OMNIS 2

Otwartość. Modernizacja. Nowoczesność. Integracja. Społeczność.

Podstawy Elektroniki
Ćwiczenia komputerowe

Filtry elektryczne

Autor: Dariusz Tefelski

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa

www.pw.edu.pl

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa
www.pw.edu.pl

1 Cel ćwiczenia

Celem niniejszego ćwiczenia jest poznanie aktywnych filtrów elektrycznych, ich zastosowań, parametrów i sposobu projektowania.

2 Wstęp teoretyczny

2.1 Wprowadzenie do filtrów aktywnych

Filtry elektryczne to układy, których zadaniem jest przepuszczanie sygnałów o określonych częstotliwościach i tłumienie pozostałych. W odróżnieniu od filtrów pasywnych zbudowanych wyłącznie z elementów RLC, filtry aktywne wykorzystują elementy wzmacniające, takie jak wzmacniacze operacyjne (np. typ LM741). Pozwala to na:

- eliminację gabarytowych cewek (szczególnie kłopotliwych i drogich przy bardzo niskich częstotliwościach),
- uzyskanie dodatkowego wzmocnienia wielkości sygnału w paśmie przepustowym,
- pełne odseparowanie impedancyjne poszczególnych stopni układu, dzięki czemu poszczególne bloki filtra nie wpływają wzajemnie na swoje częstotliwości graniczne.

W układach wykorzystuje się model standardowego wzmacniacza operacyjnego, np. LM741, który w podstawowych obliczeniach można i warto traktować jako wzmacniacz idealny o bardzo dużej rezystancji wejściowej i otwartym wzmocnieniu.

2.2 Topologia Sallen-Key dla filtrów 2. rzędu

Architektura Sallen-Key to najbardziej znana i praktyczna realizacja sekcji aktywnego filtru elektrycznego. Zawiera jeden wzmacniacz operacyjny pracujący z dodatnim sprzężeniem wewnątrz pętli o strukturze RC. Dla układu dolnoprzepustowego (ang. *LPF – Low Pass Filter*) funkcja transmitancji wygląda jak postać układu rezonansowego:

$$H(s) = \frac{K \cdot \omega_0^2}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2} \tag{1}$$

gdzie K to wzmocnienie początkowe (często równe 1 w układzie wtórnika napięciowego), a ω_0 i dobroć Q zależą od dobranych wartości rezystorów i kondensatorów.

2.3 Klasyczne typy filtrów

Charakterystykę układu narzuca przyjęta aproksymacja idealnego kształtu tzw. prostokąta. Istnieją trzy fundamentalne podejścia omawiane w ramach ćwiczeń, stosowane w symulacjach rzędów 3 i 4:

1. **Butterwortha** – zapewnia płaskie wejście w paśmie przenoszenia (bez falowania amplitudy). Cechuje je stromość asyptotyczna na poziomie $-20n$ dB/dek gdzie n to rząd filtru.

2. **Czebyszewa (Typu I)** – gwarantuje szybsze opadanie charakterystyki kosztem występowania pulsacji (falowania) amplitudy przed odcięciem. Skutkuje to dłuższym tzw. *dzwonieniem* po impulsie skokowym układu.
3. **Bessela** – skupia się nie na amplitudzie, a na stałym opóźnieniu grupowym dla całego spektrum przepustowego (liniowa modyfikacja fazy). Układ ten wyróżnia brak niepożądanych zniekształceń przebiegu prostokątnego w dziedzinie czasu – idealny do transmisji impulsowej. Opadanie transmitancji staje się jednak bardzo łagodne.

2.4 Filtry wyższych rzędów (Sallen-Key rzędu 3 i 4)

Realizacja filtrów 3. i 4. rzędu odbywa się poprzez kaskadowe włączenie osobnych układów. Obliczenia poszczególnych bloków wymagają wyznaczenia osobnych wartości elementów, by sprzężony rezultat odpowiadał wielomianom teoretycznym.

- **Rząd 3**: Jest to konfiguracja sekcji pasywnej filtra dopasowującego RC (1. rzędu) wpiętej przed kaskadą drugiego rzędu Sallen-Key o charakterystyce wielomianu odpowiedniego typu z pominięciem zniekształceń dopasowania oporności, np. poprzez użycie dodatkowego bufora - wzmacniacza np. LM746 w konfiguracji wtórnika.
- **Rząd 4**: Składa się z dwóch połączonych niezależnie sekcji drugiego rzędu Sallen-Key'a zawierających 2 wzmacniacze operacyjne. Pomiędzy modułami uważa się obciążenie za zerowe właśnie ze względu na separujące układy operacyjne.

2.5 Inne topologie

Aby uzyskać filtry innych typów, np. filtr eliptyczny, konfiguracja Sallen-Key'a jest niewystarczająca, gdyż jest to konfiguracja obejmująca tylko bieguny. W tej konfiguracji transmitancja nie ma "zer". Filtr eliptyczny wymaga właśnie obecności także zer. Potrzeba wtedy konfiguracji MFB (Multiple Feedback). Można zastosować np. topologię Baintera, State-Variable (KHN) lub Tow-Thomas albo ewentualnie zmodyfikowany Sallen-Key (Twin-T lub Bridged-T), ale w tym przypadku układ jest bardzo czuły na tolerancję elementów.

3 Środowisko obliczeniowe i moduły

Obliczenia analityczne i projektowe wykonujemy w terminalu `Jupyter Lab` opartym o język Python wspomagając się następującymi bibliotekami:

- **Scipy** (moduł `scipy.signal`) oraz **Numpy** – potężne funkcje budujące transformatę za nas, np. `signal.bode`, `signal.step`, wbudowane funkcje tworzące funkcje przejścia dla każdego wielomianów (funkcje `butter`, `cheby1`, `bessel`).
- **Sympy** – środowisko algebry komputerowej. Wykorzystane do symbolicznego rozwiązania matematycznego układu oczek z układem Sallen-Key i wyciągnięcia wielomianów transmitancji elementarnej.
- **Lcapy** – znakomita biblioteka rozszerzająca funkcjonalność standardowych bibliotek o zdolność rysowania układów operacyjnych i definicję modeli obwodów z poziomu kodu (tzw. zautomatyzowane wyliczanie równań Kirchhoffa dla opamp-ów).

4 Część praktyczna – Zadania

Uruchom środowisko Jupyter-Lab, utwórz nowy `Notebook` i wykonaj poszczególne podpunkty korzystając z opisanych funkcji analitycznych po stronie Pythona.

Zadanie 1. Symulacja charakterystyk Bodego rzędu 3 i 4

Wykorzystując funkcję `scipy.signal.butter`, wygeneruj ciągły (parametr `analog=True`) model transmitancji Butterwortha. Zaprojektuj też za pomocą funkcji `cheby1` i `bessel` filtr Czebyszewa oraz Bessela IV rzędu dla częstotliwości odcięcia wynoszącej 1000 Hz. Następnie:

1. Oblicz widmo i fazy funkcji przejścia (komenda `scipy.signal.bode`).
2. Przedstaw na wspólnym wykresie logarytmicznym w module `matplotlib` widmo wielkości amplitudowej w decybelach dla filtru Butterwortha 3. rzędu na tle Butterwortha 4. rzędu, jak i Czebyszewa oraz Bessela.
3. Zaobserwuj i opisz asyptotyczne różnice wielkości za odcięciem (pasmo zaporowe) a zakresem pasma przenoszenia.

Listing 4.1: Symulacja charakterystyk filtrów Bodego 3 i 4 rzędu oraz Czebyszewa I-go rodzaju 4-tego rzędu oraz Bessela 4-tego rzędu.

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

# Przygotowanie osi modelowej w zakresie 10 Hz do 10 kHz
freq_hz = np.logspace(1, 5, 1000)
freq_rad = 2 * np.pi * freq_hz
fc = 1000  # czestotliwosc graniczna
wc = 2 * np.pi * fc

# Generowanie definicji ukladow ciaglych
b3, a3 = signal.butter(3, wc, btype="low", analog=True)
b4, a4 = signal.butter(4, wc, btype="low", analog=True)
# Czebyszew: Wn to poczatek spadku (poczatek transition band).
# Aby -3dB bylo przy wc, musimy przesunac Wn.
rp = 3.0  # ripple 3dB - czyli zafalowanie 3dB
wp = wc / np.cosh(1 / 4 * np.acosh(1 / np.sqrt(10 ** (rp / 10) - 1)))
ch_b, ch_a = signal.cheby1(4, rp, wp, btype="low", analog=True)
# Bessel: norm='mag' sprawia, ze wc to czestotliwosc dla -3dB.
be_b, be_a = signal.bessel(4, wc, btype="low", analog=True, norm="mag")

# Uzyskanie wektorow amplitud
_, mag_b3, _ = signal.bode((b3, a3), freq_rad)
_, mag_b4, _ = signal.bode((b4, a4), freq_rad)
_, mag_ch, _ = signal.bode((ch_b, ch_a), freq_rad)
```

Listing 4.1: Symulacja charakterystyk filtrów Bodego 3 i 4 rzędu oraz Czebyszewa I-go rodzaju 4-tego rzędu oraz Bessela 4-tego rzędu. (c.d.)

```
_, mag_be, _ = signal.bode((be_b, be_a), freq_rad)

# Rysowanie wykresów pół-logarytmicznych
plt.figure(figsize=(10, 6))
plt.semilogx(freq_hz, mag_b3, "--", label="Butterworth (rzad 3)")
plt.semilogx(freq_hz, mag_b4, label="Butterworth (rzad 4)")
plt.semilogx(freq_hz, mag_ch, label="Czebyszew typu I (rzad 4,
↪  ripple=3dB)")
plt.semilogx(freq_hz, mag_be, label="Bessel (rzad 4)")

plt.axvline(fc, color="gray", linestyle=":", label="Fc=1kHz")
plt.title("Porownanie charakterystyk Bodego: Filtry Aktywne LP")
plt.ylabel("Tlumienie [dB]")
plt.xlabel("Częstotliwość [Hz]")
plt.ylim(-60, 5)
plt.legend()
plt.grid(True, which="both", ls="--")
plt.show()
```

Listing 4.2 zawiera dodatkowy kod przedstawiający opóźnienie grupowe w zależności od filtru.

Listing 4.2: Symulacja charakterystyk filtrów Bodego 3 i 4 rzędu oraz Czebyszewa I-go rodzaju 4-tego rzędu oraz Bessela 4-tego rzędu.

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

# Przygotowanie osi modelowej w zakresie 10 Hz do 10 kHz
freq_hz = np.logspace(1, 5, 1000)
freq_rad = 2 * np.pi * freq_hz
fc = 1000  # czestotliwosc graniczna
wc = 2 * np.pi * fc

# Generowanie definicji ukladow ciaglych
b3, a3 = signal.butter(3, wc, btype="low", analog=True)
b4, a4 = signal.butter(4, wc, btype="low", analog=True)
# Czebyszew: Wn to poczatek spadku (poczatek transition band).
# Aby -3dB bylo przy wc, musimy przesunac Wn.
rp = 1.0
wp = wc / np.cosh(1 / 4 * np.acosh(1 / np.sqrt(10 ** (rp / 10) - 1)))
ch_b, ch_a = signal.cheby1(4, rp, wp, btype="low", analog=True)
# Bessel: norm='mag' sprawia, ze wc to czestotliwosc dla -3dB.
be_b, be_a = signal.bessel(4, wc, btype="low", analog=True, norm="mag")
```

Listing 4.2: Symulacja charakterystyk filtrów Bodego 3 i 4 rzędu oraz Czebyszewa I-go rodzaju 4-tego rzędu oraz Bessela 4-tego rzędu. (c.d.)

```
# Funkcja pomocnicza do obliczania charakterystyki amplitudowej i opoznienia grupowego
def get_filter_stats(b, a, w_rad):
    w, h = signal.freqs(b, a, w_rad)
    mag = 20 * np.log10(np.abs(h))
    phase = np.unwrap(np.angle(h))
    # Opoznienie grupowe to ujemna pochodna fazy po czestotliwosci (tau = -d_phase / d_omega)
    gd = -np.diff(phase) / np.diff(w_rad)
    return mag, gd


# Uzyskanie danych
mag_b3, gd_b3 = get_filter_stats(b3, a3, freq_rad)
mag_b4, gd_b4 = get_filter_stats(b4, a4, freq_rad)
mag_ch, gd_ch = get_filter_stats(ch_b, ch_a, freq_rad)
mag_be, gd_be = get_filter_stats(be_b, be_a, freq_rad)

# Rysowanie wykresów pół-logarytmicznych
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10), sharex=True)

# Wykres Amplitudy
ax1.semilogx(freq_hz, mag_b3, "--", label="Butterworth (rzad 3)")
ax1.semilogx(freq_hz, mag_b4, label="Butterworth (rzad 4)")
ax1.semilogx(freq_hz, mag_ch, label="Czebyszew (rzad 4, ripple=1dB)")
ax1.semilogx(freq_hz, mag_be, label="Bessel (rzad 4, norm=mag)")
ax1.axvline(fc, color="gray", linestyle=":", label="Fc=1kHz")
ax1.set_title("Charakterystyka Amplitudowa (Punkt -3dB @ 1kHz)")
ax1.set_ylabel("Tłumienie [dB]")
ax1.set_ylim(-60, 5)
ax1.legend()
ax1.grid(True, which="both", ls="--")

# Wykres Opoznienia Grupowego
# Uzywamy freq_hz[:-1] bo diff skraca wektor o 1
ax2.semilogx(freq_hz[:-1], gd_b3 * 1000, "--", label="Butterworth (rzad 3)")
ax2.semilogx(freq_hz[:-1], gd_b4 * 1000, label="Butterworth (rzad 4)")
ax2.semilogx(freq_hz[:-1], gd_ch * 1000, label="Czebyszew (rzad 4)")
ax2.semilogx(freq_hz[:-1], gd_be * 1000, label="Bessel (rzad 4)")
ax2.axvline(fc, color="gray", linestyle=":")
ax2.set_title("Opóźnienie Grupowe (Group Delay)")
ax2.set_ylabel("Opóźnienie [ms]")
ax2.set_xlabel("Częstotliwość [Hz]")
ax2.legend()
```

Listing 4.2: Symulacja charakterystyk filtrów Bodego 3 i 4 rzędu oraz Czebyszewa I-go rodzaju 4-tego rzędu oraz Bessela 4-tego rzędu. (c.d.)

```
ax2.grid(True, which="both", ls="--")

plt.tight_layout()
plt.show()
```

Zadanie 2. Uproszczone projektowanie filtrów

Zakładając wzmacniacz operacyjny w konfiguracji struktury Sallen-Key 2. rzędu:

1. Skonstruuj matematyczny wzór transmitancji w dziedzinie `s` używając `Sympy.symbol()`.
2. Zbuduj wyrażenie algebraczjne dla wariantu układu o jednakowych wartościach rezystancji oporników $R = R_1 = R_2$ i jednakowych wartościach pojemności kondensatorów $C = C_1 = C_2$, dążąc do zredukowanego wielomianu.
3. Wylicz częstotliwość graniczną, korzystając z funkcji `subs` postawiającej wartości modelowych elementów $R = 10k\Omega$ i $C = 10nF$

Listing 4.3: Projekt filtru Sallen-Key'a o tych samych wartościach rezystorów i kondensatorów.

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt

s = sp.Symbol("s")
R = sp.Symbol("R", real=True, positive=True)
C = sp.Symbol("C", real=True, positive=True)

Rv = 10.0e3  # Rezystory 10 kOhm
Cv = 10.0e-9  # Kondensatory 10 nF

# Dla ukladu r1=r2=R=Rv i c1=c2=C=Cv
w0 = 1 / (R * C)
Q = 0.5  # Wspolczynnik dobroci dla wzmacniacza o K=1


Hs = w0**2 / (s**2 + (w0 / Q) * s + w0**2)
print("Funkcja macierzysta symboliczna dla narzuconych uproszczen
↪  Sallen-Key:")
sp.pprint(Hs.simplify())

# Parametry urzadzenia z zadania: R = 10k, C = 10n
wartosci = {R: Rv, C: Cv}
Hs_param = Hs.subs(wartosci)
```

Listing 4.3: Projekt filtru Sallen-Key'a o tych samych wartościach rezystorów i kondensatorów. (c.d.)

```
print("\nWielomian LPF po podstawieniu wartosci rezystancji i
↪  pojemności:")
sp.pprint(Hs_param)

# --- WYKREŚLANIE CHARAKTERYSTYKI ---
# Konwersja symbolicznej transmitancji na funkcję numeryczną
f_hz = np.logspace(1, 5, 10000)
w_rad = 2 * np.pi * f_hz
h_func = sp.lambdify(s, Hs_param, "numpy")
h_vals = h_func(1j * w_rad)

mag = 20 * np.log10(np.abs(h_vals))

plt.figure(figsize=(10, 6))
plt.semilogx(f_hz, mag)
plt.title(
    "Charakterystyka amplitudowa filtru Sallen-Key (R=10k, C=10n, Q=0.5)"
)
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Amplituda [dB]")
plt.grid(True, which="both", ls="--")

# Wyznaczenie i zaznaczenie punktu -3dB

idx = np.argmin(np.abs(mag + 3))

print(f"Częstotliwość graniczna filtru: {f_hz[idx]:.1f} Hz")
print(f"Amplituda dla f={f_hz[idx]:.1f} Hz k={mag[idx]:.1f} dB")

plt.axvline(
    f_hz[idx], color="r", linestyle="--", label=f"fc ~ {f_hz[idx]:.1f} Hz"
)
plt.legend()

plt.tight_layout()
# plt.savefig('charakterystyka_z2.png')
# print(f"\nWykres został zapisany do pliku: charakterystyka_z2.png")
plt.show()
```

Zadanie 3. Analiza odpowiedzi filtrów na pobudzenie skokiem jednostkowym

Skorzystaj z opracowanych w zadaniu nr 1 filtrów 4. rzędu (Butterworth, Czebyszew I z dopuszczalnym falowaniem na poziomie 3 dB, Bessel) i zasymuluj dla każdego z nich odpowiedź czasową na fukcję skoku jednostkowego(*Heaviside step response*). Wykorzystaj

wywołanie `step` z modułu scipy. Który system najstabilniej reaguje na skok jednostkowy? Przeanalizuj zjawisko rezonansowego oscylowania.

Listing 4.4: Projekt filtru Sallen-Key'a o tych samych wartościach rezystorów i kondensatorów.

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

fc = 1000  # czestotliwosc zalamania granicznego
wc = 2 * np.pi * fc

# Generowanie definicji ukladow ciaglych
b3, a3 = signal.butter(3, wc, btype="low", analog=True)
b4, a4 = signal.butter(4, wc, btype="low", analog=True)
# Czebyszew: Wn to poczatek spadku (poczatek transition band).
# Aby -3dB bylo przy wc, musimy przesunac Wn.
rp = 3.0  # ripple 3dB - czyli zafalowanie 3dB
wp = wc / np.cosh(1 / 4 * np.acosh(1 / np.sqrt(10 ** (rp / 10) - 1)))
ch_b, ch_a = signal.cheby1(4, rp, wp, btype="low", analog=True)
# Bessel: norm='mag' sprawia, ze wc to czestotliwosc dla -3dB.
be_b, be_a = signal.bessel(4, wc, btype="low", analog=True, norm="mag")

# Odpowiedź na skok jednostkowy z scipy.
time_vector = np.linspace(0, 0.005, 500)  # Próbkuj w dziedzinie czasu do
↪  5ms
sys_b4 = signal.TransferFunction(b4, a4)
sys_ch = signal.TransferFunction(ch_b, ch_a)
sys_be = signal.TransferFunction(be_b, be_a)

# Wyznaczenie odpowiedzi na skok jednostkowy f. Heaviside
t, out_b4 = signal.step(sys_b4, T=time_vector)
_, out_ch = signal.step(sys_ch, T=time_vector)
_, out_be = signal.step(sys_be, T=time_vector)

plt.figure(figsize=(10, 5))
plt.plot(t * 1000, out_b4, label="Butterworth (rzad 4)")
plt.plot(t * 1000, out_ch, label="Czebyszew (rzad 4), ripple=3dB")
plt.plot(t * 1000, out_be, label="Bessel (rzad 4)")

plt.axhline(1.0, color="k", linestyle="--", linewidth=0.8)
plt.title("Odpowiedź czasowa filtrów pobudzonych skokiem jednostkowym")
plt.xlabel("Czas [ms]")
plt.ylabel("Oczekiwana wielkość napięcia wyjsciowego [V]")
plt.legend()
plt.grid()
plt.show()
```

5 Projektowanie filtru

Zadanie

Na podstawie numeru zadania z laboratorium do ćwiczenia „Filtry elektryczne" odczytaj parametry filtra elektrycznego, który należy zaprojektować i wykonaj projekt - wyznacz potrzebne rezystory, wykonaj symulację i wykreśl charakterystykę amplitudową zaprojektowanego filtru.

W przypadku filtrów rzędu 3-ciego należy zbudować sekcję filtru 2-giego rzędu w konfiguracji Sallen-Key'a oraz sekcję filtru pasywnego 1-go rzędu R-C z dołączonym na jej wyjściu wzmacniaczem w konfiguracji wtórnika (o wzmocnieniu 1), celem separacji. Sekcja 2-giego rzędu ma transmitancję typu:

$$K(S) = \frac{k_0}{S^2 \cdot b + S \cdot a + 1} \tag{2}$$

Sekcja 1-go rzędu ma transmitancję typu:

$$K(S) = \frac{1}{S \cdot a + 1} \tag{3}$$

W przypadku filtra 4-tego rzędu należy przygotować 2 sekcje 2-giego rzędu.

Do danego typu filtra i danego rzędu potrzebne jest wyznaczenie współczynników, które można pobrać jako wartości stabelaryzowane - dostępne m.in. w instrukcji do ćwiczenia laboratoryjnego, albo można skorzystać ze skryptu 5.1, który wyznacza je na podstawie typu filtru i jego rzędu oraz zafalowania w przypadku filtra Czebyszewa.

Mając wyznaczone współczynniki, możemy przyrównać je do właściwych członów z wielomianu biegunów transmitancji. Wyznaczenie transmitancji filtru 2-go rzędu oraz wyznaczanie wartości rezystorów przedstawiono na listingu 5.2.

Listing 5.1: Wyznaczenie współczynników a i b poszczególnych sekcji filtra aktywnego w konfiguracji Sallen-Key'a w zależności od rzędu filtru.

```
import numpy as np
from math import factorial
from scipy.optimize import fsolve


def bessel_coeff_explicit(n):
    coeffs = []
    # k biegnie od n do 0, aby zachować kolejność [s^n, s^n-1, ..., s^0]
    for k in range(n, -1, -1):
        num = factorial(2 * n - k)
        den = (2 ** (n - k)) * factorial(k) * factorial(n - k)
        coeffs.append(num / den)  # //
    return coeffs
```

Listing 5.1: Wyznaczenie współczynników a i b poszczególnych sekcji filtra aktywnego w konfiguracji Sallen-Key'a w zależności od rzędu filtru. (c.d.)

```
def find_bessel_scaling(n):
    coeffs = bessel_coeff_explicit(n)

    poly = np.poly1d(coeffs)
    gain_at_zero = coeffs[-1]  # To jest a_0 (np. 105 dla n=4)

    # 2. Zdefiniuj funkcję błędu: |H(j*w)| - 1/sqrt(2) = 0
    def objective(w):
        s = 1j * w
        h_jw = gain_at_zero / poly(s)
        return np.abs(h_jw) - 1 / np.sqrt(2)

    # 3. Szukaj pierwiastka (częstotliwości w-3dB)
    # Startujemy od w=1.0 jako punktu wyjścia
    w_3dB = fsolve(objective, 1.0)[0]

    return w_3dB


def get_filter_coeffs_tested(ftype, n=4, ripple_db=1):
    # 1. Wyznaczenie bazowych biegunów (tylko n biegunów w lewej
    ↪  półpłaszczyźnie)
    if ftype == "butter":
        k = np.arange(n)
        angles = np.pi * (0.5 + (2 * k + 1) / (2 * n))
        poles = np.exp(1j * angles)
        f_3db = 1.0

    elif ftype == "cheby1":
        eps = np.sqrt(10 ** (ripple_db / 10) - 1)
        mu = np.arcsinh(1 / eps) / n
        k = np.arange(n)
        # Kąty pomocnicze
        theta = np.pi * (2 * k + 1) / (2 * n)
        poles = -np.sinh(mu) * np.sin(theta) + 1j * np.cosh(mu) *
        ↪  np.cos(theta)
        f_3db = np.cosh(np.acosh(1 / eps) / n)

    elif ftype == "bessel":
        poles = np.roots(bessel_coeff_explicit(n))
        f_3db = find_bessel_scaling(n)

    # 2. SELEKCJA UNIKALNYCH SEKCJI
```

Listing 5.1: Wyznaczenie współczynników a i b poszczególnych sekcji filtra aktywnego w konfiguracji Sallen-Key'a w zależności od rzędu filtru. (c.d.)

```
    # Musimy wziąć bieguny rzeczywiste (imag=0)
    # ORAZ tylko jeden biegun z każdej pary sprzężonej (imag > 0)

    selected_poles = []
    for p in poles:
        # Skalujemy od razu do -3dB
        p_scaled = p / f_3db

        # Jeśli biegun jest rzeczywisty (z tolerancją)
        if abs(p_scaled.imag) < 1e-8:
            selected_poles.append(("1st", p_scaled))
        # Jeśli biegun ma dodatnią część urojoną (bierze jedną z pary
        ↪  sprzężonej)
        elif p_scaled.imag > 1e-8:
            selected_poles.append(("2nd", p_scaled))

    # Sortowanie: najpierw 1. rzędu, potem 2. rzędu wg rosnącego 'a'
    selected_poles.sort(key=lambda x: (x[0] != "1st", -x[1].real))

    print(f"\n--- {ftype.upper()} n={n} ---")
    for i, (kind, p) in enumerate(selected_poles):
        if kind == "1st":
            a = -1.0 / p.real
            print(f"Sekcja {i + 1} (1. rzędu): a = {a:.4f}, b = 0.0000")
        else:
            B = np.abs(p) ** 2
            A = -2 * p.real
            a_book = A / B
            b_book = 1 / B
            print(
                f"Sekcja {i + 1} (2. rzędu): a = {a_book:.4f}, b =
                ↪  {b_book:.4f}"
            )


# TESTY:
get_filter_coeffs_tested(
    "butter", n=4
)  # Oczekiwane: 2. rzędu + 2. rzędu (RAZEM 2 sekcje)
get_filter_coeffs_tested(
    "cheby1", n=3, ripple_db=1
)  # Oczekiwane: 1. rzędu + 2. rzędu (RAZEM 2 sekcje)
get_filter_coeffs_tested(
    "bessel", n=4
)  # Oczekiwane: 2. rzędu + 2. rzędu (RAZEM 2 sekcje)
```

Listing 5.2: Wyznaczenie transmitancji filtru w konfiguracji Sallen-Key'a 2-giego rzędu. Wyznaczenie rezystorów sekcji 2-giego rzedu oraz 1-wszego rzędu.

```
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
import sympy as sp
from IPython.display import display, Math


def wyprowadz_transmitancje_sympy():
    """
    Demonstracja wykorzystania pakietu sympy do analitycznego
    wyprowadzenia transmitancji topologii Sallen-Key LPF.
    """
    print("--- WYPROWADZENIE TRANSMITANCJI ZA POMOCĄ SYMPY ---")
    s, R1, R2, C1, C2, K = sp.symbols("(S*wc) R1 R2 C1 C2 K",
    ↪  positive=True)
    # V2 - węzeł między R1 i R2, V3 - węzeł między R2 a wejściem
    ↪  nieodwracającym wzmacniacza (i także kondensatorem C1)
    V2, V3, Vout, Vin = sp.symbols("V2 Vy Vout Vin")
    # Równania z prawa prądowego Kirchhoffa w dziedzinie S
    eq1 = sp.Eq((V2 - Vin) / R1 + (V2 - Vout) * s * C2 + (V2 - V3) / R2,
    ↪  0)
    eq2 = sp.Eq((V3 - V2) / R2 + V3 * s * C1, 0)
    eq3 = sp.Eq(Vout, K * V3)

    rozwiazania = sp.solve([eq1, eq2, eq3], (V2, V3, Vout))

    Hs = sp.simplify(rozwiazania[Vout] / Vin)
    num, den = sp.fraction(Hs)
    Hs_standard = sp.expand(num) / sp.collect(sp.expand(den), s)

    sp.pprint(Hs_standard)
    print("\nPo zastosowaniu C1=C2=C")
    Hs_cap = Hs_standard.subs({C2: C, C1: C}).simplify()
    sp.pprint(Hs_cap)
    print("Jest to wyrażenie postaci:")
    display(Math(r"\frac{1}{S^2 \cdot b + S \cdot a + 1}"))
    print(
        "Wystarczy wykorzystać współczynniki a,b danej sekcji filtru celem
        ↪  wyznaczenia rezystancji R1, R2"
    )


# Wyznacz R sekcji 1-wszego rzędu filtru pasywnego
# Parametry wejściowe: a - odpowiadające danej sekcji, danego typu filtru
```

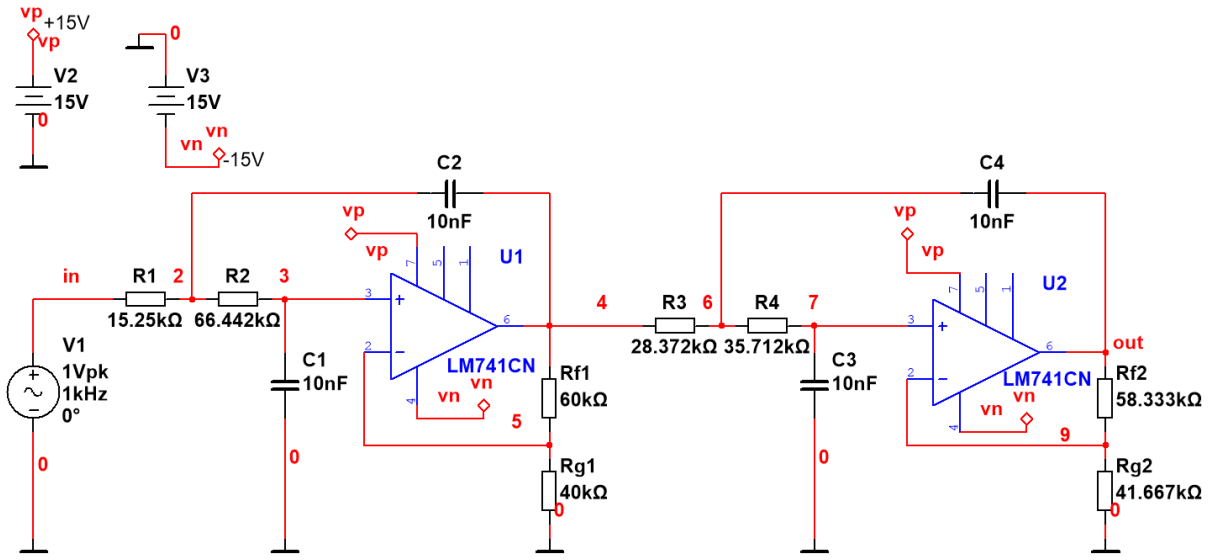
Listing 5.2: Wyznaczenie transmitancji filtru w konfiguracji Sallen-Key'a 2-giego rzędu. Wyznaczenie rezystorów sekcji 2-giego rzedu oraz 1-wszego rzędu. (c.d.)

```
# fc - częstotliwość graniczna filtru
# Cv - pojemność kondensatora, u nas: 10e-9 F
def wyznacz_r(a, fc, Cv):
    wc = 2 * np.pi * fc
    R = a / (wc * Cv)
    print("Sekcja 1-go rzędu:")
    print(rf"R = {R:.0f} Ohm")


# Wyznacz R1 i R2 sekcji 2-giego rzędu filtru w konfiguracji Sallen-Key'a
# Parametry wejściowe: a i b - odpowiadającej danej sekcji, danego typu
↪  filtru
# fc - częstotliwość graniczna filtru
# Ku - wzmocnienie danej sekcji filtru
# Cv - pojemność użytych kondensatorów, u nas: 10nF
def wyznacz_r1_r2(a, b, fc, Ku, Cv):
    wc = 2 * np.pi * fc
    wg, R1, R2, C, K = sp.symbols("wg R1 R2 C K", real=True,
    ↪  positive=True)
    eq1 = sp.Eq(wg * wg * C * C * R1 * R2, b).subs({wg: wc, C: Cv})
    eq2 = sp.Eq(wg * C * (-K * R1 + 2 * R1 + R2), a).subs(
        {wg: wc, C: Cv, K: Ku}
    )
    rozwiazania = sp.solve([eq1, eq2], (R1, R2))
    R1 = rozwiazania[0][0]
    R2 = rozwiazania[0][1]
    print("Sekcja 2-go rzędu:")
    print(f"R1={R1:.0f} Ohm\nR2={R2:.0f} Ohm")
    display(Math(rf"R_1={R1:.0f}\,\Omega  \qquad R_2={R2:.0f}\,\Omega"))

    if R1 > 100.0e3:
        print(
            "Uwaga!!! R1 jest większy niż 100 kOhm, tzn. jest poza\
zakresem cyfrowych potencjometrów używanych w laboratorium. Spróbuj
↪  zmienić wzmocnienie sekcji!"
        )
    if R2 > 100.0e3:
        print(
            "Uwaga!!! R1 jest większy niż 100 kOhm, tzn. jest poza\
zakresem cyfrowych potencjometrów używanych w laboratorium. Spróbuj
↪  zmienić wzmocnienie sekcji!"
        )
    A = [1, 1.2]
    war = 2 - a * a / b
    print(rf"warunek miękki na k {war * A[1]:.1f}")
```

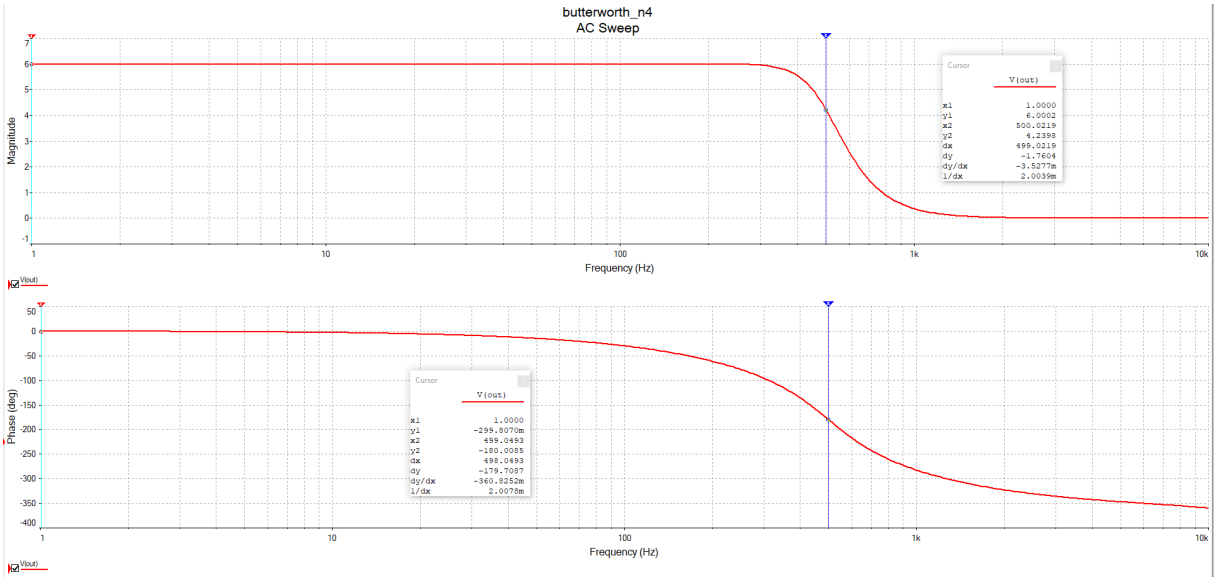
Listing 5.2: Wyznaczenie transmitancji filtru w konfiguracji Sallen-Key'a 2-giego rzędu. Wyznaczenie rezystorów sekcji 2-giego rzedu oraz 1-wszego rzędu. (c.d.)

```
    print(rf"warunek twardy na k {war * A[0]:.1f}")
    print(rf"ku={Ku}")
    if Ku < A[0] * war:
        print(
            "Warunek stabilności nie spełniony. Układ może wpaść w
            ↪ oscylacje."
        )
    elif Ku < A[1] * war:
        print(
            "Miękki warunek stabilności nie spełniony. Układ może wpaść w
            ↪ oscylacje."
        )
    else:
        print("Warunek stabilności spełniony.")


# --------------
wyprowadz_transmitancje_sympy()
# 1-wsza sekcja filtru Butterworth, rzędu 4, fc=500 Hz, wzmocnienie
↪ całkowite=6, wzmocnienie sekcji 1: 2.5
print("--- Sekcja 1 ---")
wyznacz_r1_r2(1.8478, 1.0, 500.0, 2.5, 10.0e-9)
# 2-ga sekcja filtru Butterworth, rzędu 4, fc=500 Hz, wzmocnienie
↪ całkowite=6, wzmocnienie sekcji 1: 2.4
print("--- Sekcja 2 ---")
wyznacz_r1_r2(0.7654, 1.0, 500.0, 2.4, 10.0e-9)

print("\nFiltr Butterworth 3-rzędu, fc=500Hz wzmocnienie = 4")
wyznacz_r(1, 500.0, 10.0e-9)
wyznacz_r1_r2(1, 1, 500.0, 4, 10.0e-9)
```

Na rys. 1 przedstawiony jest schemat układu filtru 4-tego rzędu w konfiguracji Sallen-Key'a z ustawionymi wartościami rezystorów dobranych dla filtra Butterworth'a o częstotliwości fc=500 Hz i wzmocnieniu całkowitym 6 V/V. Za pomocą symulacji `AC sweep` wyznaczono charakterystykę amplitudową filtru, co przedstawia rys. 2.



Rysunek 1: Multisim: Schemat do symulacji typu `AC sweep` filtru Butterworth'a 4-tego rzędu fc=500 Hz, całkowite wzmocnienie 6 V/V (wzmocnienie sekcji 1: 2.5 V/V, sekcji 2: 2.4 V/V), zbudowanego w konfiguracji Sallen-Key'a z wykorzystaniem 2 wzmacniaczy operacyjnych LM741.



Rysunek 2: Multisim: Wynik symulacji - charakterystyka amplitudowa (Bode'go) filtru Butterworth'a 4-tego rzędu, fc=500 Hz, całkowite wzmocnienie 6 V/V (wzmocnienie sekcji 1: 2.5 V/V, sekcji 2: 2.4 V/V), zbudowanego w konfiguracji Sallen-Key'a z wykorzystaniem 2 wzmacniaczy operacyjnych LM741.

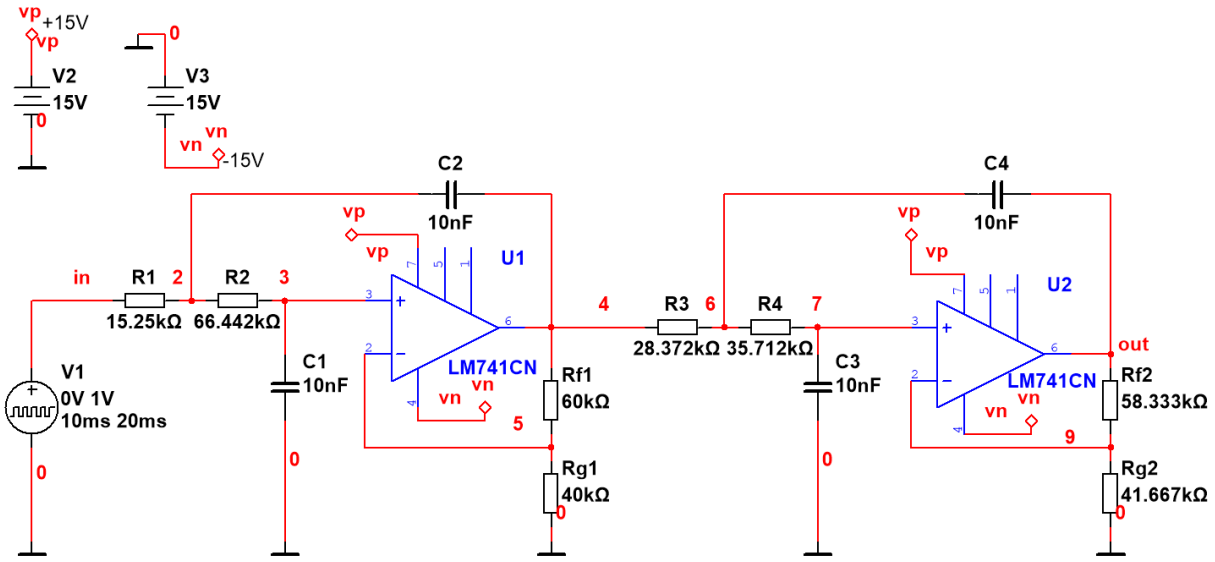
Uwaga!

Wykonanie symulacji typu `Transient` w Multisim może się nie powieść. Jeśli spróbujemy wykonać symulację typu `Interactive` pojawi sę błąd dotyczący braku zbieżności (Convergence). Można wtedy uruchomić asystenta, który pomoże znaleźć rozwiązanie. W tym konkretnym przypadku problem rozwiązała zmiana rezystancji do masy wszystkich węzłów analogowych. Domyślnie jest ona ustawiona na $10^{12}\Omega$. Jej zmiania na $10^{9}\Omega$ przywraca stabilność symulacji nie tylko w trybie `Interactive`, ale także w trybie `Transient`. Aby ustawić ten parametr należy w oknie ustawień symulacji, w zakładce `Analysis options` w `SPICE options` wybrać `Use custom settings` i kliknąć przycisk `Customize...`, znajdujący się obok. Dalej, w zakładce `Global` należy w wierszu `Shunt resistance from analog nodes to ground [RSHUNT]` zaznaczyć `ON` jeśli nie było wcześniej i zmienić wartość na `1e+09`. Potwierdzić naciśnięciem klawisza `OK`. Po tej zmianie symulacja powinna wykonywać się bez problemów.

Na rys. 3 przedstawiono schemat filtru przygotowany do symulacji odpowiedzi czasowej na skok jednostkowy. Na rys. 4 przedstawiono wynik symulacji odpowiedzi czasowej.

Warto wiedzieć

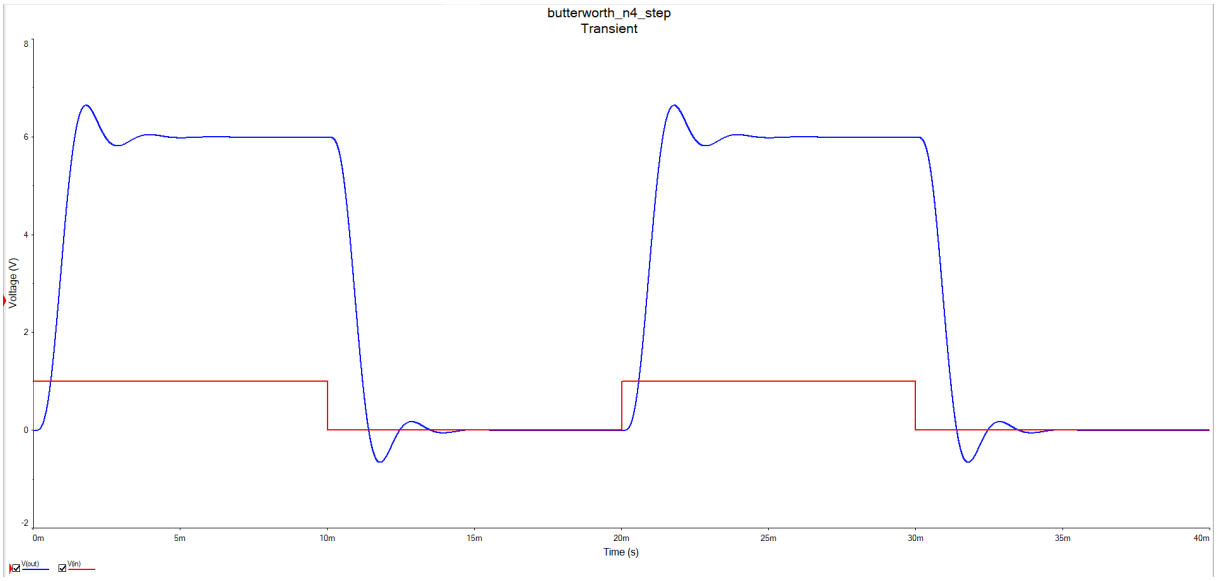
W symulacji typu `Transient` warto w `Analysis parameters` zaznaczyć opcję `Maximum time step (TMAX)` i ustawić wartość na `1e-06`. celem poprawienia szczegółowości wykresu.



Rysunek 3: Multisim: Schemat do symulacji typu `Transient` filtru Butterworth'a 4-tego rzędu fc=500 Hz, całkowite wzmocnienie 6 V/V (wzmocnienie sekcji 1: 2.5 V/V, sekcji 2: 2.4 V/V), zbudowanego w konfiguracji Sallen-Key'a z wykorzystaniem 2 wzmacniaczy operacyjnych LM741.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa
www.pw.edu.pl



Rysunek 4: Multisim: Wynik symulacji - odpowiedź na sygnał prostokątny (przybliżony skok jednostkowy) filtru Butterworth'a 4-tego rzędu, fc=500 Hz, całkowite wzmocnienie 6 V/V (wzmocnienie sekcji 1: 2.5 V/V, sekcji 2: 2.4 V/V), zbudowanego w konfiguracji Sallen-Key'a z wykorzystaniem 2 wzmacniaczy operacyjnych LM741.

Informacje dodatkowe

Więcej informacji dostępne jest na naszym Moodle: https://study.engined.eu/mod/lesson/view.php?id=778. W szczególności polecam zajrzeć do filmiku z symulacjami w LTSpice na stronie: https://study.engined.eu/mod/lesson/view.php?id=779. Można go także obejrzeć bezpośrednio na serwisie Youtube: https://www.youtube.com/watch?v=FJXcgPk4Fv0.

5.1 Wymagane oprogramowanie

Do wykonania ćwiczenia wymagany jest NI Multisim, Jupyter-lab i Python 3 z zainstalowanymi modułami:

- `Sympy` — do obliczeń symbolicznych i transformat,
- `Lcapy` — do modelowania obwodów elektrycznych,
- `Matplotlib` — do wizualizacji wyników,
- `Numpy` — do podstawowych struktur danych,
- `Scipy` — do obliczeń na sygnałach

6 Autorzy i historia opracowania

- dr inż. Dariusz Tefelski - wersja z 2026r.