Normalizacja wzmocnienia kaskadowego filtru IIR

Projekt w języku python - w Matlabie robi się to podobnie.

In [1]:
```
import numpy as np
import scipy.signal as sig
import matplotlib.pyplot as plt
np.set_printoptions(suppress=True)
```

Filtr dolnoprzepustowy IIR

- rząd 8
- częstotliwość graniczna 1 kHz
- charakterystyka Butterwortha

In [2]:
```
# Projekt filtru
N = 8
fc = 1000 / (48000 / 2)

char = 'lowpass'
type = 'butter'
b, a = sig.iirfilter(N, fc, btype=char, ftype=type)
sos = sig.iirfilter(N, fc, btype=char, ftype=type, output='sos')
z, p, k = sig.iirfilter(N, fc, btype=char, ftype=type, output='zpk')
sos_ = sos.copy()

print('k = ', k)
print(sos)
```

```
k =  2.43444901944e-10
[[ 0.          0.          0.          1.         -1.75785265  0.77302109]
 [ 1.          2.          1.          1.         -1.78875835  0.80419348]
 [ 1.          2.          1.          1.         -1.84881984  0.86477323]
 [ 1.          2.          1.          1.         -1.93365048  0.95033587]]
```

In [3]:
```
# Zaczynamy od pierwszej sekcji SOS
sos1 = sos_[:1, :]
# Obliczamy maksymalny poziom charakterystyki amplitudowej
b1, a1 = sig.sos2tf(sos1)
_, h1 = sig.freqz(b1, a1)
k1 = np.max(np.abs(h1))
print('k1 = ', k1)

# dzielimy współczynniki b sekcji 1 przez k1
sos_[0, :3] /= k1

print(sos_)
```

```
k1 =  6.41977375956e-08
[[ 0.00379211  0.00758422  0.00379211  1.         -1.75785265  0.77302109]
 [ 1.          2.          1.          1.         -1.78875835  0.80419348]
 [ 1.          2.          1.          1.         -1.84881984  0.86477323]
 [ 1.          2.          1.          1.         -1.93365048  0.95033587]]
```

In [4]:
```
# Teraz bierzemy pierwsze dwie sekcje SOS
sos2 = sos_[:2, :]
# Obliczamy maksymalny poziom charakterystyki amplitudowej
b2, a2 = sig.sos2tf(sos2)
_, h2 = sig.freqz(b2, a2)
k2 = np.max(np.abs(h2))
print('k2 = ', k2)

# dzielimy współczynniki b sekcji 2 przez k2
sos_[1, :3] /= k2

print(sos_)
```

```
k2 =  259.149175922
[[ 0.00379211  0.00758422  0.00379211  1.         -1.75785265  0.77302109]
 [ 0.00385878  0.00771756  0.00385878  1.         -1.78875835  0.80419348]
 [ 1.          2.          1.          1.         -1.84881984  0.86477323]
 [ 1.          2.          1.          1.         -1.93365048  0.95033587]]
```

In [5]:
```
# Następnie pierwsze trzy sekcje SOS
sos3 = sos_[:3, :]
# Obliczamy maksymalny poziom charakterystyki amplitudowej
b3, a3 = sig.sos2tf(sos3)
_, h3 = sig.freqz(b3, a3)
k3 = np.max(np.abs(h3))
print('k3 = ', k3)

# dzielimy współczynniki b sekcji 3 przez k3
sos_[2, :3] /= k3

print(sos_)
```

```
k3 =  250.730353735
[[ 0.00379211  0.00758422  0.00379211  1.         -1.75785265  0.77302109]
 [ 0.00385878  0.00771756  0.00385878  1.         -1.78875835  0.80419348]
 [ 0.00398835  0.0079767   0.00398835  1.         -1.84881984  0.86477323]
 [ 1.          2.          1.          1.         -1.93365048  0.95033587]]
```

In [6]:
```
# I została jeszcze ostatnia sekcja
sos4 = sos_[:4, :]
# Obliczamy maksymalny poziom charakterystyki amplitudowej
b4, a4 = sig.sos2tf(sos4)
_, h4 = sig.freqz(b4, a4)
k4 = np.max(np.abs(h4))
print('k4 = ', k4)

# dzielimy współczynniki b sekcji 4 przez k4
sos_[3, :3] /= k4

print(sos_)
```

```
k4 =  239.730636283
[[ 0.00379211  0.00758422  0.00379211  1.         -1.75785265  0.77302109]
 [ 0.00385878  0.00771756  0.00385878  1.         -1.78875835  0.80419348]
 [ 0.00398835  0.0079767   0.00398835  1.         -1.84881984  0.86477323]
 [ 0.00417135  0.0083427   0.00417135  1.         -1.93365048  0.95033587]]
```

In [7]:
```
# Sprawdźmy maksymalne wzmocnienie całego filtru
b, a = sig.sos2tf(sos_)
_, h = sig.freqz(b, a)
k = np.max(np.abs(h))
print('k = ', k)
```

```
k =  1.0
```

In [8]:
```
# Kwantyzacja współczynników do formatu Q1.14
qsos = np.around(16384 * sos_).astype(np.int16)
print(qsos)
```

```
[[    62    124     62  16384 -28801  12665]
 [    63    126     63  16384 -29307  13176]
 [    65    131     65  16384 -30291  14168]
 [    68    137     68  16384 -31681  15570]]
```

In [9]:
```
# Sprawdzenie maksymalnego wzmocnienia po każdej sekcji (po kwantyzacji)
for i in range(1, 5):
    bi, ai = sig.sos2tf(qsos[:i, :])
    _, hi = sig.freqz(bi, ai)
    ki = np.max(np.abs(hi))
    print('k{} = {}'.format(i, ki))
```

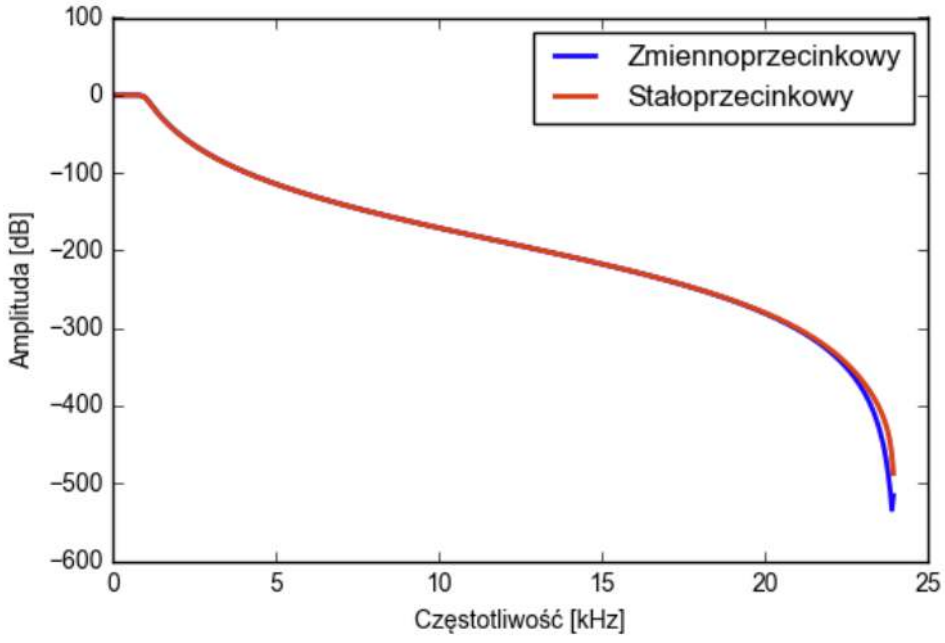
```
k1 = 1.0
k2 = 0.9960474308300394
k3 = 0.9960474308300394
k4 = 0.9960473720122682
```

In [10]:
```
# Sprawdzenie czy filtr jest stabilny - moduł biegunów nie może przekraczać 1
z, p, k = sig.sos2zpk(qsos)
print(np.abs(p))
```

```
[ 0.87921002  0.87921002  0.89677155  0.89677155  0.92991725  0.92991725
  0.97484223  0.97484223]
```

In [11]:
```
# Wykreślmy jeszcze charakterystykę obu filtrów
w, hn = sig.freqz(*sig.sos2tf(sos))
w = w * 48000 / (2 * np.pi) / 1000
hn = 20 * np.log10(np.abs(hn))
_, hq = sig.freqz(*sig.sos2tf(qsos))
hq = 20 * np.log10(np.abs(hq))

fig, ax = plt.subplots()
ax.plot(w, hn, color='b', linewidth=2, label='Zmiennoprzecinkowy')
ax.plot(w, hq, color='r', linewidth=2, label='Stałoprzecinkowy')
ax.set_xlabel('Częstotliwość [kHz]')
ax.set_ylabel('Amplituda [dB]')
ax.legend()
plt.show()
```



I chyba wszystko jest w porządku.