241auswertung

February 16, 2025

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

plt.rcParams.update({'font.size': 12})

%run ../lib.ipynb
```

[3]:
```
class aufgabe1:
    f = ValErr(165, 1) # Hz
    halbw = [
        # C, R, T_1/2
        [ 470, 1, ValErr(0.3, 0.1) * 1000 ], # nF, kO, µs
        [ 4.7, 10, ValErr(45, 1) ], # nF, kO, µs
        [ 47, 1, ValErr(34, 1) ], # nF, kO, µs
        [ 47, 1, ValErr(32, 1) ], # nF, kO, µs
    ]

class aufgabe3tp:
    f_intersect = ValErr(9.46, 0.02) # kHz

class aufgabe3hp:
    f_intersect = ValErr(3.31, 0.02) # kHz
    phase_shift = [
        # f [kHz], Dt [µs], phi [°]
        [ 1, ValErr(0.22, 0.02) * 1000, ValErr(79, 8) ],
        [ 2, ValErr(86, 2), ValErr(61.9, 1.5) ],
        [ 3, ValErr(46, 2), ValErr(49.7, 2.2) ],
        [ 4, ValErr(30, 2), ValErr(43.2, 2.9) ],
        [ 5, ValErr(20, 2), ValErr(36, 4) ],
        [ 6, ValErr(15, 2), ValErr(34, 5) ],
        [ 7, ValErr(11, 2), ValErr(27, 5) ],
        [ 8, ValErr(9, 2), ValErr(26, 6) ],
        [ 9, ValErr(7, 2), ValErr(23, 7) ],
        [ 10, ValErr(5, 2), ValErr(18, 7) ],
    ]
```

```python
class aufgabe4:
    C = 47 #nF
    freqgang = [
        # R [Ohm], f_r [kHz], U_A [Vrms], U_E [Vrms], Df [kHz]
        [ 1000, ValErr(4.02, 0.02), ValErr(0.64, 0.02), ValErr(0.661, 0.001), ValErr(4.92, 0.03) ],
        [ 220, ValErr(3.80, 0.02), ValErr(0.53, 0.02), ValErr(0.650, 0.001), ValErr(1.29, 0.03) ],
        [ 47, ValErr(3.75, 0.02), ValErr(0.27, 0.02), ValErr(0.627, 0.001), ValErr(0.56, 0.03) ],
    ]

class aufgabe5:
    f_R = ValErr(3.91, 0.02) * 10**3 # Hz (Schwarz)
    f_L = ValErr(4.04, 0.02) * 10**3 # Hz (Blau)
    f_C = ValErr(3.80, 0.02) * 10**3 # Hz (Rot)

class aufgabe6:
    f = 100 # Hz
    amplitude = [
        # U_P [V], T [ms]
        [ ValErr(0.74, 0.02), ValErr(0.26, 0.02 ) ],
        [ ValErr(0.50, 0.02), ValErr(0.26, 0.02 ) ],
        [ ValErr(0.36, 0.02), ValErr(0.26, 0.02 ) ],
        [ ValErr(0.26, 0.02), ValErr(0.26, 0.02 ) ],
        [ ValErr(0.18, 0.02), ValErr(0.26, 0.02 ) ],
    ]

class aufgabe7:
    f_res = ValErr(3.94, 0.02) * 10**3 # Hz

class aufgabe8t1oF:
    R = 220 # ohm
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-3.06, 0.02), ValErr(100.71, 10) ],
        [ ValErr(-8.06, 0.02), ValErr(3600, 10) ],
        [ ValErr(-22.13, 0.02), ValErr(6790, 10) ]
    ]

class aufgabe8t2hp:
    R = 220 # ohm
    C = 470 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-26.81, 0.02), ValErr(100.71, 10) ],
        [ ValErr(-8.69, 0.02), ValErr(3600, 10) ],
```

```python
        [ ValErr(-22.44, 0.02), ValErr(6790, 10) ]
    ]

class aufgabe8t2tp:
    R = 220 # ohm
    C = 470 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-2.75, 0.02), ValErr(100.71, 10) ],
        [ ValErr(-15.88, 0.02), ValErr(3600, 10) ],
        [ ValErr(-51.19, 0.02), ValErr(6790, 10) ]
    ]

class aufgabe8t2lctp:
    C = 470 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-2.56, 0.02), ValErr(100.71, 10) ],
        [ ValErr(9.94, 0.02), ValErr(3600, 10) ],
    ]

##############

class aufgabe8t3bpC1ko:
    R = 1 # kOhm
    C = 47 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-3.19, 0.02), ValErr(100.71, 10) ],
        [ ValErr(-8.81, 0.02), ValErr(3590, 10) ],
    ]

class aufgabe8t3bpC47o:
    R = 47 # Ohm
    C = 47 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-2.87, 0.02), ValErr(100.71, 10) ],
        [ ValErr(8.06, 0.02), ValErr(3590, 10) ],
    ]

##############

class aufgabe8t3bpL1ko:
    R = 1 # kOhm
    C = 47 # nF
    sig = [
```

```
        # U [dBV], f [Hz]
        [ ValErr(-11.50, 0.02), ValErr(3600, 10) ],
        [ ValErr(-22.75, 0.02), ValErr(6790, 10) ],
    ]

class aufgabe8t3bpL47o:
    R = 47 # Ohm
    C = 47 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(6.81, 0.02), ValErr(3590, 10) ],
        [ ValErr(-22.25, 0.02), ValErr(6790, 10) ],
    ]

##############

class aufgabe8t3bpR1ko:
    R = 1 # kOhm
    C = 47 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-32.44, 0.02), ValErr(100.7, 10) ],
        [ ValErr(-8.06, 0.02), ValErr(3590, 10) ],
        [ ValErr(-43.38, 0.02), ValErr(6800, 10) ],
    ]

class aufgabe8t3bpR47o:
    R = 47 # Ohm
    C = 47 # nF
    sig = [
        # U [dBV], f [Hz]
        [ ValErr(-57.56, 0.02), ValErr(100.71, 10) ],
        [ ValErr(-18.50, 0.02), ValErr(3590, 10) ],
    ]
```

0.0.1 Aufgabe 1

[4]:
```
# Aufgabe 1
# Zeitkonstanten der RC-Kombinationen berechnen und in Tabelle Eintragen
# C, R, f, _exp, _theo

for dat_1 in aufgabe1.halbw[0:3]:
    C_a1 = ValErr.fromValPerc(dat_1[0], 10)
    R_a1 = ValErr.fromValPerc(dat_1[1], 5)

    tau_theo = C_a1 * 10**(-9) * R_a1 * 10**3
```

```
    tau_exp = dat_1[2] * 10**(-6) / np.log(2)

    diff = np.abs(tau_theo.val - tau_exp.val) / np.sqrt(tau_theo.err**2 +␣
↪tau_exp.err**2)

    print_all(C_a1.strfmtf2(2, 0, 'C'), R_a1.strfmtf2(2, 0, 'R'), aufgabe1.f.
↪strfmtf2(2, 0, 'f'), tau_theo.strfmtf2(5, -5, ' _theo'), tau_exp.strfmtf2(5,␣
↪-5, ' _exp'), tau_theo.sigmadiff_fmt(tau_exp), '---')
```

```
C = 470.00 ± 47.00
R = 1.00 ± 0.05
f = 165.00 ± 1.00
 _theo = (47.00000 ± 5.25476)e-5
 _exp = (43.28085 ± 14.42695)e-5
0.25
---
C = 4.70 ± 0.47
R = 10.00 ± 0.50
f = 165.00 ± 1.00
 _theo = (4.70000 ± 0.52548)e-5
 _exp = (6.49213 ± 0.14427)e-5
3.29
---
C = 47.00 ± 4.70
R = 1.00 ± 0.05
f = 165.00 ± 1.00
 _theo = (4.70000 ± 0.52548)e-5
 _exp = (4.90516 ± 0.14427)e-5
0.38
---
```

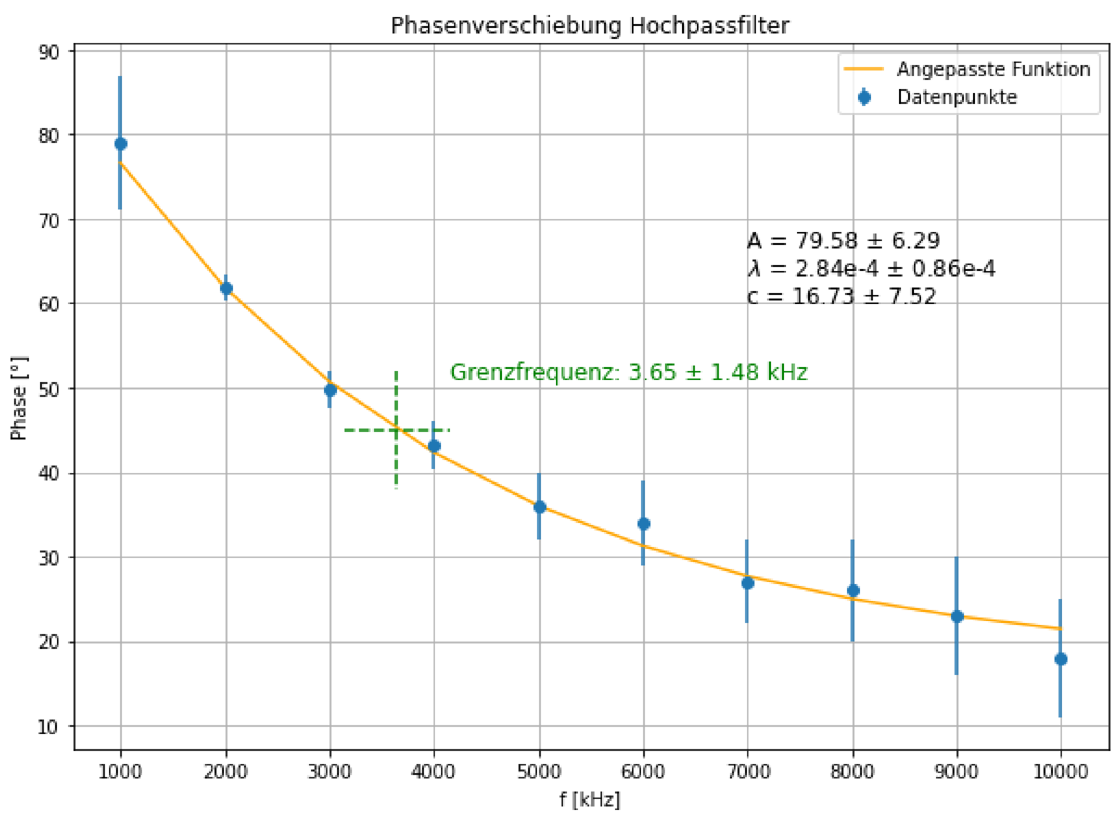
0.0.2 Aufgabe 3

[5]:
```
# f [kHz], Dt [µs], phi [°]

f_val = np.array([x[0] * 1000 for x in aufgabe3hp.phase_shift])
phi_val = np.array([x[2].val for x in aufgabe3hp.phase_shift])
phi_err = np.array([x[2].err for x in aufgabe3hp.phase_shift])

def fit_func_exp(x, A, lam, c):
    return A * np.exp(- lam * x) + c

def inv_func_exp(y, A, lam, c):
    return - (1./lam) * np.log((y - c) / A)

def inv_func_exp_err(y, A, lam, c, dA, dlam, dc):
    log_term = np.log((y - c) / A)
```

```
    dx_A = (1 / (lam * A)) * log_term * dA
    dx_lambda = (1 / lam**2) * log_term * dlam
    dx_c = (1 / (lam * (y - c))) * dc

    dx = np.sqrt(dx_A**2 + dx_lambda**2 + dx_c**2)
    return dx

p0_phase = [80, 0, 15]

popt_phase, pcov_phase = curve_fit(fit_func_exp, f_val, phi_val, p0_phase,␣
 ↪sigma=phi_err, absolute_sigma=True)

A_fit = ValErr.fromFit(popt_phase, pcov_phase, 0)
lam_fit = ValErr.fromFit(popt_phase, pcov_phase, 1)
c_fit = ValErr.fromFit(popt_phase, pcov_phase, 2)

grenzfreq_val = inv_func_exp(45, A_fit.val, lam_fit.val, c_fit.val)
grenzfreq_err = inv_func_exp_err(45,A_fit.val, lam_fit.val, c_fit.val, A_fit.
 ↪err, lam_fit.err, c_fit.err)
grenzfreq = ValErr(grenzfreq_val, grenzfreq_err)

print(f'Interpolierte Grenzfrequenz: {grenzfreq}')
print(f'Abweichung Grenzfrequenz: {aufgabe3hp.f_intersect.
 ↪sigmadiff_fmt(grenzfreq, 5)}')

plt.figure(figsize=(10,7))
plt.errorbar(f_val, phi_val, yerr=phi_err, fmt='o', label='Datenpunkte')
plt.plot(f_val, fit_func_exp(f_val, *popt_phase), color='orange',␣
 ↪label='Angepasste Funktion')
plt.plot([grenzfreq.val - 500, grenzfreq.val + 500], [45, 45], linestyle='--',␣
 ↪color='green')
plt.plot([grenzfreq.val, grenzfreq.val], [45 + 7, 45 - 7], linestyle='--',␣
 ↪color='green')
plt.ylabel(r'Phase [°]')
plt.xlabel('f [kHz]')
plt.yticks(np.arange(10, 91, 10))
plt.xticks(np.arange(1, 11, 1) * 1000)
plt.title(r'Phasenverschiebung Hochpassfilter')
plt.text(grenzfreq.val + 500, 45 + 6, f'Grenzfrequenz: {(grenzfreq * 10**(-3)).
 ↪strfmtf(2, 0)} kHz', color='green', size='large')
plt.grid()

plt.text(7000, 60, f'A = {A_fit.strfmtf(2, 0)}\n$\lambda$ = {lam_fit.strfmtf(2,␣
 ↪-4)}\nc = {c_fit.strfmtf(2, 0)}', size='large')

plt.legend(loc='upper right')
```

```
plt.savefig('phaseshift_hp_fit.png', format='png')

# theoretische werte + vergleich

# 1 / RC

C_a3 = ValErr.fromValPerc(47, 10) * 10 **(-9)
R_a3 = ValErr.fromValPerc(1, 5) * 10**(3)

w_G = (1 / ((C_a3 * R_a3) * (2 * np.pi))) * 10 ** (-3) #kHz

print(w_G , aufgabe3hp.f_intersect.sigmadiff_fmt(w_G), grenzfreq.
↪sigmadiff_fmt(w_G))
```

```
Interpolierte Grenzfrequenz: ValErr(3648.3123494678557, 1476.5500751470736)
Abweichung Grenzfrequenz: 2.4686
ValErr(3.386275384933943, 0.3785970975623283) 0.21  2.47
```




```
[ ]:
```

0.0.3 Aufgabe 4

[6]:
```
Ls = []

r47vals = dict()

cap4 = ValErr.fromValPerc(aufgabe4.C, 10) * 10**(-9)

for i in range(0, len(aufgabe4.freqgang)):
    dat4 = aufgabe4.freqgang[i]

    freq4 = dat4[1] * 1000

    L_val = 1 / ((2 * np.pi * freq4.val)**2 * cap4.val)
    L_err = L_val * np.sqrt((2 * freq4.err / freq4.val)**2 + (cap4.err / cap4.
 ↪val)**2)
    L = ValErr(L_val, L_err)
    Ls.append(L)

    print(L.strfmtf2(3, -2, f'L({i+1})'))

L_mean = ValErr(np.mean([x.val for x in Ls]), (1/len(Ls)) * np.sqrt(np.sum([x.
 ↪err**2 for x in Ls])))

print_all('=> L Mittelwert', L_mean.strfmtf2(3, -2, 'L'), '###########')

#

for i in range(0, len(aufgabe4.freqgang)):
    dat4 = aufgabe4.freqgang[i]

    freq4 = dat4[1] * 1000
    Dfreq4 = dat4[4] * 1000 * 2 * np.pi
    res4 = ValErr.fromValPerc(dat4[0], 5)
    U_A = dat4[2]
    U_E = dat4[3]


    RpRv = Dfreq4 * L_mean

    RpRv_amp = res4 * (U_E/U_A)

    print_all('Aus Bandbreite', Dfreq4.strfmtf2(3, 0, 'Δ '), RpRv.strfmtf2(3,␣
 ↪0, 'R + R_v'), res4.strfmtf2(3, 0, 'R'), (RpRv - res4).strfmtf2(3, 0,␣
 ↪'R_v'), '----')
```

```
    print_all('Aus Amplituden', U_E.strfmtf2(3, 0, 'U_E'), U_A.strfmtf2(3, 0,␣
↪'U_A'), RpRv_amp.strfmtf2(3, 0, 'R + R_v'), (RpRv_amp - res4).strfmtf2(3, 0,␣
↪'R_v'), (RpRv - res4).sigmadiff_fmt((RpRv_amp - res4), 3), '####')

    if res4.val == 47:
        r47vals['RpRv'] = RpRv
```

```
L(1) = (3.335 ± 0.335)e-2
L(2) = (3.732 ± 0.375)e-2
L(3) = (3.832 ± 0.385)e-2
=> L Mittelwert
L = (3.633 ± 0.211)e-2
###########
Aus Bandbreite
Δ  = 30913.272 ± 188.496
R + R_v = 1123.154 ± 65.668
R = 1000.000 ± 50.000
R_v = 123.154 ± 82.537
----
Aus Amplituden
U_E = 0.661 ± 0.001
U_A = 0.640 ± 0.020
R + R_v = 1032.813 ± 60.917
R_v = 32.813 ± 78.809
0.792
####
Aus Bandbreite
Δ  = 8105.309 ± 188.496
R + R_v = 294.485 ± 18.443
R = 220.000 ± 11.000
R_v = 74.485 ± 21.474
----
Aus Amplituden
U_E = 0.650 ± 0.001
U_A = 0.530 ± 0.020
R + R_v = 269.811 ± 16.907
R_v = 49.811 ± 20.170
0.838
####
Aus Bandbreite
Δ  = 3518.584 ± 188.496
R + R_v = 127.839 ± 10.108
R = 47.000 ± 2.350
R_v = 80.839 ± 10.377
----
Aus Amplituden
```

```
U_E = 0.627 ± 0.001
U_A = 0.270 ± 0.020
R + R_v = 109.144 ± 9.756
R_v = 62.144 ± 10.035
1.296
####
```

0.0.4 Aufagbe 5

[7]:
```
dekrs = []
dekrs_err = []

for i in range(0, len(aufgabe6.amplitude) - 1):
    U_P1 = aufgabe6.amplitude[i][0]
    U_P2 = aufgabe6.amplitude[i+1][0]
    d = np.log(U_P1.val / U_P2.val)
    d_err = np.sqrt(U_P1.relerr()**2 + U_P2.relerr()**2)
    dekrs.append(d)
    dekrs_err.append(d_err)
    print(U_P1.strfmtf(4, 0, 'Λ(n)'), U_P2.strfmtf(4, 0, 'Λ(n+1)'), f'\tΛ_{i+1}␣
↪=', d, d_err)

dekrement = ValErr(np.mean(dekrs), (1/len(dekrs_err)) * np.sqrt(np.sum([x**2␣
↪for x in dekrs_err])))
print(dekrement.strfmtf(4, 0, 'Λ'))

T = ValErr(0.26, 0.02) * 10**(-3)

RpRv_dekr_val = (2 * L_mean.val * dekrement.val) / T.val
RpRv_dekr_err = RpRv_dekr_val * np.sqrt(np.sum([L_mean.relerr()**2, dekrement.
↪relerr()**2, T.relerr()**2]))

RpRv_dekr = ValErr(RpRv_dekr_val, RpRv_dekr_err)

# Λ =  T,   = R/2L => Λ = TR/2L <=> R = 2LΛ/T

print_all(RpRv_dekr.strfmtf(4, 0, 'R + R_v'), 'vorher:', r47vals['RpRv'].
↪strfmtf(4, 0, 'R + R_v'), 'abw', RpRv_dekr.sigmadiff_fmt(r47vals['RpRv'], 3))
```

```
Λ(n) = 0.7400 ± 0.0200 Λ(n+1) = 0.5000 ± 0.0200         Λ_1 = 0.3920420877760237
0.048274840133548345
Λ(n) = 0.5000 ± 0.0200 Λ(n+1) = 0.3600 ± 0.0200         Λ_2 =
0.32850406697203605 0.06845743022555273
Λ(n) = 0.3600 ± 0.0200 Λ(n+1) = 0.2600 ± 0.0200         Λ_3 =
0.32542240043462795 0.09488719363749794
Λ(n) = 0.2600 ± 0.0200 Λ(n+1) = 0.1800 ± 0.0200         Λ_4 = 0.3677247801253175
0.13514007094736666
```

```
Λ = 0.3534 ± 0.0463
R + R_v = 98.7748 ± 16.0650
vorher:
R + R_v = 127.8387 ± 10.1075
abw
1.532
```

0.0.5 Aufgabe 6

[8]:
```
# w_R = sqrt(1/LC)
#   = R / 2L
# w_C = sqrt(w_R^2 - 2 ^2)
# w_L = sqrt(w_R^2 + 2 ^2)

cap6 = ValErr.fromValPerc(47, 10) * 10**(-9)
res6 = ValErr.fromValPerc(220, 5)
ind6 = L_mean

LC6 = ind6*cap6

wR6_val = np.sqrt(1 / LC6.val)
wR6_err = (1 / 2) * (LC6.err / LC6.val**(3/2))
wR6 = ValErr(wR6_val, wR6_err)
fR6 = wR6 / (2*np.pi)

delta6 = (1 / 2) * (res6 / ind6)

wC6_val = np.sqrt(wR6.val**2 - 2*delta6.val**2)
wC6_err = np.sqrt(((wR6_val*wR6_err)/wC6_val)**2 + ((delta6.val*delta6.err)/
 ↪wC6_val)**2)
wC6 = ValErr(wC6_val, wC6_err)
fC6 = wC6 / (2*np.pi)

wL6_val = np.sqrt(wR6.val**2 + 2*delta6.val**2)
wL6_err = np.sqrt(((wR6_val*wR6_err)/wL6_val)**2 + ((delta6.val*delta6.err)/
 ↪wL6_val)**2)
wL6 = ValErr(wL6_val, wL6_err)
fL6 = wL6 / (2*np.pi)


print_all(delta6.strfmtf(4, 0, ' '),
          '---',
          fR6.strfmtf(4, 0, ' _R'),
          aufgabe5.f_R.strfmtf(4, 0, ' _R_exp'),
          aufgabe5.f_R.sigmadiff_fmt(fR6, 5),
          '---',
          fC6.strfmtf(4, 0, ' _C'),
```

```
          aufgabe5.f_C.strfmtf(4, 0, ' _C_exp'),
          aufgabe5.f_C.sigmadiff_fmt(fC6, 5),
          '---',
          fL6.strfmtf(4, 0, ' _L'),
          aufgabe5.f_L.strfmtf(4, 0, ' _L_exp'),
          aufgabe5.f_L.sigmadiff_fmt(fL6, 5))
```

```
 = 3027.5991 ± 232.1858
---
 _R = 3851.4480 ± 222.7632
 _R_exp = 3910.0000 ± 20.0000
0.2618
---
 _C = 3790.6832 ± 226.3829
 _C_exp = 3800.0000 ± 20.0000
0.041
---
 _L = 3911.2690 ± 219.4034
 _L_exp = 4040.0000 ± 20.0000
0.58431
```

0.0.6 Aufagbe 7

[9]:
```
cap7 = ValErr.fromValPerc(47, 10) * 10**(-9)
ind7 = L_mean

# w_0 = 1 / sqrt(LC)

LC7 = cap7*ind7

w07_val = np.sqrt(1 / LC7.val)
w07_err = (1 / 2) * (LC7.err / LC7.val**(3/2))
w07 = ValErr(w07_val, w07_err)
f07 = w07 / (2*np.pi)

print_all(aufgabe7.f_res.strfmtf(4, 0, 'f_0_exp'),
          f07.strfmtf(4, 0, 'f_0_theo'),
          f'abw {aufgabe7.f_res.sigmadiff_fmt(f07, 4)}')
```

```
f_0_exp = 3940.0000 ± 20.0000
f_0_theo = 3851.4480 ± 222.7632
abw 0.396
```

0.0.7 Aufgabe 8

[10]:
```python
print('RLC, 47 ohm, Widerstand')
print('4kHz', aufgabe8t1oF.sig[1][0], aufgabe8t3bpR47o.sig[1][0], aufgabe8t1oF.
 ↪sig[1][0]/aufgabe8t3bpR47o.sig[1][0])
print('100Hz', aufgabe8t1oF.sig[0][0], aufgabe8t3bpR47o.sig[0][0], aufgabe8t1oF.
 ↪sig[0][0]/aufgabe8t3bpR47o.sig[0][0])

print('\n', 'RC Hochpass')
print('4kHz', aufgabe8t1oF.sig[1][0], aufgabe8t2hp.sig[1][0], aufgabe8t1oF.
 ↪sig[1][0]/aufgabe8t2hp.sig[1][0])
print('8kHz', aufgabe8t1oF.sig[2][0], aufgabe8t2hp.sig[2][0], aufgabe8t1oF.
 ↪sig[2][0]/aufgabe8t2hp.sig[2][0])
print('100Hz', aufgabe8t1oF.sig[0][0], aufgabe8t2hp.sig[0][0], aufgabe8t1oF.
 ↪sig[0][0]/aufgabe8t2hp.sig[0][0])

print('\n', 'RC Tiefpass')
print('4kHz', aufgabe8t1oF.sig[1][0], aufgabe8t2tp.sig[1][0], aufgabe8t1oF.
 ↪sig[1][0]/aufgabe8t2tp.sig[1][0])
print('8kHz', aufgabe8t1oF.sig[2][0], aufgabe8t2tp.sig[2][0], aufgabe8t1oF.
 ↪sig[2][0]/aufgabe8t2tp.sig[2][0])
print('100Hz', aufgabe8t1oF.sig[0][0], aufgabe8t2tp.sig[0][0], aufgabe8t1oF.
 ↪sig[0][0]/aufgabe8t2tp.sig[0][0])
```

```
RLC, 47 ohm, Widerstand
4kHz ValErr(-8.06, 0.02) ValErr(-18.5, 0.02) ValErr(0.4356756756756757,
0.0011792277099718577)
100Hz ValErr(-3.06, 0.02) ValErr(-57.56, 0.02) ValErr(0.05316191799861014,
0.00034795416878131134)

 RC Hochpass
4kHz ValErr(-8.06, 0.02) ValErr(-8.69, 0.02) ValErr(0.9275028768699656,
0.0031390427356231265)
8kHz ValErr(-22.13, 0.02) ValErr(-22.44, 0.02) ValErr(0.9861853832442067,
0.0012517639252519503)
100Hz ValErr(-3.06, 0.02) ValErr(-26.81, 0.02) ValErr(0.11413651622528907,
0.0007508336411163078)

 RC Tiefpass
4kHz ValErr(-8.06, 0.02) ValErr(-15.88, 0.02) ValErr(0.5075566750629723,
0.001412385144501711)
8kHz ValErr(-22.13, 0.02) ValErr(-51.19, 0.02) ValErr(0.4323109982418441,
0.0004256480161859309)
100Hz ValErr(-3.06, 0.02) ValErr(-2.75, 0.02) ValErr(1.1127272727272728,
0.01088035483979233)
```

[ ]:
```python

```