Grundlagen der Elektrotechnik – Prüfung 25.06.2018

Andre Mitterbacher

Die Musterlösung ist mit Hilfe von Python in einem Jupyter Notebook gerechnet. Alle Ergebnisse sind ohne Gewähr.

Aufgabe 1 Resonanz

Kann Resonanz auftreten?

Ja, da zwei verschiedenartige Energiespeicher vorhanden sind (L1, C1).

Auslegen von C1 für Resonanz

Bei Resonanz ist der Blindanteil von $\underline{Y}$ bzw. $\underline{Z}$ gleich Null.

$$\underline{Z} = (R + j\omega L) \,||\, \frac{1}{j\omega C}$$

$$\underline{Y} = \frac{1}{R+j\omega L} + j\omega C$$

Der Ausdruck für $\underline{Y}$ scheint einfacher zu sein... $\underline{Y} = \frac{1}{R+j\omega L} + j\omega C = \frac{R-j\omega L}{R^2+\omega^2 L^2} + j\omega C$

Damit ergibt sich der Blindleitwert (Suszeptanz) zu:

$$\frac{-\omega L}{R^2+\omega^2 L^2} + \omega C$$

Nullsetzen der Suszeptanz ergibt:

$$\frac{-\omega L}{R^2+\omega^2 L^2} + \omega C = 0$$

$$\omega L = \omega C \cdot (R^2 + \omega^2 L^2)$$

$$L = C \cdot (R^2 + \omega^2 L^2)$$
$$L = CR^2 + \omega^2 L^2 C$$

$$\omega^2 = \frac{L-CR^2}{L^2 C}$$

$$\omega = \sqrt{\frac{L-CR^2}{L^2 C}} = \sqrt{\frac{1}{LC} - \frac{R^2}{L^2}}$$

Um C für eine Resonanz bei f_{res} zu dimensionieren kann nach C gelöst werden:

$$C = \frac{L}{R^2+\omega^2 L^2}$$

In [1]:
```
import math as math
import cmath as cmath
import numpy as np
import matplotlib.pyplot as plt

def print_c(x):
    return  " = %.3g+j%.3g = %.3g/_%.3g" % (x.real,x.imag,abs(x),(cmath.phase
j   = 1j
pi  = math.pi
R1  = 1e-3    #ohm
L1  = 10e-9 #H
U   = 3.     #V
phi = 0.   #deg
Uc  = U * np.exp(j*phi)
print("U = " + print_c(Uc))
fres = 5e6;
print("fres = %.6g Hz" % fres)
wres = (2*pi*fres)
# Calculation of C1
C1 = L1 / ( R1**2 + wres**2 * L1**2 )
print("C1 = %.6g F" % C1)
# Check ob der Blindleitwert auch wirklich Null ist
Bres = -wres*L1/(R1**2 + wres**2 * L1**2) + wres*C1
print("Bres = %.6g" % Bres)
# Check ob die Resonanzfrequenz stimmt
wres_check = math.sqrt( (L1 - C1*R1**2)/(L1**2*C1) )
fres_check = wres_check/(2*pi)
print("fres_check = %.6g" % fres_check)
```

```
U =  = 3+j0 = 3/_0
fres = 5e+06 Hz
C1 = 1.0132e-07 F
Bres = 0
fres_check = 5e+06
```

Berechnung des Stroms im Resonanzfall

Für $f = f_{res}$ ist nur noch der Wirkanteil des komplexen Leitwerts wirksam.

$$Y_{res} = \frac{R}{R^2+\omega_{res}^2 L^2}$$

Der Strom kann dann mit dem ohm'schen Gesetz berechnet werden: $I_{res} = U \cdot Y_{res}$

Nachdem Y eine reele Zahl ist, kann nur mit den Beträgen gerechnet werden. Die Phasenlage des Stroms entspricht der der Spannung.

In [2]:
```
Yres = R1 / ( R1**2 + wres**2 * L1**2 )
print("Yres = %.6g S" % Yres)
Ires = Uc * Yres
print("Ires = " + print_c(Ires))
```

```
Yres = 0.010132 S
Ires =  = 0.0304+j0 = 0.0304/_0
```

Aufgabe 2

In [3]:
```
%reset
import math as math
import cmath as cmath
import numpy as np
import matplotlib.pyplot as plt

def print_c(x):
    return  " = %.6g+j%.6g = %.6g/_%.6g" % (x.real,x.imag,abs(x),(cmath.phase
j   = 1j
pi  = math.pi
R1  = 100.    #ohm
R2  = 200.    #ohm
L2  = 318.30988e-3  #H
L3  = 318.30988e-3  #H

U1   = 38.    #V
phi1 = 0.     #rad
U1c  = U1 * np.exp(j*phi1)
print("U1 = " + print_c(U1c))
U2   = 38.    #V
phi2 = 120./180*pi     #rad
U2c  = U2 * np.exp(j*phi2)
f = 50.       #Hz
w = 2*pi*f    #1/s
print("U2 = " + print_c(U2c))
ZL2  = j*w*L2
ZL3  = j*w*L3
print("ZL2 = " + print_c(ZL2))
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
U1 =  = 38+j0 = 38/_0
U2 =  = -19+j32.909 = 38/_120
ZL2 =  = 0+j100 = 100/_90
ZL3 =  = 0+j100 = 100/_90
```

ZL2 und ZL3 auf $+j100\Omega$ runden:

In [4]:
```
ZL1 = j*100.
ZL2 = j*100.
print("ZL2 = " + print_c(ZL2))
```

```
ZL2 =  = 0+j100 = 100/_90
ZL3 =  = 0+j100 = 100/_90
```

Maschenstromverfahren

Zwei Maschen jeweils im Uhrzeigersinn.

In [5]:
```
Rmat= np.array([[R1+R2+ZL2, -(R2+ZL2)],[-(R2+ZL2), R2+ZL2+ZL3]])
print("Widerstandsmatrix =")
print(Rmat)

Uvec = np.array([[U1c],[-U2c]])
print("Spannunsvektor:")
print(Uvec)
```

```
Widerstandsmatrix =
[[ 300.+100.        j -200.-100.        j]
 [-200.-100.        j  200.+199.99999806j]]
Spannunsvektor:
[[38. +0.        j]
 [19.-32.90896534j]]
```

Lösen des linearen Gleichungssystem $Rmat \cdot Ivec = Uvec$:

In [6]:
```
Ivec=np.linalg.solve(Rmat,Uvec)
print("Maschenströme:")
print(Ivec)
```

```
Maschenströme:
[[ 0.15508073-0.32850223j]
 [-0.00058742-0.41491927j]]
```

In [7]:
```
Im1 = Ivec[0][0]
Im2 = Ivec[1][0]
print("Im1 = " + print_c(Im1))
print("Im2 = " + print_c(Im2))
```

```
Im1 =  = 0.155081+j-0.328502 = 0.363268/_-64.7287
Im2 =  = -0.00058742+j-0.414919 = 0.41492/_-90.0811
```

Die gesuchten Größen können nun aus den Maschengrößen berechnet werden:

- IL2 = Im1 - Im2
- IR1 = Im1
- UL3 = ZL3*Im2

In [8]:
```
IR1 = Im1
print("IR1 = " + print_c(IR1) + " A")
IL2 = Im1 - Im2
print("IL2 = " + print_c(IL2) + " A")
UL3 = Im2 * ZL3
print("UL3 = " + print_c(UL3) + " V")
```

```
IR1 =  = 0.155081+j-0.328502 = 0.363268/_-64.7287 A
IL2 =  = 0.155668+j0.086417 = 0.178046/_29.0362 A
UL3 =  = 41.4919+j-0.0587419 = 41.492/_-0.0811161 V
```

Mit Überlagerungssatz

Erster Fall: U1 aktiv, U2 passiv

In [9]:
```
ZR2L2_A = R2 + ZL2
ZR2L2L3_A = ZR2L2_A * ZL3 / (ZR2L2_A + ZL3)
Zges_A = ZR2L2L3_A + R1

print("ZR2L2_A = " + print_c(ZR2L2_A))
print("ZR2L2L3_A = " + print_c(ZR2L2L3_A))
print("Zges_A = " + print_c(Zges_A))
#----------------------------
IR1_A = U1c / Zges_A
IL2_A = IR1_A * ZL3 / (ZL3 + ZR2L2_A)
IL3_A = IR1_A - IL2_A
UL3_A = IL3_A * ZL3

print("IR1_A = " + print_c(IR1_A))
print("IL2_A = " + print_c(IL2_A))
print("IL3_A = " + print_c(IL3_A))
print("UL3_A = " + print_c(UL3_A))
```

```
ZR2L2_A =  = 200+j100 = 223.607/_26.5651
ZR2L2L3_A =  = 25+j75 = 79.0569/_71.5651
Zges_A =  = 125+j75 = 145.774/_30.9638
IR1_A =  = 0.223529+j-0.134118 = 0.260678/_-30.9638
IL2_A =  = 0.0894118+j0.0223529 = 0.0921635/_14.0362
IL3_A =  = 0.134118+j-0.156471 = 0.206084/_-49.3987
UL3_A =  = 15.6471+j13.4118 = 20.6084/_40.6013
```

Zweiter Fall: U1 passiv, U2 aktiv

Achtung auf die Strom- und Spannungsrichtung!

In [10]:
```
ZR2L2_B = R2 + ZL2
ZR1R2L2_B = ZR2L2_B * R1 / (ZR2L2_B + R1)
Zges_B = ZR1R2L2_B + ZL3

print("ZR2L2_B = " + print_c(ZR2L2_B))
print("ZR1R2L2_B = " + print_c(ZR1R2L2_B))
print("Zges_B = " + print_c(Zges_B))
#----------------------------
IL3_B = U2c / Zges_B
UL3_B = IL3_B * ZL3
IR1_B = IL3_B * ZR2L2_B / (R1 + ZR2L2_B)
IL2_B = IL3_B - IR1_B

print("IL3_B = " + print_c(IL3_B) + " Achtung andere Stromrichtung")
print("UL3_B = " + print_c(UL3_B) + " Achtung andere Spannungsrichtung")
print("IR1_B = " + print_c(IR1_B) + " Achtung andere Stromrichtung")
print("IL2_B = " + print_c(IL2_B) + " hier passt die Richtung")
```

```
ZR2L2_B =  = 200+j100 = 223.607/_26.5651
ZR1R2L2_B =  = 70+j10 = 70.7107/_8.1301
Zges_B =  = 70+j110 = 130.384/_57.5288
IL3_B =  = 0.134705+j0.258449 = 0.291447/_62.4712 Achtung andere Stromrich
tung
UL3_B =  = -25.8449+j13.4705 = 29.1447/_152.471 Achtung andere Spannungsri
chtung
IR1_B =  = 0.0684487+j0.194385 = 0.206084/_70.6013 Achtung andere Stromric
htung
IL2_B =  = 0.0662564+j0.0640641 = 0.0921635/_44.0362 hier passt die Richtu
ng
```

Jetzt richtig überlagern (Richtung beachten):

$$IR1 = IR1_A - IR1_B$$

$$IL2 = IL2_A + IL2_B$$

$$UL3 = UL3_A - UL3_B$$

In [11]:
```
IR1_H = IR1_A - IR1_B
IL2_H = IL2_A + IL2_B
UL3_H = UL3_A - UL3_B
print("IR1 Helmholtz = " + print_c(IR1_H))
print("IL2 Helmholtz = " + print_c(IL2_H))
print("UL3 Helmholtz = " + print_c(UL3_H))
```

```
IR1 Helmholtz =  = 0.155081+j-0.328502 = 0.363268/_-64.7287
IL2 Helmholtz =  = 0.155668+j0.086417 = 0.178046/_29.0362
UL3 Helmholtz =  = 41.4919+j-0.0587419 = 41.492/_-0.0811161
```

Aufgabe 3 - Leistung im Wechselstromkreis

Berechnung von $\underline{I}_1$

Der Gesamtwiderstand ergibt sich $\underline{Z} = \frac{R2 \cdot j\omega L1}{R2 + j\omega L1} - j\frac{1}{\omega C} + R1$

Der Strom $\underline{I}_{R1} = \frac{\underline{U}}{\underline{Z}}$

In [12]:
```
%reset
import math as math
import cmath as cmath
import numpy as np
import matplotlib.pyplot as plt

def print_c(x):
    return  " = %.6g+j%.6g = %.6g/_%.6g" % (x.real,x.imag,abs(x),(cmath.phase
j   = 1j
pi  = math.pi
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

In [13]:
```
R1 = 100.
R2 = 10.
C1 = 10e-6
L1 = 100e-3
f = 50.
w = 2*pi*f
U = 230.
#--------------------------
Z = (R2*j*w*L1)/(R2 + j*w*L1) - j*1/(w*C1) + R1
print("Z = " + print_c(Z))
```

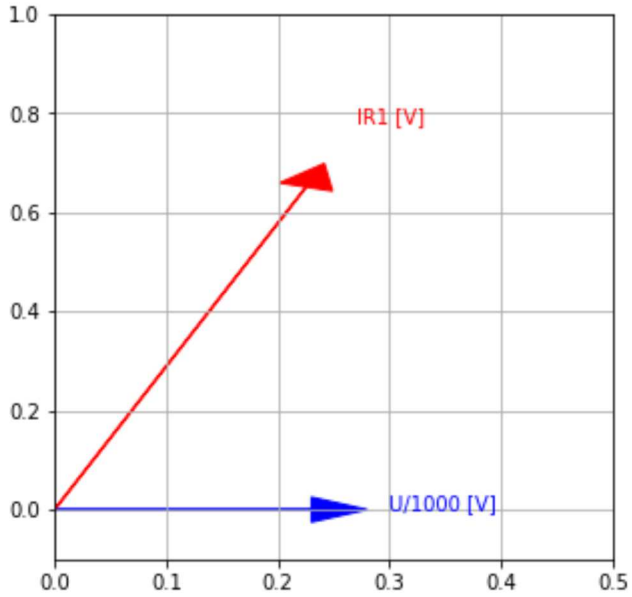
```
Z =  = 109.08+j-315.42 = 333.748/_-70.9233
```

In [14]:
```
IR1 = U / Z
print("IR1 = " + print_c(IR1))
```

```
IR1 =  = 0.225234+j0.651296 = 0.689142/_70.9233
```

Zeigerdiagramm

In [15]:
```
f1 = plt.figure(figsize=(5,5))
ax = plt.axes()
ax.arrow(0, 0, U.real/1000, U.imag/1000, head_width=0.05, head_length=0.05, f
ax.arrow(0, 0, IR1.real/1, IR1.imag/1, head_width=0.05, head_length=0.05, fc=
plt.xlim([0, 0.5])
plt.ylim([-0.1, 1.0])
plt.text(U.real/1000*1.3, U.imag/1000*1.2,"U/1000 [V]", color='b')
plt.text(IR1.real/1*1.2, IR1.imag/1*1.2,"IR1 [V]", color='r')
plt.grid(1)
```



Leistungen

Berechnung von S, P, Q über die komplexe Leistung $\underline{S}$:

$$\underline{S} = \underline{U} \cdot \underline{I_{R1}}^*$$

Danach: $S = |\underline{S}|$, $P = real(\underline{S})$ und $Q = imag(\underline{S})$.

In [16]:
```
Sc = U * IR1.conjugate()
S = abs(Sc)
P = Sc.real
Q = Sc.imag

print("Sc = " + print_c(Sc) + " W")
print("S = " + print_c(S) + " VA")
print("P = " + print_c(P) + " W")
print("Q = " + print_c(Q) + " var")
```

```
Sc =  = 51.8039+j-149.798 = 158.503/_-70.9233 W
S =  = 158.503+j0 = 158.503/_0 VA
P =  = 51.8039+j0 = 51.8039/_0 W
Q =  = -149.798+j0 = 149.798/_180 var
```

Die Augenblicksleistung muss aus den Augenblickswerten u und i_{R1} mit $P(t) = u * i_{R1}$ berechnet werden.

Dazu muss der Zeitverlauf des Strom i_{R1} aus dem komplexen Zeiger gewonnen werden:

- Amplitude = $\sqrt{2} \cdot I_{R1}$ (Wir rechnen ja mit Effektivwertzeigern.)
- Frequenz ist 50Hz (Angabe)
- Nullphasenwinkel ist der Winkel des Zeigers $\underline{I}_{R1}$

In [17]:
```
amp_iR1 = math.sqrt(2)*abs(IR1)
phi_iR1 = np.angle(IR1)

amp_u = math.sqrt(2)*U
phi_u = 0.

print("iR1 = %.6g * cos(2*pi*%.6g*t + %.6g)" % (amp_iR1,f,phi_iR1*180/pi))
```

```
iR1 = 0.974594 * cos(2*pi*50*t + 70.9233)
u   = 325.269 * cos(2*pi*50*t + 0)
```

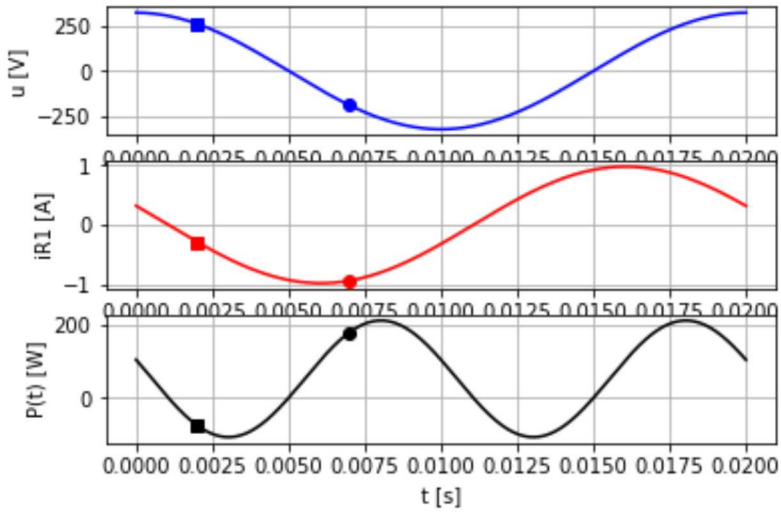
Die Augenblicksleistung soll zu den Zeiten $t1 = 2ms$ und $t2 = 7ms$ berechnet werden. Dazu müssen die Augenblickswert von Spannung und Strom bestimmt werden:

In [18]:
```
t1 = 2e-3
t2 = 7e-3
iR1_t1 = amp_iR1 * math.cos(2*pi*f*t1 + phi_iR1)
u_t1   = amp_u   * math.cos(2*pi*f*t1 + phi_u  )
iR1_t2 = amp_iR1 * math.cos(2*pi*f*t2 + phi_iR1)
u_t2   = amp_u   * math.cos(2*pi*f*t2 + phi_u  )
P_t1   = u_t1 * iR1_t1
P_t2   = u_t2 * iR1_t2
print("iR1(t1) = %.6g A" % iR1_t1)
print("iR1(t2) = %.6g A" % iR1_t2)
print("u(t1) = %.6g V" % u_t1)
print("u(t2) = %.6g V" % u_t2)
print("P(t1) = %.6g W" % P_t1)
```

```
iR1(t1) = -0.283696 A
iR1(t2) = -0.932389 A
u(t1) = 263.148 V
u(t2) = -191.188 V
P(t1) = -74.6542 W
P(t2) = 178.262 W
```

In [19]:
```
t = np.linspace(0,20e-3,10000) #erzeugt einen Zeitvektor
u   = amp_u   * np.cos(2*pi*f*t + 0)
iR1 = amp_iR1 * np.cos(2*pi*f*t + phi_iR1)
Pt = u * iR1

f2 = plt.figure(figsize=(6,4))
plt.subplot(3,1,1)
plt.plot(t,u,t1,u_t1,'s',t2,u_t2,'o',color='b')
plt.ylabel("u [V]")
plt.grid(1)
plt.subplot(3,1,2)
plt.plot(t,iR1,t1,iR1_t1,'s',t2,iR1_t2,'o',color='r')
plt.ylabel("iR1 [A]")
plt.grid(1)
plt.subplot(3,1,3)
plt.plot(t,Pt,t1,P_t1,'s',t2,P_t2,'o',color='k')
plt.ylabel("P(t) [W]")
plt.grid(1)
plt.xlabel("t [s]")
```

Out[19]:
```
Text(0.5,0,'t [s]')
```



Induktiv oder kapazitiv?

Das Netzwerk verhält sich kapazitiv. Das lässt sich anhand folgender Ergebnisse belegen:

- $\angle Z < 0$
- $Q < 0$
- Strom eilt im Zeitverlauf vor.

Aufgabe 4 - Kraft im Magnetfeld

In [20]:
```
%reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

In [25]:
```
import math as math
import numpy as np
import matplotlib.pyplot as plt

pi = math.pi

qe = -1.602e-19 #C
me = 9.11e-31   #kg
a  = 10e-3      #m
b  =  7e-3      #m
d  =  8e-3      #m
B  =  3e-3      #T
```

Kraft bei Eintritt

Die Kraft auf das Elektron lässt sich mit Hilfe der Lorentz-Kraft berechnen:

$$\vec{F}_L = q_e \cdot (\vec{v} \times \vec{B})$$

Im vorliegenden Beispiel sind die Vektoren $\vec{v}$ und $\vec{B}$ rechtwinklig aufeinander.

Damit kann mit Beträgen gerechnet werden. Die Richtung von $\vec{F}$ ergibt sich mit der rechten Hand Regel.

$$F_L = q_e \cdot v \cdot B$$

Die Richtung ist in die in die positive y-Richtung. Durch das negative Vorzeichen der Ladung jedoch in die negative y-Richtung.

In [26]:
```
FL1 = -qe * ve * B
print("FL bei Eintritt = %.6g N nach unten" % FL1)
```

```
FL bei Eintritt = 9.612e-15 N nach unten
```

Bahn im Magnetfeld

Die Bahn im Magnetfeld ist eine Kreisbahn. Der Radius ergibt sich durch Gleichsetzen der Zentrifugalkraft $\vec{F}_Z$ mit der Lorentzkraft $\vec{F}_L$.

$$\vec{F}_Z = \frac{m_e \cdot v_e^2}{r}$$

Setzt man die Beträge gleich, so ergibt sich: $\frac{m_e \cdot v_e^2}{r} = q_e \cdot v_e \cdot B$

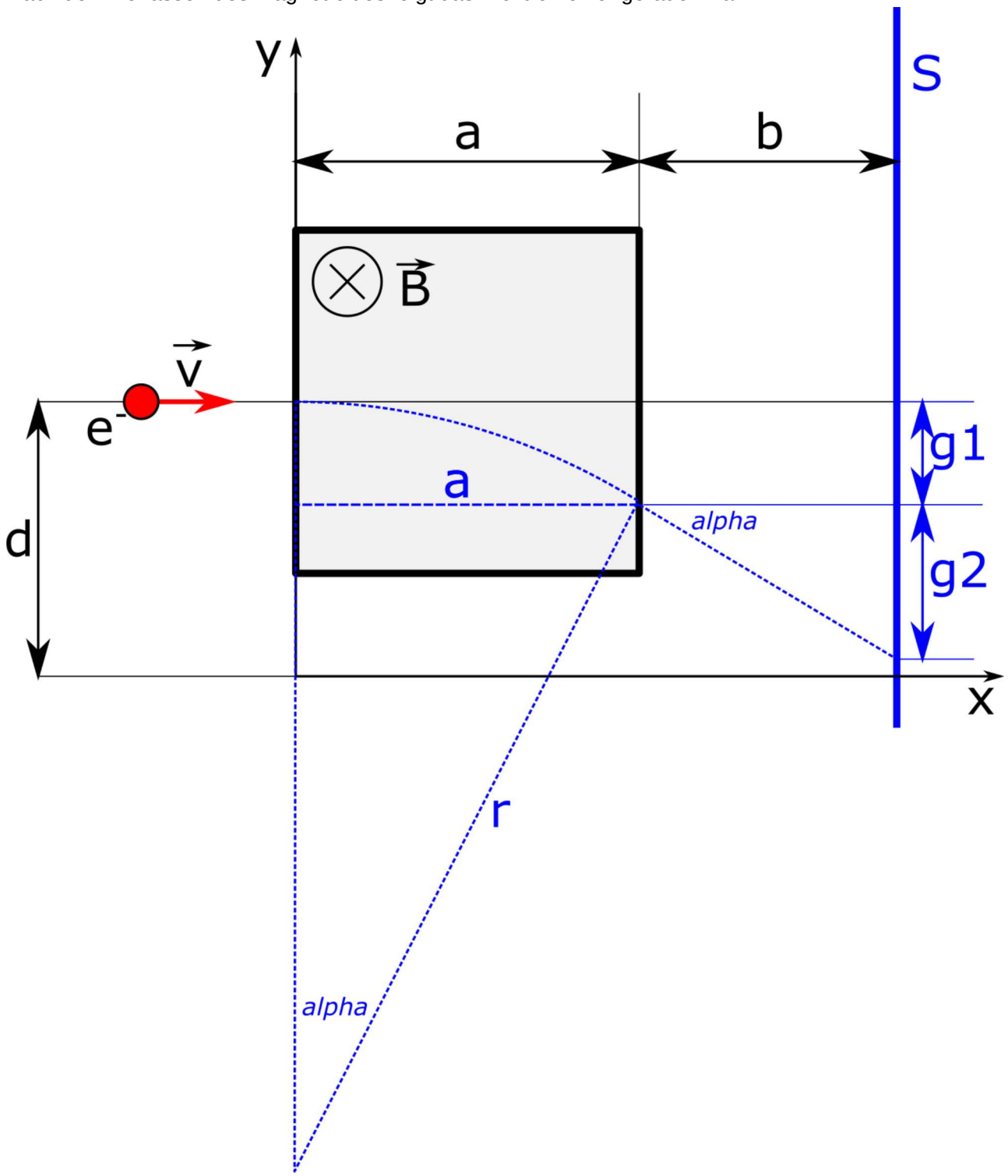
Der Radius ist damit $r = \frac{m_e \cdot v_e}{q_e \cdot B}$

In [27]:
```
r = (me*ve) / (-qe*B)
print("r = %.6g m" % r)
```

```
r = 0.0379109 m
```

Auftrittspunkt

Nach dem Verlassen des Magnetfeldes folgt das Elektron einer geraden Bahn.



Um den Einschlagspunkt am Schirm zu bestimmen, muss der Austrittswinkel α aus dem Magnetfeld berechnet werden.

Der Radius der Kreisbahn und die Länge des Magnetfeldes spannen ein rechtwinkeliges Dreieck auf. Der Winkel des Kreissegments innerhalb des Magnetfelds ist $\alpha = \arcsin(\frac{a}{r})$

Der Austrittspunkt liegt um g_1 unterhalb des Eintrittspunkts. Dieser Abstand kann als $g1 = r - r\cos(\alpha)$ berechnet werden.

In [28]:
```
alpha = math.asin(a/r)
g1 = r - r*math.cos(alpha)
print("alpha = %.6g °" % (alpha*180/pi))
print("g1 = %.6g m" % g1)
```

```
alpha = 15.2942 °
g1 = 0.00134266 m
```

Die weitere Flugbahn kann ebenfalls mit einem rechtwinkeligen Dreieck bestimmt werden. Die Ankathete ist dabei der Abstand b. Der Winkel ist α. Die Gegenkathete $g2$ ist der y-Abstand von der Eintrittsgerade.

$$\tan(\alpha) = \frac{g2}{b}$$

In [29]:
```
g2 = math.tan(alpha)*b
print("g2 = %.6g m" % g2)
```

```
g2 = 0.00191423 m
```

Die Koordinaten (P_x, P_y) des gesuchte Auftrittspunkts P sind damit:

$$P_x = a + b$$

$$P_y = d - g1 - g2$$

In [30]:
```
px = a+b
py = d-g1-g2
print("P = (%.6g,%.6g) mm" %(px*1e3,py*1e3))
```

```
P = (17,4.74312) mm
```

Aufgabe 5 - Induktivität

Es soll mithilfe eines ringförmigen Eisenkerns eine Induktivität von L = 1mH aufgebaut werden. Der Kern besitzt einen mittleren Durchmesser D=1cm. Der Querschnitt A = 0.14cm². Die relative Permeabilität des Kernmaterials beträgt μr = 1000. Die Toleranz der Permeabilität beträgt ±20%. Es wird in den Kern ein kurzer Luftspalt von g=0.5mm eingefräst.

In [31]:
```
%reset
import math as math
import numpy as np
import matplotlib.pyplot as plt
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

In [32]:
```
pi  = math.pi
mu0 = 0.4*pi*1e-6 #Vs/Am
# --- Geometrie ---
L   = 1e-3      #H
D   = 1e-2      #m
A   = 0.14e-4   #m²
g   = 0.5e-3
# --- Kernmaterial ---
mur = 1000.
tol = 0.2
# --- Strom ---
I   = 100e-3    #A
```

Magnetischer Kreis

Der magnetische Kreis besteht aus einer Serienschaltung von Durchflutung Θ mit den magnetischen Widerständen des Eisens R_{me} und der Luftspalts R_{ml}.

Die Induktivität dieser Anordnung ergibt sich als $L = N^2\Lambda$.

Der Summenleitwert Λ ergibt sich aus $\frac{1}{\Lambda} = R_{me} + R_{ml}$.

$R_m = \frac{l}{\mu_0\mu_r A}$

In [33]:
```
le  = D*pi
Rme = le/(mu0*mur*A)
Rml = g/(mu0*1*A)
Lambda = 1./(Rme + Rml)

print("Eisenweg le = %.6g m" % le)
print("Rme = %.6g 1/H" % Rme)
print("Rml = %.6g 1/H" % Rml)
print("Lambda = 1/(Rme+Rml) = %.6g H" % Lambda)
```

```
Eisenweg le = 0.0314159 m
Rme = 1.78571e+06 1/H
Rml = 2.84205e+07 1/H
Lambda = 1/(Rme+Rml) = 3.31057e-08 H
```

Benötigte Windungen

Die benötigten Windungen ergeben sich zu $N = \sqrt{\frac{L}{Lambda}}$

In [34]:
```
N = math.sqrt(L/Lambda)
print("N = %.6g" %N)
```

```
N = 173.799
```

Nachdem nur ganze Windungen gewickelt werden können wird zum Nächsten gerundet.

In [35]:
```
Nround = round(N)
print("N gerundet = %.6g" %Nround)
```

```
N gerundet = 174
```

Maximale Induktivität L_{max}

Aufgrund der Toleranz des Eisenmaterials schwankt R_{me} erheblich!

Der Luftspalt hat aber konstantes μ_r!

Die maximale Induktivität erhält man für $R_{me}^{min} = \frac{l}{\mu_0\mu_r(1+tol)A}$.

In [36]:
```
Rme_min = le/(mu0*mur*(1+tol)*A)
Lambda_max = 1./(Rme_min + Rml)
L_max = Nround**2 * Lambda_max

print("Rme_min = %.6g 1/H" % Rme_min)
print("Lambda_max = 1/(Rme_min+Rml) = %.6g H" % Lambda_max)
print("L_max = %.6g H" % L_max)
```

```
Rme_min = 1.4881e+06 1/H
Lambda_max = 1/(Rme_min+Rml) = 3.34352e-08 H
L_max = 0.00101228 H
```

Flussdichte im Eisenspalt

Mit Hilfe des magnetischen Kreises kann die Flussdichte bestimmt werden.

$\Phi = \Theta \cdot \Lambda$

Die Durchflutung ergibt sich zu $\Theta = N \cdot I$.

Der Fluss ist im Eisen und Luftspalt gleich groß!

In [37]:
```
Theta = Nround * I
Phi = Theta * Lambda

print("Theta = %.6g A" % Theta)
print("Phi = %.6g Vs" % Phi)
```

```
Theta = 17.4 A
Phi = 5.7604e-07 Vs
```

Die Flussdichte ergibt sich aus $\Phi = \int_A \vec{B} d\vec{A}$.

Nachdem der Luftspalt sehr kurz ist, der Fluss senkrecht zum Querschnitt des Toroids verläuft und keine Luftspaltaufweitung angenommen werden darf, kann die Formel vereinfacht werden: $\Phi = B \cdot A$.

In [38]:
```
B = Phi / A

print("B = %.6g T" % B)
```

```
B = 0.0411457 T
```

Aufgabe 7

Energieinhalt eines Kondensators:

$E = \int P dt$

$P = ui$

$i = C\frac{du}{dt}$

Damit: $E = \int uC\frac{du}{dt}dt = \int Cudu = \frac{1}{2}Cu^2$

In [39]:
```
%reset
import math as math
C1 = 10e-6      #F
u1 = 400.       #V

C2 = 100e-6     #F
u2 = 40.        #V

E1 = 0.5 * C1 * u1**2
E2 = 0.5 * C2 * u2**2

print("E1 = %.6g J" % E1)
print("E2 = %.6g J" % E2)
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
E1 = 0.8 J
E2 = 0.08 J
```

Aufgabe 8 - Blindleistungskompensation

Die Spannungsquelle U0 (400V, 50Hz) ist an der dargestellten induktiven Last R1, L1 angeschlossen. R1=10Ω, L1=20mH.

- Berechnen Sie den Leistungsfaktor $\lambda 1$ ohne Blindleistungskompensation (ohne C1).
- Legen Sie danach den Kondensator C1 für einen verbesserten Leistungsfaktor $\lambda 2 = 0.95$ aus.
- Um wie viele Prozent sinkt der Effektivwert des Stroms I0 (den die Quelle liefert) durch die Blindleistungskompensation?

In [40]:
```
%reset
import math as math
import cmath as cmath
import numpy as np
import matplotlib.pyplot as plt

def print_c(x):
    return  " = %.6g+j%.6g = %.6g/_%.6g" % (x.real,x.imag,abs(x),(cmath.phase
j   = 1j
pi  = math.pi
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

Berechnung des Leistungsfaktor ohne Kompensation $\lambda 1$

Viele Möglichkeiten. Eine ist den Strom durch L1, R1 zu berechen und dann die komplexe Leistung zu bestimmen. Daraus erhält man S und P und $\lambda = \frac{P}{S}$

In [41]:
```
U0   = 400.   #V
f    = 50.    #Hz
w    = 2*pi*f #1/s
R1   = 10.    #ohm
L1   = 20e-3  #H

Z1 = R1 + j*w*L1
I1 = U0/Z1
S1c = U0 * I1.conjugate()
S1 = abs(S1c)
P1 = S1c.real
lambda1 = P1/S1
phi1 = np.angle(Z1)

print("Z1 = " + print_c(Z1))
print("I1 = " + print_c(I1))
print("S1c = " + print_c(S1c))
print("S1 = " + print_c(S1))
print("P1 = " + print_c(P1))

print("lambda1 = %.6g" % lambda1)
print("phi1 = %.6g" % (phi1*180/pi))
```

```
Z1 =  = 10+j6.28319 = 11.8101/_32.1419
I1 =  = 28.6783+j-18.0191 = 33.8693/_-32.1419
S1c =  = 11471.3+j7207.64 = 13547.7/_32.1419
S1 =  = 13547.7+j0 = 13547.7/_0
P1 =  = 11471.3+j0 = 11471.3/_0
lambda1 = 0.846733
phi1 = 32.1419
```

Verbesserung auf $\lambda 2 = 0.95$

Zwei Formeln kommen zum Einsatz:

$$Qc = P \cdot (\tan(\varphi 2) - \tan(\varphi 1))$$

$$Qc = -U^2 \cdot \omega C \rightarrow C = -\frac{Qc}{U^2 \cdot \omega}$$

In [42]:
```
lambda2 = 0.95
phi2 = math.acos(lambda2)
Qc = P1 * (math.tan(phi2) - math.tan(phi1))
C  = -Qc/(U0**2 * w)

print("lambda2 = %.6g" % lambda2)
print("phi2 = %.6g°" % (phi2*180/pi))
print("Qc = %.6g var" % Qc)
print("C = %.6g F" % C)
```

```
lambda2 = 0.95
phi2 = 18.1949°
Qc = -3437.2 var
C = 6.83809e-05 F
```

Reduktion des Stroms

Der Strom nach der Kompensation lässt sich mit dem neuen komplexen Widerstand $Z2 = Z1 \| \frac{1}{j\omega C}$ bestimmen:

In [43]:
```
Z2 = (Z1 * 1/(j*w*C) ) / (Z1 + 1/(j*w*C) )
I2 = U0 / Z2

print("Z2 = " + print_c(Z2))
print("I2 = " + print_c(I2))
```

```
Z2 =  = 12.5879+j4.13745 = 13.2504/_18.1949
I2 =  = 28.6783+j-9.42609 = 30.1877/_-18.1949
```

Damit ergibt sich die prozentuelle Abnahme aus $\frac{I2-I1}{I1} \cdot 100$

In [44]:
```
delta_rel = (abs(I2) - abs(I1)) / abs(I1) * 100

print("relative Änderung delta_rel = %.6g %%" % delta_rel)
```

```
relative Änderung delta_rel = -10.8702 %
```

Aufgabe 9

RC Schaltung soll analysiert werden.

- R1 = 1kΩ
- C1 = 470nF

In [46]:
```
%reset
import math as math
import numpy as np
import matplotlib.pyplot as plt

pi = math.pi
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

DGL

Das Aufstellen der DGL erfolgt durch eine Masche:

$$-u1 + i \cdot R + u_C = 0$$

Mit $i = C\frac{du}{dt}$ ergibt sich die DGL:

$$RC\frac{du_C}{dt} + u_C = u1$$

Mit der Definition der Zeitkonstanten $\tau = RC$:

$$\tau\frac{du_C}{dt} + u_C = u1$$

Harmonische Lösung

$\tau\frac{du_C}{dt} + u_C = 0$ ist eine separierbare DGL.

$$\frac{1}{\tau}du_C = -u_C dt$$

$$\frac{1}{u_C}du_C = -\frac{1}{\tau}dt$$

Integration ergibt:

$\ln(u_C) = -\frac{t}{\tau} + D$ wobei D eine noch zu bestimmende Konstante ist.

Exponieren ergibt:

$$u_C^h = D \cdot e^{-\frac{t}{\tau}}$$

Partikuläre Lösung

$$u_C^P = u_1$$

Gesamtlösung

$$u_C = u_C^h + u_C^P = D \cdot e^{-\frac{t}{\tau}} + u_1$$

Anfangsbedingung zur Bestimmung von D:

$$u_C(t = 0) = 0V$$

$$D \cdot e^{-\frac{0}{\tau}} + u_1 = 0V$$

$$D = -u1$$

Damit ergibt sich $u_C(t) = -u_1 \cdot e^{-\frac{t}{\tau}} + u_1$ oder $u_C(t) = u_1 \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$

u_C nach 3τ und 5τ

$$u_C(3\tau) = 5V \cdot \left(1 - e^{-3}\right)$$

$$u_C(5\tau) = 5V \cdot \left(1 - e^{-5}\right)$$

In [47]:
```
uC3 = 5 * (1 - math.exp(-3))
uC5 = 5 * (1 - math.exp(-5))

print("Nach 3 tau liegt uC bei %.6g V" % uC3)
print("Nach 5 tau liegt uC bei %.6g V" % uC5)
```

```
Nach 3 tau liegt uC bei 4.75106 V
Nach 5 tau liegt uC bei 4.96631 V
```

Bodeplot des RC Tiefpass

Mit $\underline{H} = \frac{\underline{U_C}}{\underline{U_1}}$

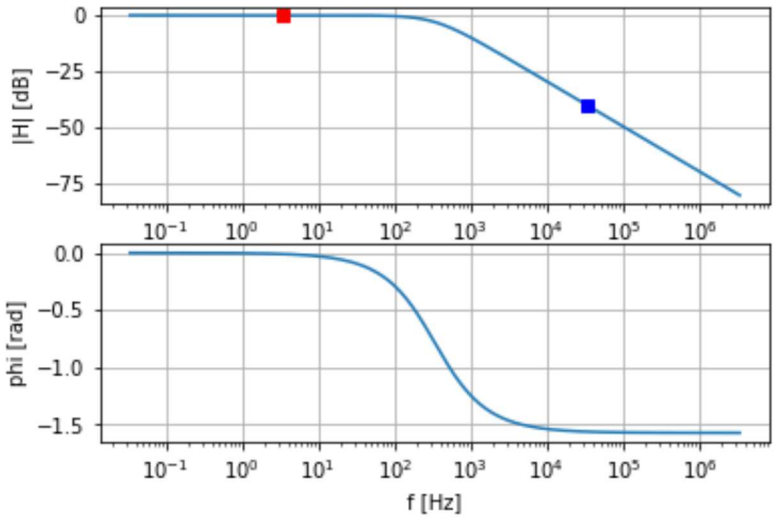
- Bei $f = f_g \rightarrow H = -3dB$ und $\angle H = -45^o$
- Bei $f \ll fg \rightarrow H = 0dB$ und $\angle H = 0^o$
- Bei $f \gg fg \rightarrow H = -20dB/Dekade$ und $\angle H = -90^o$

Abschätzung von UC1 für zwei Frequenzen:

- $f = \frac{fg}{10} \rightarrow$ 0dB
- $f = fg \cdot 100 \rightarrow$ -20dB pro Dekade und es sind zwei Dekaden $\rightarrow$ -40dB. -40dB sind $\frac{1}{100}$. Damit ist $UC1(f = fg \cdot 100) = 10mV$

In [48]:
```
R = 1e3
C = 470e-9
pi = math.pi

tau = R*C
fg = 1/(2*pi*tau)


print("tau = %.6g s" % tau)
print("fg = %.6g s" % fg)



f_01 = fg/100
H_01 = 1./(np.sqrt(1 + (0.1)**2))
f_100 = fg*100
H_100 = 1./(np.sqrt(1 + (100)**2))
print("Bei f=fg/10: H = %.6g (%.6g dB)" % (H_01,20*np.log10(H_01)))
print("Bei f=fg*100: H = %.6g (%.6g dB)" % (H_100,20*np.log10(H_100)))

fvec = np.logspace(math.log10(fg/10000),math.log10(fg*10000),1000)
H = 1./(np.sqrt(1 + (fvec/fg)**2))
phi = 0 - np.arctan(fvec/fg)


plt.subplot(2,1,1)
plt.semilogx(fvec,20*np.log10(H),f_01,20*np.log10(H_01),'rs',f_100,20*np.log1
plt.grid(1)
plt.ylabel("|H| [dB]")
plt.subplot(2,1,2)
plt.semilogx(fvec,phi)
plt.ylabel("phi [rad]")
plt.xlabel("f [Hz]")
plt.grid(1)
```

```
tau = 0.00047 s
fg = 338.628 s
Bei f=fg/10: H = 0.995037 (-0.0432137 dB)
Bei f=fg*100: H = 0.0099995 (-40.0004 dB)
```



Aufgabe 10 Toleranz Widerstand

Ein SMD Widerstand der Bauform 0805 hat bei T=25°C einen Wert von 47Ω. Er hat eine Initialtoleranz von ±1% und wird bei einer Leistung P70 für eine Zeitdauer von 8000h betrieben. Was ist der dann zu erwartende Minimal- bzw. Maximalwiderstand aufgrund Temperaturdrift und Alterung?

In [49]:
```
%reset
import math as math
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

Der gegebene 47Ω Widerstand hat laut Datenblatt folgende Toleranzen:

- Initial $\pm 1\%$
- Temperaturkoeffizient $\pm 50 ppm/K$
- Alterung $\pm 0.5\%$ nach 8000h bei P_{70}. (P_{70} bedeutet, dass der Widerstand bei einer Leistung betrieben wird so, dass er eine Temperatur von $70^{o}C$ hat.)

Maximaler Widerstandswert mit allen Toleranzen

In [50]:
```
R = 47.           #ohm
initial = 1./100  #1%
tc = 50e-6        #50ppm/K
dT = 70.-25.      #Delta Temperatur = 70C-25C
aging = 0.5/100   #0.5 %

err_sum = initial + tc*dT + aging
print("Maximale Toleranz = %.6g %%" %(err_sum*100))

Rmax = R * (1 + initial + tc*dT + aging)
print("Rmax = %.6g ohm" % Rmax)

Rmin = R * (1 - initial - tc*dT - aging)
print("Rmin = %.6g ohm" % Rmin)

Rrel = (R - Rmin)/R
print("Relative Abweichung = %.6g %%" % (Rrel*100))
```

```
Maximale Toleranz = 1.725 %
Rmax = 47.8107 ohm
Rmin = 46.1893 ohm
Relative Abweichung = 1.725 %
```

Fehler des Spannungsteilers

Der maximale Fehler tritt auf, wenn einer der beiden Widerstände minimal und der andere maximal ist.

Das nominelle Verhältnis ist $\frac{R1}{R1+R2} = \frac{R}{2R} = \frac{1}{2}$

Bei einer maximalen Abweichung err ergibt sich für den größten Fehler:

$$\frac{R(1\pm err)}{R(1\pm err)+R(1\pm err)}$$

Wenn sich beide Widerstände in die gleiche Richtung ändern, so heben sich die Fehler auf:

$$\frac{R(1+err)}{R(1+err)+R(1+err)} = \frac{R(1+err)}{2R(1+err)} = \frac{1}{2}$$

Der maximale Fehler ergibt sich somit, wenn R1 minimal und R2 maximal oder vice versa.

$$\frac{R(1+err)}{R(1+err)+R(1-err)} = \frac{R(1+err)}{R(1+err+1-err)} = \frac{1}{2}(1+err)$$

Damit ist der maximale relative Fehler ebenfalls err.

Nachfolgend noch mit Zahlenwerten:

In [51]:
```
Uin = 5.
UR2_nom = Uin * R/(R+R)
print("UR2_nom = %.6g" % UR2_nom)

print("R2 = max, R1 = min")
UR2_tol1 = Uin * Rmax / (Rmin + Rmax)
UR2_rel1 = (UR2_nom - UR2_tol1) / UR2_nom
print("UR2_tol1 = %.6g" % UR2_tol1)
print("Relativer Fehler für R2max und R1min = %.6g %%"%(UR2_rel1*100))
print("")
print("R2 = min, R1 = max")
UR2_tol2 = Uin * Rmin / (Rmax + Rmin)
UR2_rel2 = (UR2_nom - UR2_tol2) / UR2_nom
print("UR2_tol2 = %.6g" % UR2_tol2)
print("Relativer Fehler für R2min und R1max = %.6g %%"%(UR2_rel2*100))
```

```
UR2_nom = 2.5
R2 = max, R1 = min
UR2_tol1 = 2.54312
Relativer Fehler für R2max und R1min = -1.725 %

R2 = min, R1 = max
UR2_tol2 = 2.45688
Relativer Fehler für R2min und R1max = 1.725 %
```

In [ ]:

In [ ]: