Digital Signal Processing 2/ Advanced Digital Signal Processing, Audio/Video Signal Processing
Lecture 9,

Gerald Schuller, TU Ilmenau

Allpass Filters

So far we specified the magnitude of our frequency response and didn't care much about the phase. For allpass filters, it is basically the other way around.
In the beginning of filter design, we saw that we can write a transfer function as

$$H\left(e^{j\Omega}\right)=e^{j\varphi(\Omega)}\cdot A\left(e^{j\Omega}\right)$$

Here we specify, or rather, alter the phase, and keep the the magnitude of our frequency response at constant 1, meaning

$$A\left(e^{j\Omega}\right)=1$$

Hence we would like to have a filter with transfer function H of magnitude constant 1,

$$\left|H\left(e^{j\Omega}\right)\right|=1$$

The simplest allpass filter has one pole and one zero in the z-domain for the transfer function,

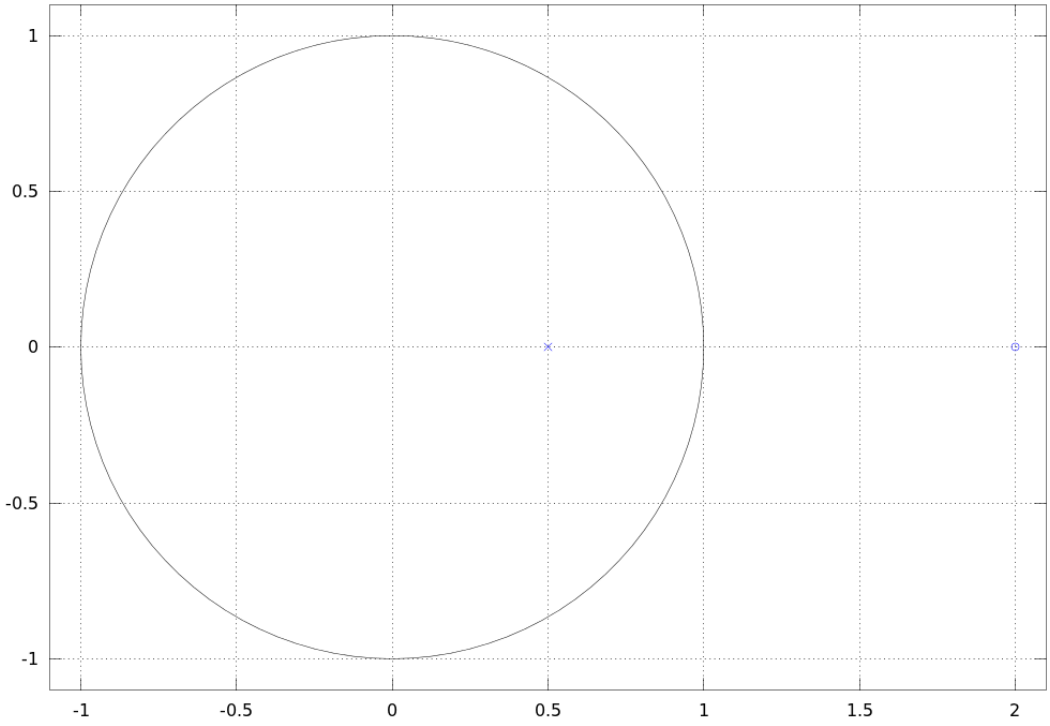
$$H_{ap}(z)=\frac{z^{-1}-\bar{a}}{1-a\,z^{-1}}$$

where a is a complex number, and $\bar{a}$ specifies the conjugate complex number.

Observe that here we have a zero at $z=\frac{1}{\bar{a}}$ and a pole at $z=a$! The **pole and the zero are at conjugate reverse locations!**

Example: If a=0.5, we obtain the pole/zero plot with Matlab/Octave,

```
a=0.5;
B=[-a' 1]; %the numerator polynomial of H_AP
A=[1 -a]; %the denominator polynomial
zplane(B,A); %plot the pole/zero diagram
axis([-1.1 2.1 -1.1 1.1],'equal') %have equal aspect ratio
```



In this plot, the cross at 0.5 is the pole, and the circle at 2 is the zero.

How can we see that the magnitude of the frequency response $H\left(e^{j\cdot\Omega}\right)$ is 1? We can re-write it as

$$H_{ap}\left(e^{j\Omega}\right)=\frac{e^{-j\Omega}-\bar{a}}{1-a\,e^{-j\Omega}}=e^{-j\Omega}\,\frac{1-\bar{a}\,e^{j\Omega}}{1-a\,e^{-j\Omega}}$$

Here you can see that the expression in the numerator is the conjugate complex of the denominator, hence their magnitude cancels to one. The exponential before the fraction also has magnitude 1, hence the entire expression has magnitude 1,

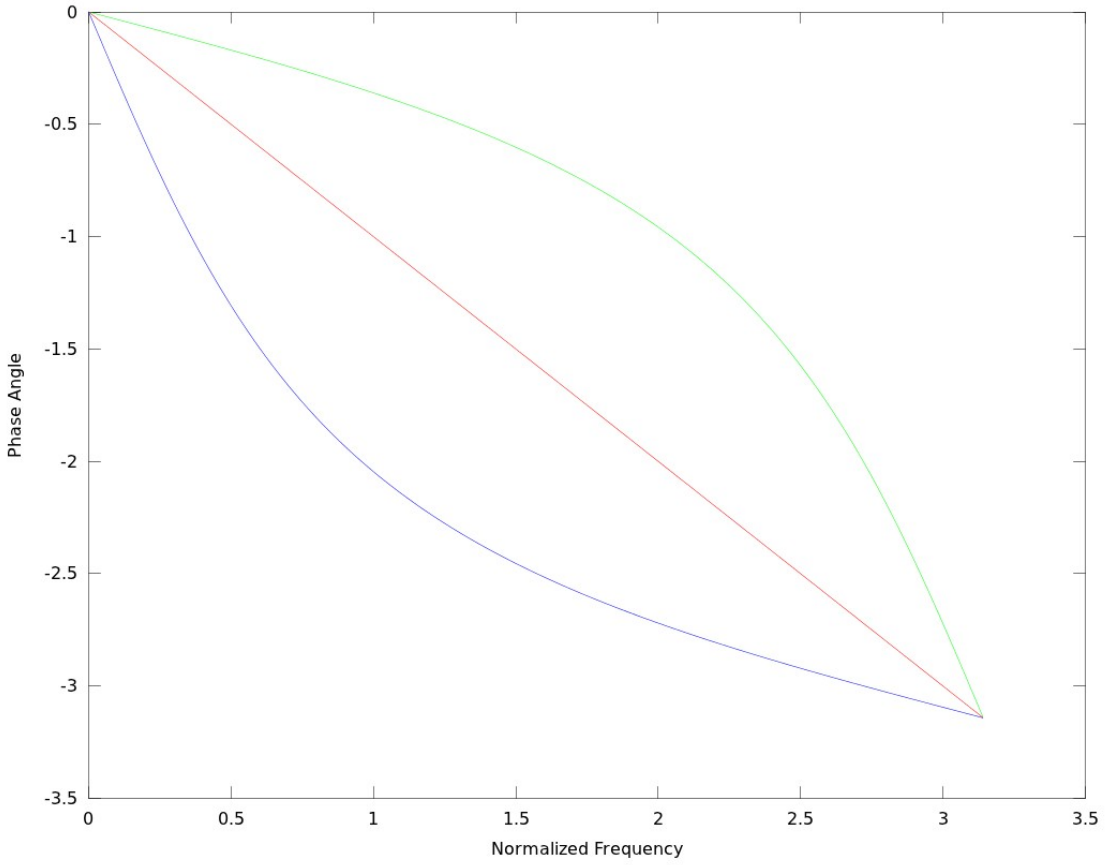
$$\left|H_{ap}\left(e^{j}\Omega\right)\right|=1$$

Here we can see, using just 1 pole and one zero, we can obtain a magnitude of constant 1. More interesting now is the resulting phase. The phase function can be found in the book Oppenheim/Schafer, “Discrete Time Signal Processing”:

$$\varphi\left(\Omega\right)=-\Omega-2\arctan\left(\frac{r\sin\left(\Omega-\theta\right)}{1-r\cos\left(\Omega-\theta\right)}\right)$$

where r is the magnitude of a and θ is the phase angle of a (hence $a=r\cdot e^{-j\theta}$). Observe that so far we assumed the phase to be linearly dependent on the frequency ($\varphi\left(\Omega\right)=-\Omega\cdot d$), and here we see it to be quite non-linear, with the trigonometric functions!

We can now plot the resulting phase over the normalized frequency, and compare it with the phase of a delay of 1 sample (of z^{-1}), where we get $\varphi\left(\Omega\right)=-\Omega$. This can be seen in the following plot, for $r=0.5$ and $r=-0.5$:



Here, the blue line is the allpass phase for r=0.5, the green line for r=-0.5, and the red line is for r=0, the phase of a pure 1 sample delay z^{-1}. Here it can be seen that the beginning and end of the curves are identical (at frequencies 0 and pi), and only in between the allpass phase deviates from the 1 sample delay! For a=0 the allpass indeed becomes identical to z^{-1}, a delay of 1 sample. So we can see that it behaves very **similar to a delay**.

The plot was produced with a simple Matlab/Octave function for the phase function,

```
function wy=warpingphase(w,a);
%produces phase wy for an allpass filter
%w: input vector of normlized frequencies (0..pi)
%a: allpass coefficient

%phase of allpass zero/pole :
theta=angle(a);
%magnitude of allpass zero/pole :
r=abs(a);
wy=-w-2*atan((r*sin(w-theta))./(1-r*cos(w-theta)));
```

The **phase** at the output of our phase function can also be **interpreted as a normalized frequency**.
An interesting observation is, that an allpass with coefficient $-\bar{a}$ is the inverse function of the allpass with coefficient a !
We can try this in Matlab/Octave.

```
%frequency range:
w=(0:0.01:pi);
a=0.5*(1+i)
wyy=(warpingphase(warpingphase(w,a),-a'));
plot(w,wyy)
xlabel('Normalized Frequency')
ylabel('Phase Angle')
```

If we define

$$H_{ap}(a, z)$$

as the z-transform of our allpass with coefficient a. If we use $z = e^{j\Omega}$ as its input (like for obtaining the frequency response). We know:

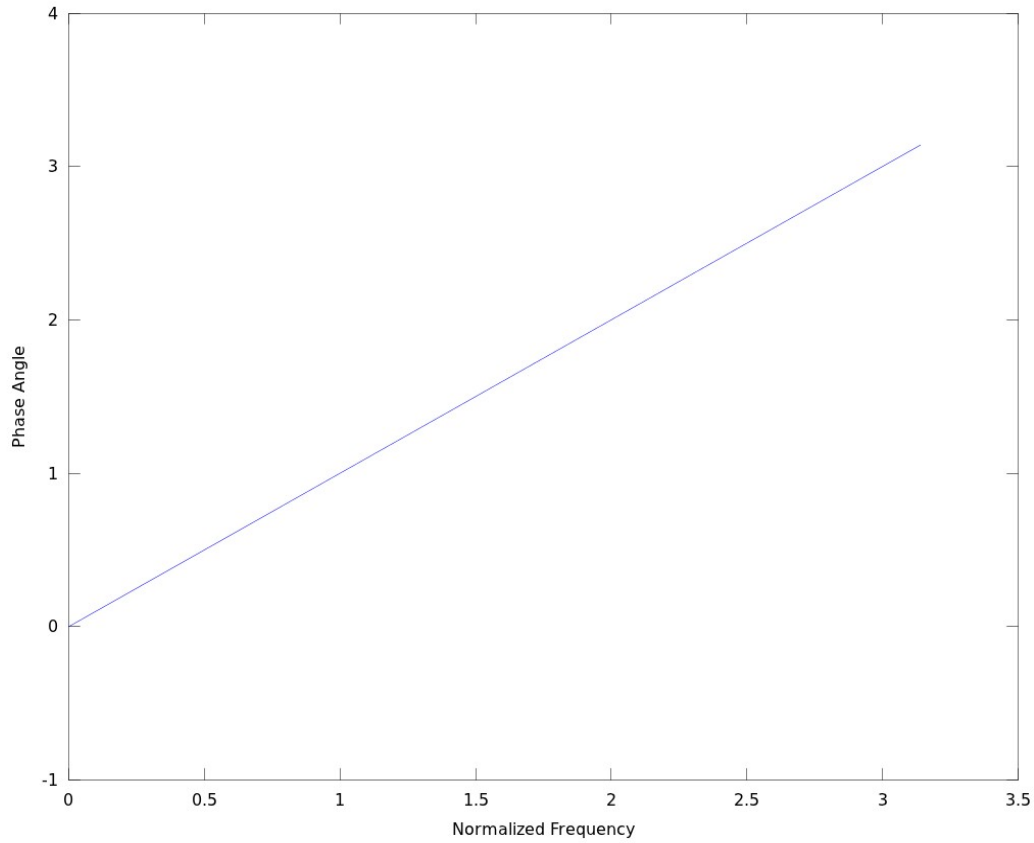
$$H_{ap}(e^{j\Omega}) = e^{j\varphi(\Omega)} \cdot A(e^{j\Omega}) = e^{j\varphi(\Omega)}$$

so replacing $z \leftarrow H_{ap}(a, z)$ and then setting $z = e^{j\Omega}$ for the frequencency response is like the total replacement $z = e^{j\varphi(\Omega)}$ where we now have this non-linear function $\varphi(\Omega)$ for the phase.

This function plots the **phase angle** of the frequency response of the concatenated system, where we replaced the z by our allpass, and in this way obtain our non-linear phase input,

$$H_{ap}(a, H_{ap}(-\bar{a}, e^{j\Omega}))$$

Observe that the **output** of the first allpass is i**nterpreted as frequency** here. The resulting plot is



Here we see that it is the **identity** function. This shows that interpreting the allpass as a normalized frequency “warper”, the allpass with a is inverse to the allpass with -a'.

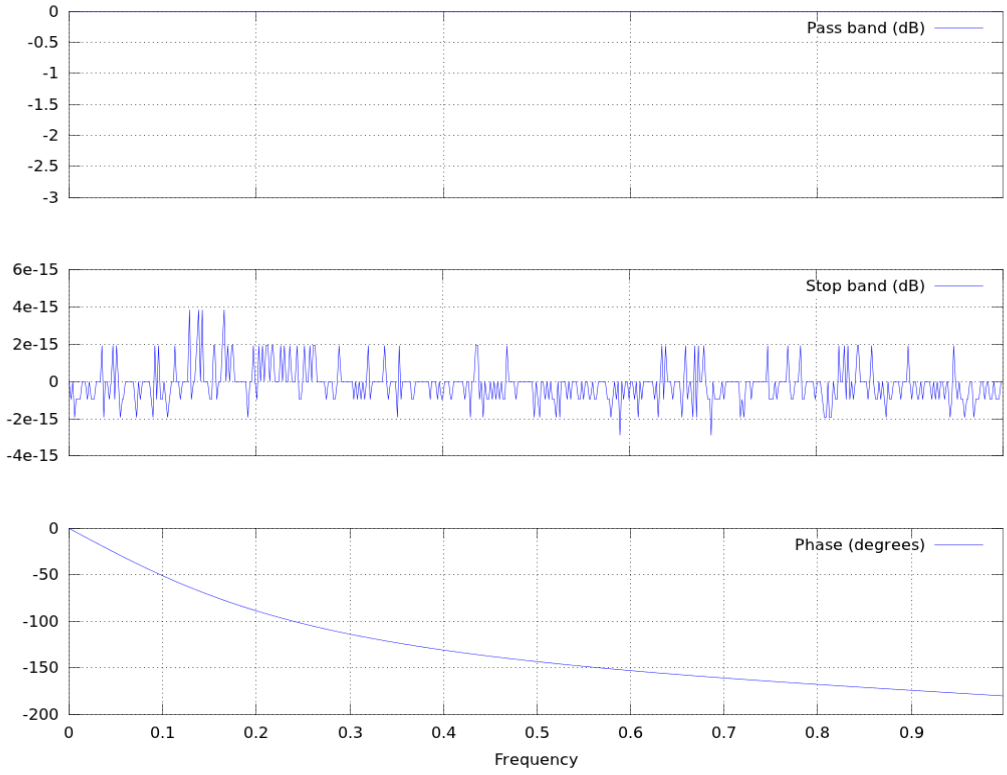
What is the frequency response of an example allpass filter? For a=0.5, we can use freqz. Looking at the z-transform

$$H_{ap}(z)=\frac{z^{-1}-\bar{a}}{1-a\,z^{-1}}$$

we get our coefficient vectors to

```
a=0.5;
B=[-a' 1];
A=[1 -a];
```

(observe that for freqz the higher exponents of z^-1 appear to the right)
Now plot the frequency response and impulse response:
freqz(B,A);
And we get

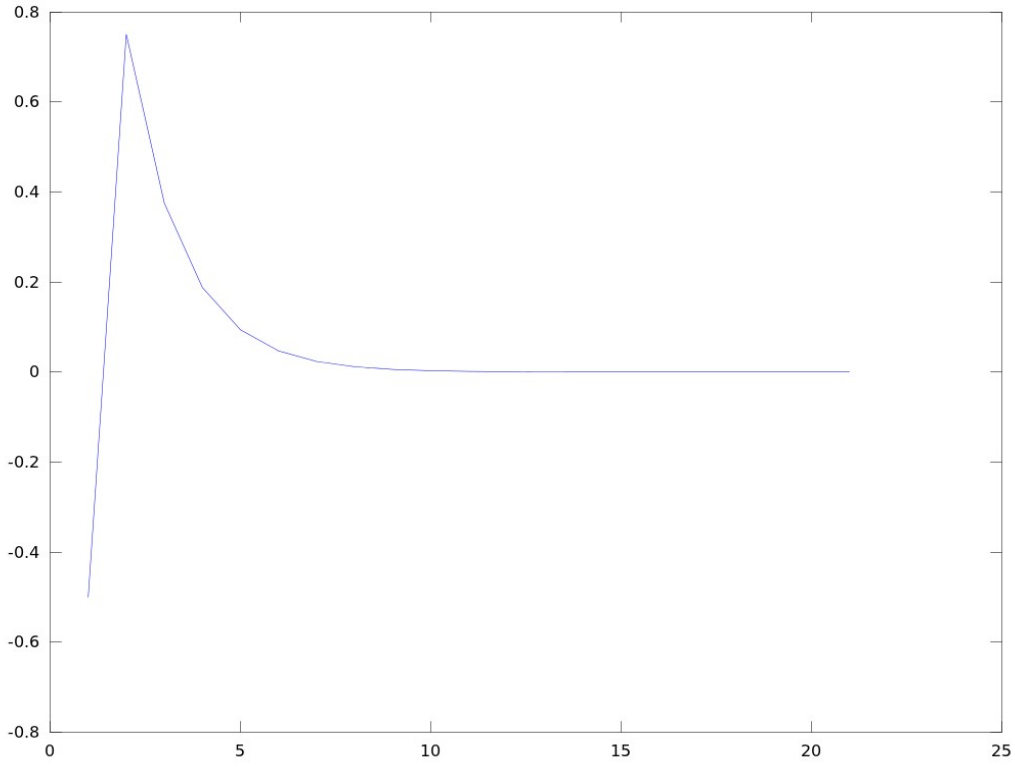


Here we can see in the above plot of the magnitude, that we indeed obtain a constant 1 (which is 0 dB), and that we have the **non-linear** phase in the lower plot, as in the phase plots before.

To obtain the impulse response, we can use the function “filter”, and input a unit impulse into it.

```
Imp=[1,zeros(1,20)];
h=filter(B,A,Imp);
plot(h);
```

we obtain the following impulse response plot



Here we can see that we have the first, non-delayed, sample not at zero, but at -0.5. This

can also be seen by plotting the first 4 elements of our impulse response:

```
h(1:4)
ans =
-0.50000 0.75000 0.37500 0.18750
```

The second element corresponds to the delay of 1 sample, our z^{-1} , with a factor of 0.75. But then there are more samples, going back into the past, exponentially decaying. This means, not only the past samples goes into our filtering calculation, but also more past samples, and even the **non-delayed** sample, with a factor of -0.5. This is actually a problem for the so-called frequency warping (next section), if we want to use frequency warping in IIR filters, because here we would get delay-less loops, which are difficult to implement! (With **FIR filters** this is **no problem** though)

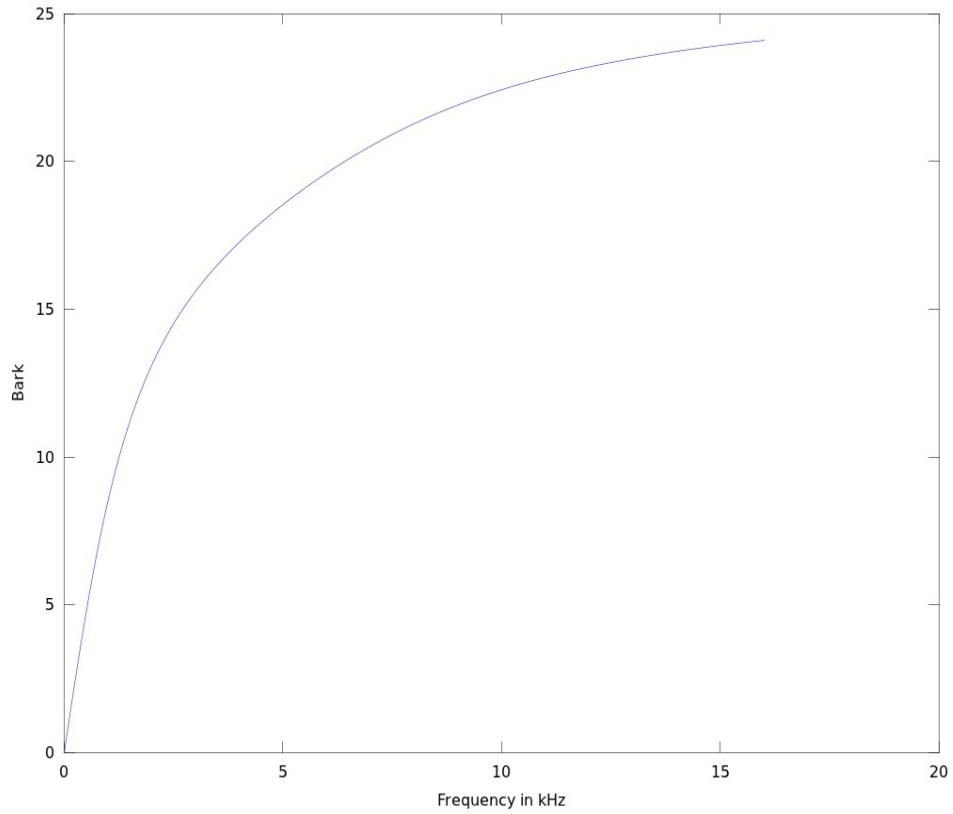
Frequency Warping

These properties of the allpass can now be used to “warp” the frequency scale of a filter (by effectively replacing $e^{j\Omega} \leftarrow e^{j\varphi(\Omega)}$ in our frequency response), for instance to map it according to the so-called **Bark scale**, used in psycho-acoustics.
A common **approximation** of the Bark scale is

$$Bark = 13 \cdot \arctan(0.0076 \cdot f) + 3.5 \cdot \arctan(((f/7500)^2))$$

(From Wikipedia, Bark scale), where f is the frequency in Hz. The Bark scale can be seen as an approximation of the changing frequency resolution over frequency of the inner ear filters of the human cochlea.

Because of the structure of our chochlea, the ear has different sensitivites for different frequencies and different signals. The signal dependent threshold of audibility of the ear is called the **Masking Threshold.** It has more spectral detail at lower than at higher frequencies, according to the Bark scale.



Here we can see, that 1 bark at lower frequency has a much lower bandwidth than at

higher frequencies. This means the ear can be seen as having a higher frequency resolution at lower frequencies than at higher frequencies. Imagine, we want to **design a filter** or system for **hearing purposes**, for instance, we would like to model the masking threshold of the ear for any given signal by some linear filter (FIR or IIR). Then it would be useful, to give this filter a **higher frequency resolution at lower frequencies**, such that it matches the smaller details of the **masking threshold** at lower frequencies. Then it would be useful, to give this filter a **higher frequency resolution at lower frequencies**, such that it matches the smaller details of the masking threshold at lower frequencies. But if we look at **the usual design methods**, they distribute the filter de**tails independent of the frequency range** (for instance what we saw with the remez method, where we have equally distributed ripples). Here we can now use frequency warping, such that we **enlarge the low frequency range** and shrink the high frequency range accordingly, such that our filter now works on the **warped frequency**, and **“sees” the lower frequencies in more detail**, the lower frequencies are more spread out in comparison to the higher frequencies.

How do we do this? For some frequency response $H\left(e^{j\Omega}\right)$ we would like to warp the

frequency Ω with some function $\varphi(\Omega)$ according to our desired frequency scale, such that we get

$$H\left(e^{j\cdot\varphi(\Omega)}\right)$$

But this is exactly the principle of an **allpass filter**, which has the frequency response

$$H_{ap}\left(e^{j\,\Omega}\right)=e^{j\cdot\varphi_{ap}(\Omega)}$$

Usually we would like to map positive frequencies to again positive frequencies, and we saw that $\varphi_{ap}(\Omega)$ becomes negative, hence we take the approach to **replace** z in the argument of our transfer function with the reverse of our **allpass** transfer function:

$$z^{-1}\leftarrow H_{ap}(a,z)$$

This is replacing all delays by our allpass filter. In this way we replace our linear function on the unit circle in z with the non-linear, warped function on the unit circle H_{ap}.
Hence we get the warped transfer function as

$$H_{warped}(z)=H\left(H_{ap}(a,z)^{-1}\right)$$

and the resulting frequency response becomes

$$H_{warped}\left(e^{j\Omega}\right)=H\left(e^{-j\cdot\varphi_{ap}(\Omega)}\right)$$

Here we can now see that we obtained the **desired frequency warping**.
What does this mean for the filter implementation? We know that our FIR filters always consist of many delay elements z^{-1}. Our replacement for z in the argument of our

transfer function means that we need to replace each delay element in our filter with

$$z^{-1} \leftarrow H_{ap}(a, z)$$

Example: Take an FIR filter,

$$H(z) = \sum_{m=0}^{L} b(m) \cdot z^{-m}$$

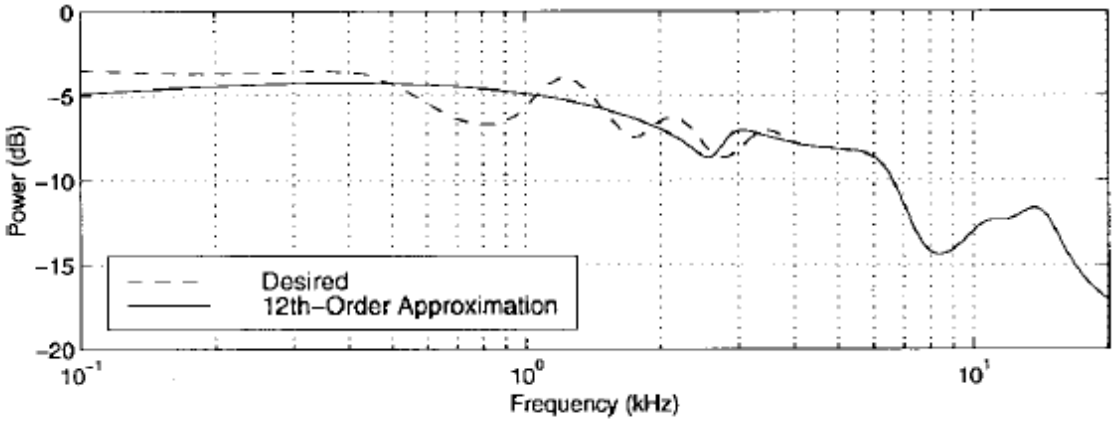
its warped version is:

$$H\left(H_{ap}(a, z)^{-1}\right) = \sum_{m=0}^{L} b(m) \cdot H_{ap}^{m}(a, z)$$

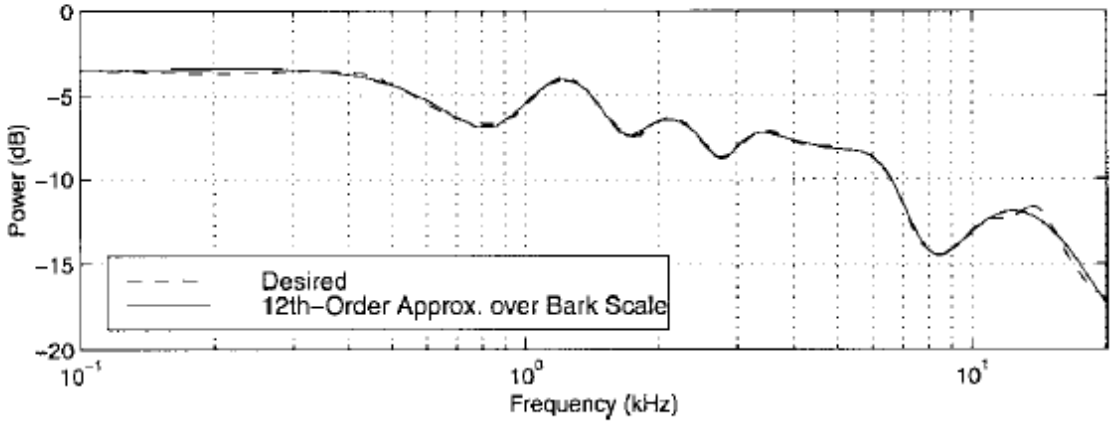
To obtain a desired filter, we now first have to **warp our desired filter,** and then **design** our filter in the **warped domain**.

Observe that the warping turns an **FIR filter into an IIR filter**, because the allpass has poles outside of zero.

An example of this kind of design can be seen in the following picture.



(a)



(b)

Fig. 10. Filter design example: Overlay of measured and modeled magnitude transfer functions, where the model is a twelfth-order filter designed by Prony's method. (a) Results without prewarping of the frequency axis. (b) Results using the Bark bilinear transform prewarping.

(From [1])

Here we can see that the 12^{th} order filter successfully approximated the more detailed curve at low frequencies, using the warping approach.

[1] Julius O. Smith and Jonathan S. Abel, “Bark and ERB Bilinear Transforms,” IEEE Transactions on Speech and Audio Processing, vol. 7, no. 6, pp. 697 – 708, November 1999.

[2] S. Wabnik, G. Schuller, U. Kraemer, J. Hirschfeld: "Frequency Warping in Low Delay Audio Coding", IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, PA, March 18–23, 2005