Mathematica

Michael Kugler, N, TGM DSK-404\MMC

Mathematica gesellt sich in die Reihe der Mathematik-Programme wie Derive oder Mathcad. Es muß nicht unbedingt als Konkurrent angesehen werden; jedes dieser Programme hat seine Anwender. Dieser Beitrag - weitere werden folgen - zeigt an Hand des Beispiels eines Serienschwingkreises, wozu Mathematica in der Lage ist.

Bemerkungen

Mathematica

Eine der Besonderheiten liegt in der Möglichkeit, Trickfilme zu erstellen. Da es in einer Zeitung nicht möglich ist, einen Film ablaufen zu lassen, habe ich aus den jeweiligen Sequenzen die beiden ersten und letzten Bilder dargestellt.

Die Syntax von Mathematica erscheint auf den ersten Blick ungewohnt. Ich habe versucht, dort, wo es mir wesentlich erschienen ist, einige Bemerkungen zu machen. Diese Bemerkungen lassen sich im Notebook, bedingt durch eine raffinierte Zellstruktur, ausblenden, so daß der jeweilige Benutzer nur die für ihn relevanten Bereiche sieht. Im vorliegen Notebook sind alle Ebenen eingeblendet.

Mathematica und Winword

Mathematica arbeitet auf den verschiedensten Plattformen, Windows ist nur eine von ihnen. Damit die Kompatibilität gewahrt bleibt, verwendet Mathematica Postscript zur Formatierung von Text und Graphik. Daraus ergibt sich ein Problem in der Zusammenarbeit mit Winword. Ein direktes Einlesen des Notebooks in Winword ist nicht möglich. Es ist jedoch über einige Umwege möglich. Bei der Graphik gibt es jedoch ein Problem: Die Größenformatierung der in den Graphiken vorkommenden Zeichen werden von Winword schlichtweg ignoriert und mit einem relativ kleinen Wert belegt; sogar wenn in Mathematica mit Formatanweisungen die Graphiken mit größeren Fonts ausgestattet werden. Daher ist die Beschriftung der Graphiken nicht optimal.

Das vorliegende Notebook

In jeder interaktiven Sitzung mit Mathematica werden die Eingabezeilen z.B. mit `In[3]:=` bezeichnet. Jeder Output erhält entsprechend `Out[3]=`. Damit Ein- und Ausgaben unterscheidbar sind, habe ich daher für die Eingaben **`Lucida Sans Typewriter fett`**, für die Ausgaben `Lucida Sans Typewriter` benutzt. Im Notebook sieht das ganze dann so aus:

```
In[23]:=lsg1=Solve[Im[zGes]==0,omega]
                                1
Out[23]={{omega -> -(---------------)}
                    Sqrt[c] Sqrt[l]
```

Da in Mathematica alle eingebauten Funktionen mit einem Großbuchstaben beginnen, ist es sinnvoll alle selbstdefinierte Namen mit einem Kleinbuchstaben beginnen zu lassen, daher z. B. `omega` statt `Omega`.

Der Mathreader

Die Firma Wolfram-Research in Illinois, USA, als Urheber von Mathematica hat ein frei verfügbares Produkt herausgegeben, den Mathreader, mit dessen Hilfe Notebooks gelesen und ausgedruckt werden können. Dieser Mathreader ist ab sofort beim Klub erhältlich. Er enthält zusätzlich weitere informative Notebooks über Mathematica.

>>> Schluß des Beitrags Mathcad

6. Zusatzsoftware

Die bis jetzt beschriebenen Beispiele geben einen kleinen Einblick, wie Berechnungsbeispiele mit Mathcad gelöst werden können. Für komplexere Aufgaben zeigt es sich nun, daß das Auffinden der entsprechenden Definitions-Gleichungen des jeweiligen Problems etc. und das Formulieren des Problems in mathematischer Form nun der zeitraubendste Teil der Lösung ist. (Was für das Berechnen "zu Fuß" natürlich auch gilt.)

Um dem Anwender zeitraubende Eingaben zu ersparen, sind in der Zwischenzeit eine Reihe von Zusatzpaketen auf den Markt gekommen. Es gibt im wesentlichen zwei Arten von Zusatzsoftware:

- Elektronische Handbücher: Zu diesen zählen im wesentlichen Tabellen mit einer großen Anzahl von physikalischen Konstanten.
- Zusatzpakete: Hier werden Aufgaben aus Teilgebieten der Mathematik beispielhaft gelöst. Diese "Fertig-Dokumente" können für spezielle Fragestellungen leicht abgewandelt werden. Es sind beispielsweise folgende Pakete verfügbar: Statistik, Differentialrechnung, Bauingenieurwesen, Elektrotechnik, Mechanik usw.

Mit diesen Zusatzpaketen ist es sogar möglich, gewöhnliche, auch nichtlineare Differentialgleichungen (mechanische und elektr.Schwinger etc.) zu lösen, sei es, daß keine explizite, allgemeine Lösungsfunktion existiert oder daß es zu aufwendig wäre, diese ausfindig zu machen. Derartige Aufgaben erfordern allerdings einen entsprechend schnellen Rechner, um z.B. bei einer Runge-Kutte Näherungsrechnung mit 400 Stützstellen innerhalb vernünftiger Zeit zu einem Ergebnis zu kommen.

7. Ausblick

Abschließend kann ich feststellen, daß mit Mathcad die rechnerische Lösung einer Vielzahl von Problemen wiederum ein Stück einfacher geworden ist. Dennoch dürfen wir die Tatsache nicht aus den Augen verlieren, daß eine Rechnung nur so gute Ergebnisse liefern kann, wie die Eingangsdaten "verläßlich" sind. Es macht also beispielsweise wenig Sinn, eine Trägerdurchbiegung auf 0.01mm zu berechnen, wenn die wirkende Kraft nur grob bekannt ist. Genauso muß die Gültigkeit von Gleichungen für entsprechende Werte beachtet werden usw. Zusammengefaßt, wir dürfen uns nicht dazu verleiten lassen, wegen leistungsfähiger "Rechen-Tools" den gesunden Menschenverstand außer acht zu lassen. Grundlegende Kenntnisse der Mathematik oder Mechanik kann auch noch so leistungsfähige Berechnungssoftware nicht ersetzen!

Sollten Sie weitere Fragen oder Anregungen zu meinen Ausführungen haben, können Sie mich über CompuServe (100276,1244) erreichen.

Literaturhinweise

D.Donnelly, MathCAD for Introductory Physics, Addison-Wesley, 1992
J.Rowell, Mathematical Modeling with MathCAD, Addison-Wesley, 1990
Div.Autoren, Quarterly Electronic Magazine, Mathsoft Inc., 1993
P.Voit, Mathcad 3.1 für Windows, Toolbox-Magazin, Red.DOS, 4/1992
D. Reiermann, Mathcad für Windows, eine gute Lösung, PC·NEWS-31, S.38.
H. Schwarz, Vektoralgebra mit Derive, PC·NEWS-35, S.18 □

Hinweise

Alle hier dargestellten Beispiele sind sowohl im Mathcad-Format der Version 3 als auch im Format der Version 4 auf der Diskette in den Verzeichnissen `MCAD\3` und `\MCAD\4` enthalten.

[1]) Der Textfehler bei v_{out} entstand durch Konversion der in Version 4 erstellten Dokumente in die Version 3 mit einer nicht ganz vollständig installierten MathCad-Version in der Redaktion.

Resonanzkurve eines Serienschwingkreises

```
Remove["Global`*"];     (*löscht alles*)
Needs["Algebra`ReIm`"]  (*liest dieses Paket ein*)
```

Was soll hier passieren?

Durch Berechnen des komplexen Widerstandes wird der Strom durch diesen Widerstand bestimmt. Die Abhängigkeit des Stromes von der Frequenz (die Resonanzkurve) und der Phase ist zu zeichnen.

Für beide Zeichnungen ist ein Film zu erstellen, in dem der Serienwiderstand der zu verändernde Parameter ist.

Das ganze Paket soll so allgemein wie möglich gehalten werden. Die speziellen Daten des Schwingkreises sollen erst bei der Auswertung der Zeichnungen eingesetzt werden.

Berechnung des Widerstandes

Im allgemeinen rechnet Mathematica mit komplexen Zahlen. Da üblicherweise R, C, L und die Frequenz reelle Werte sind, wird dies in der folgenden Regel Mathematica explizit mitgeteilt. Damit sind Vereinfachungen erst sinnvoll.

```
Im[r]^=0;
Im[l]^=0;
Im[c]^=0;
Im[omega]^=0;
```

Der Gesamtwiderstand `zGes` ist die Summe der einzelnen Widerstände.

```
zGes:=r + l omega I + 1/(c omega I)
```

Berechnung der Resonanzfrequenz

Wird der Imaginärteil des Gesamtwiderstandes Null, so herrscht Resonanz.

```
lsg1=Solve[Im[zGes]==0,omega]

                    1
{{omega -> -(---------------)},
             Sqrt[c] Sqrt[l]

                   1
 {omega -> ---------------}}
           Sqrt[c] Sqrt[l]
```

Da die erste Lösung einen negativen Wert hat, wird `omega0` der zweiten Lösung zugewiesen.

```
omega0=omega/.lsg1[[2]]

      1
---------------
Sqrt[c] Sqrt[l]
```

Eingabe der Daten

Alle für den Schwingkreis wichtigen Daten werden in einer Liste `daten` abgelegt. Wann immer es notwendig ist, Werte einzusetzen, kann diese Liste benutzt werden. Die Daten werden als Gleitkommazahlen eingegeben, damit wird jeder Ausdruck automatisch numerisch und nicht symbolisch ausgewertet.

```
daten={
          r->100.,
          l->40 10^-3,
          c->0.6 10^-6,
          uEin->1
      };
```

Die Resonanzfrequenz ergibt sich dann durch Anwenden der Liste `daten` auf den Ausdruck `omega0`. (Genauer: Im Ausdruck `omega0` werden alle Namen die in der Liste `daten` vorkommen durch die Werte ersetzt, und anschließend wird der Ausdruck solange vereinfacht, bis sich keine Änderung mehr ergibt.)

```
omega0/.daten
6454.97
```

Zur Kontrolle wird der Gesamtwiderstand bei der Resonanzfrequenz berechnet.

```
zGes/.omega->omega0 /.daten
100.
```

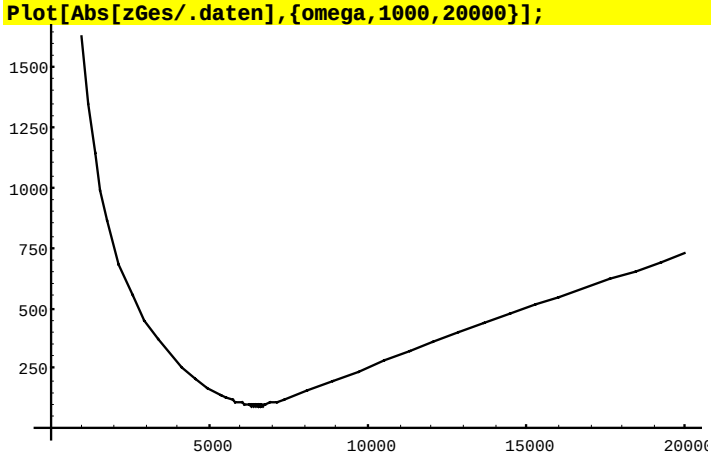
Bei einer kleinen Verstimmung muß der Widerstand wieder komplex sein:

```
zGes/.omega->(omega0 * 1.1) /.daten
100. + 49.2925 I
```

Zeichnen des Frequenzganges des Widerstandes

Mit diesen Daten soll nun der Frequenzgang des Gesamtwiderstandes (das ist der Betrag des komplexen Widerstandes) gezeichnet werden.

```
Plot[Abs[zGes/.daten],{omega,1000,20000}];
```



Aus FIDO/SCHULBRETT: Ein Vorschlag zur Rechtschreibreform:

ERSTER SCHRITT:

generelle kleinschreibung (werbeleute und graphiker benutzen sie bereits).

ZWEITER SCHRITT:

wegfall der dehnungen und schärfungen (dise masname eliminirt di meisten feler in den schulen).

DRITER SCHRIT:

v, w und ph ersezt durch f; z ersezt durch s; sch ersezt durch s; x ersezt durch ks (das alfabet fird um fir buchstaben redusiert; sreib- und setsmasinen fereinfachen sich; fertfole arbeitskrefte können der firtsaft sugefürt ferden).

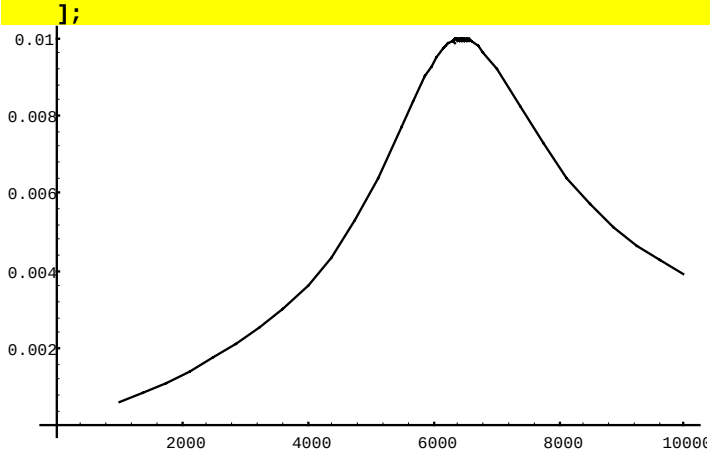
FIRTER SRIT:

q, c und ch ersest durch k; j und y ersest durch i; pf ersest durch f; ferner k durch g; t durch d; p durch b (es sind son elf buksdaben ausgesalded, in den sulen könen sdad aksig brosend rekdsreibung nüslikere feker fi fisik, kemi und mademadig mer geflegt ferden).

Der Stromverlauf

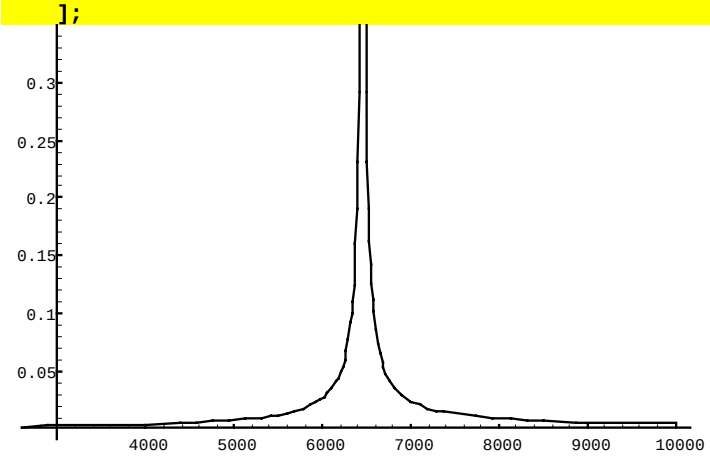
Spannung durch Widerstand ergibt den Strom, dieser wird nun mit den gegebenen Daten gezeichnet. (Der Befehl `Evaluate` in der Plot Anweisung dient nur zu der schnelleren Auswertung.)

```
Plot[Evaluate[uEin/Abs[zGes]/.daten],
     {omega,1000,10000}
    ];
```



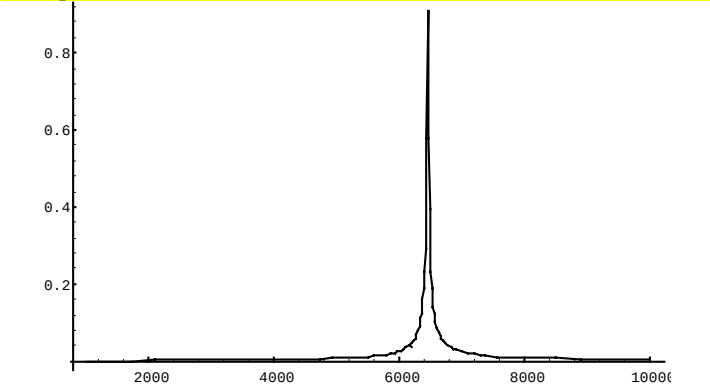
Will man diese Kurve mit einem anderen Widerstand zeichnen, so muß die Variable `r` zuerst mit einem anderen `[m1]` Wert belegt werden. Auf diese Art kann die oben angelegte Liste `daten` verwendet werden, da `r` zu diesem Zeitpunkt der Substituierung bereits durch einen Wert ersetzt wurde.

```
Plot[Evaluate[uEin/Abs[zGes]/.r->1/.daten],
     {omega,1000,10000}
    ];
```



Mathematica sucht einen vernünftigen Zeichenbereich; dabei kann es vorkommen, daß einzelne Punkte nicht mehr gezeichnet werden. Mit der Option `PlotRange->All` werden alle Punkte gezeichnet.

```
Plot[Evaluate[uEin/Abs[zGes]/.r->1/.daten],
     {omega,1000,10000},
       PlotRange->All
    ];
```



Und nun ein Movie

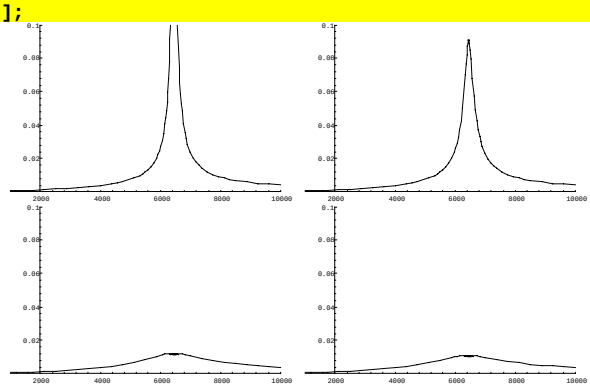
In dieser Bildfolge soll der Widerstand verändert werden. Dazu ist es sinnvoll, eine neue Funktion zu definieren, die zwei Parameter hat; die Frequenz und den Widerstand. Alle anderen Werte werden wieder von Parameterliste übernommen.

Dies ist überdies effektiver, da nur einmal die Substituierung vorgenommen wird, und nicht bei jeder Berechnung im Plot-Kommando.

```
strom[omega_,rw_]=uEin/Abs[zGes]/.r->rw/.daten
                    1
-----------------------------------
                  6
    -1.66667 10  I   I
Abs[-------------- + -- omega + rw]
        omega        25
```

Nun können die Bilder gezeichnet werden. `PlotRange` ist notwendig, damit alle Bilder den gleichen Maßstab haben.

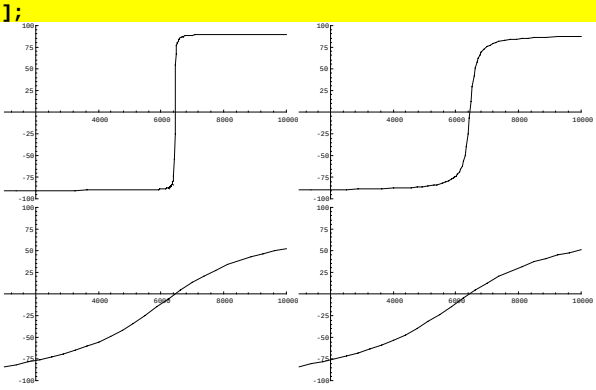
```
Table[
      Plot[strom[omegaBereich,rBereich],
           {omegaBereich,1000,10000},
              PlotRange->{{1000,10000},{0,0.1}}
          ],
       {rBereich,1,100,10}
      ];
```



Der Phasengang

Für den Phasengang wird in ähnlicher Weise vorgegangen. Die Division durch `Degree` ist wegen der Umrechnung von Radianten notwendig.

```
phase[omega_,rw_]=Arg[zGes]/Degree/.r->rw/.daten
                    6
    -1.66667 10  I   I
Arg[-------------- + -- omega + rw]
        omega        25
-----------------------------------
                Degree
Table[
      Plot[phase[omegaBereich,rBereich],
           {omegaBereich,1000,10000},
              PlotRange->{{1000,10000},{-100,100}}
          ],
       {rBereich,1,200,10}
      ];
```



Die normierte Darstellung

Zunächst normieren wir die Frequenz. Damit es übersichtlicher wird, ist es sinnvoll, die Belegung von `omega0` zu löschen.

```
Clear[omega0]
```

Die Verstimmung `v` ist folgendermaßen definiert:

```
v==omega/omega0-omega0/omega
     omega    omega0
v == ------ - ------
     omega0   omega
```

dies nach `omega` aufgelöst ergibt:

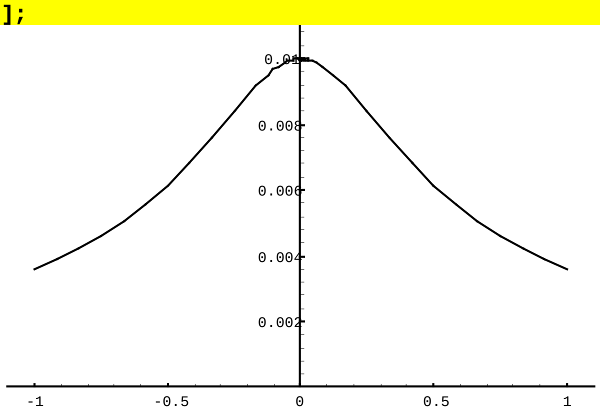
```
lsg2=Solve[v==omega/omega0-omega0/omega,omega]
                                   2          2  2
           omega0 v - Sqrt[4 omega0  + omega0  v ]
{{omega -> ---------------------------------------},
                              2

                                   2          2  2
           omega0 v + Sqrt[4 omega0  + omega0  v ]
 {omega -> ---------------------------------------}}
                              2
```

In unsere Formel für den Strom müssen wir nun `omega` durch diesen Ausdruck ersetzen, da nun `v` der Parameter zum Zeichnen ist. Die vorhin gelöschte Belegung für die Resonanzfrequenz `omega0` wird wieder installiert. (Dies war nur für die Übersichtlichkeit, nicht aber für die Rechnung an sich notwendig).

```
omega0=omega/.lsg1[[2]]
       1
---------------
Sqrt[c] Sqrt[l]
```

Mit der Funktion `strom1` wird nun der auf die Frequenz normierte Strom bezeichnet, und `omega` durch die erste der beiden Lösungen ersetzt, anschließend werden wieder die Daten eingesetzt.

```
strom1[v_,rVariabel_]=
     strom[omega,rVariabel]/.lsg2[[1]]/.daten;
Plot[strom1[vBereich,100],
     {vBereich,-1,1},
     PlotRange->{{-1.1,1.1},{0,0.011}}
    ];
```



Nun zur Normierung des Stromes. Dazu ist es notwendig, den maximalen Strom zu wissen. Dieser fließt bei der Resonanzfrequenz `omega0`.

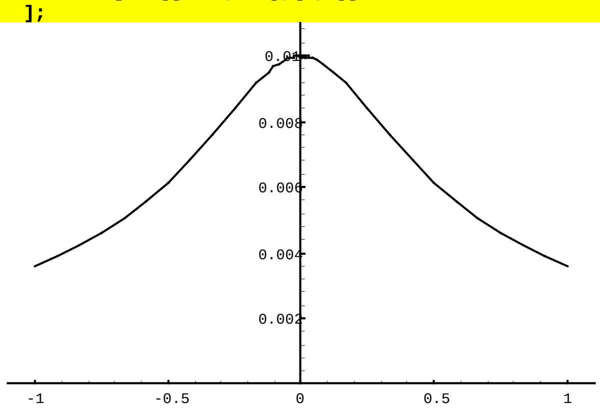
```
iMax[rVariabel_]=strom[omega0,rVariabel]/.daten
         1
--------------------
Abs[0. I + rVariabel]
```

Die Funktion `stromNormiert[v_,rVariabel_]` liefert die Werte des normierten Stromes bezüglich der Verstimmung `v`.

```
stromNormiert[v_,rVariabel_]=
    strom1[v,rVariabel] / iMax[rVariabel]
Abs[0. I + rVariabel] /

 Abs[rVariabel -

                            6
                   3.33333 10  I
    ------------------------------------------- +
                             8            7  2
    6454.97 v - Sqrt[1.66667 10  + 4.16667 10  v ]

   I                            8            7  2
   -- (6454.97 v - Sqrt[1.66667 10  + 4.16667 10  v ])
   50

   ]
```

Und nun die graphische Darstellung für einen Serienwiderstand von 100 Ohm.

```
 Plot[stromNormiert[vBereich,100],
      {vBereich,-2,2},
      PlotRange->{{-2.1,2.1},{0,1}}
     ];
```



Der Übergang zu einem Movie ist nun nicht schwer. Zusätzlich werden in die Plotanweisung noch einige Verschönerungen eingebaut.

Info zum folgenden Plot

`AxesLabel` liefert die Achsenbeschriftung. `PlotStyle` wird benutzt um mit der Funktion Text einen String in der Graphik auszugeben. Mit `StringForm` lassen sich - ähnlich wie in C - Text und Werte von Variablen formatiert ausgeben.

```
Table[Plot[stromNormiert[vBereich,rVariabel],
              {vBereich,-2,2},
              PlotRange->{{-2.1,2.1},{0,1}},
              AxesLabel->
                {"Verstimmung","normierter Strom"},
              PlotStyle->
                Text[StringForm["R=`` Ohm",rVariabel],
                     {1,1}
                  ]
          ],
      {rVariabel,10,200,10}
     ];
```