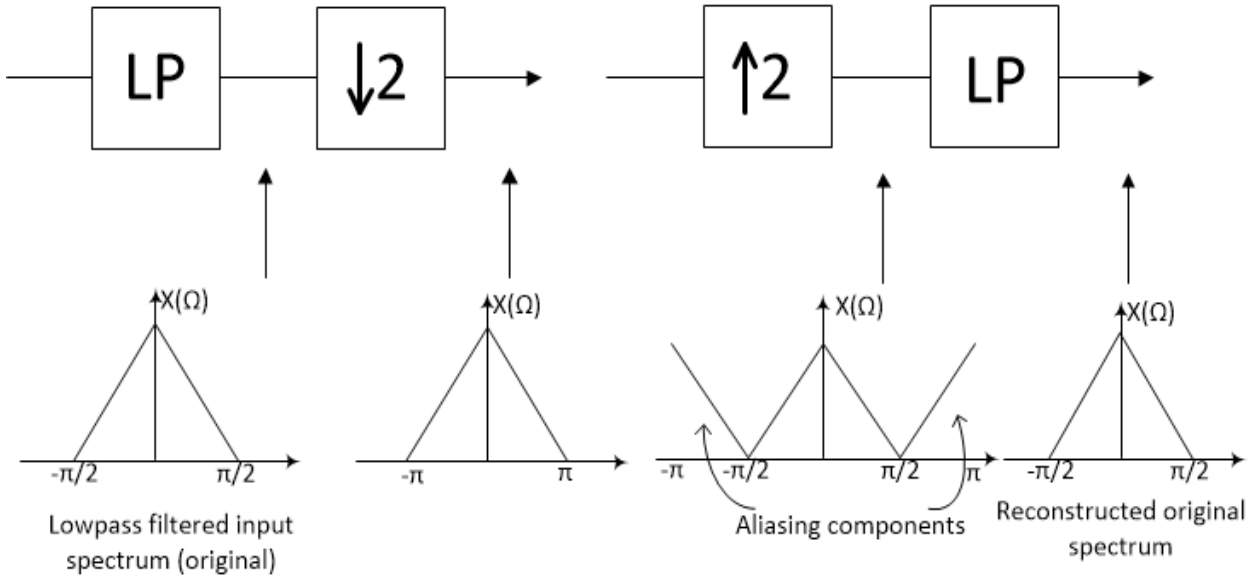
Lecture 9, Multirate Signal Processing, Sampling in the z-Domain

The standard low pass Nyquist case: here we have a lowpass signal which we downsample and reconstruct:

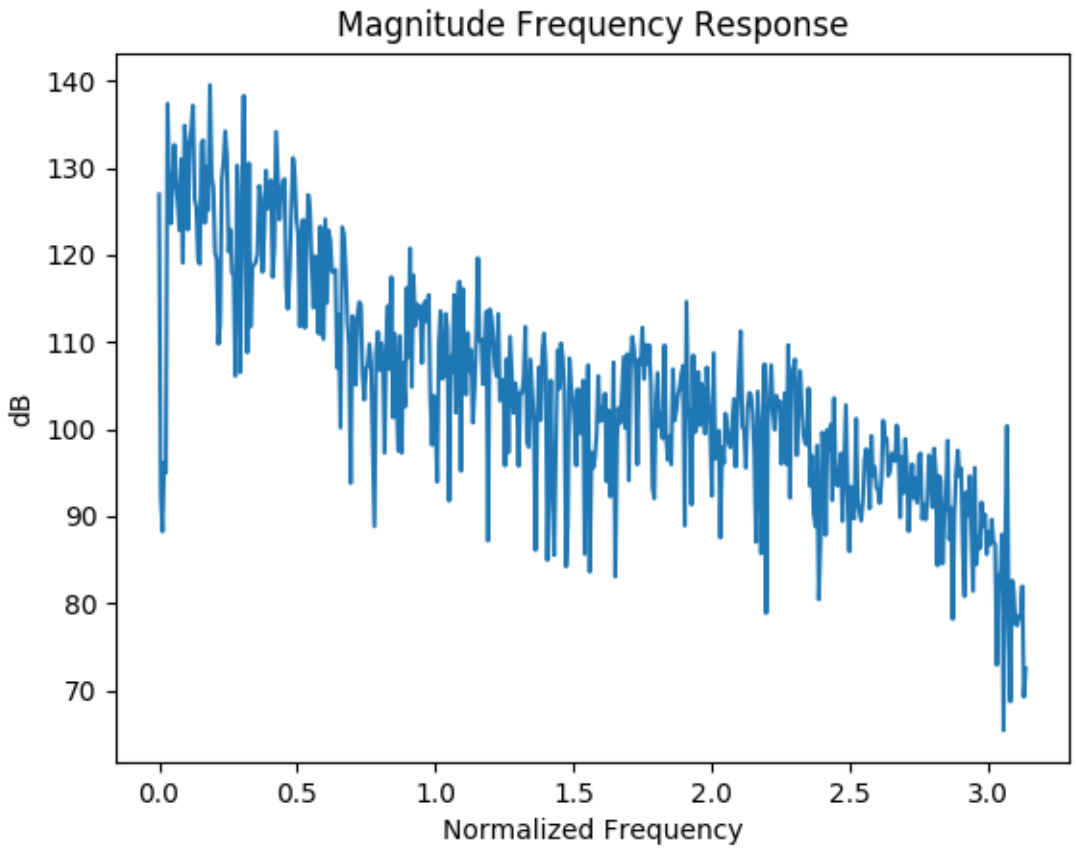


Python Example:

Just take an audio signal and read it into Python, and filter it with our previous low pass filter (e.g. Kaiser Window with Beta=8),

```
ipython –pylab
from scipy.io import wavfile
import scipy.signal as sig
from sound import sound
```

```
fs, x = wavfile.read('04_topchart.wav');
%listen to it:
sound(x,fs)
%Look at its spectrum:
w,H=sig.freqz(x)
plot(w,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Frequency Response')
```

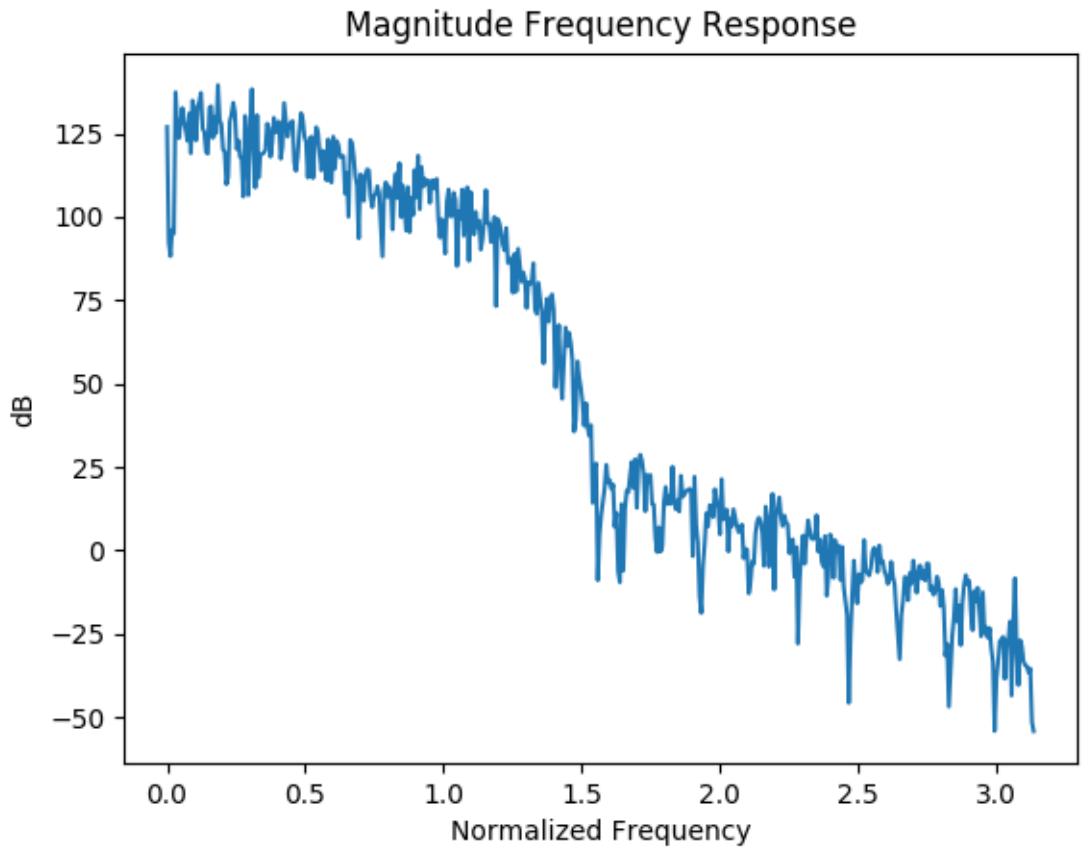


This is the spectrum of our original. Observe its already low pass characteristic.

Now we can low pass filter it. We take our Kaiser Window low pass filter design from slides Nr. 6,

```
n = arange(32)
w_c = 0.33*pi
n_d = (len(n)-1)/2.0
#ideal impulse response:
h = sin(w_c*(n-n_d))/(pi*(n-n_d))
#Kaiser window:
hk = np.kaiser(32,8)
#multiply ideal filter and Kaiser
window:
hfilt = hk*h
xlp=convolve(x,hfilt)

w,H=sig.freqz(xlp)
plot(w,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Frequency Response')
```



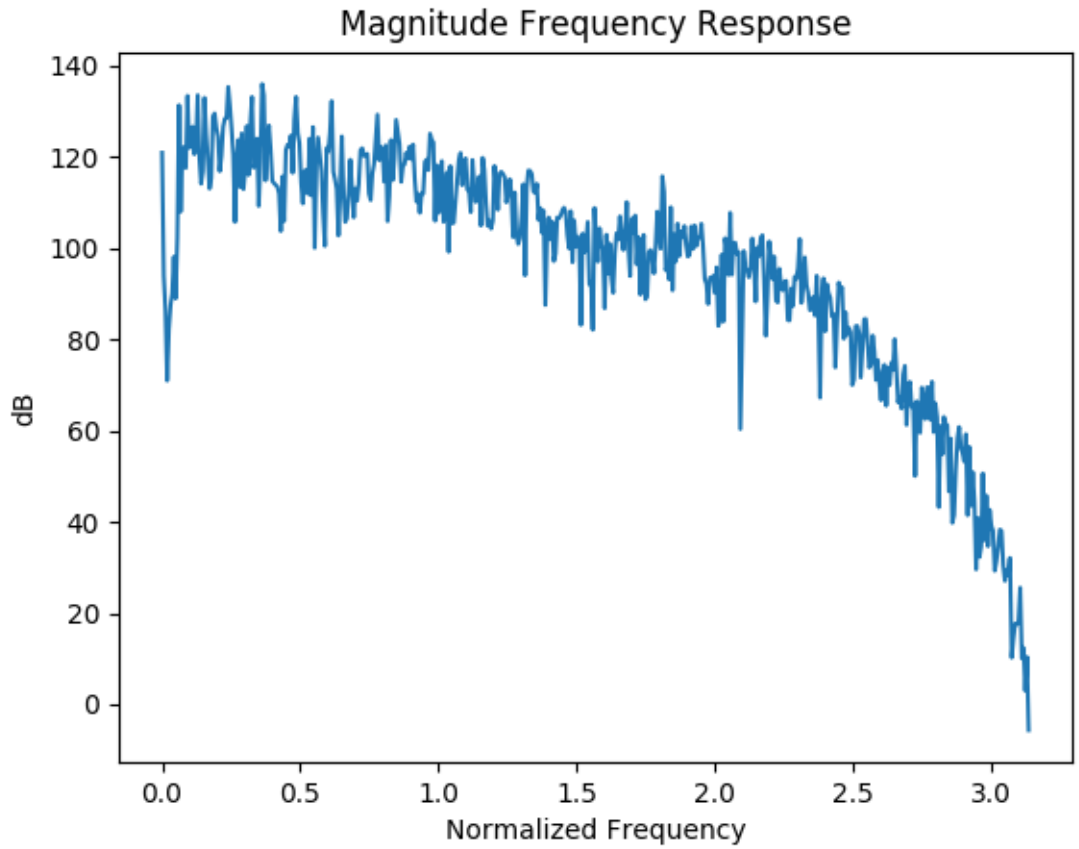
Observe that beginning at frequency 1.5 we have indeed much attenuation and hardly any signal left.
Listen to the low pass version:

```
sound(xlp,fs);
```

Now we can down-sample it by a factor of N=2, including the removal of the zeros,

```
xds = xlp[::2]
w,H=sig.freqz(xds)
plot(w,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
```

```
title('Magnitude Frequency Response')
```



Observe that we now obtain the “streched” spectrum.
Listen to it:

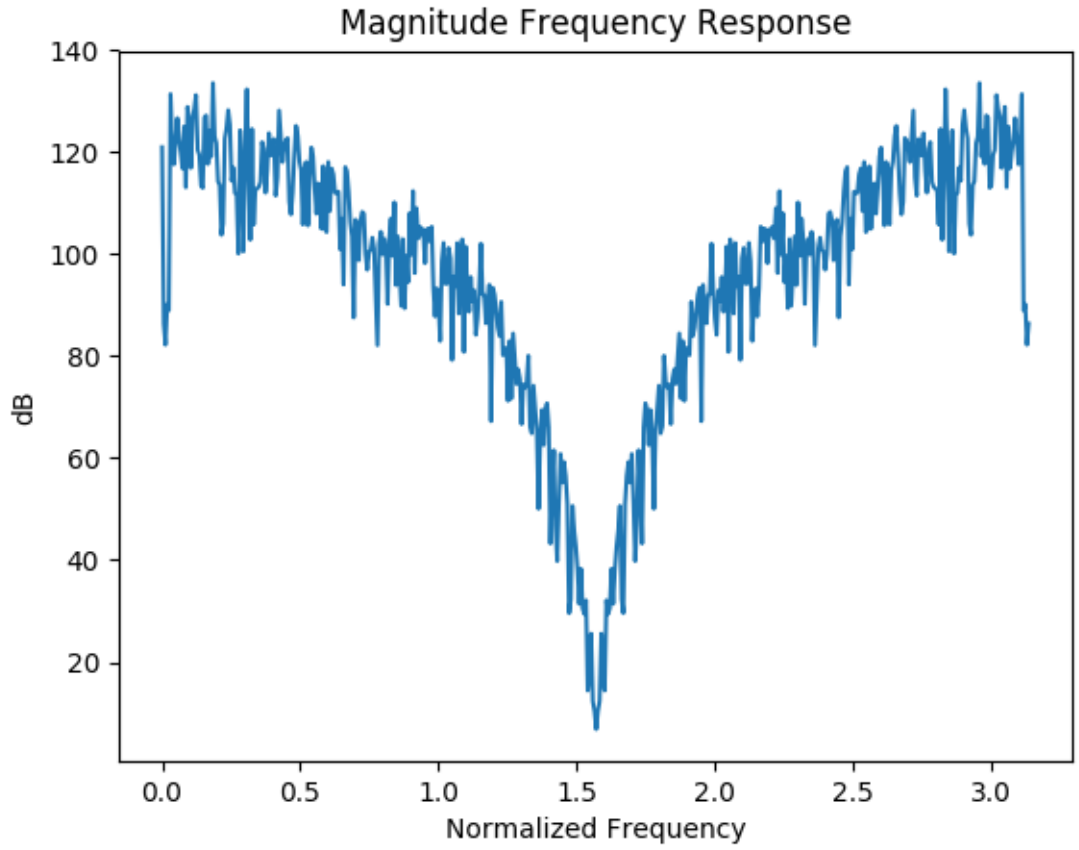
```
sound(xds,fs/2)
```

It should sound more “muffled”, but otherwise the same, but at now half the sampling rate!

Now we can upsample again, using the same indexing trick, now just on the receiving side, effectively inserting a zero after each sample. Observe that in this way we can avoid the

function “downsample” or “upsample”, which makes it clearer to see and check what happens,

```
xups = np.zeros(2*max(xds.shape))
xups[::2] = xds
w,H=sig.freqz(xups)
plot(w,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Frequency Response')
```



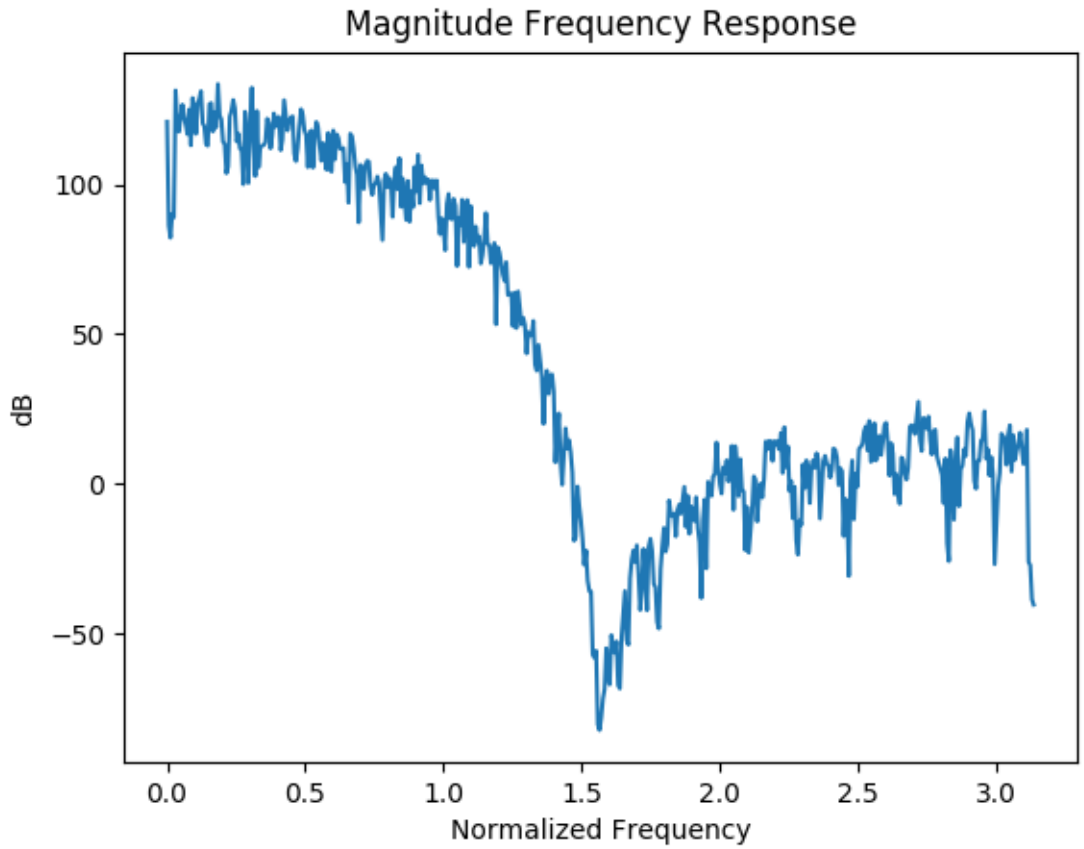
Observe the “shrinking” and periodic

continuation of the spectrum, the aliasing component at the high frequencies. Listen to the signal including aliasing,

```
sound(xups,fs);
```

Now low pass filter the result,

```
xupslp = np.convolve(xups,hfilt)
w,H=sig.freqz(xupslp)
plot(w,20*log10(abs(H)))
xlabel('Normalized Frequency')
ylabel('dB')
title('Magnitude Frequency Response')
```



Observe that now we removed the aliasing component at high frequencies.
Now listen to it,

```
sound(xupslp,32000);
```

Observe: it should now sound the same as at the lower sampling rate, but now at the higher sampling rate of 32 kHz! (Possible differences are due to not sufficiently attenuated aliasing).

Effects in the z-Domain

The z-Transform is a more general transform than the Fourier transform, and we will use it to obtain perfect reconstruction in filter banks and wavelets. Hence we will now look at the effects of sampling and some more tools in the z-domain.

Since we usually deal with causal systems in practice, we use the 1-sided z-Transform, defined as

$$X(z)=\sum_{n=0}^{\infty} x(n)z^{-n}$$

First observe that we get our usual frequency response (the Discrete Time Fourier Transform) if we evaluate the z-tranform along the unit circle in the z-domain,

$$z=\mathrm{e}^{j\Omega}$$

What is now the effect of **multiplying our signal with the unitpulse** train in the z-domain? To see this we simply apply the z-transform, and use our sum formulation for the delta impulse train,

$$X^{d}(z)=\sum_{n=0}^{\infty} x(n)\Delta_{N}(n)z^{-n}$$

Using

$$\Delta_{N}(n)=\frac{1}{N}\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}\cdot k\cdot n}$$

this becomes

$$\frac{1}{N}\sum_{k=0}^{N-1}\sum_{n=0}^{\infty} x(n)\left(\mathrm{e}^{-j\frac{2\pi}{N}\cdot k}\cdot z\right)^{-n}$$
$$=\frac{1}{N}\sum_{k=0}^{N-1} X\left(\mathrm{e}^{-j\frac{2\pi}{N}\cdot k}\cdot z\right)$$

(using the z-Transfrom definition and replacing z by $e^{-j\frac{2\pi}{N}\cdot k}\cdot z$:

$$X\left(e^{-j\frac{2\pi}{N}\cdot k}\cdot z\right)=\sum_{n=0}^{\infty} x(n)\left(e^{-j\frac{2\pi}{N}\cdot k}\cdot z\right)^{-n}$$)

or, in short,

$$X^d(z)=\frac{1}{N}\sum_{k=0}^{N-1} X\left(\mathrm{e}^{-j\frac{2\pi}{N}\cdot k}\cdot z\right)$$

This is very similar to the Discrete Time Fourier transform formulation. We get a sum with aliasing components, just that we now don't have frequency shifts for the aliasing terms, but a **multiplication of their exponential functions** to z. Here we effectively shift the phase (or use rotation) of the complex number z using this complex expoential. This also makes sense since the frequency information is contained in the phase of z , which we see if we replace $z=e^{j\Omega}$.

The next effect is the **removal or re-insertion of the zeros** from or into the signal. Let's again use our definition $y(m)=x^d(mN)$, meaning the y(m) is the signal without the zeros. Then the z-transform becomes,

$$Y(z)=\sum_{m=0}^{\infty} y(m)z^{-m}=$$
$$=\sum_{m=0}^{\infty} x^d(mN)z^{-m}=$$

Replacing the sum index m (the lower sampling rate) by the higher sampling rate n=mN, and observing that the sequence $x^d(n)$ contains the zeros, with $x^d(n)=0$ for $n\neq mN$, this results in (using the indiex substitution $n=mN$)

$$=\sum_{n=0}^{\infty} x^d(n)z^{-n/N}=\sum_{n=0}^{\infty} x^d(n)(z^{1/N})^{-n}=X^d(z^{1/N})$$

Observe the 1/N in the exponent of z! In short, we get

$$Y(z)=X^d(z^{1/N})$$

This **exponent 1/N** of z now correponds to the stretching of our frequency scale in the Fourier spectrum.

Modulation

Another very useful tool, which we already saw, is the **modulation**. This is the **multiplication** of our signal with a periodic function, for instance an exponential function. It can be written as

$$x_M(n):=x(n)\cdot \mathrm{e}^{-j\Omega_M\cdot n}.$$

(where "M" denotes the modulation). Observe that the modulation function here has a periodicity of $2\pi/\Omega_M$. Its z-transform hence

becomes

$$X_M(z)=\sum_{n=0}^{\infty} x(n)\cdot e^{-j\Omega_M\cdot n}\cdot z^{-n}$$

$$X_M(z)=X(\mathrm{e}^{j\Omega_M}\cdot z)$$

here again we have this rotation (or phase shift) of z by an exponential factor.

Time-Reversal

Another important tool is the **reversal of the ordering** of a finite length signal sequence, with length L (meaning x(n) is non-zero only for $n=0,\ldots,L-1$),

$$x_r(n):=x(L-1-n).$$

Its z-transform is

$$X_r(z)=\sum_{n=0}^{\infty} x(L-1-n)\cdot z^{-n}$$

we can now reverse the order of the summation (of course without affecting the result) by starting at the highest index, going to the lowest, replacing the index n by the expression $L-1-n'$ (index substitution),

$$X_r(z)=\sum_{n'=0}^{\infty} x(n')\cdot z^{-(L-1-n')}$$

$$=z^{-(L-1)}\cdot X(z^{-1})$$

or, in short,

$$X_r(z)=z^{-(L-1)}\cdot X(z^{-1})$$

(Remenber the z-transform was

$X(z)=\sum_{n=0}^{\infty} x(n)z^{-n}$)

So what we obtain is the inverse of z in the z-transform (which signifies the time reversal), and a factor of $z^{-(L-1)}$, which is simply a delay of L-1 samples! Important here is the inverse of z.

What difference does this make in our Fourier spectrum, replacing z by $e^{j\Omega}$? We obtain $X(-\Omega)$ instead of $X(\Omega)$. For **real valued** signals this only makes a **difference for the phases** of our frequency responses (they are sign flipped), because of the spectral symmetries for real valued signals. The **magnitudes are identical**.

This can still be of importance, for instance in filter banks with **aliasing cancellation**. Here the different signs also change the sign of the aliasing components, and that can make the difference between aliasing components cancelling between different bands or adding up!

For **complex valued** signals, the negative and positive frequencies can be **completely different**, and hence time-reversal would make a bigger difference.