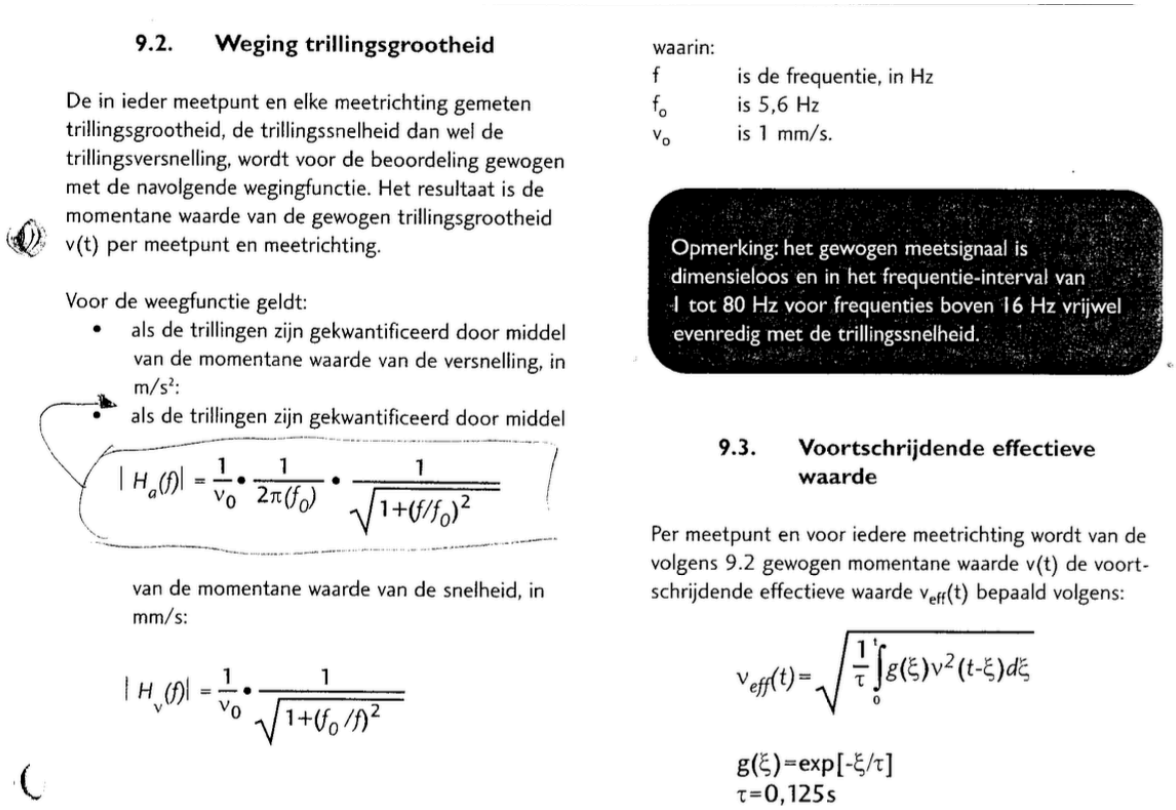
```
import numpy as np; import matplotlib.pyplot as plt; from scipy import s
plt.rcParams['axes.grid'] = True; plt.rcParams['figure.figsize'] = [10,
fs = 1000
```

SBR B Wegingsfilter

SBR B 9.2 specificeert twee verschillende filters. Een filter om de weging toe te passen in het acceleratiedomein en een filter voor als het trillingssignaal in het snelheidsdomein is.



9.2. Weging trillingsgrootheid

De in ieder meetpunt en elke meetrichting gemeten trillingsgrootheid, de trillingssnelheid dan wel de trillingsversnelling, wordt voor de beoordeling gewogen met de navolgende wegingfunctie. Het resultaat is de momentane waarde van de gewogen trillingsgrootheid v(t) per meetpunt en meetrichting.

Voor de weegfunctie geldt:

- als de trillingen zijn gekwantificeerd door middel van de momentane waarde van de versnelling, in m/s^2:
- als de trillingen zijn gekwantificeerd door middel van de momentane waarde van de snelheid, in mm/s:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi(f_0)} \cdot \frac{1}{\sqrt{1+(f/f_0)^2}}$$

$$|H_v(f)| = \frac{1}{v_0} \cdot \frac{1}{\sqrt{1+(f_0/f)^2}}$$

waarin:

f	is de frequentie, in Hz
f_o	is 5,6 Hz
v_o	is 1 mm/s.

Opmerking: het gewogen meetsignaal is dimensieloos en in het frequentie-interval van 1 tot 80 Hz voor frequenties boven 16 Hz vrijwel evenredig met de trillingssnelheid.

9.3. Voortschrijdende effectieve waarde

Per meetpunt en voor iedere meetrichting wordt van de volgens 9.2 gewogen momentane waarde v(t) de voortschrijdende effectieve waarde $v_{eff}(t)$ bepaald volgens:

$$v_{eff}(t) = \sqrt{\frac{1}{\tau}\int_0^t g(\xi)v^2(t-\xi)d\xi}$$

$g(\xi) = \exp[-\xi/\tau]$
$\tau = 0{,}125\,s$

Acceleratiedomein

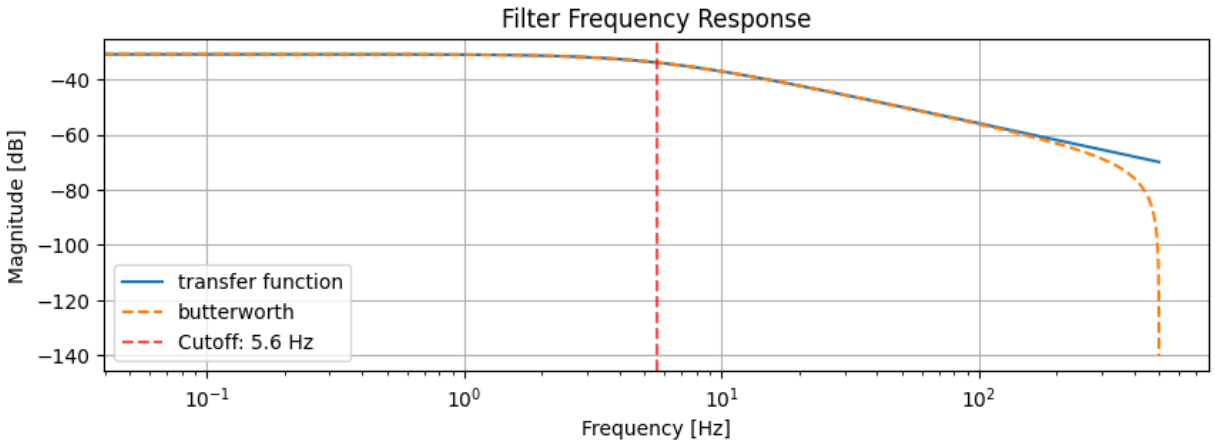
De filter voor het acceleratiedomein is gedefinieerd als

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi(f_0)} \cdot \frac{1}{\sqrt{1+(f/f_0)^2}}$$

Met $v_0 = 1$ en $f_0 = 5.6\text{Hz}$

Dit is een eerste orde Butterworth lowpass filter ($\frac{1}{\sqrt{1+(f/f_0)^2}}$) met een extra verzwakkingsfactor $\frac{1}{2\pi(f_0)}$.

```
# SBRB low pass 5.6 Hz filter
v0 = 1
f0 = 5.6

# Create the filter using scipy's signal.butter
# Wn is the normalized cutoff frequency (cutoff / nyquist)
Wn = f0 / (fs/2)
lp_b, lp_a = signal.butter(1, Wn, btype='low', analog=False)

# apply the extra attenuation factor to the nominator
lp_b = lp_b / (v0 * 2 *np.pi * f0)

w, h = signal.freqz(lp_b, lp_a, worN=8000)
f = w * fs / (2 * np.pi)

# Calculate the frequency domain transfer function for the same frequenc
H = 1 / v0 * 1 / (2 *np.pi * f0) * 1 / (np.sqrt(1 + (f / f0)**2))

plt.semilogx(f, 20*np.log10(H), label="transfer function")
plt.semilogx(f, 20 * np.log10(abs(h)), label="butterworth", linestyle="-
plt.title('Filter Frequency Response'); plt.xlabel('Frequency [Hz]'); pl
plt.axvline(f0, color='red', linestyle='--', alpha=0.7, label=f'Cutoff:
plt.legend(); plt.show()
```



We zien dat de scipy butterworth filter en de transfer function zoals gedefinieerd in de SBR B richtlijn exact overeenkomen behalve voor de hoogste frequenties. Deze afwijking komt doordat we een analoge filter vergelijken met een digitale filter waarbij we gelimiteerd zijn tot de Nyquist frequentie.

Snelheidsdomein

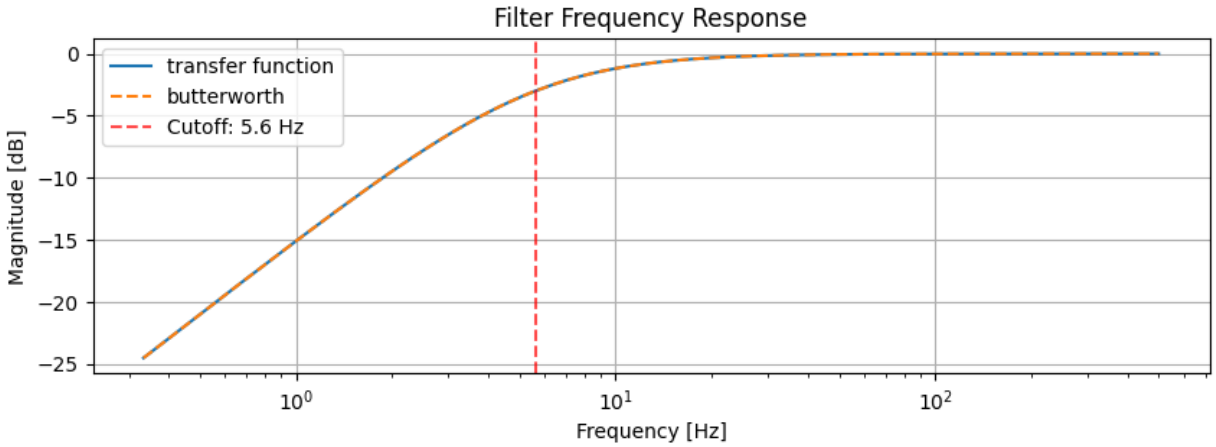
De wegingsfilter die gedefinieerd is voor het snelheidsdomein is gedefinieerd als

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{\sqrt{1 + (f_0/f)^2}}$$

Met $v_0 = 1$ en $f_0 = 5.6\text{Hz}$

Dit is een eerste orde Butterworth highpass filter.

```
# SBRB high pass 5.6 Hz filter
v0 = 1
f0 = 5.6

# Create the filter using scipy's signal.butter
# Wn is the normalized cutoff frequency (cutoff / nyquist)
Wn = f0 / (fs/2)
hp_b, hp_a = signal.butter(1, Wn, btype='high', analog=False)

w, h = signal.freqz(hp_b, hp_a, worN=1500)
f = w * fs / (2 * np.pi)

#  H_a(f) = (1/v0) * (1/sqrt(1+(f0/f)^2))
H = 1 / v0 * 1 / (np.sqrt(1 + (f0 / f[1:])**2))

plt.semilogx(f[1:], 20*np.log10(H), label="transfer function")
plt.semilogx(f[1:], 20 * np.log10(abs(h[1:])), label="butterworth", lines
plt.title('Filter Frequency Response'); plt.xlabel('Frequency [Hz]'); pl
plt.axvline(f0, color='red', linestyle='--', alpha=0.7, label=f'Cutoff:
plt.legend(); plt.show()
```



Nu hebben we aangetoond dat we het wegingsfilter inderdaad kunnen implementeren als

een digitale eerste orde Butterworth IIR filter.

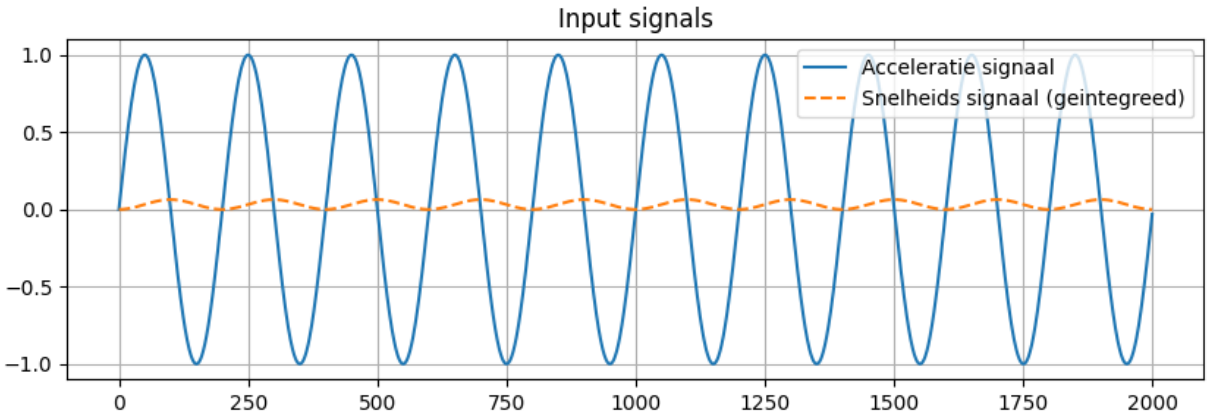
Acceleratie vs Snelheidsdomein

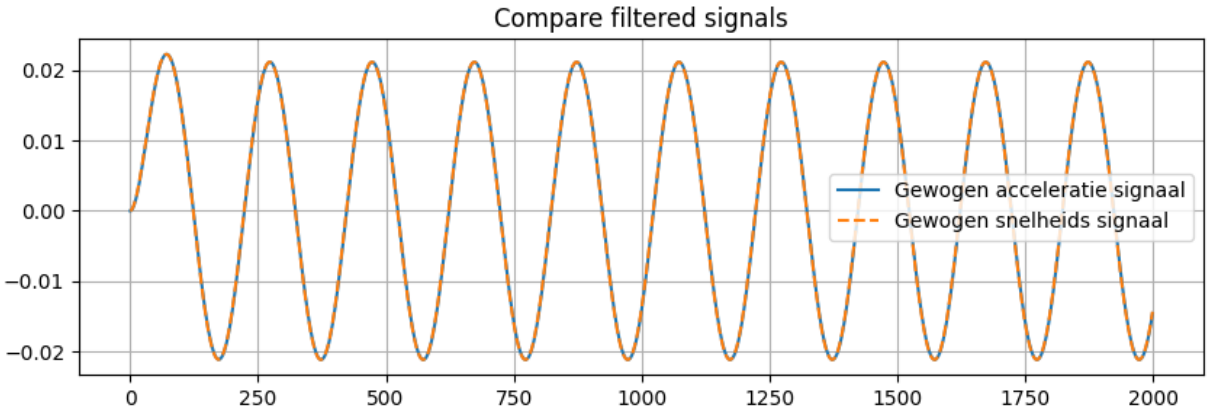
Nu gaan we laten zien dat deze twee filters op het zelfde resultaat uitkomen. We definieren een test signaal in het acceleratiedomein: een sinusvorm met frequentie 5Hz.

- We passen we de wegingsfilter toe voor het acceleratie domein.
- We integreren het signaal en passen dan het wegingsfilter toe voor het snelheidsdomein.

Uiteindelijk leggen we de gefilterde signalen over elkaar heen.

```
T = 2 # sec
t = np.arange(0, T, 1/fs)

x_accel = np.sin(2*np.pi*5*t)

# For the acceleration test signal we use the low pass filter
y1 = signal.lfilter(lp_b, lp_a, x_accel)

# Now we integrate the acceleration signal
x_velocity = np.cumsum(x_accel) / fs

# for the velocity signal we use the highpass filter
y2 = signal.lfilter(hp_b, hp_a, x_velocity)

plt.plot(x_accel, label="Acceleratie signaal")
plt.plot(x_velocity, label="Snelheids signaal (geintegreed)", linestyle=
plt.title("Input signals")
plt.legend(); plt.show()
plt.figure()
plt.plot(y1, label="Gewogen acceleratie signaal")
plt.plot(y2, label="Gewogen snelheids signaal", linestyle="--")
plt.title("Compare filtered signals")
plt.legend(); plt.show()
```





Waarom zijn ze precies gelijk?

Als we kijken naar de formule voor de acceleratiedomein filter:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi(f_0)} \cdot \frac{1}{\sqrt{1 + (f/f_0)^2}}$$

We kunnen dit hetschrijven als

$$|H_a(f)| = \frac{1}{2\pi f} \cdot \left(\frac{1}{v_0} \cdot \frac{1}{\sqrt{1 + (f_0/f)^2}} \right)$$

En nu zien we dat de eerste term $\frac{1}{2\pi f}$ een integrator is, gevolg door een eerst orde *highpass* Butterworth filter. Hieronder de wiskunde die dit laat zien.

Stap 1: Herschrijven van de wortelterm

De term binnen de wortel kan als volgt worden herschreven:

$$1 + (f/f_0)^2 = \frac{f_0^2}{f_0^2} + \frac{f^2}{f_0^2} = \frac{f_0^2 + f^2}{f_0^2}$$

Daarom geldt:

$$\sqrt{1+(f/f_0)^2} = \sqrt{\frac{f_0^2+f^2}{f_0^2}} = \frac{\sqrt{f_0^2+f^2}}{\sqrt{f_0^2}} = \frac{\sqrt{f_0^2+f^2}}{f_0}$$

Stap 2: Invoegen in de oorspronkelijke functie

We substitueren dit terug:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi f_0} \cdot \frac{1}{\frac{\sqrt{f_0^2+f^2}}{f_0}}$$

Dit kunnen we vereenvoudigen:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi f_0} \cdot \frac{f_0}{\sqrt{f_0^2+f^2}}$$

De factor f_0 in teller en noemer heffen elkaar op:

Stap 3: Introduceren van de factor f/f

Om de integrator te herkennen, vermenigvuldigen we teller en noemer met f/f:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi} \cdot \frac{1}{\sqrt{f_0^2+f^2}} \cdot \frac{f}{f}$$

Omdat f een factor van de wortel kan worden:

$$\sqrt{f_0^2+f^2} = f \cdot \sqrt{(f_0/f)^2+1}$$

krijgen we:

$$|H_a(f)| = \frac{1}{v_0} \cdot \frac{1}{2\pi f} \cdot \frac{1}{\sqrt{1+(f_0/f)^2}}$$

Stap 4: Identificeren van componenten

Nu zien we duidelijk:

$$\frac{1}{2\pi f}$$

is een **integrator** en

$$\frac{1}{\sqrt{1+(f_0/f)^2}}$$

is een **hoogdoorlaatfilter** met kantelfrequentie f_0.

Conclusie

De oorspronkelijke overdrachtsfunctie kan dus worden herschreven als:

$$|H_a(f)| = \left(\frac{1}{2\pi f}\right) \cdot \left(\frac{1}{v_0} \cdot \frac{1}{\sqrt{1+(f_0/f)^2}}\right)$$

waarbij:

- $\frac{1}{2\pi f}$ een **integrator** is.
- $\frac{1}{\sqrt{1+(f_0/f)^2}}$ een **eerste-orde hoogdoorlaat Butterworth-filter** is.

Dit bevestigt dat het oorspronkelijke filter een **integrator gevolgd door een hoogdoorlaatfilter** is.