3.4 C-Datentypen/Komplexe Zahlen in C99

3.4.1 Problem

In der Wechselstromtechnik hat man oft Schaltungen wie in Abbildung 1. L, C und R_v sind bekannt. Nun sollen bei einer bestimmten Frequenz f der Scheinwiderstand Z und der Phasenwinkel φ (Phasenverschiebung zwischen U und I) ausgerechnet werden.

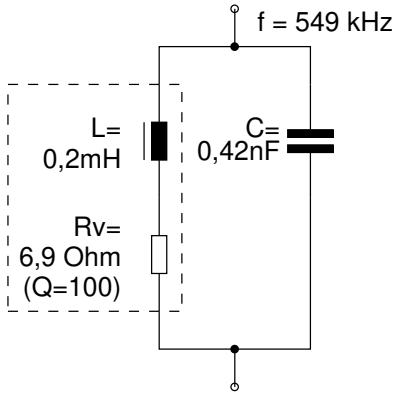

Abbildung 1: Zu berechnende Schaltung

Bei der Berechnung von Hand könnte man sich auf Zeigerdiagramme stützen. Bei der Berechnung mit einem Computer hilft dagegen die Verwendung sogenannter *komplexer Größen*.

Komplexe Zahlen sind zweiteilig: Sie haben einen Realteil und einen Imaginärteil. Beide sind erst einmal voneinander unabhängig, wie zweidimensionale Vektoren (oder auch wie die Koordinaten in einem zweidimensionalen Koordinatensystem). Eine komplexe Zahl k mit dem Realteil a und dem Imaginärteil b könnte man also als Vektor darstellen:

$$k = \begin{pmatrix} a \\ b \end{pmatrix}$$

Nun ist aber eine komplexe Zahl *mehr* als eine Zusammenfassung zweier voneinander unabhängiger Zahlen. Man schreibt sie deshalb so:

$$k = a + i \cdot b$$

Die Zahl i ist dabei die imaginäre Einheit. In der Elektrotechnik nennt man sie lieber j, um sie nicht mit der Stromstärke I zu verwechseln. Man hat i als Wurzel der Zahl -1 definiert:

$$i \cdot i = -1$$

Eine relle Zahl könnte diese Gleichung niemals erfüllen, i dagegen kann es. Mit dieser Festlegung hat man die Möglichkeit, zwei komplexe Zahlen miteinander zu multiplizieren (oder zu dividieren):

$$(a + i \cdot b) \cdot (c + i \cdot d) = ac + iad + ibc - bd$$

Der absolute Betrag einer Zahl k wird berechnet zu:

$$|k| = \sqrt{a^2 + b^2}$$

Den Phasenwinkel φ erhält man mit:

$$\varphi = \arctan \frac{b}{a}$$

Diese etwas willkürliche Festlegung für i, $|k|$ und φ (und damit für komplexe Zahlen) hat sich bewährt, denn sie führt zu Rechenregeln und Rechenergebnissen, die man in Physik und Technik (besonders in der Wechselstromtechnik) gut verwenden kann.

In der Elektrotechnik kann man komplexe Zahlen nämlich so einsetzen:

a) Der Wirkanteil eines Widerstands wird zum Realteil der komplexen Größe Scheinwiderstand Z.

b) Der Blindanteil X desselben Widerstands wird zum Imaginärteil des Scheinwiderstands Z. Dabei wird ein induktiver Anteil X_L positiv gerechnet und ein kapazitiver Anteil X_C negativ.

c) Damit gilt:

$$Z = R + j \cdot X$$

Dann gilt das Ohm'sche Gesetz für komplex dargestellte Wechselspannungen, Wechselströme und Scheinwiderstände genau so wie bei den entsprechenden Gleichgrößen:

$$U = Z \cdot I$$

3.4.2 Lösung bei Wirkwiderständen und Gleichspannung

Das Rechnen mit Scheinwiderständen (komplexen Zahlen) macht die Wechselspannungstechnik also so einfach wie die Gleichspannungstechnik.

Man kann sogar die gleichen Regeln für Reihen-, Parallel-, Brücken- und sonstige Schaltungen verwenden, wie man sie aus der Gleichspannungstechnik kennt. Also kann man Abbildung 1 umwandeln in Abbildung 2 links. Für die linke Schaltung (Wirkwiderstände bei Gleichspannung)

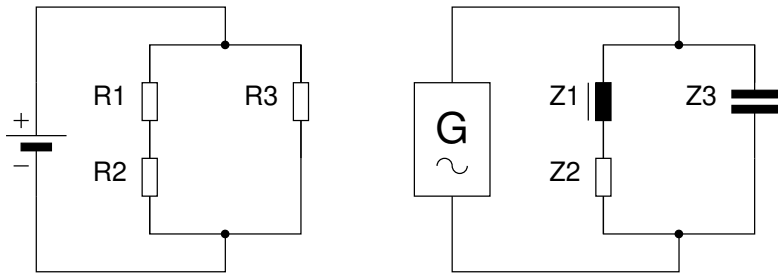

Abbildung 2: Links: DC-Schaltung, rechts: AC-Schaltung

ergibt sich:

$$R_{ers} = (R_1 + R_2)||R_3$$

Für die rechte Schaltung (Scheinwiderstände bei Wechselspannung) ergibt sich:

$$Z_{ers} = (Z_1 + Z_2)||Z_3$$

3.4.3 Umsetzung von R, C und L in Scheinwiderstände

Wenn man Widerstand, Spule und Kondensator in Scheinwiderstände umwandeln will, benutzt man die Formeln (hier mit dem Formelzeichen j für die imaginäre Einheit und $\omega = 2 \cdot \pi \cdot f$):

- $Z_R = R \leftarrow$ Wirkanteil
- $Z_L = j\omega L \leftarrow$ Blindwiderstand, positiv
- $Z_C = \frac{-j}{\omega C} = \frac{1}{j\omega C} \leftarrow$ Blindwiderstand, negativ

3.4.4 Datentypen und Konstanten für komplexe Zahlen in C

Für komplexe Zahlen gibt es seit C99 die folgenden drei Datentypen:

a) `complex float`

b) `complex double`

c) `complex long double`

Meistens verwendet man `complex double`, weil dieser Datentyp in den meisten Fällen eine genügende Genauigkeit bereitstellt. Die imaginäre Einheit wird durch die Konstante `_Complex_I` dargestellt. Mit dem Einbinden von `<complex.h>` darf man für die imaginäre Einheit das Makro `I` verwenden. Damit kann man die Formeln für R, L und C im Quelltext so darstellen:

```
1    double R=6.9, L=0.2e-3, C=0.42e-9, f=549e3, omega=2*3.14*f;
2    complex double ZR, ZL, ZC;
3    ZR=R;        /* implizite Typumwandlung double nach complex double */
4    ZL=I*omega*L;
5    ZC=1.0/(I*omega*C); /* oder: -I/(omega*C)   */
```

3.4.5 Reihen- und Parallelschaltung von Scheinwiderständen

Bei der Reihenschaltung ist:

$$Z_{ers} = Z_a + Z_b$$

Bei der Parallelschaltung ist:

$$Z_{ers} = \frac{Z_a \cdot Z_b}{Z_a + Z_b}$$

Deshalb bietet es sich an, für die Parallelschaltung je eine Funktion zur Berechnung von Z_{ers} zu bauen (und wenn man will, auch für die Reihenschaltung):

```
1  complex double parallel(complex double Za, complex double Zb)
2  {
3     return (Za*Zb)/(Za+Zb);
4  }
5  complex double reihe(complex double Za, complex double Zb)
6  {
7     return Za+Zb;
8  }
```

3.4.6 Ein- und Ausgabe

Für die Ein- und Ausgabe einer komplexen Zahl gibt es in C leider keine Vereinfachungen. Es gibt aber Funktionen, die den Realteil und den Imaginärteil der komplexen Zahl liefern:

a) `double creal(complex double z)` liefert den Realteil von `z`

b) `double cimag(complex double z)` liefert den Imaginärteil von `z`

Wenn man `<math.h>` einbindet, kann man noch zwei weitere nützliche Funktionen verwenden (bei `gcc` mit der Option `-lm` compilieren):

c) `double cabs(complex double z)` liefert den Betrag $|z| = \sqrt{a^2 + b^2}$

d) `double carg(complex double z)` liefert den Winkel $\varphi = \arctan \frac{b}{a}$

Damit kann man etwas umständlich das Ergebnis `Z` ausgeben; in Zeile 1 als Wirk- und Blindanteil und in Zeile 2 als Absolutwert und Phasenwinkel:

```
1     printf("Z=%f Ohm + j * %f Ohm\n", creal(Z), cimag(Z));
2     printf("|Z|=%f Ohm, phi=% Grad\n", cabs(Z), carg(Z)/M_PI*180.0);
```

3.4.7 Lösung

```c
#include <complex.h>
#include <math.h>
#include <stdio.h>

complex double parallel(complex double za, complex double zb)
{
    complex double erg;
    erg = za*zb/(za+zb);
    return erg;
}

complex double l_nach_z(double L, double f)
{
    complex double zl;
    zl = I * 2 * M_PI * f * L;
    return zl;
}

complex double c_nach_z(double C, double f)
{
    complex double zc;
    zc = -I/(2 * M_PI * f * C);
    return zc;
}

int main(void)
{
    double L = 0.2e-3;
    double Rv= 6.9;
    double C = 0.42e-9;
    double f = 549000;
    complex double Z1, Z2, Z3, Z_ers;

    Z1 = l_nach_z(L,f);
    Z2 = Rv;
    Z3 = c_nach_z(C,f);

    Z_ers = parallel(Z1+Z2 , Z3);
    printf("Z_ers=%lf Ohm\n", cabs(Z_ers));
    printf("phi=%lf Grad\n", carg(Z_ers)*180.0/M_PI);
    return 0;
}
```

Wenn man gcc benutzt, muss man das obige Programm mit der Option `-lm` (bedeutet: Einbinden der Mathe-Bibliothek `libm.a`) compilieren:

Terminal

```
schueler@debian964:~$ gcc rlcgemischt.c -lm
schueler@debian964:~$ a.out
Z_ers=68931.117229 Ohm
phi=2.278345 Grad
```