High pass filter

A high-pass filter allows frequencies above a certain cutoff frequency to pass through while attenuating frequencies below that cutoff. Here's a mathematical explanation of how a high-pass filter works, particularly the Butterworth filter used in the provided code.

### **High-Pass Filter Design**

#### **Butterworth Filter**

A Butterworth filter is designed to have a flat frequency response in the passband. The high-pass filter can be designed using the following steps:

1. **Determine the cutoff frequency**: This is the frequency below which the signal will be attenuated.
2. **Normalize the cutoff frequency**: This is done by dividing the cutoff frequency by the Nyquist frequency (half the sampling rate).

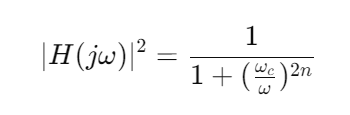
Where ​ is the cutoff frequency and is the sampling rate.

3. **Calculate filter coefficients**: Using the normalized cutoff frequency, the filter coefficients are calculated. For a Butterworth filter, these coefficients can be obtained using the bilinear transformation.

#### 

#### **Butterworth High-Pass Filter**

The Butterworth filter is characterized by its maximally flat frequency response in the passband. The transfer function H(s) of an n-th order Butterworth filter is given by:



where:

* ω is the angular frequency,
* ωc​ is the cutoff angular frequency,
* n is the order of the filter.

In the digital domain, the Butterworth filter can be implemented using difference equations derived from its transfer function.

### **Implementing the Filter**

The Butterworth high-pass filter can be implemented using the following steps:

1. **Design the filter coefficients**: Using a digital filter design method like the bilinear transform, the continuous-time filter coefficients are transformed to discrete-time filter coefficients.
2. **Apply the filter to the signal**: The discrete-time signal x[n] is filtered using the designed coefficients to obtain the filtered signal y[n].

The difference equation for a first-order Butterworth high-pass filter is given by:

y[n] = ​x[n] + ​x[n−1] − ​y[n−1]

where ​, ​, and​ are the filter coefficients calculated from the normalized cutoff frequency.

### **Example Code Explanation**

In the provided code, the butter function from scipy.signal is used to design a first-order Butterworth high-pass filter. The filtfilt function applies this filter to the input signal in both the forward and reverse directions to achieve zero phase distortion. Here’s the breakdown of the filter design and application:

nyquist = 0.5 \* sample\_rate

normal\_cutoff = cutoff\_freq / nyquist

b, a = butter(1, normal\_cutoff, btype='high', analog=False)

The Nyquist frequency is calculated as half the sample rate.

The normalized cutoff frequency is computed.

The butter function designs a first-order Butterworth high-pass filter, returning the filter coefficients b and a.

filtered\_data = filtfilt(b, a, data)

The filtfilt function applies the filter to the input data in a forward and backward direction, ensuring zero phase distortion.