [2].

The filters are usually specified to be 2 og 4 pole filters, that is, they have a 12dB/24dB rolloff per octave respectively [3,10]. The most common filter parameters is cutoff(freq), resonance(reso), modulation (mod) and keyboard tracking/follow (kybd). The Cutoff parameter is used to set the cutoff frequency of the specific filter (usually at -3dB attenuation). For a lowpass filter the resulting frequency response can be shown graphically as shown in Figure 2. Figures like this are usually shown in synthesizer manuals [10].

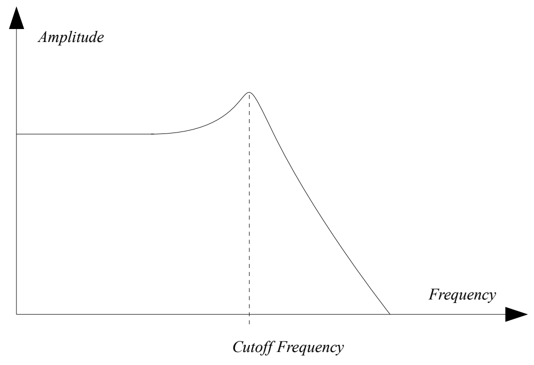


Figure 2

The resonance parameter or quality factor (Q-factor) is used to enhance certain frequency components, i.e. introduce resonance much like a digital resonator at the cutoff frequency. Changing the resonance parameter affects the frequency response as shown in Figure 3. A high resonance setting introduces a frequency peak at the cutoff point.

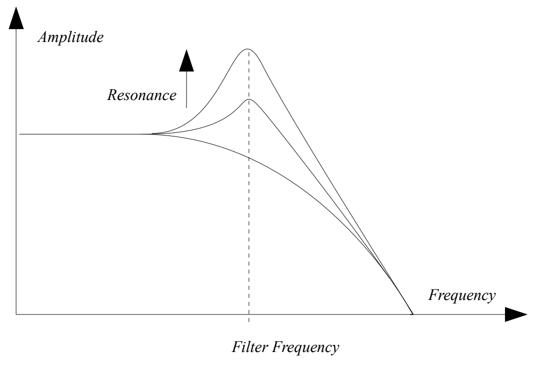


Figure 3

The Modulation parameter is used to control the amount of modulation performed on the cutoff parameter, that is, scale the amount of the cutoff filter- sweep. The sweeping of the filter is usually controlled by an envelope or low frequency oscillator [8,2]. This effect is shown in Figure 4. When the modulation parameter is set to zero no modulation is performed.

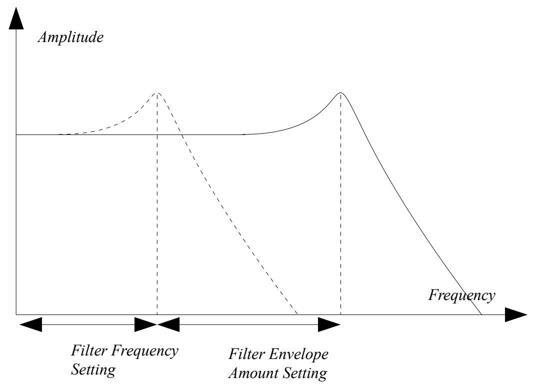


Figure 4

The Kybd parameter is used to set the amount of keyboard tracking, which is a simple adjustment of the cutoff frequency to make it scale according to the frequency of the oscillator [2,10]. Thus the filters cutoff frequency can be controlled by the pitch of a key played on the synthesizer. This is shown graphically in Figure 5.

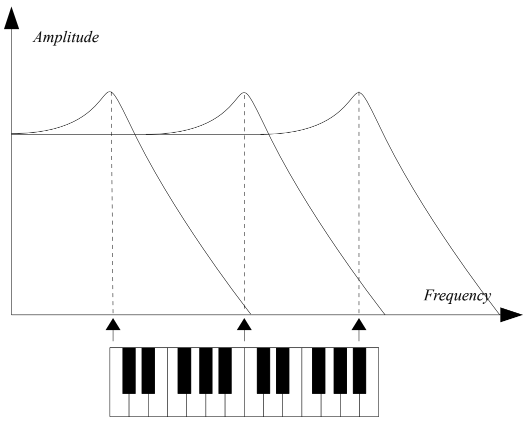


Figure 5

As we can see the mod and kybd parameters are simply used to adjust the cutoff parameter and therefore not related to the implementation of the actual filter. Thus the filter needs to have two parameters: *Cutoff* and *Resonance*. And we will initially limit our filter designs to *lowpass* filters, since highpass, bandpass and notch filters can be derived from this.

To specify exactly what a designed filter should be able to do, we will try to list a series of design specifications.

Design specifications:

1. The filter must be lowpass filter. For simplicity we will start by concentrating on discussing construction of a lowpass filter. Later on we should be able to convert our design to a highpass filter.

2. The filter must behave and sound much like a synthesizer filter. This is actually an important point, since our goal is to simulate a synthesizer filters *look and feel* in sound. We should note however that filters are implemented very differently on different synthesizers. It is common that two apparently similar filters do not sound the same on different synthesizers.

3. The filter must have a *cutoff parameter*. The range of the cutoff frequency is usually between the lowest and the highest allowed frequency. If we are using a samplerate of say 44.100 kHz, then a sensible maximum for the cutoff frequency is around 22.050 kHz (*Nyquist*). Thus when the cutoff is at it highest for a lowpass filter the filter should have basically no effect on the signal, i.e. no frequencies are cut off. Furthermore it should be possible to sweep the cutoff frequency.

4. The filter must have a *resonance parameter*. The range of the resonance parameter differs from synthesizer to synthesizer. Some synthesizers let the resonance be tweaked so high that the filter introduces self-oscillation, and can at this state actually be used as a sinewave generator. When resonance is set to zero, no resonance is applied to the signal. The change of cutoff or resonance in a filter is generally known as *tuning* the filter.

5. The filter's roll-off should either be 12dB or 24dB or something in between. This is not a strict design criteria though, since this might unnecessarily limit our filter design. It would be an advantage if the filter could be designed so it easily scales to different roll-offs, for instance by cascading two, three or four similar filters. This would also make it possible to connect similar filters in both parallel and series, which would be an advantage when creating actual filter-banks.

6. The filter must be fast to apply to a signal. This is a very relative design specification, but basically this tells us that we should try to limit the amount of CPU processing used when applying the filter, i.e., limit the amount of multiplications/additions. Furthermore since the filter is sweepable, some internal calculations of the filter-coefficients may be necessary , which in a worst-case scenario will need to be evaluated for each sample. These calculations should also be fast. The goal is be able to comfortably apply one or more filters in real-time on a standard Pentium PC while other sound processing tasks run concurrently.

7. The implemented filter should avoid quantization effects to minimize quantization noise and quantization of filter coefficients, i.e. have a high dynamic range for amplitude for improved SNR. This is however a design criteria for the implementation of the filter which may vary, and will be discussed further in the implementation section. In this paper we will assume an overall high dynamic range.

8. The filter will have some kind of phase response. We will not limit our design to a specific phase response here. Since we are not aware of the phase response in real synthesizer filters, we will discuss phase related issues later on.

9. The filter should have some kind of automatic gain control that will ensure that the filtered sound will sound just as loud as the unfiltered sound. This is directly related to normalization of the frequency response.

3. DESIGN

In Digital Signal Processing there exists many different methods for filtering a signal. We are interested in exploring methods for designing filters to obtain a desired frequency response.
In general we want to design a system that convolves a time-domain input signal *x(n)* with an impulse response *h(n)* that describes the filter, to obtain a time-domain output signal *y(n)*. See Figure 6.

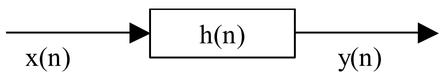


Figure 6

Mathematically we write this as

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

In the digital z-domain we can write this equation as

$$Y(z) = X(z)H(z) \Leftrightarrow H(z) = \frac{Y(z)}{X(z)}$$

where *H(z)* is the filters system function.

At first we will look at a filter as if all coefficients are constant, that is, the filters response isn't changed over time. This simplifies things, since we can view the filter as time invariant and focus our discussion on causal linear-time-invariant (LTI) systems.

Various methods exist for achieving this type of filtering in practice: by designing FIR and IIR filters directly in the digital domain; by working in the frequency domain using FFT and different methods for handling overlaps or by using well-known analog prototype filters: Butterworth, Chebychev, etc.

3.1 DIRECT DIGITAL DESIGN

To begin with we will concentrate on using a simple technique for designing a filter directly in the digital domain in an ad-hoc manner. The goal is to quickly and easily design a simple working filter that meets our design specifications. This will also work as a foundation for understanding any further filter designs.

We would like to directly implement a 2 pole or 4 pole filter in the digital domain. Due to the presence of poles, a corresponding pole-zero system is an IIR system [1].
The runtime of IIR filters is in general superior to FIR (all zero filter), since IIR has a substantial lower amount of coefficients and thus lower runtime than a FIR filter equivalent for obtaining the same frequency response [12].
We will start by stating some of the basic properties of IIR filters, and design some simple, but useful filters.

A LTI system described by a constant coefficient difference equation has a rational system function:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = b_0 \frac{\prod_{k=1}^{M}(1 - z_k z^{-1})}{\prod_{k=1}^{N}(1 - p_k z^{-1})}$$

where a_k and b_k are the filter coefficients, and p_k and z_k are the complex pole and zero coefficients respectively. b_0 is the overall gain constant which can be selected to normalize the frequency response at a specific frequency. As shown, the poles and zeros are the roots of the filter coefficients polynomial expressions [1].

For a first order system this equation becomes

$$H_1(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} = b_0 \frac{1 - z_1 z^{-1}}{1 - p_1 z^{-1}}$$

For a second order system (M=N=2) this transfer function becomes

$$H_2(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = b_0 \frac{(1 - z_1 z^{-1})(1 - z_2 z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})}$$

This function can be implemented as:

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2)$$

which in general is known as the *canonical second order structure* or *biquad* [5]. These second order structures (2 pole, 2 zero) can be used as building blocks for realizing higher order systems [1]. The direct form II structure for this system is shown in Figure 7.

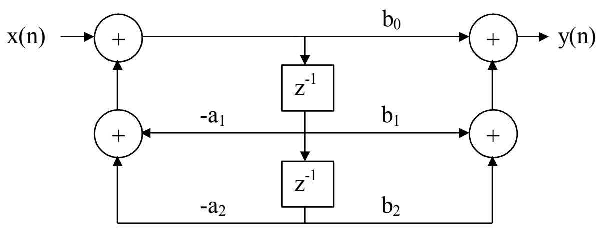


Figure 7

3.1.1. One Pole Filter

We will start by creating a simple one-pole adjustable lowpass filter. This type of filter can be realized by sweeping a pole from the center of the unit circle in the z-plane and along the x-axis to the unit circles peripheral. Since the pole is real we can represent it by the scalar *a*. Furthermore we select $b_0 = 1-a$ so the filter has unity gain at DC. The system function becomes [1]:

$$H_1(z) = \frac{1-a}{1-az^{-1}}$$

The filter frequency and phase response, along with the corresponding z-plane diagram for different values of *a* is shown in appendix A. We see that we, by adjusting the poles position, have trivial control of the filters cutoff frequency. The frequency response is shown as both magnitude response in a linear frequency scale and as a gain response in a log scale. We see that the filter has a roll-off of 6dB/octave.

The system is realized by the difference equation:

$$y(n) = (1-a)x(n) + ay(n-1)$$

Note that when $a=0$, the filter has no effect.

Thus the systems first order coefficients are $b_0 = 1-a$, $b_1=0$ and $a_1=a$.

There is a fixed relationship between *a* and the -3dB *cutoff frequency* f_c of the filter [13]:

$$a = e^{-2\pi f_c / F_s}$$

where F_s is the sample rate.

The structure of the filter can be sketched as shown in Figure 8.

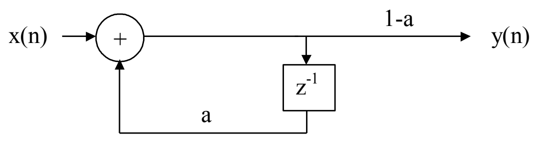


Figure 8

Usually this filter is used as

$$y(n) = cx(n) + (1-c)y(n-1) = y(n-1) + c(x(n) - y(n-1))$$

where $c = 1-a$, so that $c=1$ corresponds to $f=F_s/2$ and $c=0$ to $f=0$.

In this design we only use one pole for changing the frequency response. What we are actually doing is using the pole to enhance low frequencies (at ω=0) as *a* approaches 1. However, the normalization at DC turns this into a variable attenuation at higher frequencies. It is not possible to enhance frequencies at other positions than DC or Nyquist using only one pole [12]. To do this we will need to place a pole that does not lie on the x-axis, but since we want the filter to produce real output and not complex output, we need to place a conjugate pole at the corresponding negative frequency. Thus we have a two pole system.

3.1.2. Digital Resonator

We will now look at how we can design a two pole filter which introduces resonance at a specific frequency: a digital resonator. A resonator can be constructed using a pair of complex conjugate poles placed near the unit circle. The angular position ω_0 of the poles determine the resonant frequency [1], and the radius r determines the amount of resonance. As the poles get closer to the unit circle the filter will start to oscillate. As the poles move towards the center of the unit circle the resonance dies away. If the poles are moved outside the unit circle the filter will become unstable [1,12].

A resonator has a good resemblance to a bandpass filter, where r controls the bandwidth and ω_0 controls the center frequency of the filter. The bandwidth is defined as the width of the magnitude response curve at the points where the curve has decreased to half the power it has as its peak. Since power is proportional to the square of the amplitude, the half-power points correspond to amplitude(=1) values of $1/\sqrt{2} \approx 0.707$, which again correspond to the points at –3 dB attenuation. So r determines the bandwidth of the resonance. It can be shown that

$$r \approx 1 - b/2$$

where r is the pole radius, and b is the bandwidth measured in radians [1,12].

We should also be aware that the angular position of the poles ω_0 is not equal to the position of the peak of the magnitude response ω_r. The following relationship exists [1]:

$$\cos\omega_r = \frac{1+r^2}{2r}\cos\omega_0$$

This relationship is shown graphically in appendix B. We see that for high resonance settings (when r is close to 1) ω_r is close to ω_0.

We now know how we can design a filter which introduces resonance at a specific frequency ω_r by using a two pole system:

$$H_2(z) = \frac{b_0}{(1 - re^{j\omega_0}z^{-1})(1 - re^{-j\omega_0}z^{-1})}$$

where r is the resonance radius, ω_0 is the angle of the poles and b_0 is the gain factor.

Expanding the denominator and using Euler's formula yields:

$$1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}$$

So we have $a_1 = -2r\cos\omega_0$ and $a_2 = r^2$.
The only thing left is to normalize the gain at the resonance frequency to 1.
We do this, setting $|H_2(z)| = \sqrt{H_2(z)H_2(z^{-1})} = 1$.
When normalized to 1 at ω_0, b_0 becomes [1]:

$$b_0 = (1-r)\sqrt{1 + r^2 - 2r\cos 2\omega_0}$$

Using the derived coefficients and the ω_0, ω_r relationship we can construct a two pole filter that resonates at a specific frequency. The frequency and phase response for the filter for different values of r, at half the Nyquist frequency is shown in appendix B.

One can wonder why resonance is important for music synthesis. The reason is that most acoustic instruments exhibit *formants* in their spectra. Formants are constant spectral peaks in absolute frequency regions [6]. This characteristic appears in instruments that consist of excitation sources or resonating systems, which most acoustic instruments such as reeds, tubes or strings consist of. This tonal quality or timbre gives the instrument a characteristic sound, that is independent of the instrument's fundamental frequency and can be used to give the instrument a distinct spectral appearance.
The digital resonator can be used to place formants at selected frequencies in an instrument's spectrum.

3.1.3. Two Pole Resonant Lowpass Filter

It is possible to modify a two pole digital resonator into a filter that also attenuates frequency components, and in this way create a resonant lowpass filter.
We observe, that using a digital resonator at a low frequency gives a frequency response that has some lowpass filter similarities (see appendix B, plate 4). We can improve this characteristic by further attenuating high frequency components. In practice we do this by placing zeros in the z-plane at the Nyquist frequency ($\omega = \pi$).
Placing zeros changes the numerator of the system function, but since the zero positions are constant and real, the coefficients are constant and simple to calculate. Placing a zero at $\omega = \pi$ yields the coefficients $b_1=1$, $b_2=0$ and two zeros at the same location yields $b_1=2$, $b_2=1$.

We will try to design the filter directly by placing poles and zeros in the complex z-plane.
To illustrate the simplicity of the mathematical theory we will list all the necessary calculations.

The system is shown in Figure 9. We place two zeros at $\omega = \pi$ and sweep two conjugate poles inside the unit circle, at radius *r*.

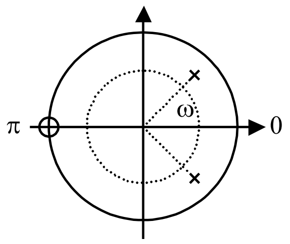


Figure 9

We start by inserting two real zeros o_1 and o_2 in the numerator of the second order transfer function:

$$\begin{aligned}&(1-o_1z^{-1})(1-o_2z^{-1})\\ &= 1-o_1z^{-1}-o_2z^{-1}+o_1o_2z^{-2}\\ &= 1+(-o_1-o_2)z^{-1}+o_1o_2z^{-2}\\ &\Rightarrow \underline{b_1=-o_1-o_2,\ \ b_2=o_1o_2}\end{aligned}$$

So we see, that two zeros at –1 gives the already shown coefficients: $b_1=2,\ b_2=1$.

Next we insert the complex poles into the denominator of the second order transfer function.
For simplicity we denote the complex conjugate poles $p = re^{\pm j\omega} = r\cos\omega \pm jr\sin\omega = (c,\pm s)$.

$$\begin{aligned}&(1-(c,s)z^{-1})(1-(c,-s)z^{-1})\\ &= 1-(\mathrm{c,s})z^{-1}-(c,-s)z^{-1}+(c,s)(c,-s)z^{-2}\\ &= 1+(-2c)z^{-1}+(c^2+s^2)z^{-2}\\ &\Rightarrow \underline{a_1=-2r\cos\omega,\ \ a_2=r^2(\cos^2\omega+\sin^2\omega)=r^2}\end{aligned}$$

So we again have $a_1 = -2r\cos\omega_0$ and $a_2 = r^2$, as with the digital resonator.
In general we can obtain the coefficients of the second order transfer function, consisting of two conjugate poles and zeros by:

$$\begin{aligned}&\mathrm{b}_0=1,\quad \mathrm{b}_1=-2r_z\cos(\omega_z),\quad \mathrm{b}_2={r_z}^2\\ &\mathrm{a}_1=-2r_p\cos(\omega_p),\quad \mathrm{a}_2={r_p}^2\end{aligned}$$

where r_z, ω_z and r_p, ω_p gives the position of the zeros and poles respectively [13].

At this point we need to normalize the gain. We choose to normalize at $\omega = 0$ (DC).

$$\begin{aligned}&H(\omega_0)=\frac{b_0+b_0 2e^{-j\omega_0}+b_0 1e^{-j2\omega_0}}{1+a_1e^{-j\omega_0}+a_2e^{-j2\omega_0}}\\ &H(0)=\frac{b_0+2b_0+b_0}{1+a_1+a_2}=1\Rightarrow\\ &\underline{b_0=\frac{1+a_1+a_2}{4}}\end{aligned}$$

The plot of this filter is shown for a sweeping cutoff frequency and sweeping radius in Appendix C, plate 1 and 2.
We see that the filter has some of the wanted characteristics. It has a 12 dB roll-off per octave, but it does not behave exactly as expected.
The amount of resonance in the output signal is scaled as a function of frequency. Furthermore we see that it is necessary for the radius to be close to the unit circle (slight resonance), if we want the frequency response to behave properly. This is seen clearly in the resonance sweep, where we observe that only the zeros affect the frequency response for a low pole radius (low resonance).
We can compensate for this by downscaling the radius for higher frequencies. In Appendix C, plate 3, we see a better result. Here the radius equals *r = 1-f*, were *f* is normalized frequency, *f=1* equals Nyquist. This solution is however far from perfect, since the relationship isn't linear, and since this completely eliminates resonance at high frequencies.
Since resonance is less sensitive at low frequencies, and more sensitive at high frequencies, we can instead replace the radius with the non linear relationship $r = 1 - q\sin(\pi \cdot f)$, where q adjusts resonance. The frequency response of the filter using this relationship, for different cutoff frequencies and for both low and high resonance is shown in Appendix C, plate 4 and 5. The corresponding z-plane configurations are also plotted. We see the effect of readjusting the radius as a function of frequency clearly in the z-plane. Please note, that the frequency is swept exponentially in the frequency

response figures, while it is swept linearly in the z-plane plots. The resulting frequency responses are exactly what we wanted.
This design method is very straightforward and intuitive, since we work directly in the digital domain. The design is however not very exact or accurate.
We will later design more accurate filters.

3.2. Design Considerations

3.2.1. IIR versus FIR filters

Up until now, we have only discussed the design of Infinite Impulse Response (IIR) filters, that is, filters with feedback. There exists other filters without feedback, Finite Impulse Response (FIR), that corresponds to an IIR all-zero filter. The filters are very popular in DSP since they are easy to control and can be used to design high order filters, with strong attenuation of unwanted frequency components [5]. However these filters often require more computation time than equivalent IIR filters, simply because they use more coefficients [12]. FIR filtering can be made almost as fast as IIR, by fast convolution using the FFT, but tuning these filters interactively is not practical [5].
Thus for music synthesis, the IIR filter is superior to FIR, because of its lower runtime and easy tuning relationships.

3.2.2. Filter Tuning,Cutoff and Resonance Control

We have seen filters with cutoff and/or resonance control. We will try to lay out guidelines for implementing these controls. In general we would like the cutoff control to be “linear” in relation to the way humans perceive frequency. Usually we look at a frequency spectrum in a log scale, since this is the way the human mind understands frequency. We can convince ourselves of this, by playing the piano. Here the frequency doubles for each octave on the piano. Thus, since we would like the cutoff control to appear linear, we must make it exponential. In this way the control will move slowly in the low frequency area and faster in the high frequency area. The range of human hearing is generally between 20Hz to 20kHz [13].
Furthermore the roll-off of a filters attenuation should in general be gentle (6 to 24dB per octave) when used for music synthesis. That is, have a wide transition band. This is simply to give the best possible control of the harmonic content of the signal.
We should be aware that the filters, when used, are not real LTI systems, since we can change cutoff and resonance and thereby change the system coefficients over time. Since we use the LTI model, this can lead to instabilities for fast changes in the cutoff frequency [5], and result in sample glitches and clicks [6]. It may be necessary to interpolate the tuning parameters to force smooth transitions.

In principle the same control of the cutoff applies for the resonance control. The resonance control is an amplification and is therefore related to loudness instead of frequency. The perception of loudness relates roughly to the sound power to an exponent of 1/3. This means that a perceived loudness increase by a factor of 2 corresponds to a increase in the sound power by a factor of 10 [13].
Furthermore the resonance brings up an important question: In which way should we normalize the frequency response of the filter ? The lowpass filter only attenuates frequencies, but the resonance can amplify a frequency range beyond a gain of 0dB . Should this be allowed or should we normalize to the cutoff frequency, and thus force the frequency peak to 0dB ? One could think that the latter would be the best solution, but this is not the case. In general we will be working with signals containing a fundamental frequency plus harmonics related to an instruments timbre. Applying a filter with resonance at a higher frequency than the fundamental, will amplify parts of the harmonic content, but the amplitude of these harmonics will seldom be larger than the amplitude of the fundamental. If we normalize to 0dB at the cutoff point, we will instead attenuate the fundamental, which is not desired. So in general a filter should be normalized to 0dB, but the resonance should be allowed to go beyond this point.

3.2.3. The Importance of Phase

We have seen frequency and phase response plots for some different filters. How should we interpret these responses in relation to the musical context in which

they will be used ? In music, a general rule of thumb is, that our primary interest is obtaining a specific frequency response, and that a resulting (non-linear) phase response is accepted [7].
The human ear is very insensitive to phase [13]. It has been shown that the frequency response is much more important than the phase response in relation to an instrument's timbre, and that a change in phase has a minimal effect on the perceived quality of the sound, even though the resulting waveform appears radically different [6]. That is why all filters in this paper are designed in relation to the frequency response.
We must remember that the phase response is in reality a time-delay, and that the human ear will hardly be able to distinguish a small time-delay. In general we will therefore not worry about a filters particular phase response. It is not possible to achieve a linear phase response with IIR filters, only with more expensive FIR filters [1].
Note however that a particular phase response *can* be used to create effects related to music synthesis. A varying filter, which only has phase response (an all-pass filter), can for example be used to create a chorus-like effect [7].

3.2.4. Transforming into other Filter types

Until now we have only considered lowpass filter design. The discussed design methods are however general enough to be used for designing other filters, like highpass or bandpass filters. It is also possible to transform a well designed lowpass filter into an equivalent highpass filter for instance. This is useful, since we can concentrate on developing a single filter-type and convert this to other filter-types when the design is finished.
We will look at the highpass filter. Spectral inversion can be used for transforming a FIR lowpass filter into an equivalent highpass filter. This method does however not always yield correct results when used with IIR filters [13], but other methods exists.
The highpass has great resemblance to the lowpass filter. Instead of attenuating the high frequencies the filter should attenuate the low frequencies. So the highpass filter is a flipped lowpass filter $H_h(\omega) = H_l(\omega - \pi)$, and we can flip the lowpass filter by substituting *z* with *-z* in the transfer function [12]. This yields a transfer function like the lowpass, but where the sign of the odd-numbered samples are flipped [1]. For the canonical second order structure this corresponds to changing the sign of the b_1 and a_1 coefficients. We will later on use this to transform a lowpass filter into a highpass filter. It is also possible to transform a designed lowpass filter into for example bandpass and notch filters. These transforms are more complicated than the highpass transformation, and will not be covered in this paper.

3.2.5. Cascade and Parallel Filter Stages

It is possible to connect filters in cascade or parallel stages, and in this way create more complex filters. See Figure 10. This is used for designing digital Butterworth filters as we shall see later on, but the method can in general be used to create new higher order filters based on existing filter designs.
Filters are cascaded by multiplying their transfer functions, and put in parallel by adding their transfer functions [13]. Knowing this, it is for example possible to create a bandpass filter by cascading a lowpass and highpass filter. If the filters are two pole, the resulting system will be a four pole filter.
The system can be combined into a single filter, but it is usually advantageous to keep the filter stages separate, and if possible use the canonical second order structure for each stage, since poles and zeros appear in conjugate pairs.

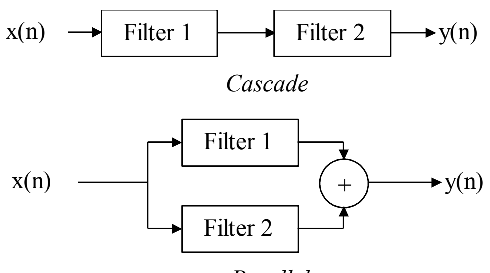


Figure 10

For cascading two second order sections into one combined fourth order section, we can derive the general equation for the transfer function by:

$$\frac{(b_0 + b_1 z^{-1} + b_2 z^{-2})(d_0 + d_1 z^{-1} + d_2 z^{-2})}{(a_0 + a_1 z^{-1} + b_2 z^{-2})(c_0 + c_1 z^{-1} + c_2 z^{-2})}$$
$$= \frac{b_0 d_0 + (b_1 d_0 + b_0 d_1) z^{-1} + (b_0 d_2 + b_1 d_1 + b_2 d_0) z^{-2} + (b_1 d_2 + b_2 d_1) z^{-3} + b_2 d_2 z^{-4}}{a_0 c_0 + (a_1 c_0 + a_0 c_1) z^{-1} + (a_0 c_2 + a_1 c_1 + a_2 c_0) z^{-2} + (a_1 c_2 + a_2 c_1) z^{-3} + a_2 c_2 z^{-4}}$$

We retrieve the fourth order coefficients directly. We can later use this for designing fourth order Butterworth Filters.

3.2.6. Digital Design versus Analog Design

All though we have seen that it is possible to design digital filters directly in the z-domain, it is in general not the preferred method for designing digital filters [6]. The response of especially IIR filters are both hard to predict and control in the digital domain. In the analog domain this is different. Analog filter design has a long history and is a well developed field [1,6], and it is therefore advantageous to design filters in the analog domain and transform these into the digital domain.

One way to do this is using the *Bilinear Transform* which relates the s-domain to the z-domain.
Analog filter design is done in the s-domain, which has infinite frequency. The Bilinear Transform maps the s-plane into the z-plane using the relation:

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

where T is the sampling interval. One of the strong points of this transform, is that it avoids aliasing of frequency components, while frequency warping the entire s-plane [1]. We note that the point $s=\infty$ maps into $z=-1$.
The frequency relationship between the two domains is

$$\omega = 2\arctan\left(\frac{\Omega T}{2}\right)$$

We should be aware that the Bilinear Transform only gives an approximation of the analog filter. The frequency response characteristics of the analog filter and the corresponding digital filter will differ slightly [6].

In the next chapter, we will use the Bilinear Transform, to design digital filters based on analog prototype filters.

3.3. Analog based filter Design

3.3.1. A Simple Second Order Analog Filter

Filtering is a well established field in the world of analog electronics. We will start by looking at the simplest possible second order lowpass filter: The Sallen & Key Filter [5, 9, 14]. See Appendix D, plate 1. The frequency response is given by:

$$H(s) = \frac{1}{\tau^2 s^2 + 2\zeta\tau s + 1}, \qquad \tau = \frac{1}{2\pi f_c}$$

where f_c is the cutoff frequency and ζ is the so called damping factor. In the analog world, ζ is used to specify the damping factor, or resonance. The lower the damping factor, the higher the resonance [5]. Simple relations exist for determining the value of the resistors/capacitors from the tuning coefficients f_c and ζ, but these relations are coupled, so it is difficult to vary one coefficient while the other is constant [5,9]. This structure is therefore not very popular for use in synthesizers [9].

By transforming this second order (2 pole) analog filter into the digital domain using the Bilinear Transform, the resulting system function is the canonical second order transfer function (2 pole, 2 zero).
Substituting with *T=1/F*$_s$ and reordering, it can be shown that [9]:

$$\begin{aligned}
&C = 1/\tan(\pi f_c / F_s) \\
&b_0 = 1/(1+2\zeta C + C^2), \qquad b_1 = 2b_0, \qquad b_2 = b_0 \\
&a_1 = 2b_0(1-C^2), \qquad a_2 = b_0(1-2\zeta C + C^2)
\end{aligned}$$

We immediately see, that this structure has resemblance to the previously designed digital filter, where we place two zeros at Nyquist and sweep two conjugate poles in the z-plane. The frequency response of this filter along with the corresponding z-plane plots are shown in Appendix D. Note that the frequency is swept exponentially in the frequency response figures, while it is swept linearly in the z-plane plots. We observe that the filter has a 12 dB roll-off per octave, and that it is normalized to 0dB at DC.

We will now try to convert this transformed filter into an equivalent highpass filter, by using the method discussed earlier. By simply changing the sign of b_1 and a_1 we obtain the highpass frequency response shown in Appendix D.

This analog filter has some nice characteristics and is very fast to apply to a signal, since it can be implemented using the canonical second order difference equation. Unfortunately the tuning relationships are coupled and the calculations for finding the filter-coefficients are rather complex. This can make the filter computational expensive when modifying the tuning of the filter, especially if this is done at a high rate [5].

3.3.2. The Butterworth Filter

Probably the best known and most commonly used method for designing IIR digital filters is the transformation of the classical analog filters: Butterworth, Chebyshev and Elliptic filters.
We will look at the Butterworth filter, since it has a maximally flat frequency response in the passband , and thus is suitable for filtering in music synthesis.
The Butterworth filter of order *2n* can be designed by cascading *n* second order sections, tuned to the same cutoff frequency, but with different damping factors [5]. Using analog hardware, such filters can therefore be realized by cascading *n* Sallen & Key Filters [7].

In the analog domain the Butterworth transfer function is:

$$H(s) = \frac{k_0}{\prod_{k=1}^{n}\left(s - e^{j\pi\left(1/2+(2k-1)/2n\right)}\right)}$$

where n is the filter order and k_0 is a normalizing constant.
The Butterworth design can be transformed into the digital domain by means of the Bilinear Transform. The resulting system function has the form [12]:

$$H(z) = \frac{A(z+1)^n}{(z-z_0)\cdots(z-z_{n-1})}$$

$$z_i = \frac{1+e^{j\theta_i}\tan(\omega_c/2)}{1-e^{j\theta_i}\tan(\omega_c/2)},$$

$$\theta_i = \frac{\pi}{2} + \left(\frac{2i+1}{4n}\right)2\pi, \qquad \text{for i = 0,1....,2n - 1}$$

However these formulas are rarely used for filtering. Instead a look-up table of different damping- or Q-factors for each biquad section is browsed, for finding the factors related to the selected filter order.

We will try to design a Butterworth filter of fourth order (4-pole), by simply cascading two second order sections based on the Sallen & Key structure. Damping factors for Butterworth filters of different orders is shown in Appendix E, plate 1 [5]. These factors approximate a flat passband response, by letting the resulting peak fill the rounded area below the cutoff point of the previous section [7]. It is thus a matter of simply scaling the damping factors to achieve more resonance in the filter.
This scaling can be applied to the first second order section, for a peaking resonance, or to all sections, for large bandwidth resonance.
The frequency response and matching z-plane plots of the 4-pole filter is shown in Appendix E, both with and without resonance. The resonance is achieved by scaling the first second order sections damping factor. We see that the roll-off is the expected 24dB per octave, and that the resonance setting functions as desired.

Using this filter and higher order Butterworth filters, it is possible to achieve steep roll-off's, and thus small transition bands, that can be advantageous for segregating narrow spectral components in complex sounds.

3.3.3. The State Variable Filter

A very popular type of filter for synthesizers, is the Analog *State Variable Filter* (SVF). This filter has supposedly been used in synthesizers from Oberheim as well as synthesizers from other manufactures. The second order SVF has gained popularity, because it is a multimode filter that has both lowpass, bandpass and highpass output at the same time. Furthermore the filter is very easy to tune with regards to cutoff and resonance, since these variables can be changed independently by simply varying different resistors in the analog network [7]. An analog second order state variable filter is shown in Appendix F, plate 1.

The analog SVF can be simulated in the digital domain by modeling each analogue component by one corresponding digital operator. The resulting digital structure is shown in Figure 11 [9].

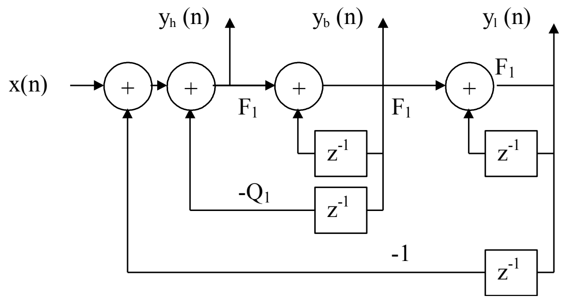


Figure 11

The values Q_1 and F_1 are related to the cutoff frequency f_c and damping factor ζ as:

$$F_1 = 2\sin\left(\frac{\pi f_c}{F_s}\right) \qquad Q_1 = 2\zeta$$

where F_s is the sampling rate.
The difference equations are derived from the structure [5]:

$$\begin{aligned} y_h(n) &= x(n) - y_l(n-1) - Q_1 y_b(n-1) \\ y_b(n) &= F_1 y_h(n) + y_b(n-1) \\ y_l(n) &= F_1 y_b(n) + y_l(n-1) \end{aligned}$$

By z-transforming the difference equations and substituting with $r = F_1$ and $q = 1 - F_1 Q_1$, we acquire the resulting system functions [9] :

$$\begin{aligned} H_l(z) &= \frac{r^2}{1 + (r^2 - q - 1)z^{-1} + qz^{-2}} \\ H_b(z) &= \frac{1 - z^{-1}}{r} H_l(z) \\ H_h(z) &= \left(\frac{1 - z^{-1}}{r}\right)^2 H_l(z) \end{aligned}$$

We note that the result is a two pole system.
The frequency response of the lowpass SVF is shown in Appendix F, for high (ζ=0.8) and low (ζ=0.1) damping factors. We observe that the filter has the familiar 12 dB per octave roll-off and that the filter is normalized at DC. Furthermore we observe that for low resonance (high damping) the frequency peaks at the Nyquist frequency. For even higher damping factors the filter becomes unstable at high frequencies, since the resulting poles are placed close to the unit circle [9]. The problem exist for both lowpass, bandpass and highpass.

By finding the roots of the denominator, we see how the poles are placed in the z-plane for the cutoff sweeps at both high and low damping factors. See appendix F.
The lowpass, bandpass and highpass frequency characteristic for this filter is shown in appendix G for a fixed low frequency at different resonance settings. We note that the bandpass filter has the expected roll-off of 6 dB per octave.

The problem with the SVF at high damping and high frequencies, can be corrected by adding a zero at $F_s/2$, and thus attenuating frequencies at the Nyquist frequency. This alters the frequency response at high frequencies, but can be advantageous if high damping at high frequencies is desired [9]. In practice we do this by multiplying the numerator by $(1+z^{-1})$. This removes the frequency peaks at $F_s/2$ in all the filter-types and produces frequency responses that looks very promising. Since this modification also results in an additional factor 2 amplification, we renormalize by 0.5.

The frequency response for lowpass, bandpass and highpass SVF, before and after adding the zero, is shown in Appendix H. We see that the high peak has been removed and replaced by a faster roll-off in the high frequency range. The modification destroys the highpass filter somewhat, since the zero attenuates the high frequency components.
The implementation of this change in the structure of the SVF is not difficult. Before giving the input to the filter we just need to apply the $+z^{-1}$ [9]. This is shown in Figure 12. In the difference equation this is implemented by adding a + *x(n-1)* to the equation containing *x(n)*, and multiplying both by 0.5 to adjust the gain.

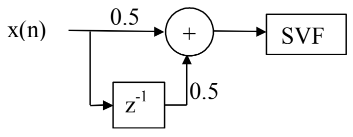


Figure 12

Another implementation of the SVF is presented in [7]. Here an extra z^{-1} delay is present in the loop before the lowpass structure. Otherwise the structure is very similar to the previously discussed. See

Figure 13. This structure also produces notch filter output.

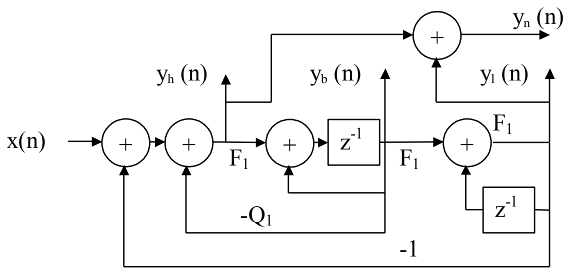


Figure 13

The difference equations are:

$$y_l(n) = F_1 y_b(n-1) + y_l(n-1)$$
$$y_h(n) = x(n) - y_l(n) - Q_1 y_b(n-1)$$
$$y_b(n) = F_1 y_h(n) + y_b(n-1)$$
$$y_n(n) = y_h(n) + y_l(n)$$

Besides the order of which the filters are calculated, we note the presence of the extra delay in the lowpass equation. The frequency response of this filter is equivalent to the already discussed [9].

The SVF allows the resonance to be tweaked so high, that the filter will start to oscillate at the cutoff frequency. By setting $Q_l=0$ the SVF can actually function as a fast sine-wave generator [7].

The State Variable Filter has some very desirable properties. It is a second order filter with independent and fast control of center frequency and resonance, and can simultaneously provide lowpass, bandpass, highpass and even notch outputs.

4. Filter Implementation

We have seen how we can design digital filters with different characteristics. We will now see how to implement these filters in a digital system like the computer. Given a systems difference equation the implementation follows directly. The *x(n)* is the current input sample, and *x(n-1)*, *x(n-2),…,x(n-M)* is the past *M* input samples, which could be stored in an array and updated for each iteration. Similarly the *y(n-1)*, *y(n-2),…,y(n-N)* is the past *N* output samples, and *y(n)* is the resulting filtered sample for the current iteration. The gives a total of *M+N* memory cells for storing past input and output samples, and a total of *M+N+1* multiplication's, when using a Direct Form I implementation:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Using a Direct Form II implementation:

$$w(n) = -\sum_{k=1}^{N} a_k w(n-k) + x(n), \qquad y(n) = \sum_{k=0}^{M} b_k w(n-k)$$

we instead use *max{M,N}* memory cells, but still *M+N+1* multiplication's [1].
Since the filters discussed in this paper are very small IIR filters, there is no real gain in using Direct Form II instead of I, and we might as well use the simpler Direct Form I implementation.

When implementing filters in a digital machine we operate with numbers of finite-word-length. This leads to quantization errors in the signals, which appear as noise in the resulting sound. The amount of noise which originates from quatization depends on the chosen number representation: i.e. integer, fixed-point, float etc., that is, the number of bits per word. IIR filters are extra sensitive to quantization noise because they are infinite, and therefore the error accumulates over time. An in-depth discussion of quantization errors in relation to IIR filtering is beyond the scope of this paper. but we will note that special implementations exist for minimizing this error. One alternative implementation method is the *coupled-form state-space realization*, which is based on the canonical second order structure, but is less sensitive to finite-word-length effects, and therefore suitable as a building block in cascaded systems [1]. Furthermore implementation of IIR filters can lead to *overflow* and especially *underflow* errors, which should be handled in some way. For dedicated DSP chips this is usually not a real problem, but for desktop computers this can introduce time-consuming floating-point *denormals*, that can lead to strange periodic slow-down errors if the programmer is not aware of this. The smallest non-denormal single precision floating point number is 1.175494350822288e-38f.

5. Results

We have discussed several different approaches for designing digital filters. We will now discuss how these filters behave when implemented in a digital computer and if they meet our previously described design specifications.

5.1. Test System and Test Procedure

All the filters have been analyzed in MATLAB, for both frequency response, phase response and z-plane configuration, for different tuning parameters. One of the developed MATLAB programs is shown in Appendix I. Furthermore the filters have been implemented on an Intel Pentium III system running Microsoft Windows, and applied to real-time band-limited signals, consisting of periodic sampled saw-, triangle- and square-waves. The systems samplerate was set to 44.1kHz and the filtering was performed on blocks of 1024 samples. Fast sine-wave modulation of cutoff at different setting for resonance was tested for some selected filters. The tuning parameters were not interpolated in any way. The results were sent directly to the computers sound-card. The implementation was written in C++, using 32 bit floats for number representation, however for internal calculations the Pentium operates with 80 bit representations. For avoiding underflow in feedback, floating-point denormals were removed using this simple macro:

```
#define DENORMALIZE(fv) (((( *(unsigned int*) &(fv))
& 0x7FFFFFFF) < 0x0da24260 ) ? 0.0f : (fv) )
```

It forces absolute floats below 1.e-30f to zero, without using any FPU instructions.

5.2. Test Specifications

The tested filters are as follows:

1. 1-pole lowpass (direct digital design)
2. 2-pole resonant lowpass (direct digital design).
3. 2-pole lowpass and highpass based on the analog Sallen & Key structure.
4. 4-pole Butterworth lowpass filter (based on analog design)
5. 2 pole State Variable Filter (lowpass, highpass, bandpass + implementation with notch)
6. 2 pole State Variable Filter with added zero for attenuation at nyquist.

Except for Filter #1, all these filters can be tuned by a cutoff and resonance parameter as described by design specification 3 and 4 (DS3,4). The filters have a roll-off of 6dB (1 pole), 12dB (2 pole) and 24dB (4 pole) to give a gentle attenuation and a long transition band, and it is a simply a matter of cascading more filter stages to achieve a faster roll-off (DS5). We have primarily tested the lowpass filter, but highpass filters are very easily implemented as we have seen (DS1). All the filters are normalized to a gain of 0dB at DC, and resonance has been allowed to go as high as +20dB (DS9). We note that all the filters have a non-linear phase response (DS8), but as discussed the phase response is not our primary concern and is therefore accepted.

5.3. Test Results And Comparison

The tested filters are generally easy to operate and function as expected. They behave and sound much like a “real” filter in a standard synthesizer (DS2), although some people might say that they lack some of the analog “warmth” that was characteristic in classic analog synthesizers. This however can also be partly related to quality of the input samples, since the filter only shapes the frequency components of incoming signal. Furthermore the filters are generally very fast to apply to a signal (DS6), even at high sampling rates, and it is possible to comfortably apply, say 20 two-pole filters (non-modulated) simultaneously and only use about 12% of the CPU on the test system.

The fastest and easiest implemented resonant filter is 2 pole State Variable Filter (#5). The combination of quick and independent calculation of tuning coefficients and simultaneous lowpass, bandpass, highpass and even notch output makes this filter stand out. In its standard form, the filter is difficult to tune at low resonance and high frequency settings, and it can be necessary to limit the resonance or cutoff interval considerably, giving less control of the timbre. However it is possible to compensate for this by adding an extra zero in the structure, and thus eliminate this restriction (#6). This causes a small attenuation in the highpass filter close to the Nyquist frequency, but in general this is not heard since harmonics seldom reach this frequency. The highpass might give different results when applied to high frequency instruments, like hihats, but this has not been tested. The change makes the lowpass filter very stable across the entire frequency range at both low and high resonance settings, and this configuration (#6) is therefore more advantageous than the standard form (#5). The actual filtering involves 4 additions and 3 multiplication's. For the configuration with an extra zero this is instead 5 additions and 4 multiplication's. For both configurations, one extra addition is needed for notch output. Note that by using one extra multiplication per output for scaling, this filter can also function as a fast 3 band equalizer with adjustable center-frequency.

Filters based on the analog Sallen & Key structure (#3), are stable across the entire frequency range, at both low and high frequencies without any modifications. The sound characteristics of this filter is almost identical to the state variable configuration (#6), which is not surprising since they have an almost similar frequency response. The filter is more difficult to tune, because of the calculations involved and because is necessary to use look-up tables for calculating the coefficients for maximum efficiency. Once the filter coefficients have been calculated the actual filtering process is almost as fast as the SVF, 5 multiplication's and 4 additions for a standard second order configuration. However only one filtered output signal is produced by this, i.e., lowpass or highpass etc.
A sine wave with variable frequency and amplitude was used to modulate the cutoff frequency of filter #3,#5 and #6. The State Variable Filters (#5,#6) provided the best audiovisual results, and the modulation sounded correct. The Sallen & Key filter suffered from distortion and hissing when the sine wave was set to a high frequency. However, this could be due to the lack of interpolation of the cutoff-parameter in the test system.
The direct digital designed two pole filter (#2) has almost the same properties as the Sallen & Key structure. The filter is a little simpler to tune, because of the approximated coefficient relationships. However since the filter design is not completely accurate, it should not be used for cascading second order sections for example.

The Butterworth filter is very easy to design and use, when based on an existing second order structure, like the Sallen & Key filter. It is a simple matter of cascading the existing second order filter in *n* stages. The 4-pole Butterworth (#4), therefore has almost exactly the same properties as #3, only with a steeper roll-off. The effect of the steeper roll-off is easy to hear in the resulting sound, the attenuation is more profound, and the filter makes the instrument sound more “hollow” when resonance is applied.

The one pole filter (#1) is generally not suitable for audio filtering alone, but it can be cascaded with the described two-pole filters for achieving a steeper roll-off, thereby creating a 3-pole filter for example (18 dB roll-off). Furthermore, since this filter is very fast to apply to a signal, it can generally be used to a vast variety of tasks related to the smoothing of digital signals that appear in programs related to music synthesis. This can for example be removal of sample-glitches, smooth volume ramping and audio filtering in feedback, needed for echo and reverb simulations.

To summarize, two of the tested second order filters have properties that are very desirable for use in music synthesis. The test shows, that the best all-round filter implementation is the State Variable Filter, with an extra added zero (#6). The filter is faster, easier to tune, and has more possibilities than any of the other tested filters. Filters based on the analog Sallen & Key filter (#3), are not quite as fast, and not as easy to tune, but provide a slightly better frequency response than the SVF, which makes it the next best implementation.. Finally, the Butterworth design (#4) can be considered a good extension to second order filters for achieving steeper roll-off’s. We have tested this using the Sallen & Key implementation, but it should also be possible to achieve a Butterworth response, by cascading second order SVF sections, although we have not tested this.

6. Conclusion

We have described theory for designing digital filters for music synthesis, directly in the digital domain and based on analog prototypes. We have seen, that it is possible to design simple and useful filters directly in the digital domain, but that it can be easier and more practical to design filters based on existing analog filters. We have also seen, that it is possible to improve the transformed analog filters directly in the digital domain, thereby making the designed filter more functional, as in the case of the State Variable Filter.

We have described methods for improving the filters by using more filter stages, and transforming filters into different types. Furthermore we have implemented and tested the described filters on a computer and used them for real-time filtering.

We have seen that different filters have both advantages and disadvantages, and that a selection of a particular filter therefore should depend on the application. However, a particular filter seems to stand out, with regards to both functionality and speed, that in relation to music synthesis, makes this filter very desirable. This is the State Variable Filter. It is surprising that this filter is not described more thoroughly in the literature. Only one book [7] of the used references describes this filter, but does not cover the improved variation of this filter, that we have seen in this paper. The SVF is instead covered in technical articles [5,9]. Books related to music synthesis seems to focus mainly on the two pole Butterworth design, which corresponds to the described Sallen & Key filter.

We have primarily covered the resonant lowpass, highpass, bandpass and notch filters, since these are widely used filters in synthesizers. We should note however, that filtering for music synthesis is a large topic, an many other different filtering operations can be of interest. As computers get faster in the future, lots of different filtering operations related to music synthesis and audio-effects will inevitably be implemented in computer software, and it will be interesting to witness this musical evolution.

7. References

[1] J.G.Proakis, D.G.Manolakis, “Digital Signal Processing, Principles, Algorithms, and Applications”, 3rd edition, 1996.

[2] Dean Friedman, “Synthesizer Basics”, 1986.

[3] David Crombie, “The New Complete Synthesizer, A Comprehensive Guide to the World of Electronic Music”, 1986.

[4] L.R.Rabiner, B.Gold, “Theory And Applications of Digital Signal Processing”, 1975.

[5] P.Dutilleux, “Filters, Delays, Modulations and Demodulations A Tutorial”, 1998.

[6] Charles Dodge, Thomas A. Jerse, “Computer Music”, 1997.

[7] H.Chamberlin, “Musical Applications of Microprocessors”, 1980.

[8] Lane, Hoory, Martinez, and Wang, “Modeling Analog Synthesis with DSPs”, Computer Music Journal, 21:4, pp.23-41, 1997.

[9] P.Dutilleux, “Simple To Operate Digital Time Varying Filters”, Preprint 2757, 86th AES Convention, Hamburg, March, 1989.

[10] “Nord Lead, Virtual Analog, Owners Manual, Software Version 2.x English”, Clavia 1996

[11] John Lane and Garth Hillman, “Motorola's DSP56000/SPS/DSP56001, Implementing IIR/FIR Filters with Digital Signal Processors”, Motorola, 1993.

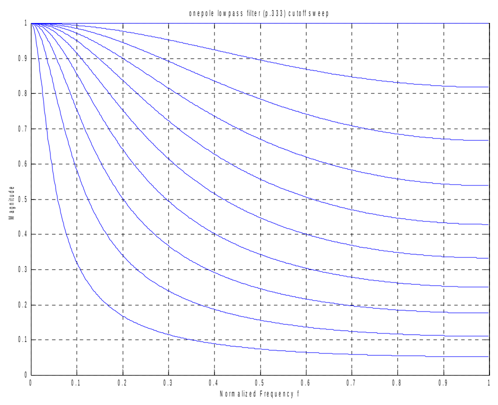
[12] Ken Steiglitz, “A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music”, 1996

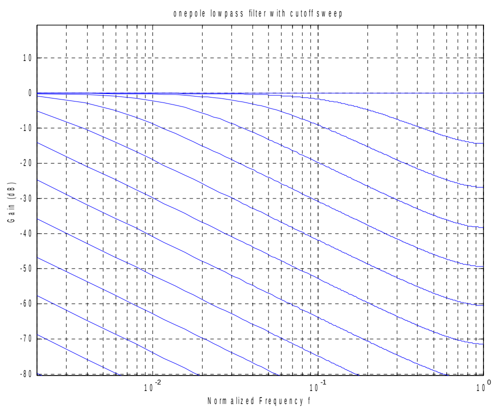
[13] Steven W.Smith, “The Scientist and Engineer’s Guide to Digital Signal Processing”, California Technical Publishing, 1997, (http://www.dspguide.com).

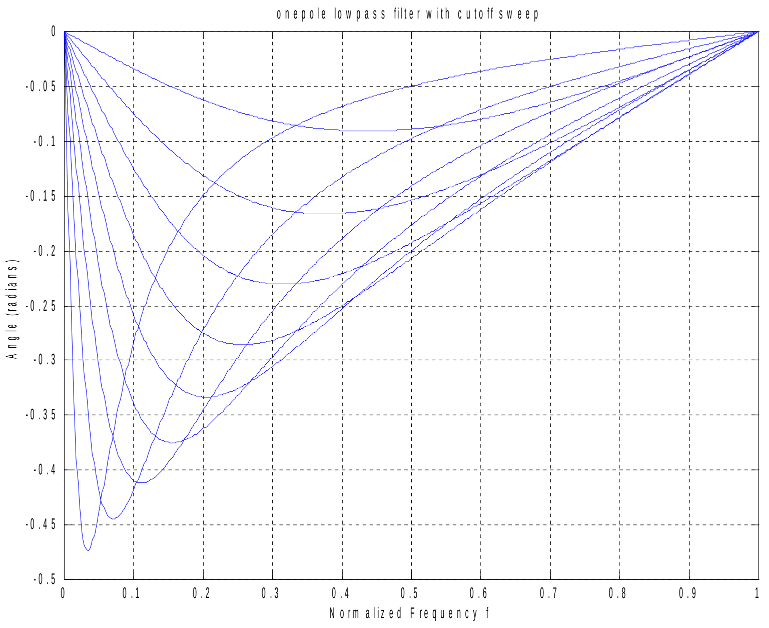
[14] Kerry Lacanette, “A Basic Introduction to Filters – Active, Passive and Switched-Capacitor”, AN-779, National Semiconductor Corporation, 1991.

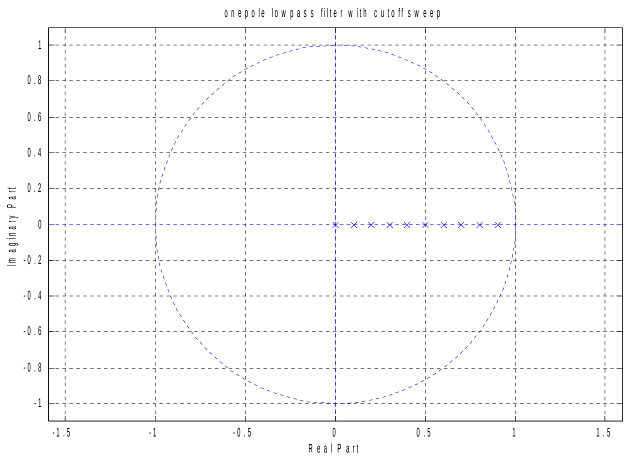
[15] Curtis Roads et al., “The Computer Music Tutorial”, MIT Press, 1996.

APPENDIX A

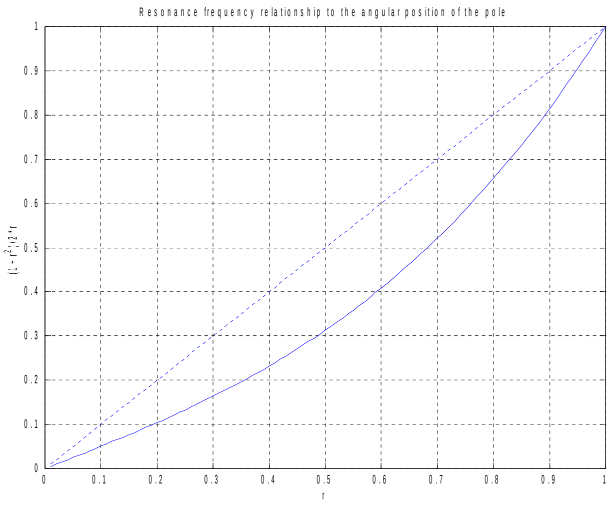

onepole lowpass filter (p.333) cutoff sweep
Magnitude
Normalized Frequency f
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


onepole lowpass filter with cutoff sweep
Gain (dB)
Normalized Frequency f
10
0
-10
-20
-30
-40
-50
-60
-70
-80
10^{-2}
10^{-1}
10^{0}

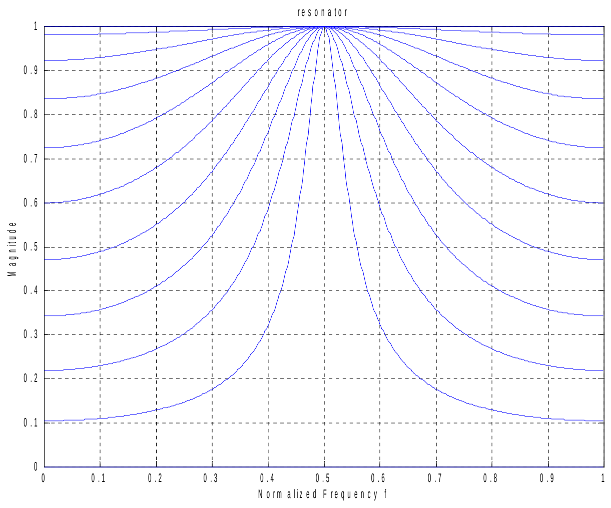

onepole lowpass filter with cutoff sweep
Angle (radians)
Normalized Frequency f
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


onepole lowpass filter with cutoff sweep
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1
1.5

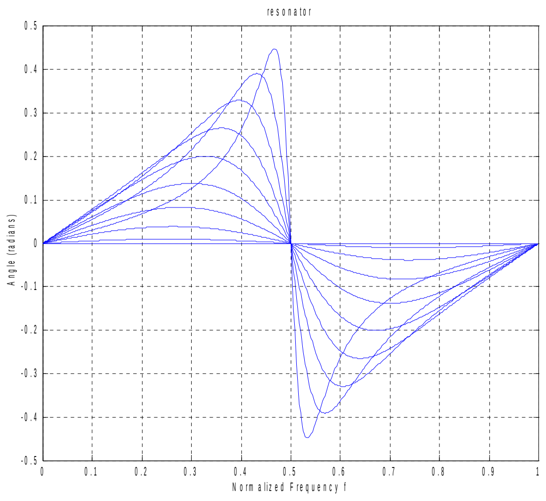
APPENDIX B



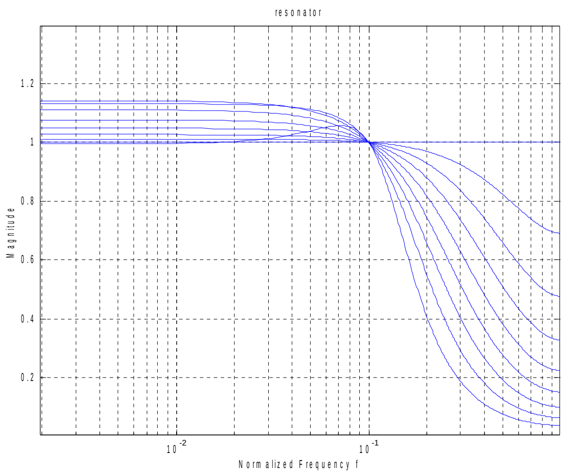
Resonance frequency relationship to the angular position of the pole
$(1+r^2)/2*r$
r
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



resonator
Magnitude
Normalized Frequency f
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

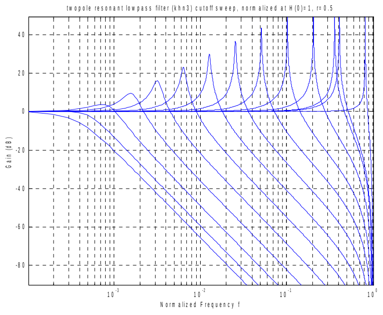


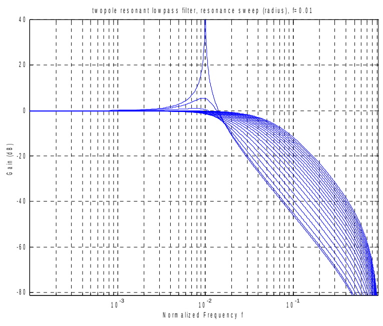
resonator
Angle (radians)
Normalized Frequency f
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5

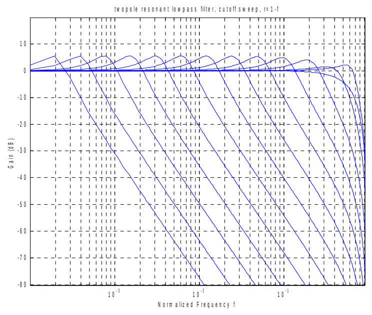


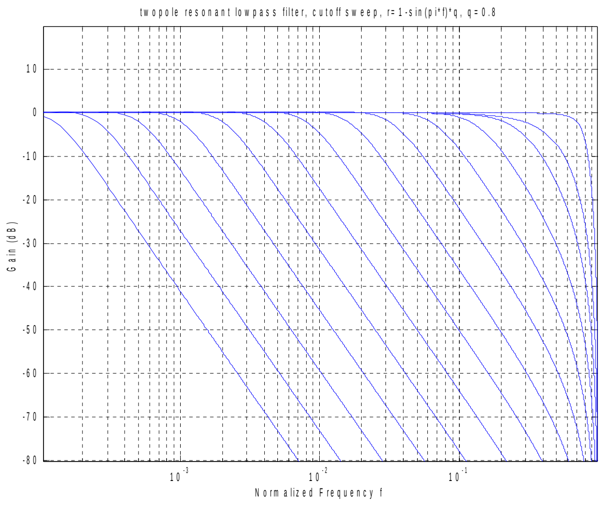
resonator
Magnitude
Normalized Frequency f
1.2
1
0.8
0.6
0.4
0.2
10^{-2}
10^{-1}

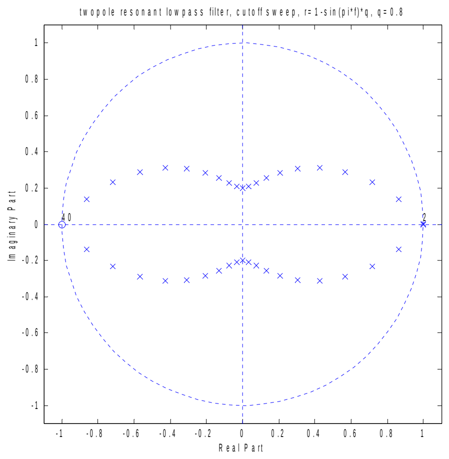
APPENDIX C

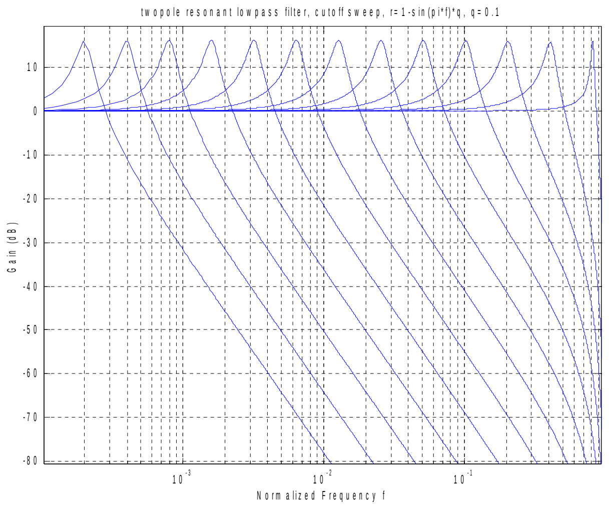

twopole resonant lowpass filter (khn3) cutoff sweep, normalized at H(0)=1, r=0.5
Gain (dB)
Normalized Frequency f

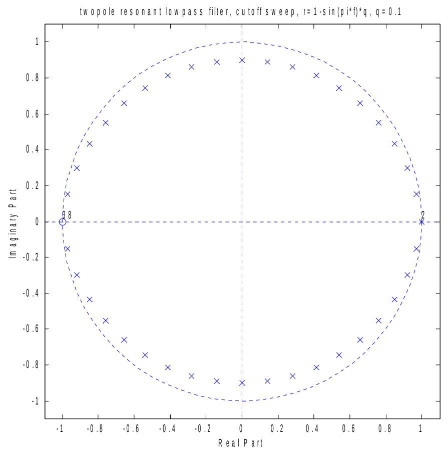

twopole resonant lowpass filter, resonance sweep (radius), f=0.01
Gain (dB)
Normalized Frequency f


twopole resonant lowpass filter, cutoff sweep, r=1-f
Gain (dB)
Normalized Frequency f

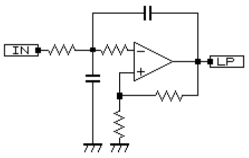

twopole resonant lowpass filter, cutoff sweep, r=1-sin(pi*f)*q, q=0.8
Gain (dB)
Normalized Frequency f


twopole resonant lowpass filter, cutoff sweep, r=1-sin(pi*f)*q, q=0.8
Imaginary Part
Real Part
40
2

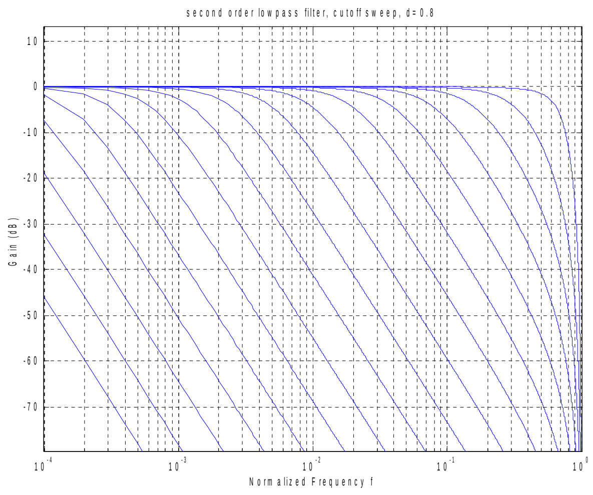

twopole resonant lowpass filter, cutoff sweep, r=1-sin(pi*f)*q, q=0.1
Gain (dB)
Normalized Frequency f

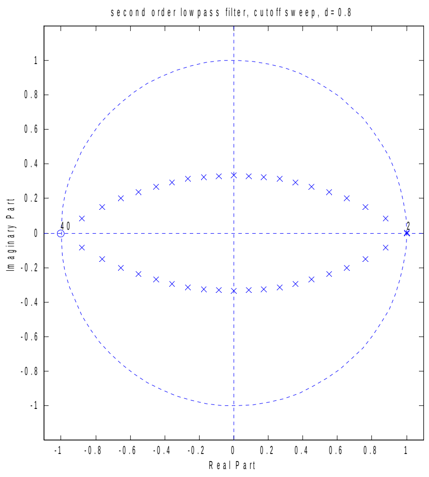

twopole resonant lowpass filter, cutoff sweep, r=1-sin(pi*f)*q, q=0.1
Imaginary Part
Real Part
38
2

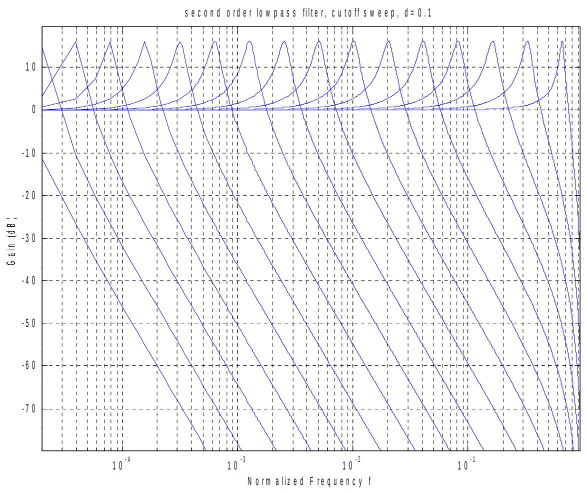
Appendix D

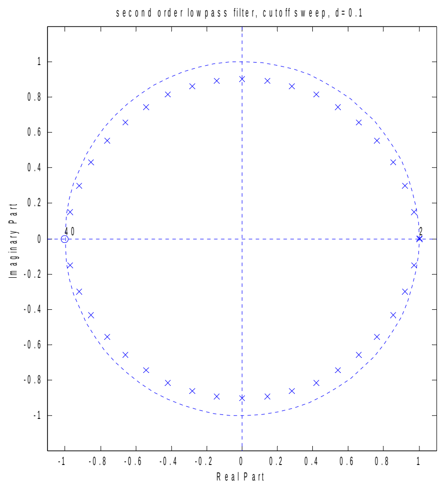


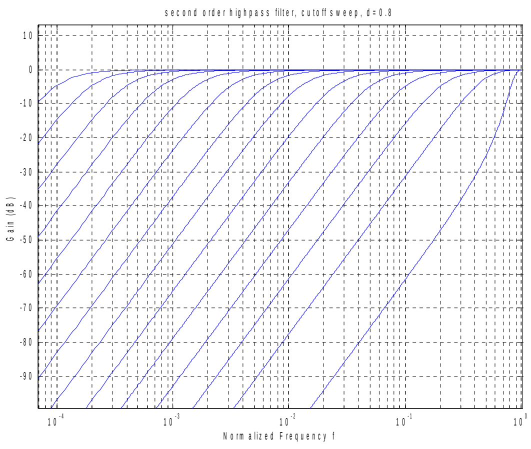
Sallen-Key 2nd Order Active Lowpass filter Network

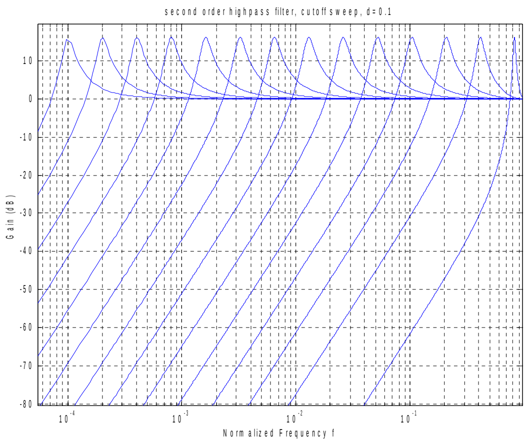








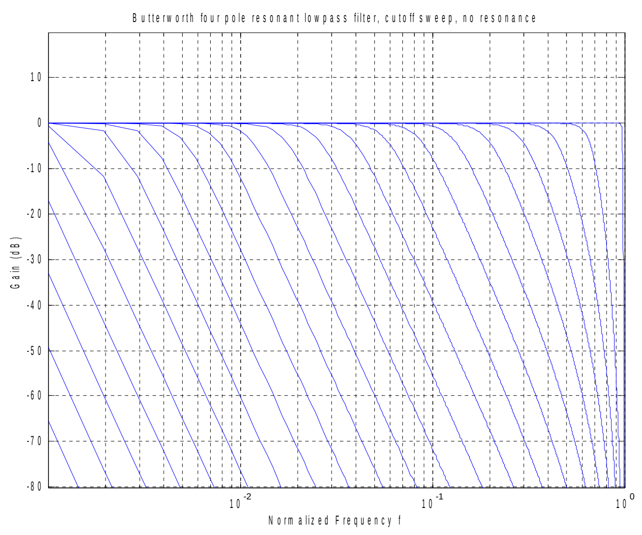


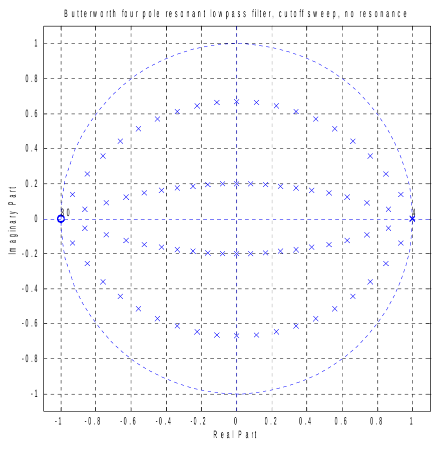


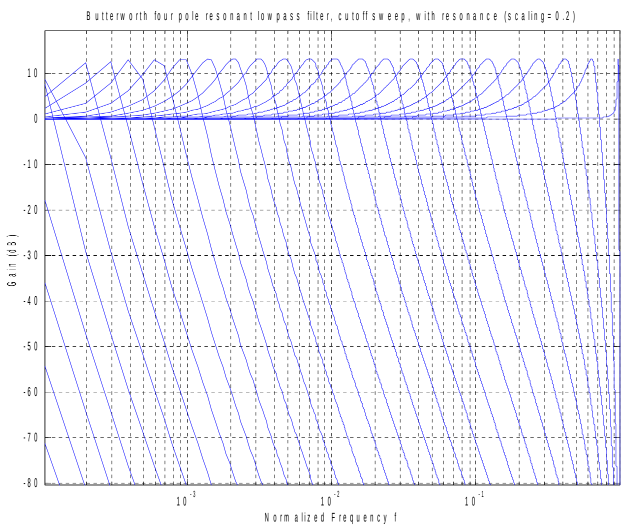
APPENDIX E

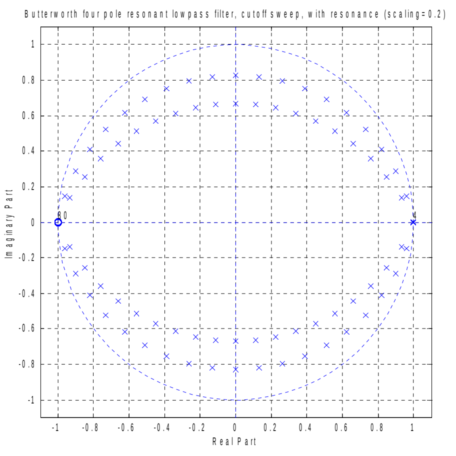
N	ζ				
2	0.707				
4	0.924	0.383			
6	0.966	0.707	0.259		
8	0.981	0.831	0.556	0.195	
10	0.988	0.891	0.707	0.454	0.156

Damping factors for second order sections in Butterworth filters

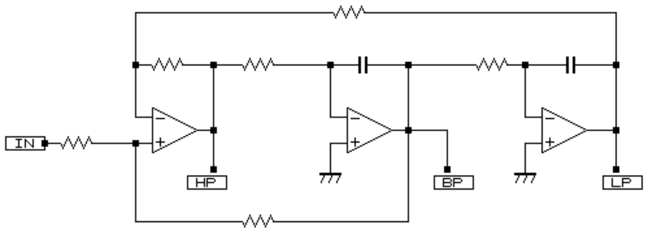




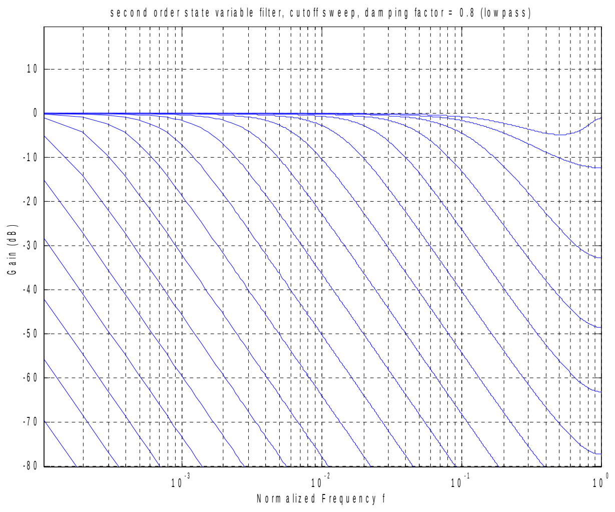


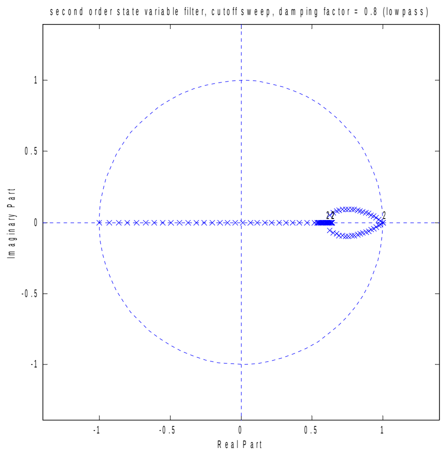


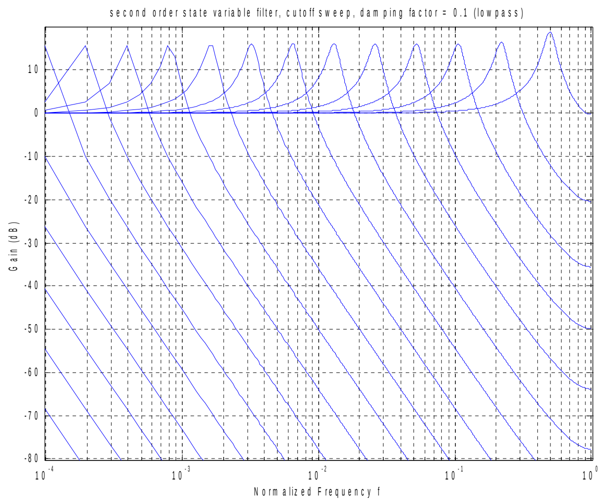
APPENDIX F

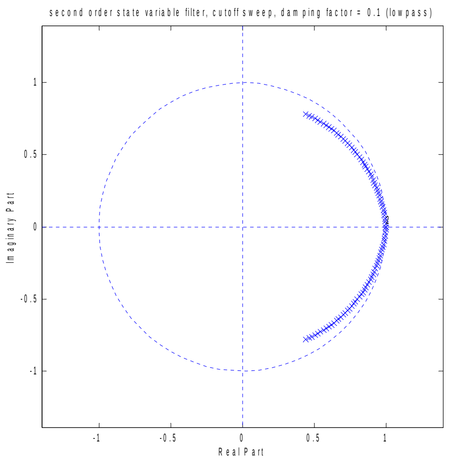


Analog 2nd order State Variable Filter Network

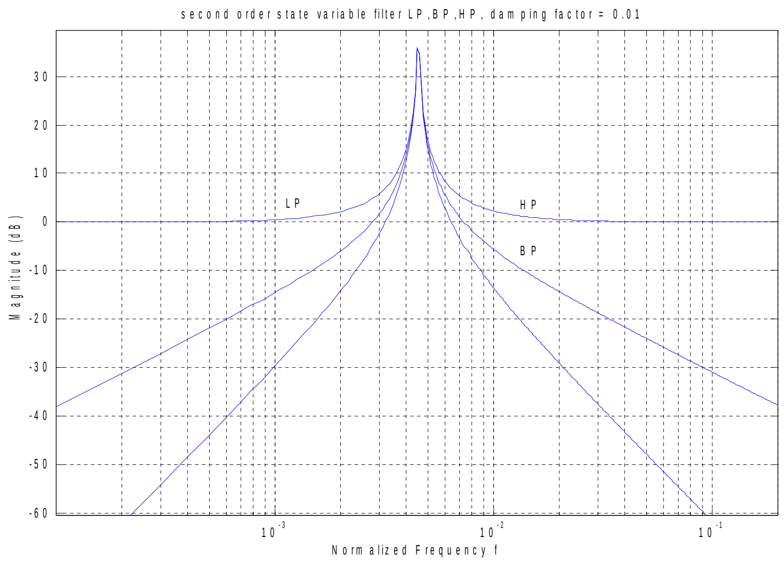


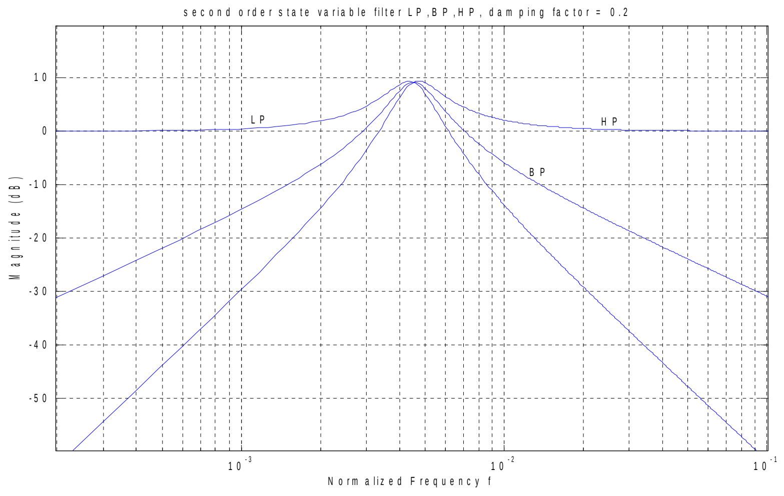


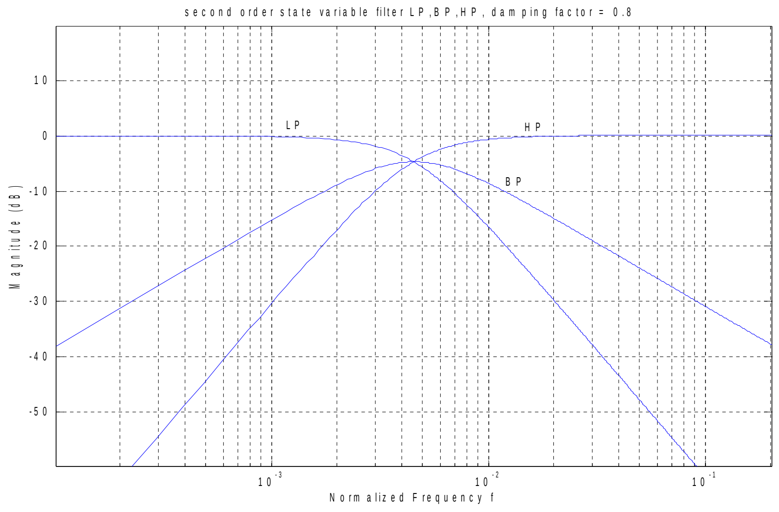




APPENDIX G


second order state variable filter LP,BP,HP, damping factor = 0.01
30
20
10
0
-10
-20
-30
-40
-50
-60
Magnitude (dB)
LP
HP
BP
10^{-3}
10^{-2}
10^{-1}
Normalized Frequency f

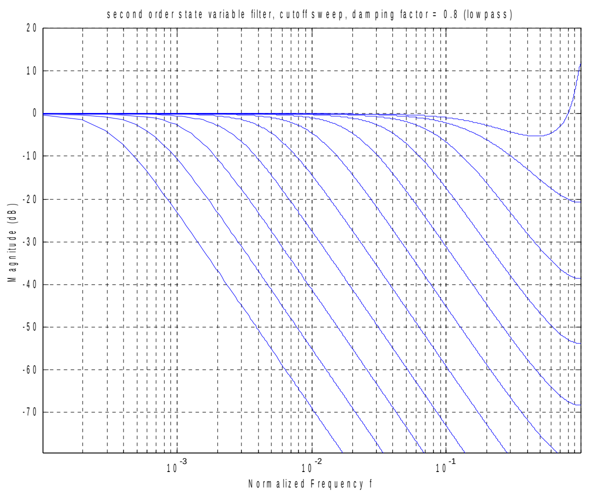

second order state variable filter LP,BP,HP, damping factor = 0.2
10
0
-10
-20
-30
-40
-50
Magnitude (dB)
LP
HP
BP
10^{-3}
10^{-2}
10^{-1}
Normalized Frequency f

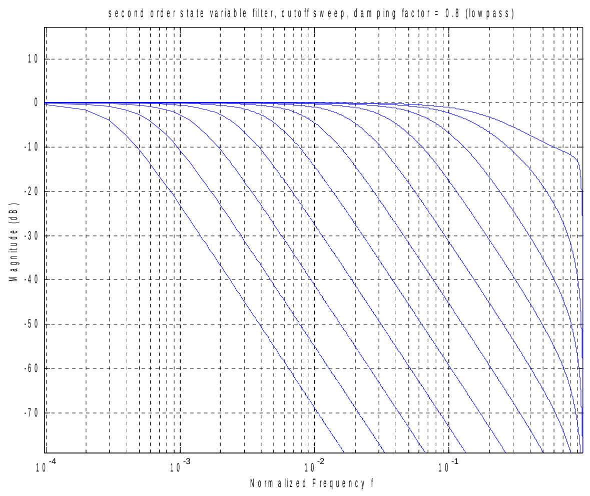

second order state variable filter LP,BP,HP, damping factor = 0.8
10
0
-10
-20
-30
-40
-50
Magnitude (dB)
LP
HP
BP
10^{-3}
10^{-2}
10^{-1}
Normalized Frequency f

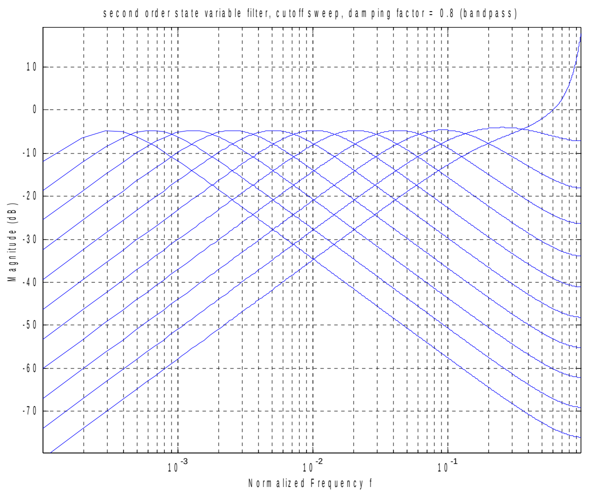
Appendix H

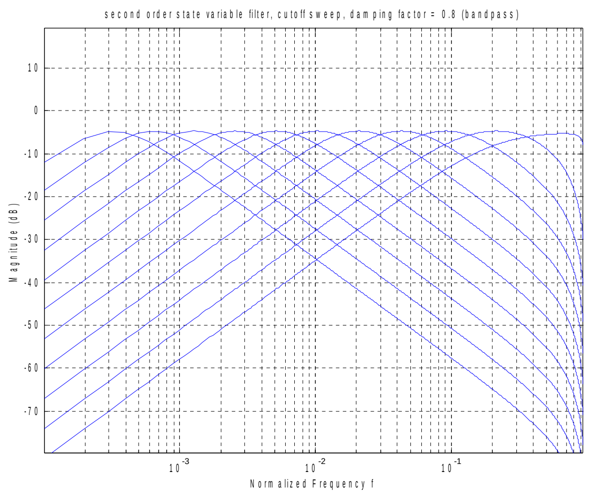
without extra zero

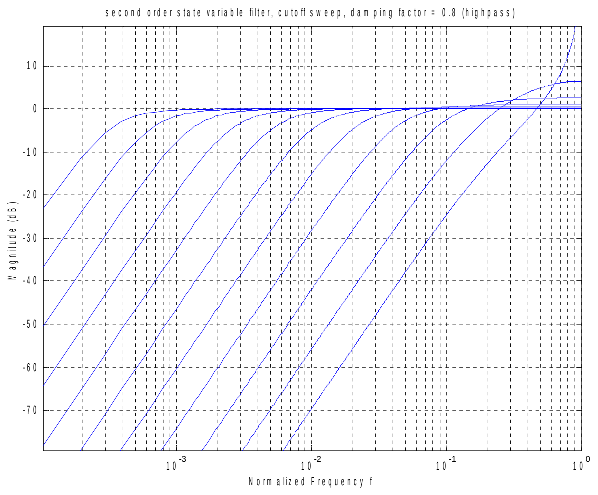
with extra zero

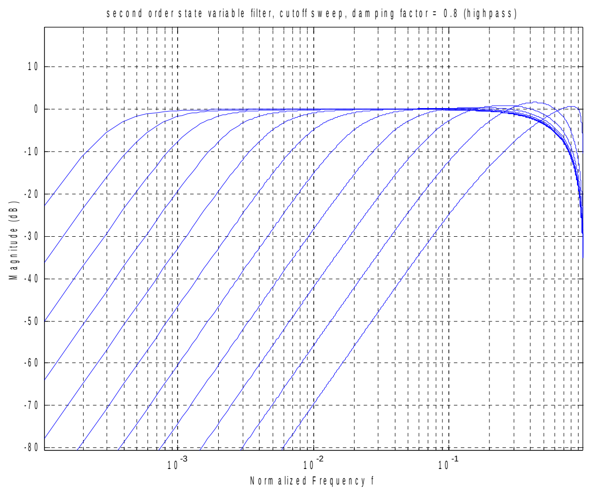












APPENDIX I

```
%
%   Second order State Variable Filter
%

comment= 'second order state variable filter, cutoff sweep, damping factor = 0.1 (lowpass)';

Q=0.1;

hold off;

kzr=[];
kpr=[];

f=0.00001;

while f < 1

%f1 = sin(pi*0.5*f);
f1 = f;

r=f1;
q = 1-f1*2*Q;

f = f + 1*f;     % obtains linear cutoff adjustment
%f= f+0.02;


b0 = r*r;
a1 = r*r-q-1;
a2 = q;

% coefficients

% ======== lowpass SVF
zc = [b0];
%zc = [0.5*b0,+b0*0.5];   % added zero
pc = [1,a1,a2];

% ======== highpass SVF
%%%%%% (1-z)(1-z) = 1 + z*z -2z
%%%%%% (1+z)(1+z*z -2z) = 1 + z*z -2z + z + z*z*z -2*z*z = 1 + -z*z -z + z*z*z
%zc = [b0,-2*b0,b0];
%zc = [0.5*b0,-b0*0.5,-b0*0.5,b0*0.5]; % added zero
%pc = [r*r,r*r*a1,r*r*a2];

% ======== bandpass SVF
%%%% (1-z)(1+z) = 1+z-z-z*z;
%zc = [b0,-b0];
%zc=[0.5*b0,0,-b0*0.5]; % added zero
%pc = [r,r*a1,r*a2];

% poles & zeros (roots of the coefficients)

zr = roots(zc);
pr = roots(pc);

kzr = [kzr;zr];
kpr = [kpr;pr];

[XX,wx] = freqz(zc,pc,1024*10);  %% high resolution
grid on;semilogx(wx/pi,10*log(abs(XX))); xlabel('Normalized Frequency f');ylabel('Gain (dB)');


title(comment);
hold on;

end

%zplane(kzr,kpr);title(comment);
```