OMNIS 2

Otwartość. Modernizacja. Nowoczesność. Integracja. Społeczność.

Podstawy Elektroniki
Ćwiczenia komputerowe

Stany nieustalone

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 zajęć jest zapoznanie się z metodami analizy stanów nieustalonych w obwodach liniowych RLC przy użyciu bibliotek `SymPy` oraz `Lcapy`. Studenci nauczą się wyznaczać transmitancję operatorową oraz wyznaczać odpowiedzi czasowe układu (impulsową i skokową) poprzez odwrotną transformatę Laplace'a.

2 Wprowadzenie

2.1 Wymagane oprogramowanie

Do wykonania ćwiczenia wymagany jest Python 3 z zainstalowanymi modułami:

- `Sympy` — do obliczeń symbolicznych i transformat,
- `Lcapy` — do modelowania obwodów elektrycznych,
- `matplotlib`, `numpy` — do wizualizacji wyników.

2.2 Stany nieustalone

Definicja

Stan nieustalony w obwodzie elektrycznym definiuje się jako proces przejściowy pomiędzy dwoma stanami ustalonymi. Pojawia się on na skutek zmian konfiguracji obwodu (np. załączenie lub wyłączenie źródła zasilania, zwarcie lub rozwarcie poszczególnych gałęzi obwodu) lub w wyniku zmian parametrów elementów obwodu zlokalizowanych w danej gałęzi.

Czas trwania stanu nieustalonego i jego charakter zależą bezpośrednio od stałych czasowych obwodu, które są zdeterminowane przez elementy inercyjne gromadzące energię: cewki (indukcyjność L) i kondensatory (pojemność C) oraz elementy rozpraszające energię, czyli rezystory (R).

W stanie nieustalonym prądy i napięcia ulegają dynamicznym (zależnym od czasu) zmianom, dążąc asymptotycznie do nowych, stałych lub okresowych, wartości ustalonych. Kluczową rolę w tej dynamice odgrywają **prawa komutacji**, które głoszą, że w zjawisku fizycznym energia nie może się zmienić ani ustać skokowo (w czasie nieskończenie krótkim):

- Prąd płynący przez cewkę nie może zmienić się skokowo. Warunek ciągłości prądu indukcyjności określa zapis:
$$i_L(0^-) = i_L(0^+) \tag{1}$$
- Napięcie na kondensatorze nie może zmienić się skokowo. Warunek ciągłości napięcia pojemności określa zapis:
$$u_C(0^-) = u_C(0^+) \tag{2}$$

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

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

gdzie chwila $t = 0^-$ następuje nieskończenie krótko przed komutacją, a $t = 0^+$ stanowi chwilę tuż po rozpoczęciu rekonfiguracji układu.

2.3 Rachunek operatorowy – Transformata Laplace'a

Klasyczna metoda analizy obwodu w stanie nieustalonym wymaga zbudowania i rozwiązania często dość uciążliwych i bardzo złożonych układów algebraiczno-różniczkowych. Alternatywą, która znakomicie upraszcza rozwiązanie, jest metoda operatorowa oparta na wyznaczeniu transformaty Laplace'a.

Transformata Laplace'a funkcji rzeczywistej, używana powszechnie w teorii układów dynamicznych i teorii obwodów, definiowana jest wzorem całkowym:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt \tag{3}$$

gdzie $s = j\omega$ to zmienna zespolona nazwana pulsacją zespoloną.

Głównym celem przekształcenia Laplace'a jest konwersja równań różniczkowych z dziedziny czasu do postaci układu prostych równań w dziedzinie częstotliwości zespolonej (liniowo-algebraicznych).

Ważniejsze własności przydatne przy przekształcaniu struktury różniczkowo-całkowej sprowadzają się do:

- Liniowości: $\mathcal{L}\{af_1(t) + bf_2(t)\} = aF_1(s) + bF_2(s)$
- Transformaty pochodnej: $\mathcal{L}\left\{\frac{df(t)}{dt}\right\} = sF(s) - f(0^+)$
  (co obrazuje stan początkowy prądu płynącego przez cewkę/napięcia zasilającego przy komutacji).
- Transformaty całki: $\mathcal{L}\left\{\int f(t)dt\right\} = \frac{1}{s}F(s) + \frac{1}{s}\int_{-\infty}^{0^-} f(t)dt$
  (co uwzględnia ewentualne początkowe napięcie na okładkach kondensatora).

2.4 Analiza obwodów w dziedzinie częstotliwości i czasu

W dziedzinie częstotliwości zespolonej (operatorowej - przestrzeń s) analiza sprowadza się do obliczenia odpowiednika zastępczego impedancji podzespołów R, L i C. Z racji przekształcenia do dziedziny s, impedancje operatorowe wyliczamy następująco:

- Rezystor: $Z_R(s) = R$
- Cewka posada operatorową odpowiednio indukcyjność: $Z_L(s) = sL$
- Kondensator jest opisany relacją proporcji powrotnych: $Z_C(s) = \frac{1}{sC}$

Układ możemy teraz łatwo analizować na podstawie operatorowych praw Kirchhoffa i prawa Ohma. Finalne równanie rozwiązuje się poprzez manipulację na otrzymanych ułamkach wymiernych (rozkład na zbiór ułamków prostych). Ostatnim punktem metody jest

powrót z przestrzeni abstrakcyjnej zmiennej s do mierzalnych w czasie funkcji odpowiednich $f(t)$ dzięki wyznaczeniu odwrotnej transformaty Laplace'a:

$$f(t) = \mathcal{L}^{-1}\{F(s)\} \tag{4}$$

3 Funkcja skoku jednostkowego (Heaviside'a)

Funkcja Heaviside'a (oznaczana symbolem $H(x)$ lub $\theta(x)$) jest funkcją nieciągłą, która przyjmuje wartość zero dla argumentów ujemnych oraz wartość jeden dla argumentów dodatnich. Jest ona szeroko wykorzystywana w analizie sygnałów, teorii sterowania oraz rachunku operacyjnym do modelowania sygnałów włączanych w danej chwili czasu.

Matematycznie funkcję tę definiuje się następująco:

$$H(x) = \begin{cases} 0 & \text{dla } x < 0 \\ 1 & \text{dla } x > 0 \end{cases} \tag{5}$$

Wartość funkcji w punkcie $x = 0$ zależy od przyjętej konwencji. Najczęściej spotykane definicje to $H(0) = 0$, $H(0) = 1$ lub $H(0) = \frac{1}{2}$ (ostatnia forma jest często stosowana w analizie Fouriera).

Formalnie, funkcja Heaviside'a jest funkcją pierwotną dystrybucji delty Diraca:

$$H(x) = \int_{-\infty}^{x} \delta(t)\, dt \tag{6}$$

4 Delta Diraca (dystrybucja δ)

Delta Diraca, oznaczana symbolem $\delta(x)$, nie jest funkcją w sensie klasycznym, lecz dystrybucją (funkcją uogólnioną). Została wprowadzona przez fizyka Paula Diraca do opisu zjawisk punktowych, takich jak masa punktowa lub ładunek punktowy.

Intuicyjnie można ją rozumieć jako funkcję, która jest nieskończenie wysoka i nieskończenie wąska w punkcie $x = 0$, a jej pole powierzchni pod wykresem wynosi dokładnie 1. Formalnie definiuje się ją poprzez jej działanie pod znakiem całki:

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1 \quad \text{oraz} \quad \delta(x) = 0 \text{ dla } x \neq 0 \tag{7}$$

Najważniejszą własnością delty Diraca jest tzw. własność próbkowania (przesiewania), która pozwala wyizolować wartość dowolnej funkcji ciągłej $f(x)$ w konkretnym punkcie:

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)\, dx = f(x_0) \tag{8}$$

Delta Diraca jest również związana z funkcją Heaviside'a $H(x)$ jako jej pochodna w sensie dystrybucyjnym:

$$\frac{d}{dx}H(x) = \delta(x) \tag{9}$$

5 Transformata Laplace'a i transformata odwrotna Laplace'a

Sposób wyznaczenia transformaty oraz transformaty odwrotnej Laplace'a przedstawia listing 5.1.

Listing 5.1: Przykład wyznaczenia transformaty Laplace'a oraz transformaty odwrtonej Laplace'a.

```
import sympy as sp
t, s = sp.symbols("t s")

a = sp.symbols("a", real=True, positive=True)

f1 = sp.Heaviside(t)
sp.plot(f1)
F1 = sp.laplace_transform(f1, t, s, noconds=True)
print("Transformata skoku jednostkowego: ",F1)

# Przebieg tłumiony
f2 = sp.exp(-a*t)
sp.plot(f2.subs(a,1.0),(t,0,10))
F2 = sp.laplace_transform(f2, t, s, noconds=True)
print("Transformata e^(-at): ", F2)

# Odwrotna transformata sprawdzająca poprawne powrotne przejście do czasu
f2_inv = sp.inverse_laplace_transform(F2, s, t)
print("Odwrotna transformata Laplace'a F2: ", f2_inv)
sp.plot(f2_inv.subs(a,1))
```

Zadanie

W komórce notatnika zdefiniuj funkcję fali harmonicznej sp.sin(omega*t), wyznacz transformatę Laplace'a a następnie przeprowadź na wyniku transformatę odwrotną Laplace'a. Przedstaw wykresy.

Warto wiedzieć

Jeśli mamy wyznaczoną transmitancję napięciową $k(s)$, to wyznaczając odwrotną transformatę Laplace'a otrzymujemy odpowiedź na impuls (Deltę Diraca):

$$k(t) = \mathcal{L}^{-1}\{k(s)\} \tag{10}$$

Jeśli zaś wyznaczymy odwrotną transformatę Laplace,a wyrażenia $h(s) = \frac{k(s)}{s}$, to otrzymujemy odpowiedź na skok jednostkowy (funkcja Heaviside'a).

$$h(t) = \mathcal{L}^{-1}\left\{\frac{k(s)}{s}\right\} \tag{11}$$

Uwaga!

Jeśli stopień wielomianu w liczniku transmitancji operatorowej jest większy bądź równy stopniowi mianownika, to w wyniku odwrotnej transformaty Laplace'a powstaną delty Dirac'a. Aby prawidłowo wyznaczyć taką transformatę należy rozłożyć transmitancję operatorową na ułamki proste. Służy do tego funkcja **apart** z modułu Sympy. Niektóre systemy obliczeń symbolicznych np. **Maxima** nie potrafią wyznaczyć odwrotnej transformaty Laplace'a z „jedynki" - czyli wprowadzenia symbolu delty Diraca $\delta(t)$. W takim przypadku przed wyznaczeniem transformaty odwrotnej trzeba się pozbyć jedynki z wyrażenia.

Uwaga!

Jeśli w skrypcie zostanie zadeklarowana zmienna czas **t** z symbolem „t" oraz opcjami **real=True** oraz **positive=True**, czyli wymuszemie, że t jest liczbą rzeczywistą i dodatnią, to w wyniku transformaty odwrotnej Laplace'a nie pojawią się symbole delty Diraca $\delta(t)$.

Listing 5.2: Przykład wyznaczenia odwrotnej transformaty Laplace'a gdy stopień licznika = stopniowi mianownika, z wykorzystaniem modułu Sympy

```
from sympy import inverse_laplace_transform, apart, symbols

t = symbols("t", real=True, positive=False)
s = symbols("s")

H = (s + 2) / (s + 5)
# Wymuś rozkład na ułamki proste (część stała + ułamek właściwy)
H_apart = apart(H, s)
display(H_apart)
h_t = inverse_laplace_transform(H_apart, s, t)
print(h_t)
```

.

6 Zadania projektowe

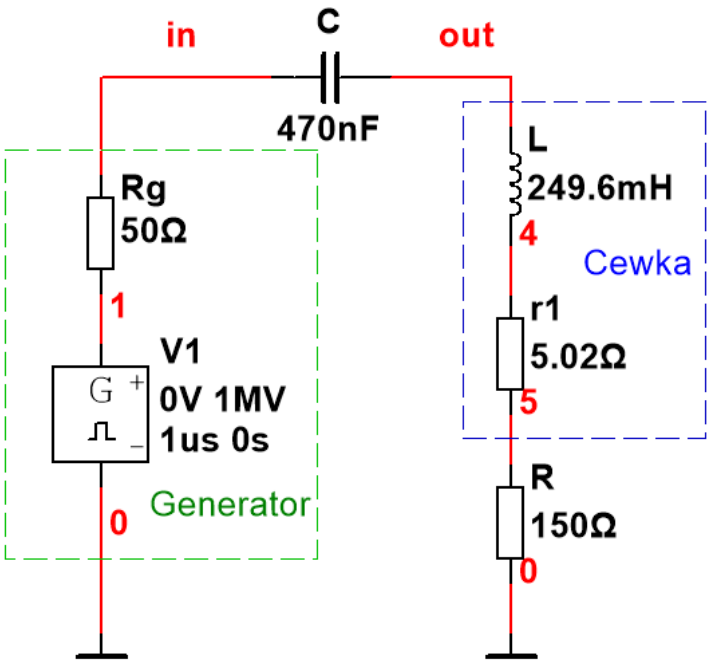
Zadanie

Wykorzystując moduły Sympy oraz Lcapy wyznaczyć transmitancję operatorową układu z projektu na laboratoria „Stany Nieustalone” w ramach przedmiotu Elektronika w Eksperymencie Fizycznym. Celem weryfikacji wykonać odpowiednie symulację w NI Multisim, albo LTSpice. W efekcie przygotować projekt na laboratoria.

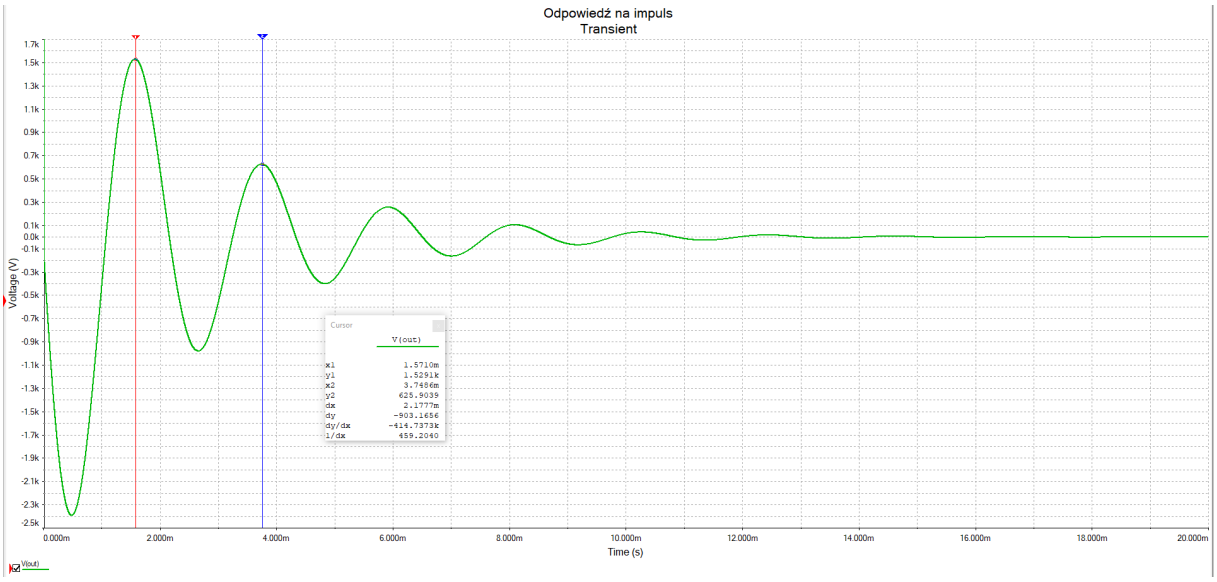
Informacje dodatkowe

Obliczenia symboliczne i numeryczne można wykonać także w oprogramowaniu **Maxima** - wraz z graficzną nakładką **wxMaxima**. Symulacje można wykonać w oprogramowaniu Analog Devices LTSpice. Warto zapoznać się z przygotowanym materiałem video: https://study.engined.eu/mod/lesson/view.php?id=773.

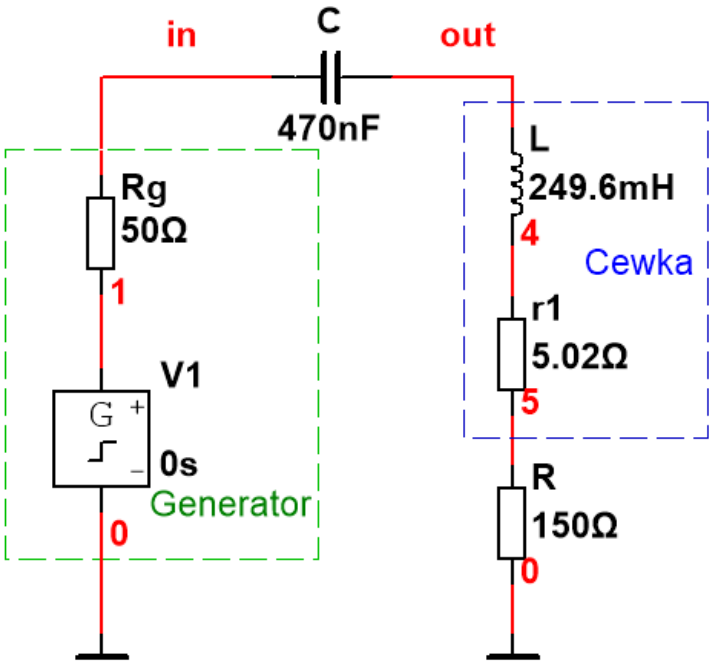
6.1 Układ typu A



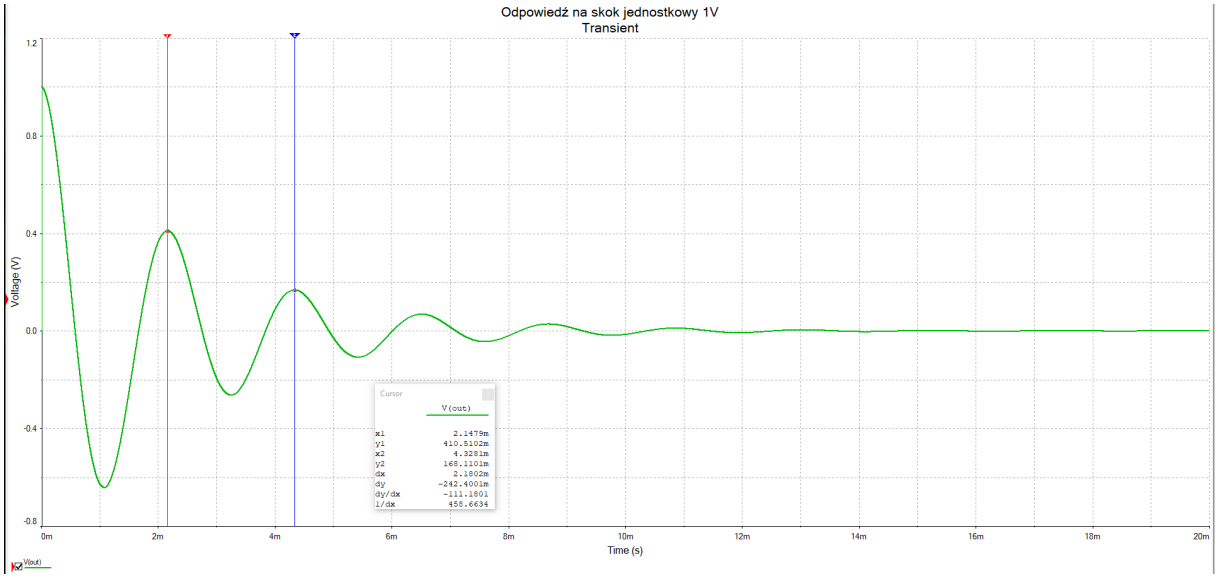
Rysunek 1: Obwód typu A do symulacji pobudzeniem impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu $1\mu\text{s}$, napięcie 1MV



Rysunek 2: Obwód typu A. Odpowiedź czasowa na pobudzenie impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu $1\mu\text{s}$, napięcie 1MV.



Rysunek 3: Obwód typu A do symulacji pobudzeniem skokiem jednostkowym 1V.



Rysunek 4: Obwód typu A. Odpowiedź czasowa na pobudzenie skokiem jednostkowym 1V.

Listing 6.1: Obwód typu A. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy

```
# Obwód typu A
import sympy as sp
r, Rg, R, L, C = sp.symbols("r Rg R L C", real=True, positive=True)
```

Listing 6.1: Obwód typu A. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
s = sp.symbols("s")
#Impedancje
ZR = R
Zr = r
ZRg = Rg
ZL = s * L
ZC = 1/ (s*C)

#Transmitancja
K = (ZL + Zr + R)/((ZRg + ZC + ZL + Zr + R))
K = sp.simplify(K)
print("Forma kanoniczna Transmitancja K(s):")
display(K.cancel())
print("Faktoryzacja Transmitancja K(s):")
display(K.factor())
licznik, mianownik = K.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

#print("cancel: ", sp.cancel(K))

zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)

print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

wartosci = { Rg: 50,
             R: 150,
             L: 249.6e-3,
             r: 5.02,
             C: 470e-9
           }

Kw = K.subs(wartosci).simplify()
print("Transmitancja K(s)=",Kw)
licznik, mianownik = Kw.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

# Wyznaczenie zer i biegunów
zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)
```

Listing 6.1: Obwód typu A. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)


# Odpowiedź na impuls (delta Diraca)

#UWAGA! Poniższa deklaracja czas - wartość rzeczywista i dodatnia,
↪  powoduje, że znikają Delty Dirac'a
t = sp.symbols('t', real=True, positive=True)
#Rozkład na ułamki proste. Odwrotna transformata Laplace'a z 1 daje
↪  DiracDelta(t)
KK = sp.apart(K,s)
print("KK: ",KK)
k_t = sp.inverse_laplace_transform(KK,s,t).simplify()
display(k_t)
k_tw = k_t.subs(wartosci)
k_tw = k_tw.simplify()
print("k(t)=")
display(k_tw)
print("Odpowiedź na impuls o polu 1 V*s, wartości nieskończonej i czasie
↪  nieskończenie małym (Delta Diraca)")
sp.plot(k_tw,(t,0,0.02))

import matplotlib.pyplot as plt

#Wyznaczenie zer i biegunów oraz przedstawienie na wykresie na
↪  płaszczyźnie zespolonej
# Wyciągamy Re i Im z biegunów
zera_re_parts = [sp.re(p.evalf()) for p in zera]
zera_im_parts = [sp.im(p.evalf()) for p in zera]
bieguny_re_parts = [sp.re(p.evalf()) for p in bieguny]
bieguny_im_parts = [sp.im(p.evalf()) for p in bieguny]

plt.scatter(bieguny_re_parts, bieguny_im_parts, marker='x', color='red',
↪  label='Bieguny')
plt.scatter(zera_re_parts, zera_im_parts, marker='o', color='blue',
↪  label='Zera')

plt.axvline(0, color='black', lw=1); plt.axhline(0, color='black', lw=1)
plt.grid(); plt.legend(); plt.title("Mapa zer i biegunów")
plt.show()

# Odpowiedź na skok (funkcja Heaviside'a)

t = sp.symbols('t', real=True, positive=True)
h_t = sp.inverse_laplace_transform(K/s,s,t).simplify()
```

Listing 6.1: Obwód typu A. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
display(h_t)
h_tw = h_t.subs(wartosci)
h_tw = h_tw.simplify()
print("h(t)=")
display(h_tw)
print("Odpowiedź na skok jednostkowy 1V w chwili t=0")
p=sp.plot(h_tw,(t,0,0.02))
```

Listing 6.2: Obwód typu A. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy

```
# Obwód typu A. Odpowiedź na impuls
import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 {delta(t)}  ; down=1.5
#V1 1 0 {1e6*(u(t) - u(t - 1e-6))} ; down
Rg 1 in 50 ; up=2
C in out 470n; right=2
L out 2 249.6m ; down=1.5
R1 2 3 5.02 ; down
R 3 0 150 ; down
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
k = cct.out.v
kw = k.evalf().simplify().evalf()
print("k(t): ",kw)
_ = k.plot((0,20e-3))
```

Listing 6.3: Obwód typu A. Wyznaczenie odpowiedzi na skok jednostkowy z wykorzystaniem modułu Lcapy

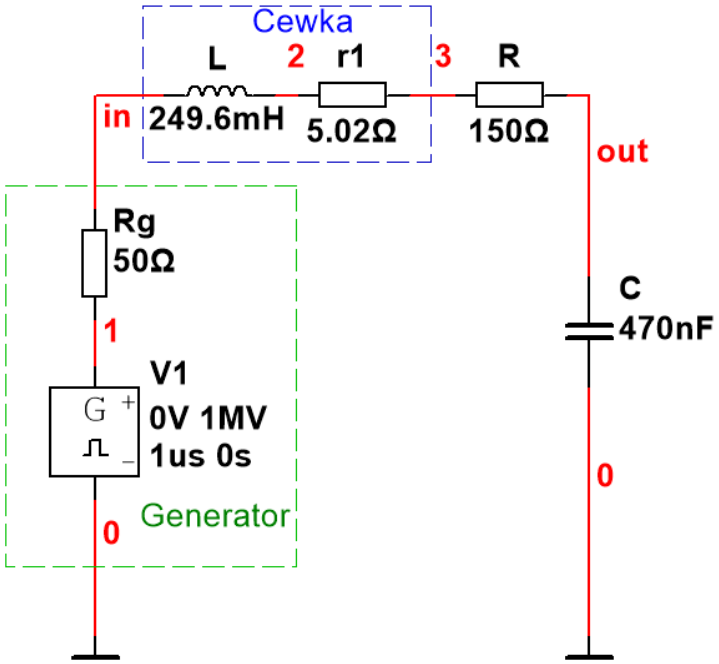
```
# Obwód typu A. Odpowiedź na skok

import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 step 1 ; down=1.5
Rg 1 in 50 ; up=2
```

Listing 6.3: Obwód typu A. Wyznaczenie odpowiedzi na skok jednostkowy z wykorzystaniem modułu Lcapy (c.d.)

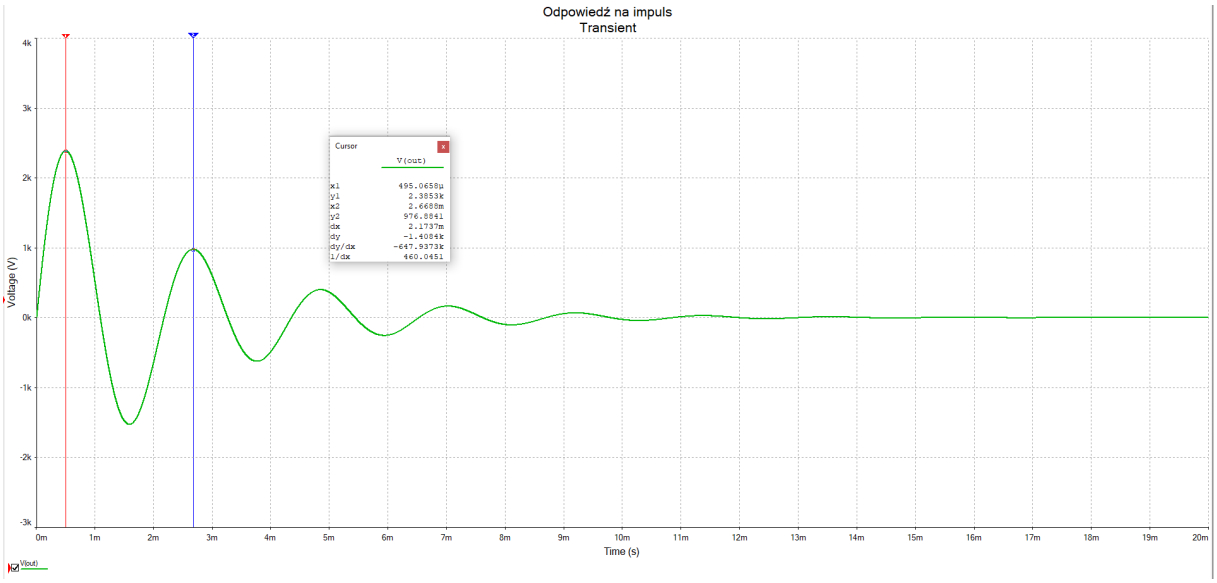
```
C in out 470n; right=2
L out 2 249.6m ; down=1.5
R1 2 3 5.02 ; down
R 3 0 150 ; down
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
h = cct.out.v
hw = h.evalf().simplify().evalf()
print("h(t): ",hw)
_=h.plot((1e-6,20e-3))
```


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

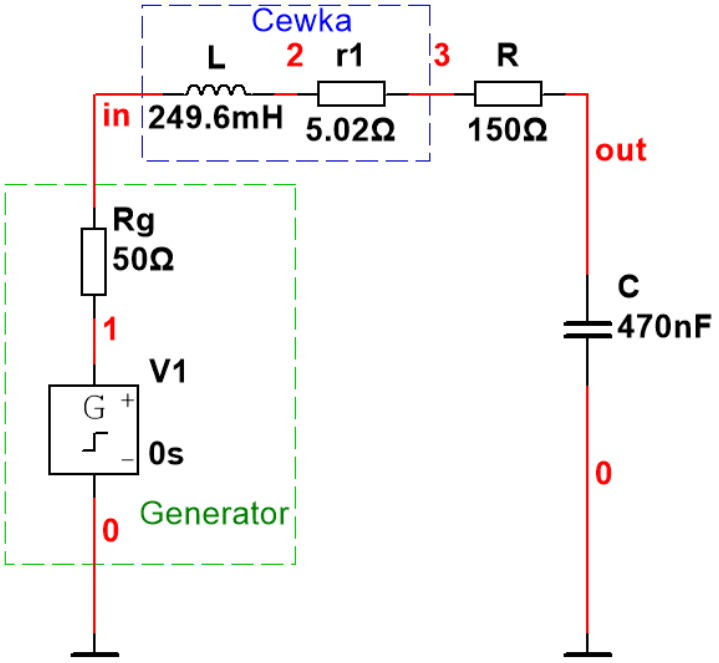
6.2 Układ typu B



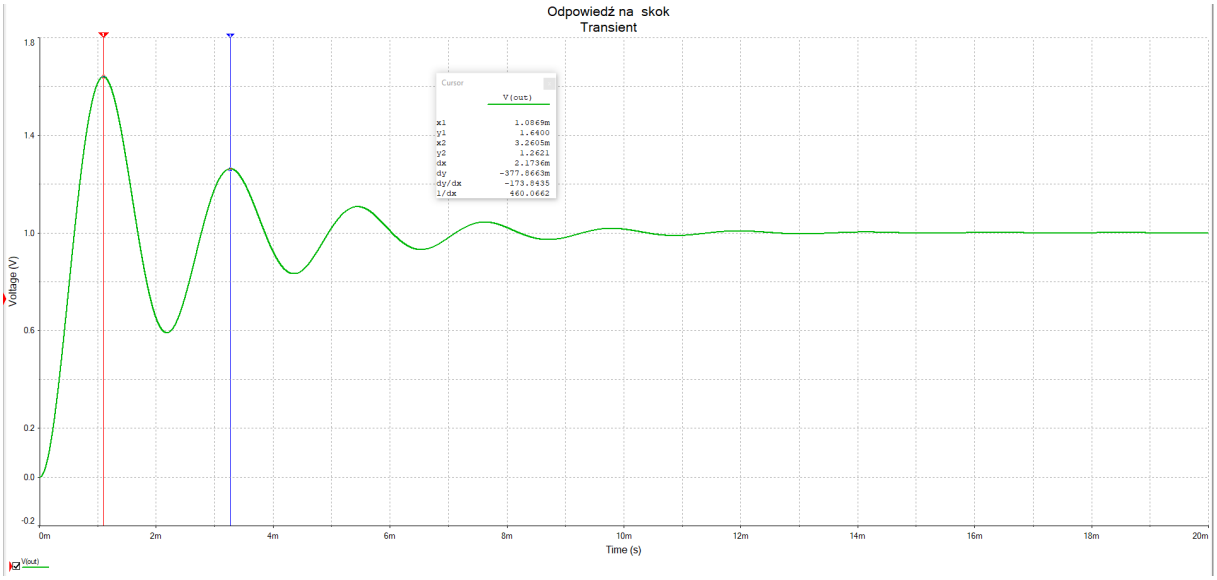
Rysunek 5: Obwód typu B do symulacji pobudzeniem impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV



Rysunek 6: Obwód typu B. Odpowiedź czasowa na pobudzenie impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV.



Rysunek 7: Obwód typu B do symulacji pobudzeniem skokiem jednostkowym 1V.



Rysunek 8: Obwód typu B. Odpowiedź czasowa na pobudzenie skokiem jednostkowym 1V.

Listing 6.4: Obwód typu B. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy

```
# Obwód typu B
import sympy as sp
r, Rg, R, L, C = sp.symbols("r Rg R L C", real=True, positive=True)
```

Listing 6.4: Obwód typu B. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
s = sp.symbols("s")
#Impedancje
ZR = R
Zr = r
ZRg = Rg
ZL = s * L
ZC = 1/ (s*C)

#Transmitancja
K = (ZC)/((ZRg + ZC + ZL + Zr + R))
K = sp.simplify(K)
print("Forma kanoniczna Transmitancja K(s):")
display(K.cancel())
print("Faktoryzacja Transmitancja K(s):")
display(K.factor())
licznik, mianownik = K.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

#print("cancel: ", sp.cancel(K))

zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)

print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

wartosci = { Rg: 50,
             R: 150,
             L: 249.6e-3,
             r: 5.02,
             C: 470e-9
           }

Kw = K.subs(wartosci).simplify()
print("Transmitancja K(s)=",Kw)
licznik, mianownik = Kw.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

# Wyznaczenie zer i biegunów
zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)
```

Listing 6.4: Obwód typu B. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

# Odpowiedź na impuls (delta Diraca)

#UWAGA! Poniższa deklaracja czas - wartość rzeczywista i dodatnia,
↪ powoduje, że znikają Delty Dirac'a
t = sp.symbols('t', real=True, positive=True)
#Rozkład na ułamki proste. Odwrotna transformata Laplace'a z 1 daje
↪ DiracDelta(t)
KK = sp.apart(K,s)
print("KK: ",KK)
k_t = sp.inverse_laplace_transform(KK,s,t).simplify()
display(k_t)
k_tw = k_t.subs(wartosci)
print("k(t)=")
display(k_tw)
print("Odpowiedź na impuls o polu 1 V*s, wartości nieskończonej i czasie
↪ nieskończenie małym (Delta Diraca)")
sp.plot(k_tw,(t,0,0.02))

# Przedstawienie zer i biegunów na płaszczyźnie zespolonej
import matplotlib.pyplot as plt
# Wyciągamy Re i Im z biegunów

zera_re_parts = [sp.re(p.evalf()) for p in zera]
zera_im_parts = [sp.im(p.evalf()) for p in zera]
bieguny_re_parts = [sp.re(p.evalf()) for p in bieguny]
bieguny_im_parts = [sp.im(p.evalf()) for p in bieguny]

plt.scatter(bieguny_re_parts, bieguny_im_parts, marker='x', color='red',
↪ label='Bieguny')
plt.scatter(zera_re_parts, zera_im_parts, marker='o', color='blue',
↪ label='Zera')

plt.axvline(0, color='black', lw=1); plt.axhline(0, color='black', lw=1)
plt.grid(); plt.legend(); plt.title("Mapa zer i biegunów")
plt.show()

# Odpowiedź na skok (funkcja Heaviside'a)

t = sp.symbols('t', real=True, positive=True)
h_t = sp.inverse_laplace_transform(K/s,s,t).simplify()
display(h_t)

h_tw = h_t.subs(wartosci)
```

Listing 6.4: Obwód typu B. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
# expr_clean = h_tw.rewrite(sp.exp).expand()
# expr = sp.collect(expr_clean, sp.exp(sp.Wild('a') * t))
# h_tw = sp.simplify(expr)

# expr = sp.expand(h_tw)
# expr = expr.rewrite(sp.cos)
# expr = sp.trigsimp(expr)
# print(expr)
# h_tw = expr

expr_clean = h_tw.rewrite(sp.exp).expand() #.rewrite(sp.sin)
h_tw = expr_clean.expand(complex=True)
h_tw = h_tw.simplify()

# h_tw = h_t.subs(wartosci)
# expr_clean = h_tw.rewrite(sp.exp).expand().rewrite(sp.sin)
# h_tw = expr_clean.expand(complex=True)
#h_tw = h_tw.simplify()
print("h(t)=")
display(h_tw)
print("Odpowiedź na skok jednostkowy 1V w chwili t=0")
p=sp.plot(h_tw,(t,0,0.02))
```

Listing 6.5: Obwód typu B. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy

```
# Obwód typu B. Odpowiedź na impuls
import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 {delta(t)} ; down
Rg 1 in 50 ; up
L in 2 249.6m ; right=1.5
R1 2 3 5.02 ; right
R 3 out 150 ; right=1.5
C out 0 470n; down=2
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
k = cct.out.v
kw = k.evalf().simplify().evalf()
```

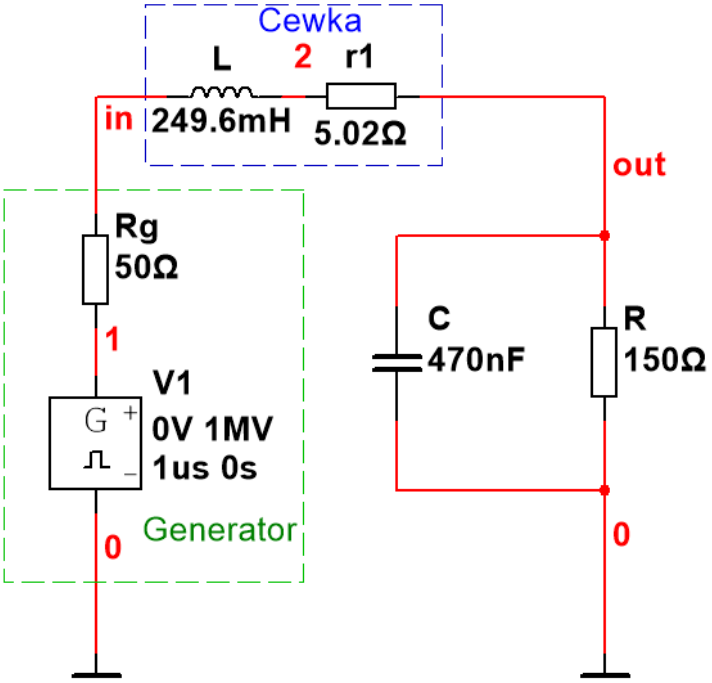
Listing 6.5: Obwód typu B. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy (c.d.)

```
print("k(t): ",kw)
_ = k.plot((0,20e-3))
```

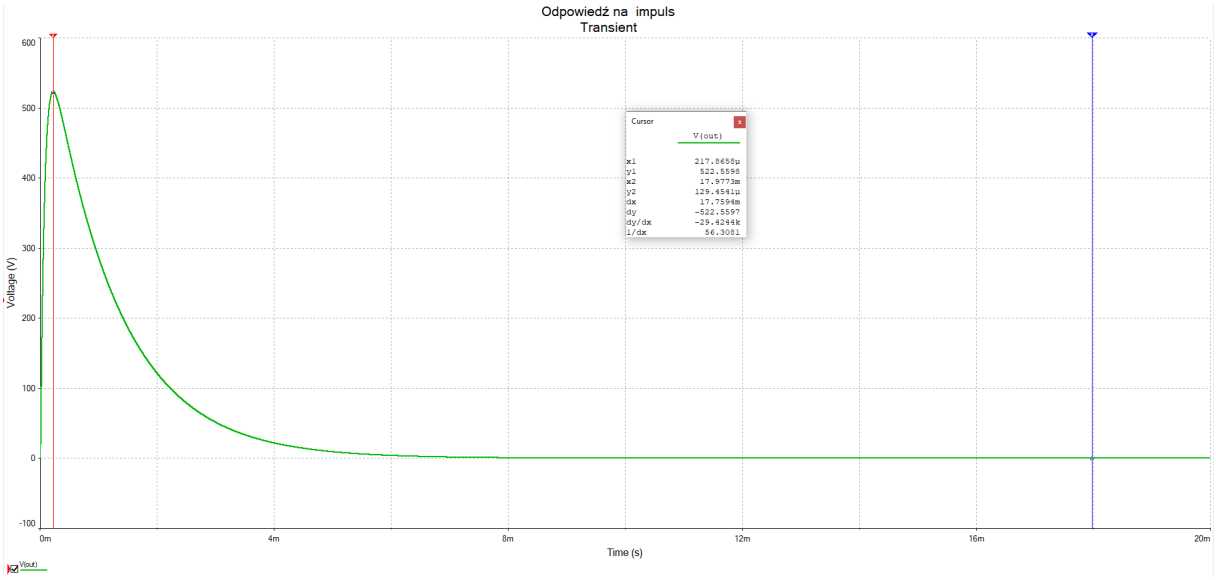
Listing 6.6: Obwód typu B. Wyznaczenie odpowiedzi na skok jednostkowy z wykorzystaniem modułu Lcapy

```
# Obwód typu B. Odpowiedź na skok

import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 step 1 ; down=1.5
Rg 1 in 50 ; up
L in 2 249.6m ; right=1.5
R1 2 3 5.02 ; right
R 3 out 150 ; right=1.5
C out 0 470n; down=2
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
h = cct.out.v
hw = h.evalf().simplify().evalf()
print("h(t): ",hw)
_=h.plot((0,20e-3))
```

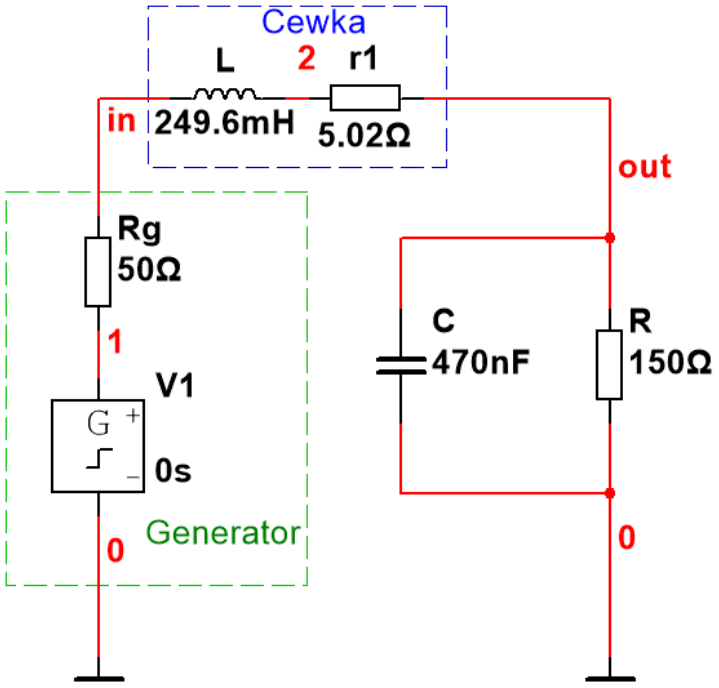
6.3 Układ typu C



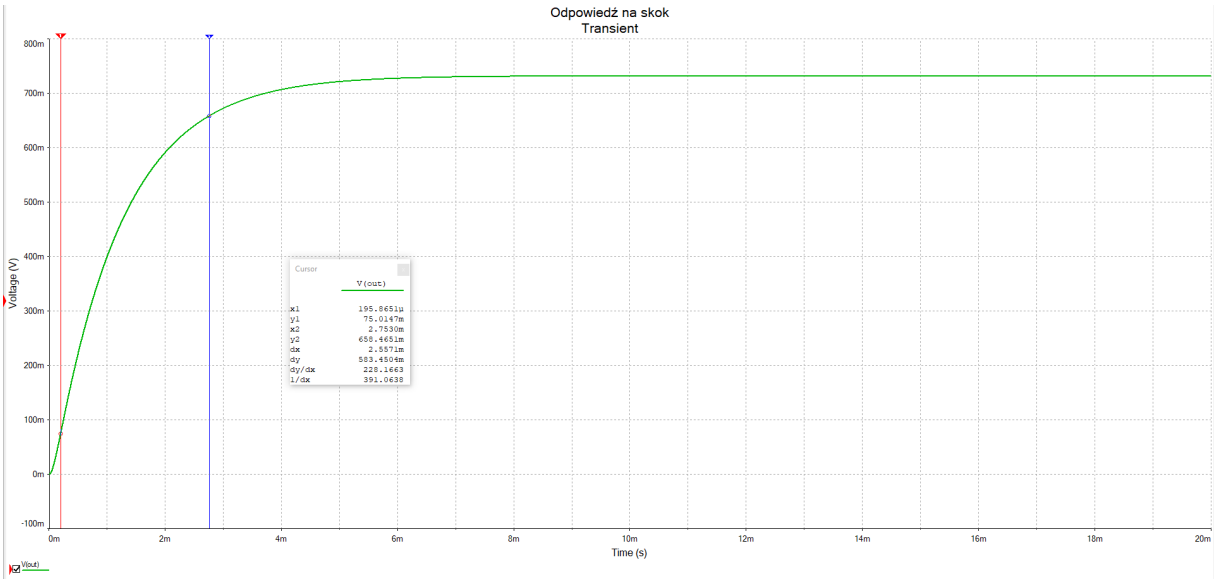
Rysunek 9: Obwód typu C do symulacji pobudzeniem impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV



Rysunek 10: Obwód typu C. Odpowiedź czasowa na pobudzenie impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV.



Rysunek 11: Obwód typu C do symulacji pobudzeniem skokiem jednostkowym 1V.



Rysunek 12: Obwód typu C. Odpowiedź czasowa na pobudzenie skokiem jednostkowym 1V.

Listing 6.7: Obwód typu C. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy

```
# Obwód typu C
import sympy as sp
r, Rg, R, L, C = sp.symbols("r Rg R L C", real=True, positive=True)
```

Listing 6.7: Obwód typu C. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
s = sp.symbols("s")
#Impedancje
ZR = R
Zr = r
ZRg = Rg
ZL = s * L
ZC = 1/ (s*C)

#Transmitancja
K = ((ZC*ZR)/(ZC+ZR))/( ZRg + ZL + Zr + (ZC*ZR)/(ZC+ZR) )
K = sp.simplify(K)
print("Forma kanoniczna Transmitancja K(s):")
display(K.cancel())
print("Faktoryzacja Transmitancja K(s):")
display(K.factor())
licznik, mianownik = K.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

#print("cancel: ", sp.cancel(K))

zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)

print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

wartosci = { Rg: 50,
             R: 150,
             L: 249.6e-3,
             r: 5.02,
             C: 470e-9
           }

Kw = K.subs(wartosci).simplify()
print("Transmitancja K(s)=",Kw)
licznik, mianownik = Kw.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

# Wyznaczenie zer i biegunów
zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)
```

Listing 6.7: Obwód typu C. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

# Odpowiedź na impuls (delta Diraca)

#UWAGA! Poniższa deklaracja czas - wartość rzeczywista i dodatnia,
↪ powoduje, że znikają Delty Dirac'a
t = sp.symbols('t', real=True, positive=True)
#Rozkład na ułamki proste. Odwrotna transformata Laplace'a z 1 daje
↪ DiracDelta(t)
KK = sp.apart(K,s)
print("KK: ",KK)
k_t = sp.inverse_laplace_transform(KK,s,t).simplify()
display(k_t)
k_tw = k_t.subs(wartosci)
expr_clean = k_tw.rewrite(sp.exp).expand() #.rewrite(sp.sin)
k_tw = expr_clean.expand(complex=True)
k_tw = k_tw.simplify()
display(k_tw)
print("Odpowiedź na impuls o polu 1 V*s, wartości nieskończonej i czasie
↪ nieskończenie małym (Delta Diraca)")
sp.plot(k_tw,(t,0,0.02))

# # Przedstawienie zer i biegunów na płaszczyźnie zespolonej
import matplotlib.pyplot as plt
# # Wyciągamy Re i Im z biegunów

zera_re_parts = [sp.re(p.evalf()) for p in zera]
zera_im_parts = [sp.im(p.evalf()) for p in zera]
bieguny_re_parts = [sp.re(p.evalf()) for p in bieguny]
bieguny_im_parts = [sp.im(p.evalf()) for p in bieguny]

plt.scatter(bieguny_re_parts, bieguny_im_parts, marker='x', color='red',
↪ label='Bieguny')
plt.scatter(zera_re_parts, zera_im_parts, marker='o', color='blue',
↪ label='Zera')

plt.axvline(0, color='black', lw=1); plt.axhline(0, color='black', lw=1)
plt.grid(); plt.legend(); plt.title("Mapa zer i biegunów")
plt.show()
print("odp na skok")
# Odpowiedź na skok (funkcja Heaviside'a)
hh = K/s
h2=hh.simplify()
h = h2.apart(s)
display(h)
```

Listing 6.7: Obwód typu C. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
t = sp.symbols('t', real=True, positive=True)
h_t = sp.inverse_laplace_transform(h,s,t).simplify()
print("H_t:",h_t)
display(h_t)
h_tw = h_t.subs(wartosci)
#h_tw=h_tw.rewrite(sp.cos)
expr_clean = h_tw.rewrite(sp.exp).expand().rewrite(sp.sin)
h_tw = expr_clean.expand(complex=True)
h_tw = h_tw.simplify()
display(h_tw)
print("Odpowiedź na skok jednostkowy 1V w chwili t=0")
p=sp.plot(h_tw,(t,0,0.02))
```

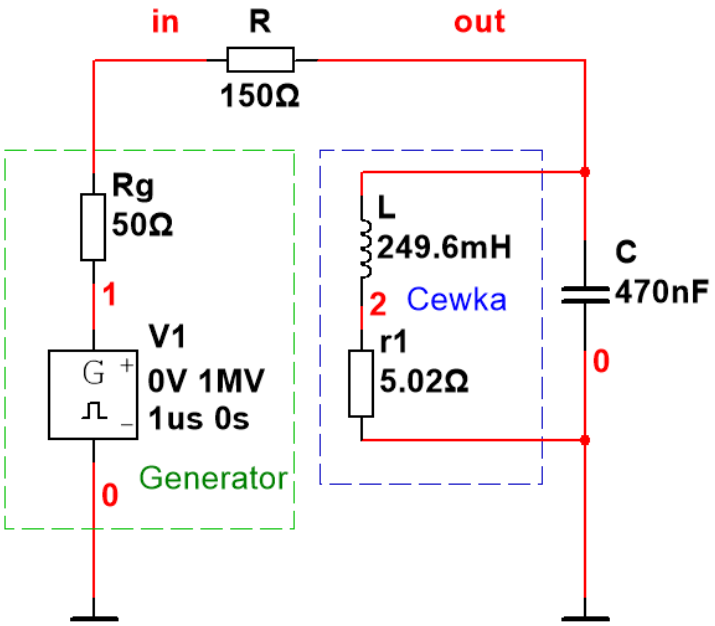
Listing 6.8: Obwód typu C. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy

```
# Obwód typu C. Odpowiedź na impuls
import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 {delta(t)}  ; down
Rg 1 in 50 ; up
L in 2 249.6m ; right=1.5
R1 2 out 5.02 ; right=1.5
C out 0 470n; down=2
W out 3; right=1.5
R 3 0 150 ; down=2
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
k = cct.out.v
kw = k.evalf().simplify().evalf()
print("k(t): ",kw)
_ = k.plot((0,20e-3))
```

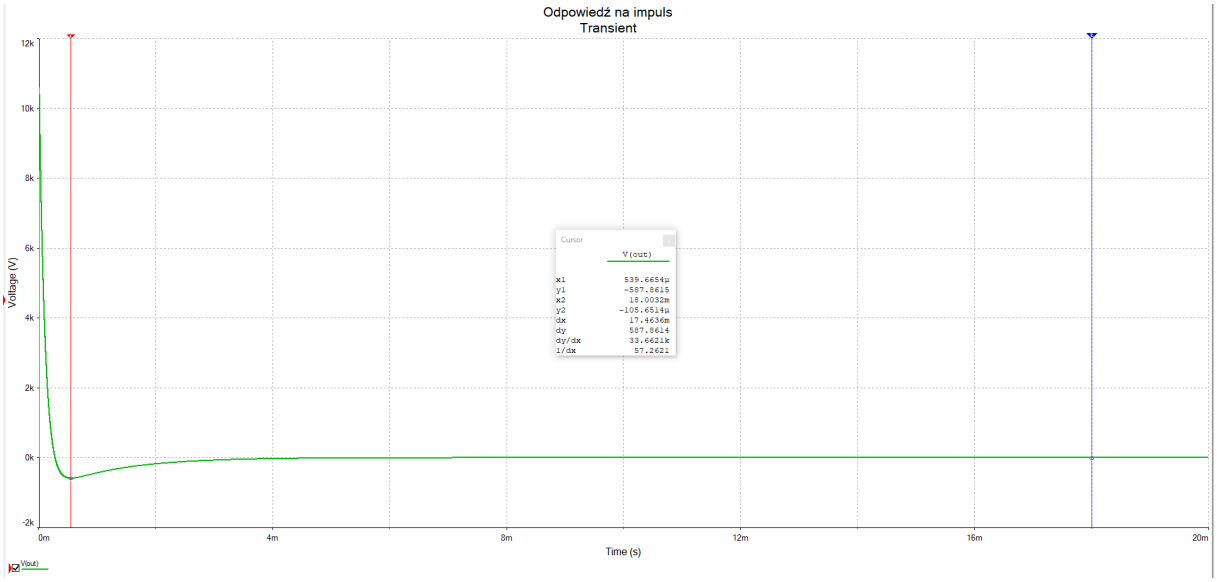
Listing 6.9: Obwód typu C. Wyznaczenie odpowiedzi na skok jednostkowy z wykorzystaniem modułu Lcapy

```
# Obwód typu C. Odpowiedź na skok

import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 step 1 ; down
Rg 1 in 50 ; up
L in 2 249.6m ; right=1.5
R1 2 out 5.02 ; right=1.5
C out 0 470n; down=2
W out 3; right=1.5
R 3 0 150 ; down=2
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
h = cct.out.v
hw = h.evalf().simplify().evalf()
print("h(t): ",hw)
_=h.plot((0,20e-3))
```

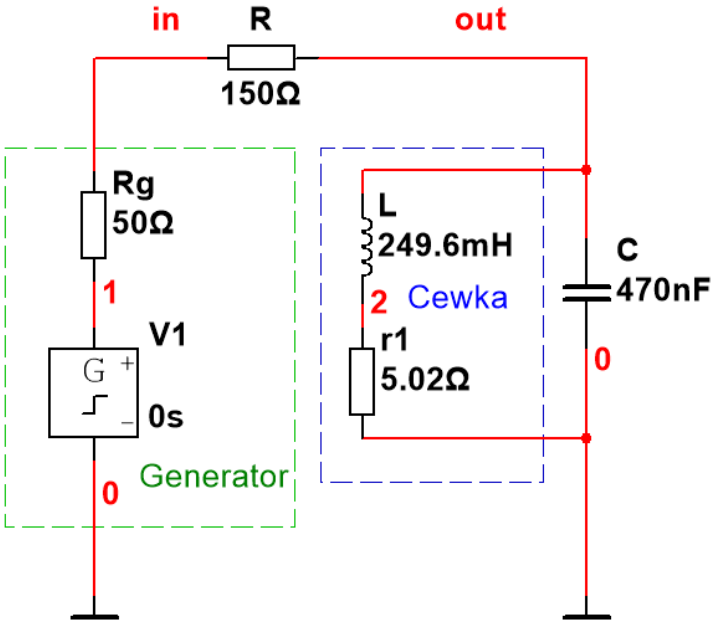
6.4 Układ typu D



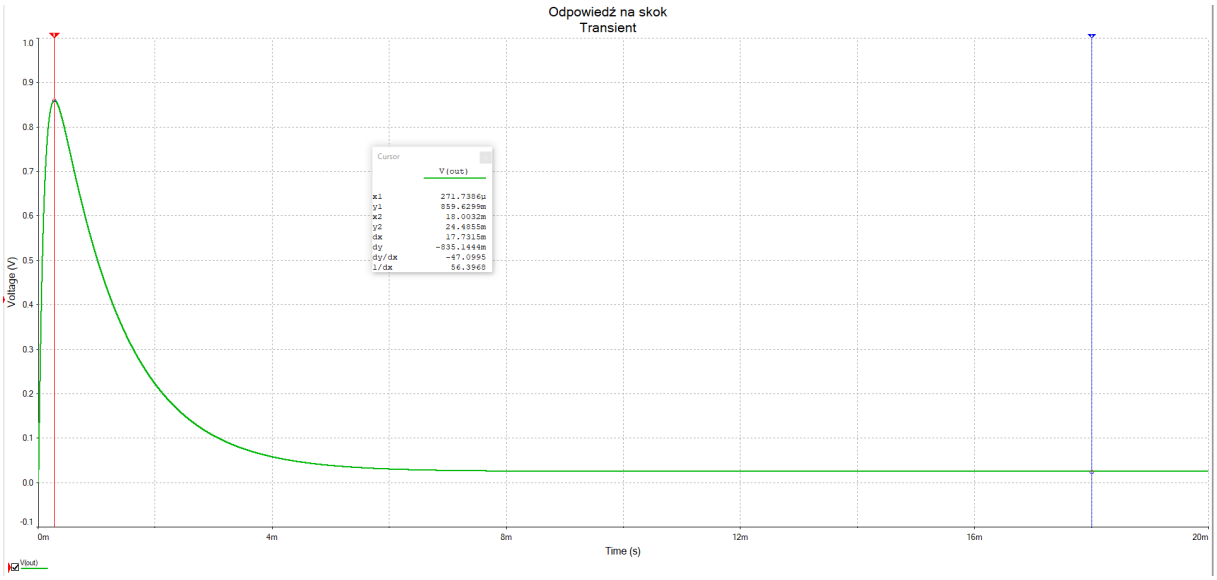
Rysunek 13: Obwód typu D do symulacji pobudzeniem impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV



Rysunek 14: Obwod typu D. Odpowiedź czasowa na pobudzenie impulsem zgodnym z impulsem Delty Diraca - pole impulsu $S = 1\text{V} \cdot \text{s}$, czas trwania impulsu 1μs, napięcie 1MV.



Rysunek 15: Obwód typu D do symulacji pobudzeniem skokiem jednostkowym 1V.



Rysunek 16: Obwód typu D. Odpowiedź czasowa na pobudzenie skokiem jednostkowym 1V.

Listing 6.10: Obwód typu D. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy

```
# Obwód typu D
import sympy as sp
r, Rg, R, L, C = sp.symbols("r Rg R L C", real=True, positive=True)
s = sp.symbols("s")
```

Listing 6.10: Obwód typu D. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
#Impedancje
ZR = R
Zr = r
ZRg = Rg
ZL = s * L
ZC = 1/ (s*C)

#Transmitancja
K = ((ZC+Zr)*ZL/(ZC+ZL+Zr))/(ZRg + ZR + (ZC+Zr)*ZL/(ZC+ZL+Zr))
K = sp.simplify(K)
print("Forma kanoniczna Transmitancja K(s):")
display(K.cancel())
print("Faktoryzacja Transmitancja K(s):")
display(K.factor())
licznik, mianownik = K.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

#print("cancel: ", sp.cancel(K))

zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)

print("Zera układu: ", zera)
print("Bieguny układu: ", bieguny)

wartosci = { Rg: 50,
             R: 150,
             L: 249.6e-3,
             r: 5.02,
             C: 470e-9
           }

Kw = K.subs(wartosci).simplify()
print("Transmitancja K(s)=",Kw)
licznik, mianownik = Kw.as_numer_denom()
print("licznik: ", licznik.expand())
#print("faktor licznik: ", licznik.collect(s))
print("mianownik: ", mianownik.expand())

# Wyznaczenie zer i biegunów
zera = sp.solve(licznik, s)
bieguny = sp.solve(mianownik, s)

print("Zera układu: ", zera)
```

Listing 6.10: Obwód typu D. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
print("Bieguny układu: ", bieguny)

# Odpowiedź na impuls (delta Diraca)

#UWAGA! Poniższa deklaracja czas - wartość rzeczywista i dodatnia,
↪  powoduje, że znikają Delty Dirac'a
t = sp.symbols('t', real=True, positive=True)
#Rozkład na ułamki proste. Odwrotna transformata Laplace'a z 1 daje
↪  DiracDelta(t)
KK = sp.apart(K,s)
print("KK: ",KK)
k_t = sp.inverse_laplace_transform(KK,s,t).simplify()
display(k_t)
k_tw = k_t.subs(wartosci)

expr_clean = k_tw.rewrite(sp.exp).expand() #.rewrite(sp.sin)
k_tw = expr_clean.expand(complex=True)
k_tw = k_tw.simplify()
print("k(t)=")
display(k_tw)
print("Odpowiedź na impuls o polu 1 V*s, wartości nieskończonej i czasie
↪  nieskończenie małym (Delta Diraca)")
sp.plot(k_tw,(t,0,0.02))

# Przedstawienie zer i biegunów na płaszczyźnie zespolonej
import matplotlib.pyplot as plt
# Wyciągamy Re i Im z biegunów

zera_re_parts = [sp.re(p.evalf()) for p in zera]
zera_im_parts = [sp.im(p.evalf()) for p in zera]
bieguny_re_parts = [sp.re(p.evalf()) for p in bieguny]
bieguny_im_parts = [sp.im(p.evalf()) for p in bieguny]

plt.scatter(bieguny_re_parts, bieguny_im_parts, marker='x', color='red',
↪  label='Bieguny')
plt.scatter(zera_re_parts, zera_im_parts, marker='o', color='blue',
↪  label='Zera')

plt.axvline(0, color='black', lw=1); plt.axhline(0, color='black', lw=1)
plt.grid(); plt.legend(); plt.title("Mapa zer i biegunów")
plt.show()

# Odpowiedź na skok (funkcja Heaviside'a)

t = sp.symbols('t', real=True, positive=True)
h_t = sp.inverse_laplace_transform(K/s,s,t).simplify()
```

Listing 6.10: Obwód typu D. Wyznaczenie odpowiedzi na impuls oraz odpowiedzi na skok jednostkowy z wykorzystaniem modułu Sympy (c.d.)

```
display(h_t)
h_tw = h_t.subs(wartosci)

expr_clean = h_tw.rewrite(sp.exp).expand() #.rewrite(sp.sin)
h_tw = expr_clean.expand(complex=True)
h_tw = h_tw.simplify()
print("h(t)=")

display(h_tw)
print("Odpowiedź na skok jednostkowy 1V w chwili t=0")
p=sp.plot(h_tw,(t,0,0.02))
```

Listing 6.11: Obwód typu D. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy

```
# Obwód typu D. Odpowiedź na impuls
import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 {delta(t)}  ; down
#V1 1 0 {1e6*(u(t) - u(t - 1e-6))} ; down
Rg 1 in 50 ; up
R in out 150 ; right=2
C out 0 470n ; down=2
W out 3 ; right=1.5
L 3 2 249.6m ; down
R1 2 0 5.02 ; down
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european

""")
cct.draw()
#V_in_s = cct.V1.V.laplace()
#V_out_s = cct.C.V.laplace()
#ks = V_out_s / V_in_s
#print("ks: ", ks.evalf().simplify())
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
display("Bieguny 'x' i zera 'O' na płaszczyźnie zespolonej")
ks.plot()


print("Odpowiedź na impuls k(t)")
k = cct.out.v
kw = k.evalf().simplify().evalf()
```

Listing 6.11: Obwód typu D. Wyznaczenie odpowiedzi na impuls z wykorzystaniem modułu Lcapy (c.d.)

```
print(kw)
k.plot((0,20e-3))
```

Listing 6.12: Obwód typu D. Wyznaczenie odpowiedzi na skok jednostkowy z wykorzystaniem modułu Lcapy

```
# Obwód typu D. Odpowiedź na skok

import lcapy as lc
from lcapy import Circuit

cct = Circuit("""
V1 1 0 step 1 ; down
Rg 1 in 50 ; up
R in out 150 ; right=2
C out 0 470n ; down=2
W out 3 ; right=1.5
L 3 2 249.6m ; down
R1 2 0 5.02 ; down
; autoground=tground, label_ids=true, draw_nodes=connections
; resistor_style=european
""")
cct.draw()
ks = cct.out.V.laplace().evalf().simplify().evalf()
print("k(s): ",ks)
ks.plot()
h = cct.out.v
hw = h.evalf().simplify().evalf()
print("h(t): ",hw)
_=h.plot((0,20e-3))
```

7 Autorzy i historia opracowania

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