Filtr wszechprzepustowy

Jest to filtr o nieskończonej odpowiedzi impulsowej posiadający charakterystykę częstotliwościową równą jeden w całym zakresie częstotliwości. Najważniejszą cechą tego filtra jest charakterystyka fazowa nie będąca funkcją stałą.

$$H(z) = \frac{g + z^{-1}}{1 + g \cdot z^{-1}}$$

$$g = re^{j\theta}$$

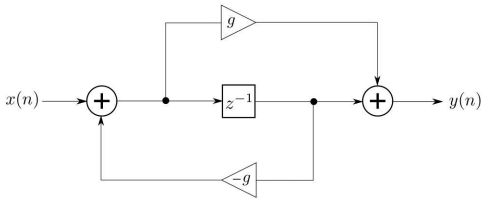
$$\phi(\omega) = -\omega - 2\,\mathrm{arc\,tg}\left[\frac{r\sin(\omega-\theta)}{1 - r\cos(\omega-\theta)}\right]$$

Opóźnienie sinunoidy o częstotliwości ω_k określa zależność:

$$\tau_k = -\frac{\phi(\omega_k)}{\omega_k}$$

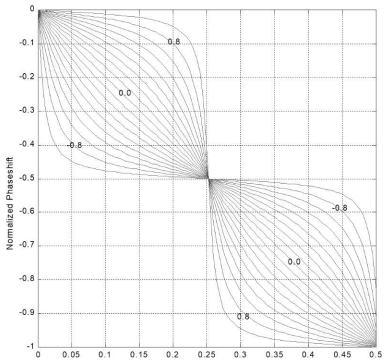
Filtr wszechprzepustowy

Struktura filtra wszechprzepustowego pierwszego rzędu:



$$y(n) = g \cdot x(n) + x(n-1) - g \cdot y(n-1)$$

Filtr wszechprzepustowy



Filtr grzebieniowy

Filtr grzebieniowy realizowany jest poprzez oddziaływanie na sygnał wejściowy wersją opóźnioną tego sygnału. W wyniku tego procesu odpowiedź filtra przyjmuje postać równoodległych szpilek w dziedzinie częstotliwości.

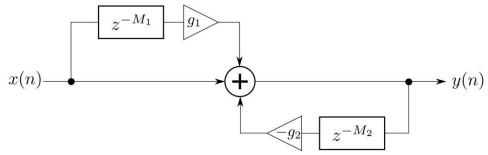
Tego typu filtr najczęściej stosowany jest w aplikacjach audio do symulacji efektów związanych z opóźnieniem oraz w systemach realizujących interpolację i decymację (filtry CIC).

Transmitancja filtra grzebieniowego (feedforward+feedback):

$$H(z) = \frac{1 + g_1 \cdot z^{-M_1}}{1 + g_2 \cdot z^{-M_2}}$$

Filtr grzebieniowy

Struktura filtra grzebieniowego:

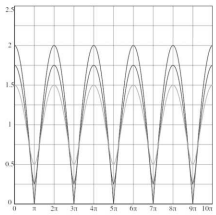


$$y(n) = x(n) + g_1 \cdot x(n - M_1) - g_2 \cdot y(n - M_2)$$

Filtr grzebieniowy

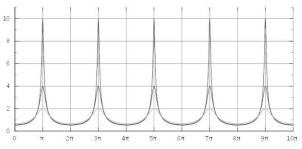
$$H_1(z) = 1 + g_1 \cdot z^{-M_1}$$

$$y(n) = x(n) + g_1 \cdot x(n - M_1)$$



$$H_2(z) = \frac{1}{1 - g_2 \cdot z^{-M_2}}$$

$$y(n) = x(n) - g_2 \cdot y(n - M_2)$$



Filtr blokujący składową stałą

Jest to filtr górnoprzepustowy, którego zadaniem jest usunięcie składowej stałej z sygnału.

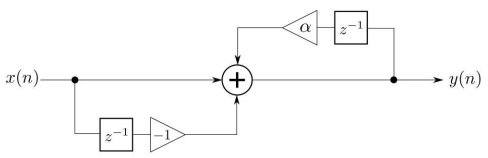
Transmitancja tego filtra opisana jest następująco:

$$H(z) = \frac{1 - z^{-1}}{1 - \alpha \cdot z^{-1}}$$

W praktycznych zastosowaniach, wartość współczynnika α dobiera się pomiędzy 0.9 a 1.

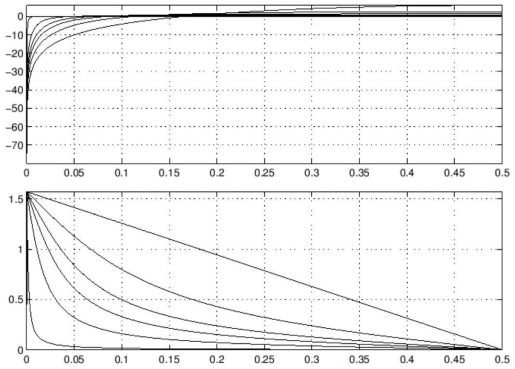
Filtr blokujący składową stałą

Struktura filtra blokującego składową stałą:



$$y(n) = x(n) - x(n-1) + \alpha \cdot y(n-1)$$

Filtr blokujący składową stałą

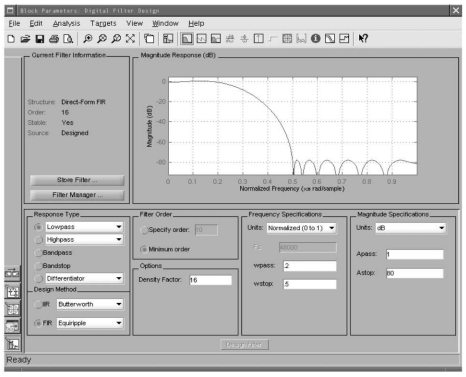


Projektowanie filtrów w praktyce

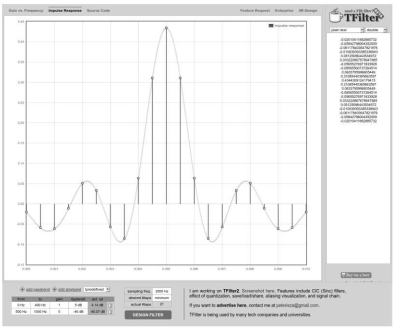
Często wykorzystywanym narzędziem do projektowania filtrów w praktyce jest środowisko **MATLAB/Simulink**. Istnieją również środowiska darmowe takie jak **SciLab** (`www.scilab.org`), lub **Octave** (`www.octave.org`) oferujące gotowe narzędzia do projektowania szerokiej gamy filtrów cyfrowych.

W przypadku środowiska **MATLAB/Simulink** do projektowania można wykorzystać narzędzie wizualne FDATool:

Projektowanie filtrów w praktyce

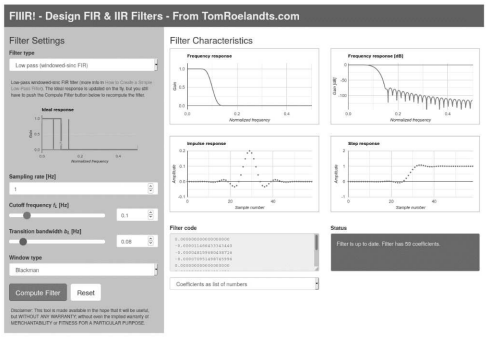


Projektowanie filtrów w praktyce



http://t-filter.engineerjs.com/

Projektowanie filtrów w praktyce



http://fiiir.com/

Projektowanie filtrów w praktyce

Przykładowe metody służące do projektowania filtrów FIR dostępne w bibliotece **scipy.signal** (Python)

SciPy.org | Docs | SciPy v1.4.1 Reference Guide | Signal processing (scipy.signal)

scipy.signal.firwin

scipy.signal.**firwin**(*numtaps, cutoff, width=None, window='hamming', pass_zero=True, scale=True, nyq=None, fs=None*)

FIR filter design using the window method.

SciPy.org | Docs | SciPy v1.4.1 Reference Guide | Signal processing (scipy.signal)

scipy.signal.firls

scipy.signal.**firls**(*numtaps, bands, desired, weight=None, nyq=None, fs=None*)

FIR filter design using least-squares error minimization.

SciPy.org | Docs | SciPy v1.4.1 Reference Guide | Signal processing (scipy.signal)

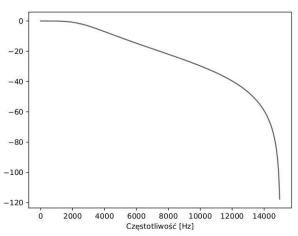
scipy.signal.remez

scipy.signal.**remez**(*numtaps, bands, desired, weight=None, Hz=None, type='bandpass', maxiter=25, grid_density=16, fs=None*)

Calculate the minimax optimal filter using the Remez exchange algorithm.

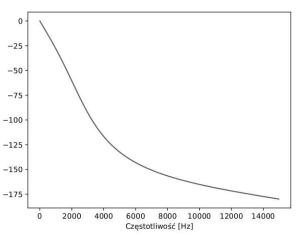
Charakterystyka filtra

```
import scipy.signal as signal
import numpy as np
import matplotlib.pyplot as plt

fs = 30000

a = [1, -1.168261, 0.424118]
b = [0.063964, 0.127929, 0.063964]

w, h = signal.freqz(b,a)

x = w * fs / (2 * np.pi)
y = 20 * np.log10(np.abs(h))

plt.plot(x, y)
plt.xlabel("Czestotliwosc [Hz]")
plt.show()
```



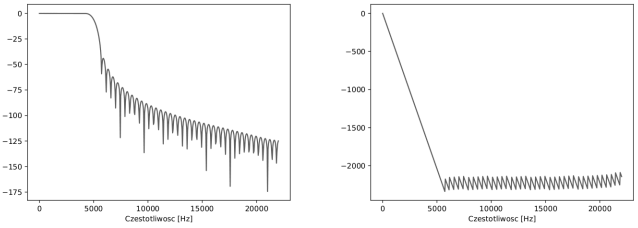
Charakterystyka filtra

```
import scipy.signal as signal
import numpy as np
import matplotlib.pyplot as plt

fs = 30000

a = [1, -1.168261, 0.424118]
b = [0.063964, 0.127929, 0.063964]

w, h = signal.freqz(b,a)

x = w * fs / (2 * np.pi)
y = np.arctan2(np.imag(h), np.real(h))
y = np.degrees(np.unwrap(y))

plt.plot(x, y)
plt.xlabel("Czestotliwosc [Hz]")
plt.show()
```



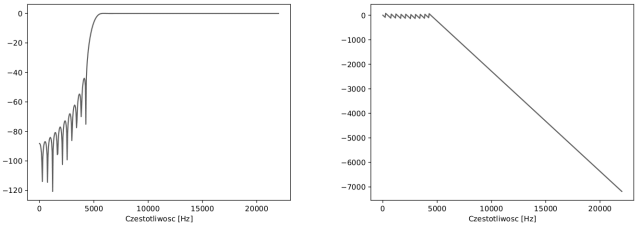
Filtr FIR dolno-przepustowy

```
fs = 44100  # czestotliwosc probkowania
n  = 101    # rzad filtra
fc = 5000   # czestotliwosc odciecia

b = signal.firwin(n, cutoff = fc, fs=fs,
                  window = "hanning",
                  pass_zero="lowpass")

w, h = signal.freqz(b, [1])
```



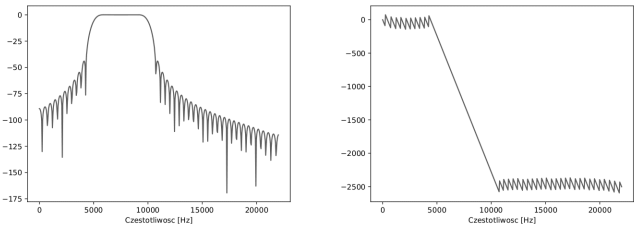
Filtr FIR górno-przepustowy

```
fs = 44100  # czestotliwosc probkowania
n  = 101    # rzad filtra
fc = 5000   # czestotliwosc odciecia

b = signal.firwin(n, cutoff = fc, fs=fs,
                  window = "hanning",
                  pass_zero="highpass")

w, h = signal.freqz(b, [1])
```



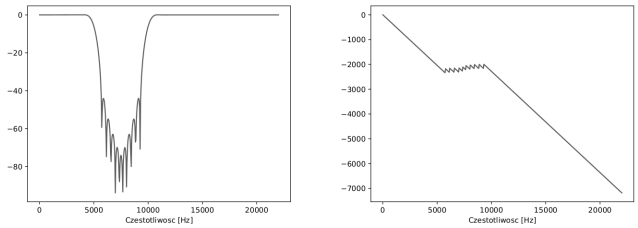
Filtr FIR pasmowo-przepustowy

```
fs  = 44100  # czestotliwosc probkowania
n   = 101    # rzad filtra
fc1 = 5000   # czestotliwosc graniczna (L)
fc2 = 10000  # czestotliwosc graniczna (R)

b = signal.firwin(n, cutoff = [fc1, fc2], fs=fs,
                  window = "hanning",
                  pass_zero="bandpass")

w, h = signal.freqz(b, [1])
```



Filtr FIR pasmowo-zaporowy

```
fs  = 44100  # czestotliwosc probkowania
n   = 101    # rzad filtra
fc1 = 5000   # czestotliwosc graniczna (L)
fc2 = 10000  # czestotliwosc graniczna (R)

b = signal.firwin(n, cutoff = [fc1, fc2], fs=fs,
                  window = "hanning",
                  pass_zero="bandstop")

w, h = signal.freqz(b, [1])
```



Projektowanie filtrów w praktyce

Przykładowe metody służące do projektowania filtrów IIR

SciPy.org | Docs | SciPy v1.6.1 Reference Guide | Signal processing (scipy.signal)

scipy.signal.iirdesign

scipy.signal.**iirdesign**(*wp, ws, gpass, gstop, analog=False, ftype='ellip', output='ba', fs=None*)

Complete IIR digital and analog filter design.

SciPy.org | Docs | SciPy v1.6.1 Reference Guide | Signal processing (scipy.signal)

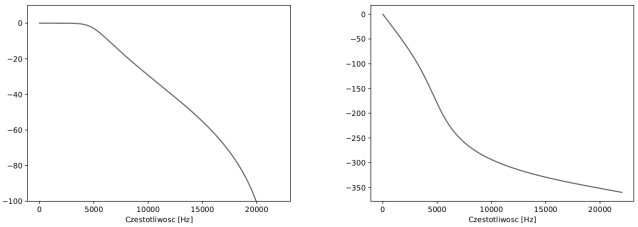
scipy.signal.iirfilter

scipy.signal.**iirfilter**(*N, Wn, rp=None, rs=None, btype='band', analog=False, ftype='butter', output='ba', fs=None*)

IIR digital and analog filter design given order and critical points.

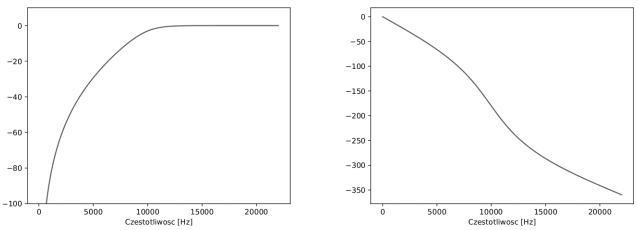
Filtr IIR dolno-przepustowy

```
fs = 44100   # czestotliwosc probkowania
n  = 4       # rzad filtra
fc = 5000    # czestotliwosc graniczna

b, a = signal.iirfilter(n, fc / (fs / 2),
                        btype='lowpass')

w, h = signal.freqz(b, a)
```



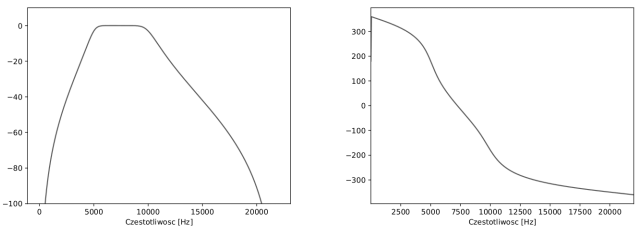
Filtr IIR górno-przepustowy

```
fs = 44100   # czestotliwosc probkowania
n  = 4       # rzad filtra
fc = 10000   # czestotliwosc graniczna

b, a = signal.iirfilter(n, fc / (fs / 2),
                        btype='highpass')

w, h = signal.freqz(b, a)
```



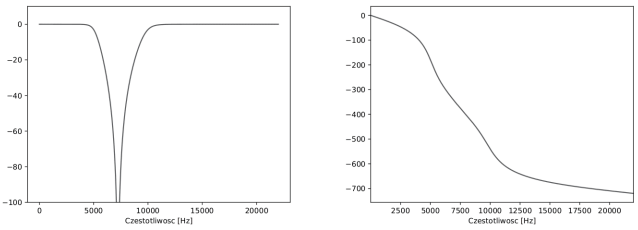
Filtr IIR pasmowo-przepustowy

```
fs  = 44100  # czestotliwosc probkowania
n   = 4      # rzad filtra
fc1 = 5000   # czestotliwosc graniczna
fc2 = 10000  # czestotliwosc graniczna

b, a = signal.iirfilter(n, [fc1 / (fs / 2), fc2 /
    (fs / 2)],
                        btype='bandpass')

w, h = signal.freqz(b, a)
```



Filtr IIR pasmowo-zaporowy

```
fs   = 44100   # czestotliwosc probkowania
n    = 4       # rzad filtra
fc1 = 5000    # czestotliwosc graniczna
fc2 = 10000   # czestotliwosc graniczna

b, a = signal.iirfilter(n, [fc1 / (fs / 2), fc2 /
    (fs / 2)], btype='bandstop')

w, h = signal.freqz(b, a)
```



Projektowanie filtrów w praktyce

Metody pomocnicze:

SciPy.org Docs SciPy v1.4.1 Reference Guide Signal processing (scipy.signal)

scipy.signal.freqz

scipy.signal.**freqz**(*b, a=1, worN=512, whole=False, plot=None, fs=6.283185307179586*)

Compute the frequency response of a digital filter.

SciPy.org Docs SciPy v1.4.1 Reference Guide Signal processing (scipy.signal)

scipy.signal.lfilter

scipy.signal.**lfilter**(*b, a, x, axis=-1, zi=None*)

Filter data along one-dimension with an IIR or FIR filter.

SciPy.org Docs SciPy v1.4.1 Reference Guide Signal processing (scipy.signal)

scipy.signal.fftconvolve

scipy.signal.**fftconvolve**(*in1, in2, mode='full', axes=None*)

Convolve two N-dimensional arrays using FFT.

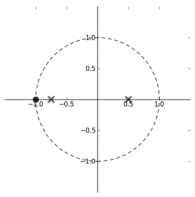
Pełna lista dostępnych metod:

https://docs.scipy.org/doc/scipy/reference/signal.html

Wizualizacja zer i biegunów

$$H(z) = \frac{z^{-1} + z^{-2}}{1 + \frac{1}{4}z^{-1} - \frac{3}{8}z^{-2}}$$

```
>>> import numpy as np
# If the code is in a file called plot_zplane.py
>>> from plot_zplane import zplane
>>> b = np.array([0, 1, 1])
>>> a = np.array([1, 1/4., -3/8.])
>>> zplane(b,a)
```



https://www.dsprelated.com/showcode/244.php