Observe that the frequency response is **symmetric** around frequency 0. This is the result of **real valued impulse responses**. To see why we get this **symmetry**, assume we change ω to $-\omega$, we obtain

$$H(-\omega)=\sum_{n=-\infty}^{\infty} h(n)\cdot e^{-jn(-\omega)}$$

If we change the sign of the exponent in the above equation, it is equivalent to taking the conjugate complex value:

$$e^{-j\omega}=\overline{e^{j\omega}}$$

where the overbar " $\overline{\ }$ " means "conjugate complex". (remember:

$\overline{e^{j\omega}}=\overline{\cos(\omega)+j\sin(\omega)}=\cos(\omega)-j\sin(\omega)=e^{-j\omega}$)

Taking the conjugate complex means to change the sign of the imaginary part. Changing the sign of the complex exponential number means to rotate in the opposite direction, and this exactly corresponds to changing the sign of the resulting complex number.

Hence we obtain

$$H(-\omega)=\sum_{n=-\infty}^{\infty} h(n)e^{-jn(-\omega)}=\sum_{n=-\infty}^{\infty} \overline{h(n)}\,\overline{e^{-jn\omega}}=\overline{H(\omega)}$$

This is true because the complex conjugate does not change the **real valued** numbers h(n). It now shows the

conjugate symmetry between **negative and positive** frequencies if we have real valued impulse responses.
In summary: If we have real valued impulse response, the **positive and negative** frequency responses are **conjugate complex** of each other.

Observe that this does not change the magnitude of our frequency response, only the phase!
Also observe that this symmetry is **no longer true** if we have **complex** valued impulse responses!

Filters

Using frequency responses and the z-Transform, we can now analyse and design more general filters than just the running average or the running difference.
Applying a filter has the effect of multiplying the frequency domain description of a signal $X(\omega)$ with the frequency response or transfer function of a filter $H(\omega)$ to obtain the filter output $Y(\omega)$:

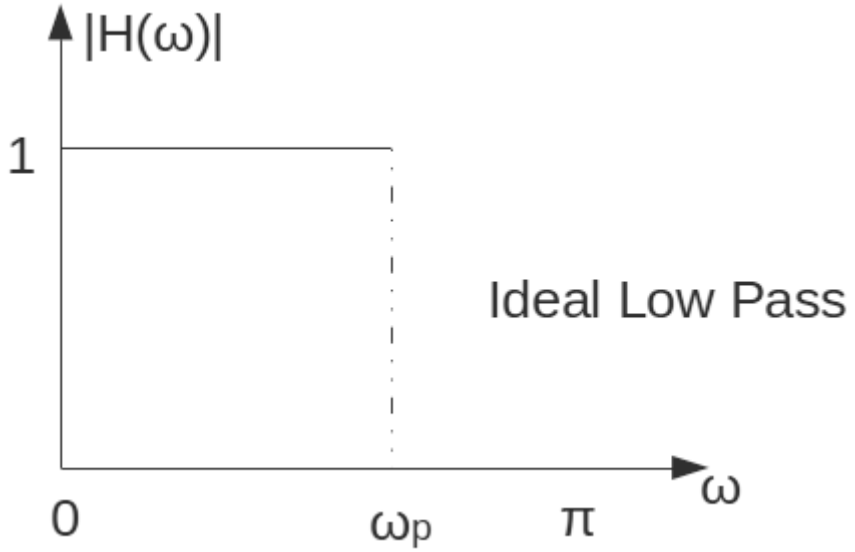
$$Y(\omega)=X(\omega)\cdot H(\omega)$$

In the time (or space) domain we get the corresponding operation by taking the inverse Discrete Time Fourier Transform. Remember that a multiplication in the frequency domain becomes a **convolution** in the time domain:

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

This is what we already saw for the example of the running average low pass filter with the matrix formulation!
This is now the more general formulation to obtain filters, which we can now use to design filters.

Example: An **ideal** low pass filter has a constant pass band, with constant magnitude of 1, and a constant stop band, with a magnitude of 0! This is also called a "brick wall filter", because of the similarity of the magnitude of the frequency response with a brick.



We have now very easily designed this filter in the **frequency domain** (this is our filter goal). We just need to know the pass band edge and the magnitudes of pass band and stop band. But usually we implement and compute our filters in the time or space domain (at

least for short filters this is more efficient to implement). How do we obtain a corresponding **impulse response**? First we also need to specify the **phase**, because the frequency response consists of 2 parts, the magnitude and the phase. There are filters where we want to specify the phase (as in all-pass filters), and there are filters where we don't care much about the phase, as in our low pass example. If we don't care about the phase, we can give it an arbitrary value, like 0. But there we find that we will get non-causal filters, so it makes sense to **add a delay** to make it causal (meaning that it starts at time 0). In that case we have a **linear phase** corresponding to that **delay**.
Why is a delay linear phase? We can write a **delay by m sample** points as multiplying the signal with z^{-m} in the z-domain:

$$H(z)\cdot z^{-m}=\sum_{n=0}^{\infty} h(n)z^{-n-m}=\sum_{n'=m}^{\infty} h(n'-m)z^{-n'}$$

with the index substitution $n':=n+m$, for a causal system. This is the z-Transform of the delayed signal $h(n-m)$, with a delay of m samples.
In summary: A **delay** of m samples in the time domain corresponds to a **multiplication** with z^{-m} in the z-domain.

Observe that this delay can also be written as a convolution with the following impulse response s(n) in the time domain:

$$s(n)=\left[\underbrace{0,0,\ldots,}_{m\,\text{zeros}}1\right]$$

Taking the z-Transform obviously again results in

$$S(z)=\sum_{n=0}^{\infty} s(n)z^{-n}=z^{-m}$$

Important to **remember**: A z-transform element of z^{-m} corresponds to a time-domain **delay of m samples**! (Because this is one of the fundamentals of multirate signal processing)

Python Example:

Define a signal like

```
ipython --pylab
x=range(1,5);
plot(x);
```

then write the shift operator:

```
s=array([0,0,0,1])
```

If we filter (convolve) our signal with this shift operator, we get a delayed signal:

```
sx=convolve(s,x);
plot(sx)
```

Here you can observe a delay by 3 samples.

We can now apply our trick to obtain the **frequency response** of the delay, we replace z by $e^{j\omega}$:

$$S(\omega)=e^{-jm\omega}$$

The magnitude of it is obviously 1, and the phase is $-m\cdot\omega$, so it is linear with a slope of -m!

In Python plot the frequency response of our shift operator above:

```
ipython -pylab
import scipy.signal as signal
omega, H= signal.freqz(s);
plot(omega, angle(H))
```

What do you observe?

So now we know what to do with the phase. If we want to have causal filters, we need to have this delay with the corresponding phase.
How do we continue with the desired magnitude? Since we now have both parts, magnitude and phase, we can simply apply the **inverse** Discrete Time Fourier Transform to obtain the corresponding **impulse response** h(n).
For our ideal low pass filter we obtain the well known sinc function as impulse response. For the phase equal to 0 and for a desired upper limit of our passband ω_p we obtain the following:

$$h(n)=\frac{\sin(n\omega_p)}{n\omega_p}$$

with $n=-\infty,\ldots,\infty$, which makes the filter indeed non-causal and it cannot be made causal.

This shows that it is not practically implementable (which is disappointing since it was a nice filter :-)).