6 Polynome

Dieses Kapitel beschäftigt sich mit der numerischen Auswertung von Polynomen. Insbesondere sind dies Funktionswerte, und numerische Differenziation mit dem Verfahren nach Horner. Ferner wird zur Vertiefung und Repetition das Thema Nullstellenbestimmung aufgegriffen. Hier wird die Newton-Methode bezüglich Polynome spezialisiert und weiter wird mit dem Verfahren nach Bairstow gezeigt wie auch komplexe Nullstellen bestimmt werden können.

Polynome haben besonders für den Elektroingenieur grosse Bedeutung, da viele Funktionen in der Elektrotechnik mit Polynomen oder als rationale Funktionen beschrieben werden. Deshalb stellt dieses Kapitel einen umfangreichen Werkzeugkasten zur numerischen Behandlung von Polynomen dar.

6.1 Einführung

Polynome sind Summen von Potenzen mit konstanten Koeffizienten der Form:

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 \qquad a_i \in \mathbb{R}, a_n \neq 0 \tag{6.1}$$

Dabei heisst n der Grad des Polynoms. Ist der Leitkoeffizient a_n=1, so nennen wir das Polynom 'normiertes Polynom' oder Normalform. Man kann die Normalform immer durch Division des gesamten Polynoms mit a_n erreichen.

Die Nullstellen eines Polynoms vom Grad < 5 können exakt mit Lösungsformeln bestimmt werden. Nullstellen in Polynomen vom Grad ≥ 5 sind nicht mehr algebraisch bestimmbar. Dies wurde durch Galois[1] und Andere bewiesen.
Jedoch wird bereits beim Polynom vierten Grades der Aufwand für eine analytische Lösung so gross, dass in den meisten Fällen auf numerische Verfahren zurückgegriffen wird.

Als analytische Lösung eines Nullstellenproblems kennen wir für Polynome zweiten Grades die Methode der quadratischen Ergänzung, die schlussendlich zu der bekannten Lösungsformel führt:

$$ax^2 + bx + c = 0 \qquad \rightarrow \qquad x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{6.2}$$

Ist die Diskriminante $b^2 - 4ac$ negativ, so erhalten wir ein konjugiert komplexes Nullstellenpaar. Grundsätzlich gilt, dass bei einem Polynom mit ausschliesslich reellen Koeffizienten, die Nullstellen immer als konjugiert komplexe Paare auftreten.

6.1.1 Fundamentalsatz der Algebra

Nullstellenbestimmungen von Polynomen werden vielfach zum Zerlegen des Polynoms in Faktoren bestimmt.

Der Fundamentalsatz besagt:
Jedes Polynom mit reellen Koeffizienten lässt sich in komplexe Linearfaktoren zerlegen. Also:

$$\begin{aligned} p_n(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots a_2 x^2 + a_1 x + a_0 \qquad a_i \in \mathbb{R}, a_n \neq 0 \\ &= (x - x_0)(x - x_1)\cdots(x - x_{n-1})(x - x_n) \qquad x_i \in \mathbb{C}, \text{ Nullstelle des Polynoms} \end{aligned} \tag{6.3}$$

[1] E. Galois: Französischer Mathematiker (1811-1832). Er begründete viele Sätze in der Algebra, insbesondere der Körper- und Gruppentheorie.

Wir können also von jedem Polynom *n* komplexe Nullstellen x_i bestimmen und *n* Linearfaktoren bilden. Man sagt auch : Das Polynom zerfällt über □ in Linearfaktoren):

$$p_n(x) = a_n(x - x_1)(x - x_2)\cdots(x - x_n) \tag{6.4}$$

Dabei können die Nullstellen ohne weiteres mehrfach auftreten.

Anmerkung: Treten bei der Zerlegung komplexe Nullstellen auf, so sind diese immer paarweise konjugiert komplex. Dies gilt generell für Polynome mit reellen Koeffizienten.

6.2 Zerlegen eines Polynoms in Linearfaktoren

Dazu werden die Nullstellen des Polynoms bestimmt. Mit den negativ genommenen Nullstellen werden dann die Linearfaktoren erzeugt.

Beispiel:

$$x^3 - 3x^2 + x + 5 = 0 \qquad x_1 = -1, x_2 = 2 + j, x_3 = 2 - j$$

Wir erhalten also das faktorisierte Polynom:

$$p(x) = (x + 1)(x - (2 + j))(x - (2 - j))$$

Ist eine Nullstelle bekannt, so können wir durch Polynomdivision den Grad des Polynoms vermindern. Haben wir beispielsweise durch Erraten in obigen Beispiel x_1=-1 als Nullstelle gefunden, so können wir das Polynom durch den Linearfaktor *(x+1)* dividieren:

$$\begin{array}{l} \left(x^3 - 3x^2 + x + 5\right) : (x + 1) = x^2 - 4x + 5 \\ \underline{-(x^3 + x^2)} \\ \quad 0 - 4x^2 \\ \quad \underline{-(-4x^2 - 4x)} \\ \qquad 0 \quad + 5x \\ \qquad \underline{-(5x + 5)} \\ \qquad\quad 0 \end{array} \qquad \Rightarrow \quad p(x) = (x + 1)(x^2 - 4x + 5)$$

Wir erhalten einen Linearfaktor und ein quadratisches Polynom.

Folgerung:

Polynome mit reellen Koeffizienten können immer in ein Produkt von linearen- und quadratischen Faktoren mit reellen Koeffizienten zerlegt werden.

Begründung:
Aus dem Fundamentalsatz der Algebra folgt, dass jedes Polynom über □ in Linearfaktoren zerfällt. Da die Koeffizienten des Polynoms nach Vorgabe reell sind, folgt dass komplexe Nullstellen immer als konjugiert komplexe Paare auftreten. Das Produkt zweier Linearfaktoren mit konjugiert komplexen Gliedern ergibt ein quadratisches Polynom mit reellen Koeffizienten.

6.2.1 Programmierte Lösung zum Abspalten von Linearfaktoren

Das Abspalten eines Linearfaktors kann man in einem Programm elegant in einer Funktion implementieren. Wir zeigen hier eine Funktion, der die Koeffizienten $a_0,..,a_n$ des Polynoms in einem Array `a` als Argument übergeben werden. Weitere Argumente sind der aktuelle Grad `n` des Polynoms (über Referenz) und die Nullstelle `x0`.

Die Funktion reduziert nun durch Abspalten eines Linearfaktors des im Array `a` bestimmten Polynoms. Im Array stehen am Schluss die Koeffizienten des im Grad um 1 reduzierten Polynoms und in `n` wird der reduzierte Grad über Zeigerreferenz zurückgeschrieben.

```
/* Abspalten eines Linearfaktors der Form (x-x0) vom
   Polynom dessen Koeffizienten in a[0]..a[n] gegeben sind.

   n verkörpert den Grad des zu reduzierenden Polynoms.n wird
   ueber Zeigerreferenz übergeben und in der Funktion ebenfalls
   um 1 reduziert.*/

void deflate(double a[], int *n, double x0)
{ int i;
  int grad;

  grad = *n;
  for (i = grad; i > 0; i--)
     a[i-1] = a[i] * x0 + a[i-1];
  for (i = 0; i < grad; i++)
     a[i] = a[i+1];
  *n = grad-1;
}
```

6.3 Abschätzen der Nullstellen

Die Lage der Nullstellen eines Polynoms können aufgrund der Koeffizienten abgeschätzt werden. Für die Nullstelle x_i gilt:

$$|x_i| \le \min(2\gamma, 1+\beta) \qquad \text{wobei:} \quad \beta := \max_i \left|\frac{a_i}{a_n}\right| \quad \gamma := \max_{i<n} \sqrt[n-i]{\left|\frac{a_i}{a_n}\right|} \tag{6.5}$$

Beispiel: Die Lage der Nullstellen des Polynoms $x^3 - 5x^2 - 28x + 32$ ist abzuschätzen:

$$n = 3$$

$$\beta = \max_i \left|\frac{a_i}{a_n}\right| = \max_i \left|\frac{a_0}{a_3}, \frac{a_1}{a_3}, \frac{a_2}{a_3}, \frac{a_3}{a_3}\right| = \max_i |32, -28, -5, 1| = 32$$

$$\gamma = \max_{i<n} \sqrt[n-i]{\left|\frac{a_i}{a_n}\right|} = \max_{i<3} \left(\sqrt[3]{\left|\frac{a_0}{a_3}\right|}, \sqrt[2]{\left|\frac{a_1}{a_3}\right|}, \left|\frac{a_2}{a_3}\right| \right) = \max_{i<3} \left(\sqrt[3]{32}, \sqrt{28}, 5 \right) = \sqrt{28}$$

$$x = \min(2\gamma, 1+\beta) = \min(2\sqrt{28}, 33) = 2\sqrt{28} \approx 10.58$$

Diese Methode ist vor allem für Polynome höheren Grades aufgrund der Extremwertbildung aufwendig, wenn sie von Hand praktiziert wird. Das Verfahren kann aber einfach in C programmiert werden.

6.3.1 Programmierte Lösung zum Abschätzen der Nullstellen

Wir definieren hierzu eine Funktion `zero_guess()`, welche die Schätzung einer Nullstelle zurückliefert:

```
/* Abschaetzen einer Nullstelle des Polynoms mit den Koeffizienten
   a[0]..a[n-1].
   n ist der Grad des Polynoms.
*/

double zero_guess(double a[], int n)
{ double gamma, beta;
  double x,w;
  int i;

  beta=1;
  for (i=0; i < n-2; i++)
    if (beta < fabs(a[i]/a[n]))
       beta =  fabs(a[i]/a[n]);

  gamma = fabs(a[n]);
  for (i=0; i< n; i++)
    { if (a[i] != 0)
       { w = exp(log(fabs(a[i]))/(n-i));
         if (w > gamma)
            gamma = w;
       }
    }
   if (2 * gamma < 1 + beta)
        x = 2 * gamma;
     else
        x = 1 + beta;

  return x;
}
```

Die Funktion erhält die Koeffizienten des Polynoms als Argument im Array `a`. Das zweite Argument `n` ist der Grad des Polynoms.
Wir werden in einem späteren Entwurf auf diese Funktion zurückkommen.

6.4 Horner-Schema

Unter dem Horner Schema versteht man Methoden zur vereinfachten Berechnung von Funktionswerten von Polynomen, Ableitungen und Abspalten von Faktoren. Das grundsätzliche Verfahren wurde von Wiliam G. Horner, 1756-1837) vorgestellt.

Besondere Bedeutung erhält die Bestimmung des Funktionswertes eines Polynomes $p_n(x)$ an der Stelle x. Klassisch wird dabei an jedem Glied der Summe das Argument x potenziert und mit dem entsprechenden Koeffizienten multipliziert. Dieses Potenzieren, was ja eigentlich eine fortgesetzte Multiplikation darstellt, wird bei so einer Berechnung unnötig oft ausgeführt.

Beispiel: Berechnen des Funktionswertes des Polynoms $3x^3 - 10x^2 + 103x - 700$ an der Stelle x=2:

Klassisch: $3 \cdot 2^3 - 10 \cdot 2^2 + 103 \cdot 2 - 700 = 3 \cdot (2 \cdot 2 \cdot 2) - 10 \cdot (2 \cdot 2) + 103 \cdot 2 - 700 = -510$

Horner: $(((3)x - 10)x + 103)x - 700) = (((3 \cdot 2 - 10)2 + 103)2 - 700) = -510$

Das Verfahren von Horner zeigt wie Funktionswerte und deren Ableitungen von Polynomen effizient bestimmt werden können. Die Effizienzsteigerung beruht vor allem darauf , dass die Anzahl der notwendigen Multiplikationen drastisch reduziert werden.

Betrachten wir den Rechenaufwand bezüglich Multiplikationen[2] bei einer Auswertung eines Polynoms der Form

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \quad \cdots \quad + a_2 x^2 + a_1 x + a_0 \qquad a_i \in R, a_n \neq 0 \tag{6.6}$$

so benötigen wir $\frac{n^2+n}{2}$ Multiplikationen für die Auswertung.

Das Verfahren von Horner beruht darauf, dass bei einem Polynom jeweils sukzessive ein x ausgeklammert werden kann:

$$\begin{aligned} p_n(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 \qquad a_i \in R, a_n \neq 0 \\ &= ((((a_n x) x + a_{n-1})x + \cdots)x + a_2)x + a_1)x + a_0 \end{aligned} \tag{6.7}$$

Dieses Verfahren benötigt nur noch n Multiplikationen, was vor allem für grosse n eine wesentliche Verbesserung darstellt.

6.4.1 Praktische Anwendung

Das Horner Schema ist sowohl für die Handrechnung wie auch zur Formulierung effizienter Auswertungen in Programmen gleichermassen geeignet.
Für die Handrechnung erstellt man zweckmässigerweise eine Tabelle wo die Koeffizienten des Polynomes eingetragen werden. Wir schreiben ein *einzeiliges Horner Schema* auf:

	a_n	a_{n-1}	...	a_1	a_0
x_0	–	$a_n x_0$	...		
	a_n	$a_n x_0 + a_{n-1}$	...		$p_n(x_0)$

Die Felder in der untersten Zeile entstehen durch Addition der jeweils oberen beiden Felder Das mittlere Feld in der zweiten Zeile erhält man durch Multiplikation mit x_0. Im letzten Feld steht dann das gewünschte Resultat, der Funktionswert der Polynomfunktion an der Stelle x_0.

Zu beachten ist, dass in dieser Tabelle alle Koeffizienten des Polynoms notiert werden müssen, auch solche mit dem Wert Null.

Beispiel: Auswertung eines Polynomes $2x^5 - 5x^3 + 4x + 1$ an der Stelle x_0=4:

	2	0	-5	0	4	1
4	–	8	32	108	432	1744
	2	8	27	108	436	1745

Ein Interessanter Fall entsteht in der Tabelle (Horner Schema), wenn wir das Verfahren für eine Nullstelle des Polynoms anwenden. Es ist klar, dass wir im Feld unten rechts Null erhalten werden. Jedoch verkörpern die restlichen Felder der dritten Zeile die Koeffizienten des Quotienten der Polynomdivision, d.h. wenn wir einen Linearfaktor vom Polynom abspalten würden.

Beispiel:

[2] Aufwandbetrachtung:
Für die Abschätzung des Rechenaufwandes wird hier nur die wesentliche Operation betrachtet. Dies ist hier die Multiplikation, da diese aufwendiger ist als die Addition und somit auch mehr ins Gewicht fällt. Generell werden bei Aufwandbetrachtungen in Verfahren nur die wesentlichen Operationen gezählt. Wichtig ist beim erhaltenen Resultat nicht nur die Aussage selbst, sondern auch das Wachstumsverhalten der Methode für grosse n.

Wir kennen durch Probieren eine Nullstelle x_0=-1 des Polynoms $2x^5 - 5x^3 + 4x + 1$ und möchten nun einen Linearfaktor abspalten:

$$p_n(x) = p_{n-1}(x)(x - x_0) + p_n(x_0) \tag{6.8}$$

Hierbei ist $p_n(x_0)$ gerade der Rest der Division des Polynoms $p_n(x)$ durch den Linearfaktor. Wir erhalten dabei ein im Grad um eins vermindertes Polynom. Das klassische Verfahren wäre eine Polynomdivision durchzuführen, wie in Kapitel 6.2 gezeigt. Diese Methode ist aber recht mühsam. Eleganter ist die Nullstelle in der Tabelle auszuwerten:

	2	0	-5	0	4	1
-1	_	-2	2	3	-3	-1
	2	-2	-3	3	1	0

$p_4(x)$

Wie oben bereits gesagt, verkörpern die Felder links von $p_5(x_0)$ die Koeffizienten des Quotienten der Polynomdivison mit dem Linearfaktor $(x\text{-}x_0)$. Der Rest der Division ist Null, da wir eine Nullstelle ausgewertet haben und tatsächlich das Feld rechts Null ergibt. Wir erhalten also das Resultat:

$$\left(2x^5 - 5x^3 + 4x + 1\right) : (x + 1) = 2x^4 - 2x^3 - 3x^2 + 3x + 1$$

6.4.2 Ableitungen mit dem Horner Schema

Mit dem Horner Schema lassen sich auch Ableitungswerte berechnen. Wie vorher gezeigt gilt:

$$p_n(x) = p_{n-1}(x)(x - x_0) + p_n(x_0)$$

Nach der Definition des Differenzialquotienten gilt:

$$p_n'(x) := \lim_{x \to x_0} \frac{p_n(x) - p_n(x_0)}{x - x_0} = p_{n-1}(x) \tag{6.9}$$

Für $x \to x_0$ gilt also: $p_n'(x_0) = p_{n-1}(x_0)$. Die Koeffizienten für $p_{n-1}(x)$ haben wir durch einmaliges Anwenden des Horner Schemas erhalten. Um die Ableitung $p_n'(x_0)$ zu berechnen müssen wir das Horner Schema ein zweites Mal anwenden. Wir erhalten ein *zweistufiges Horner Schema* der Form:

	a_n	a_{n-1}	...	a_1	a_0
x_0	_	$a_n x_0$	...		
	a_n	$a_n x_0 + a_{n-1}$	...		$p_n(x_0)$
x_0	_	$a_n x_0$	...		
	a_n	$a_n x_0 + a_n x_0 + a_{n-1}$	...	$p'_n(x_0)$	

Konkret angewandt auf unser vorheriges Beispiel $2x^5 - 5x^3 + 4x + 1$ an der Stelle x_0=4 wird dies:

	2	0	-5	0	4	1
4	_	8	32	108	432	1744
	2	8	27	108	436	1745
4	_	8	64	364	1888	
	2	16	91	472	2324	

Eine Kontrollrechnung mit den Mitteln der Analysis zeigt die Richtigkeit:

$$p_5(x) = 2x^5 - 5x^3 + 4x + 1 \qquad p_5'(x) = 10x^4 - 15x^2 + 4$$
$$p_5'(4) = 2324$$

Dieses Verfahren kann natürlich auch auf die höheren Ableitungen erweitert werden. Jedoch müssen hier zur Bildung einer höheren Ableitung immer alle tieferen Ableitungen in diesem Punkt vorliegen.

Bei höheren Ableitungen (f'',f''',etc.) ist das Resultat mit der Fakultät des Grades der Ableitung zu multiplizieren.

6.4.3 Implementierung in EXCEL mit Visual Basic (VBA)

Obwohl in EXCEL Polynome direkt mit Potenzierung ausgewertet werden können ist es sinnvoll, Polynomauswertungen mit dem Hornerschema vorzunehmen:

1. Es werden weniger zeitintensive Rechenoperationen durchgeführt. Das bringt besonders bei Berechnungen von Wertetabellen Vorteile.

2. Werden Polynome höheren Grades (in der Regel deg $p > 10$) direkt mit Potenzieren der einzelnen Polynomglieder bestimmt, so können sich Rechenungenauigkeiten kumulieren bis das Resultat unbrauchbar wird. Die Praxis zeigt aber, dass dies eher selten der Fall ist.

Wir zeigen eine mögliche Implementierung einer Polynomauswertefunktion in Visual Basic für Applikationen:

```
' Auswerten einer Polynomfunktion n-ten Grades an der Stelle x
' Die Polynomfunktion wird mit Koeffizentenarray a0,..,an definiert und an der Stelle
' x ausgewertet und als Funktionswert retourniert.
' Die Berechnung erfolgt mit dem Horner-Schema.
'
' Autor: Gerhard Krucker
' Datum: 19.8.1995, 22.9.1996
'
Function PolynomEval(a, x As Double) As Double
Dim px                      ' Polynomfunktionswert '
Dim Polynomgrad As Integer   ' Grad des Polynomes, das die Koeffizienten in a darstellen '
Dim i

  Polynomgrad = a.Count - 1  ' Grad des Polynoms bestimmen = Anzahl Koeffizienten-1 '

  px = a(Polynomgrad + 1)
  For i = Polynomgrad To 1 Step -1
       px = px * x + a(i)
  Next i

  PolynomEval = px       ' Funktionswert retournieren '

End Function
```

Die Funktion wird als Benutzerdefinierte Funktion über den Funktionsassistenten angesprochen und liefert die Auswertung des Polynoms mit den Koeffizienten $a_0..a_n$ an der Stelle x zurück.

6.4.4 Zweizeiliges Horner Schema

Sollen von einem Polynom quadratische Faktoren der Form $x^2 - px - q$ abgespalten werden, so wird hierzu ein zweizeiliges Horner Schema benötigt.
Hierbei werden in der ersten Zeile die Koeffizienten des Polynoms notiert. In den nächsten beiden Zeilen werden p und q notiert.

	a_n	a_{n-1}	a_{n-2}	...	a_2	a_1	a_0
q	–	–	$a_n q$	...			
p	–	$a_n p$		...			
	a_n	$a_n p+a_{n-1}$		...		r_1	r_0

$p_{n-2}(x)$

In der dritten Zeile wird das p-fache des vorangegangenen Elements aus der vierten Zeile notiert, in der vierten Zeile das q-fache des vorletzten Elementes aus der vierten Zeile.

Ist nun $x^2 - px - q$ kein Teiler des Polynoms $p_n(x)$, so verbleibt ein linearer Restterm bestehend aus r_0, r_1. Wir können also das Polynom folgendermassen zerlegen:

$$p_n(x) = p_{n-2}(x) + r_1(x - p) + r_0 \tag{6.10}$$

Beispiel:
Wir möchten vom Polynom $x^5 - 7x^4 + 18x^3 - 14x^2 - 15x + 25$ den quadratischen Faktor $x^2 - 4x + 5$, der durch die konjugiert komplexen Nullstellen $x_{1,2} = 2 \pm j$ entsteht, abspalten. Dafür erhalten wir folgendes Horner Schema:

	1	-7	18	-14	-15	25
-5	–	–	-5	15	-5	-25
4	–	4	-12	4	20	0
	1	-3	1	5	0	0

$p_3(x)$ r_1 r_0

Eine Kontrolle durch Ausmultiplizieren bestätigt:

$$(x^2 - 4x + 5)(x^3 - 3x^2 + x + 5) = x^5 - 7x^4 + 18x^3 - 14x^2 - 15x + 25$$

Dieses zweizeilige Horner Schema ist die Grundlage für Nullstellenverfahren, welche auch komplexe Nullstellen bestimmen können (z.B. Bairstowsche Methode). Diese arbeiten so, dass sie quadratische und lineare Faktoren abspalten und diese dann separat mit Lösungsformeln zerlegen.

6.5 Newton Methode für Polynome

Ziel des Kapitels ist ein universelles Programm zu entwickeln, das bequem alle Nullstellen eines Polynoms mit reellen Koeffizienten bestimmt.

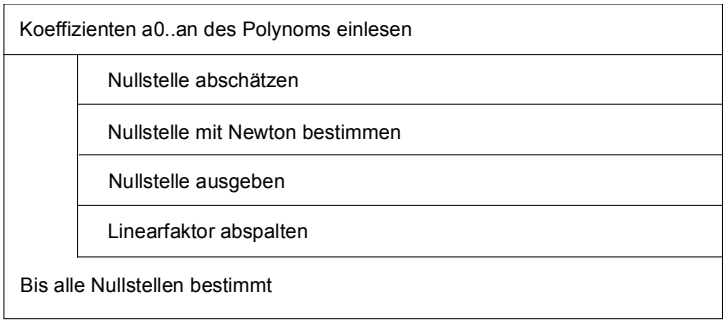
Dazu werden alle in Einzelteilen gezeigten Methoden nun zu einer Gesamtlösung zusammengefügt. Dies erfolgt modular, indem wir einzelne Funktionen implementieren: Koeffizienten einlesen, Funktionsauswertung mit dem Verfahren nach Horner, Linearfaktoren abspalten und ein Abschätzen der Nullstelle.

In einem ersten Ansatz entwickeln wir ein Programm welches ausschliesslich reelle Nullstellen bestimmt. In einer Erweiterung zeigen wir wie auch komplexe Nullstellen bestimmt werden können.

Zur Bestimmung der reellen Nullstellen kann man folgenden Ablauf charakterisieren:

1. Startwert für Nullstelle Abschätzen nach Kapitel 6.3
2. Nullstelle mit Newton-Verfahren bestimmen
3. Linearfaktor abspalten
4. Wiederholen der Punkte 1..3 für alle Nullstellen

Durch Abspalten des Linearfaktors erhalten wir das reduzierte Polynom vom Grad (*i-1*). Das garantiert, dass im nächsten Durchlauf das Newton-Verfahren nicht wieder gegen dieselbe Nullstelle konvergiert, ausser wenn sie tatsächlich eine mehrfache Nullstelle vorliegt.



Eine mögliche Implementierung dieses Entwurfes mit einigen Verfeinerungen:

```
/* Nullstellenbestimmung nach Newton fuer Polynome.
   Entwurf eines Programmes, das die reellen Nullstellen eines Polynomes bestimmt gemaess
   dem Entwurf im Skript.

   Autor: Gerhard Krucker
   Datum: 24.2.1995
   Sprache: MS Visual C V1.5 (Quick-Win)
*/

#include <stdio.h>
#include <math.h>

#define MAX 10                  /* Maximaler Grad des Polynoms */
#define ITER_MAX 100            /* Maximale Anzahl Iterationen */
#define EPSILON 1E-12           /* Genauigkeit der zu bestimmenden Nullstelle */

/* Funktionsprototypen fuer extern definierte Funktionen */
void deflate(double a[], int *n, double x0);   /* Abspalten eines Linearfaktors (x-x0) */
void get_pcoeff(double a[], int *n);           /* Einlesen der Koeffizienten */
double zero_guess(double a[], int n);          /* Abschaetzen einer Nullstelle des Polynoms */

/* Auswerten des Polynomes und der Ableitung an der Stelle x
   nach dem Verfahren nach Horner.
   pn ist das Array mit den Koeffizienten des Polynoms a[0]..a[n]
   x ist die auszuwertende Stelle
   n ist der Grad des Polynoms
   *fx ist der Funktionswert, der ueber Zeigerreferenz zurueckgegeben wird
   *dfx ist der Ableitungswert, der ueber Zeigerreferenz zurueckgegeben wird.

   Bemerkung: Die Wahl des Datentyps long double ist hier notwendig, da sonst
   der Fehler durch Stellenausloeschung zu gross wird.
*/

void horner(double pn[], double x, int n, double *fx, double *dfx)
{ int i;
  long double a[MAX];   /* Array für das Horner Schema */
  long double df;

  for (i=0; i<=n; i++) a[i]=pn[i];    /* Koeffizienten für das Horner Schema uebertragen */

  for (i=n; i > 0;i--)                /* Horner Schema durchfuehren */
       a[i-1]=a[i]* x + a[i-1];
  *fx = (double) a[0];           /* Wertrueckgabe der Auswertung pn[x] ueber Zeigerreferenz */

  df=a[n];
  for (i=n; i > 1;i--)              /* Horner Schema fuer die Ableitung */
      df=a[i-1] + df * x;
  *dfx=(double) df;             /* Wertrueckgabe der Auswertung pn'[x] ueber Zeigerreferenz */
}

main()
{ double pn[MAX];               /* Koeffizienten des Polynoms */
  int n;                        /* Grad des Polynoms */
  int iter;                     /* Iterationszaehler */
  int x0count;                  /* Nullstellenzeahler */
  double dfx,fx,x,xalt;

  get_pcoeff(pn,&n);            /* Koeffizienten des Polynoms einlesen */

  x0count=0;
  while (n > 0)
  {  x=zero_guess(pn,n);           /* Nullstelle abschaetzen */
     iter = 0;
     do {
       horner(pn,x,n,&fx,&dfx);
       xalt=x;
       x = x- fx/dfx;
       if (iter++ > ITER_MAX)
         { printf("Maximale Iterationszahl erreicht!");
           return 0;
         }
      } while (fabs(xalt-x) > EPSILON);
     printf("Nullstelle %d: %lg\n",++x0count,x);
     deflate(pn,&n,x);                     /* Linearfaktor abspalten */
  }
  return 0;
}
```

Ein Test des Programmes mit dem Polynom $x^5 - 6x^4 + 10x^3 - 11x + 6$ liefert die Nullstellen:

```
Eingabe der Koeffizienten des Polynoms:  (leere Eingabe = Beenden)
a0: 6
a1: -11
a2: 0
a3: 10
a4: -6
a5: 1
a6:
Nullstelle 1: 3
Nullstelle 2: 2
Nullstelle 3: 1
Nullstelle 4: 1
Nullstelle 5: -1
```

Bei der praktischen Anwendung sehen wir, dass das Programm nicht immer alle reellen Nullstellen bestimmen kann. Der Grund hierfür ist, dass das Newton-Verfahren bei mehrfachen Nullstellen nur langsam konvergiert.
Unterscheiden sich hier zwei aufeinander folgende Näherungswerte aufgrund der langsamen Konvergenz um weniger als die geforderte Genauigkeitschranke ε, so ist die Forderung scheinbar erfüllt obwohl der Fehler wesentlich grösser ist.

Hierzu ein konkretes Beispiel:
Wir bestimmen vom Polynom $x^8 - 4x^6 + 6x^4 - 4x^2 + 1$ die Nullstellen mit dem Programm. Wir erhalten das Resultat:

```
Nullstelle 1: 1.00001
Nullstelle 2: 0.99999
Maximale Iterationszahl erreicht!
```

Wir können durch Probieren leicht herausfinden, dass dieses Polynom vierfache Nullstellen bei -1 und +1 besitzt. Aufgrund der langsamen Konvergenz wird die Nullstelle mit einem zu grossen Fehler berechnet.

Aufgrund der ungenauen Nullstelle wird dann versucht ein Linearfaktor abzuspalten. Dies geht aufgrund der Ungenauigkeit nicht ohne Divisionsrest. Wir erhalten ein fehlerbehaftetes reduziertes Polynom. Dieser Fehler pflanzt sich fort und führt zu falschen Ergebnissen.

6.5.1 Probleme beim Newton-Verfahren

Generell kann man festhalten, dass das Newton-Verfahren bei Polynomen mit mehrfachen Nullstellen problembehaftet ist. Zwar kann durch Erhöhen der internen Rechengenauigkeit und Vorgabe kleiner Fehlerschranken eine Verbesserung erreicht werden. In der Regel häufen sich Fehler jedoch derart, dass der Newton-Algorithmus unbrauchbar wird.

Betrachten wir hierzu ein quadratisches Polynom $x^2 - 2x + 1$ mit einer doppelten Nullstelle bei 1. Hingegen besitzt das Polynom $x^2 - 2x + 1.00001$ keine reellen Nullstellen. Genau ein solches Polynom entsteht bei der Abspaltung im obigen Beispiel.

Ein Nullstellenprogramm für Polynome ist erst brauchbar, wenn es auch die komplexen Nullstellen bestimmen kann. Nachfolgend wird der Entwurf eines solchen Programmes gezeigt.

6.6 Nullstellenbestimmung nach Bairstow

Mit diesem Verfahren können auch komplexe Nullstellen von Polynomen mit reellen Koeffizienten bestimmt werden. Es arbeitet nach dem Prinzip des Abspaltens von quadratischen Faktoren mit Hilfe des zweizeiligen Horner Schemas und Newton-Iteration. Diese abgespaltenen quadratischen Faktoren werden dann mit der Lösungsformel aufgelöst und liefern dann je zwei Nullstellen. Wesentlich ist bei dieser Methode, dass ohne komplexe Rechnung gearbeitet wird.

6.6.1 Begründung des Verfahrens nach Bairstow

Komplexe Nullstellen treten bei Polynomen mit reellen Koeffizienten immer als konjugiert komplexe Paare auf. Dass heisst, wenn $u + jv$ eine Nullstelle ist, dann ist auch $u - jv$ eine Nullstelle des Polynoms $p_n(x)$.

Durch Ausmultiplizieren der beiden komplexen Linearfaktoren erhalten wir einen quadratischen Faktor mit reellen Koeffizienten:

$$(x-(u+jv)(x-(u-jv)) = x^2 - 2ux + (u^2+v^2) \tag{6.11}$$

Dieser quadratische Faktor ist ein Teiler des Polynoms $p_n(x)$.Bairstow schlägt nun folgende Methode vor:

1. Vom Polynom $p_n(x)$ wird ein quadratischer Faktor abgespalten:

   $$p_n(x) = p_{n-2}(x)(x^2 - px - q) \tag{6.12}$$

   Ein solcher quadratischer Faktor existiert immer für $n \geq 2$.

1. Hat man einen solchen quadratischen Faktor gefunden, so bestimmt man die Nullstellen mit der Lösungsformel für die quadratische Gleichung, wobei gegebenenfalls komplexe Nullstellenpaare entstehen.

2. Das Verfahren wird wiederholt bis nur noch ein Restpolynom ersten oder zweiten Grades übrig bleibt.

Die Koeffizienten p und q für die Abspaltung

$$p_n(x) = p_{n-2}(x)(x^2 - px - q)$$

werden durch ein Iterationsverfahren bestimmt. Die Idee ist dabei folgende:

Ist $x^2 - px - q$ kein Teiler des Polynoms $p_n(x)$, so verbleibt ein linearer Restterm $r_1(x-p) + r_0$. (Siehe auch zweizeiliges Horner Schema). Man variiert p und q solange bis dieser Restterm Null wird:

$$r_1(x-p) + r_0 = 0 \tag{6.13}$$

Dabei hängen sowohl r_0 wie auch r_1 von p und q ab, aber nicht von x. Es gilt, dass sowohl r_0 wie auch die Summe (der ganze Restterm) Null sein muss. Wir erhalten ein nichtlineares Gleichungssystem mit zwei Unbekannten:

$$f(p,q) := r_1(p,q) = 0$$
$$g(p,q) := r_0(p,q) + r_1(p,q)p = 0$$

Dieses Gleichungssystem kann näherungsweise mit dem Newton-Verfahren gelöst werden. Wir erhalten dafür folgende Iterationsformeln:

$$p_{n+1} = p_n - \frac{f \cdot g_q - f_q g}{f_p g_q - f_q g_p} \tag{6.16}$$

$$q_{n+1} = q_n - \frac{f_p \cdot g - f g_p}{f_p g_q - f_q g_p} \tag{6.17}$$

Die Auswertung von *f* und *g* wird mit dem zweizeiligen Horner Schema vorgenommen. Die notwendigen partiellen Ableitungen lassen sich ebenfalls mit dem Schema berechnen:

	a_n	a_{n-1}	a_{n-2}		a_3	a_2	a_1	a_0
q	_	_	$a_n q$					
p	_	$a_n p$						
	a_n	$a_n p + a_{n-1}$					f	g
q	_	_	$a_n q$					
p	_	$a_n p$						
	a_n	$a_n p + a_{n-1}$			f_q	$f_p = g_q$	g_p	

Eine mögliche Implementierung des Verfahrens in C:

```
/* Nullstellenbestimmung fuer Polynome nach dem Verfahren von BAIRSTOW.

   Autor: Gerhard Krucker
   Datum: 28.2.1995
   Sprache MS Visual C V1.5 (QuickWin)
*/

#include <stdio.h>
#include <math.h>

#define MAX 10                      /* Maximaler Grad des Polynoms */
#define ITER_MAX 100                /* Maximale Anzahl Iterationen */
#define EPSILON 1E-12               /* Genauigkeit für p und q */

/* Funktionsprototypen fuer extern definierte Funktionen */
void get_pcoeff(double a[], int *n);           /* Einlesen der Koeffizienten */
double zero_guess(double a[], int n);          /* Abschaetzen einer Nullstelle des Polynoms */

int x0count;                          /* Globale Variable, zaehlt die gefundenen Nullstellen.  */

/* Nullstellen des quadratischen Gliedes mit der Formel der quadratischen Ergaenzung berechnen.
   Hier wird die vereinfachte Formel verwendet, die voraussetzt, dass p und q bestimmt sind. */
void quad_null(double p, double q)
{ double d;

  d =p * p / 4 - q;
  p = -p / 2;
  if (d >= 0)    /* Zwei reelle Nullstellen */
    { d = sqrt(d);
      printf("Nullstelle %d:  %lg\n",++x0count,p+d);
      printf("Nullstelle %d:  %lg\n",++x0count,p-d);
    }
   else    /* Zwei konjugiert komplexe Nullstellen */
    { d = sqrt(-(double)d);
      printf("Nullstelle %d:  %lg + j%lg\n",++x0count,p,d);
      printf("Nullstelle %d:  %lg - j%lg\n",++x0count,p,d);
    }
}

main()
{ double pn[MAX];              /* Koeffizienten des Polynoms */
  int n;                       /* Grad des Polynoms */
  int iter;                    /* Iterationszaehler */
  int x0count;                 /* Nullstellenzeahler */
  long double h1[MAX];         /* Erste Stufe des Horner Schemas (Funktionswert) */
  long double h2[MAX];         /* Zweite Stufe des Horner Schemas (Ableitungswert) */
  long double p,q;             /* Aktuelle Werte fuer p, q (verbesserte Naeherung nach 1 Schritt) */
  long double palt, qalt;      /* Werte vor dem Iterationschritt */
  int i;
```

```
  long double f,g;              /* Funktionswerte */
  long double fq,fp,gq,gp;      /* Partielle Ableitungen */
  long double dJ;               /* Determinante der Jacobi-Matrix (Matrix der partiellen Ableitungen) */

  get_pcoeff(pn,&n);            /* Koeffizienten des Polynoms einlesen */
  x0count=0;
  while (n > 2)
  {  p = zero_guess(pn,n);          /* p und q abschaetzen */
     q = -(p*p) / 4;
     iter = 0;

     do {
       /* Erste Stufe des Horner Schemas: p,q in pn auswerten */
       h1[n]=(long double)pn[n];                          /* Erste Spalte erzeugen */
       h1[n-1] = (long double)pn[n-1]+h1[n]*p;            /* Zweite Spalte enthaelt nur p */
       for (i=n-1; i > 0; i--)                  /* Restliche Spalten mit p und q ausrechnen */
           h1[i-1] = (long double) pn[i-1] + h1[i]*p+h1[i+1]*q;
       f=h1[1]; g=h1[0];                        /* Funktionswerte uebertragen */

       /* Zweite Stufe des Horner Schemas: Partielle Ableitungen an p und q bestimmen */
       h2[n]=(long double)h1[n];
       h2[n-1] =(long double) pn[n-1] + h2[n]*p;
       for (i=n-1; i > 1; i--)
            h2[i-1] = h1[i-1] + h2[i]*p+h2[i+1]*q;
       fq=h2[3]; fp=gq=h2[2]; gp=h2[1];      /* Partielle Ableitungswerte uebertragen */

       /* Iterationsschritt im Newton-Verfahren für zwei Variablen ausfuehren */
       dJ = fp*gq - fq*gp;          /* Determinante der Jacobi Matrix bestimmen */
       palt = p; qalt = q;
       p = p - (f * gq - fq * g) / dJ;
       q = q - (fp * g - f * gp) / dJ;
       if (iter++ > ITER_MAX)
         { printf("Maximale Iterationszahl erreicht!");
           return 0;
         }
      } while ((fabsl(palt-p)+fabsl(qalt-q)) > EPSILON);

     quad_null((double)-p,(double)-q);

     /* Quadratischen Faktor abspalten (Deflation), mit dem Verfahren von Horner */
     h1[n]= (long double)pn[n];
     h1[n-1] =(long double) pn[n-1]+h1[n       /* Zweite Spalte enthaelt nur p */
     for (i=n-1; i > 0; i--)                   /* Restliche Spalten mit p und q ausrechnen */
           h1[i-1] =(long double) pn[i-1] + h1[i]*p+h1[i+1]*q;
     for (i=0; i <= n-2; i++)
        pn[i]=(double)h1[i+2];
     n = n-2;
  }

  /* Jetzt bleibt nur noch ein quadratisches oder lineares Glied uebrig. */
  if (n==1)
      printf("Nullstelle %d:  %lg + j%lg\n",++x0count,-pn[0]);
    else
     quad_null(pn[1],pn[0]);
  return 0;
}
```

Ein Testdurchlauf mit dem Polynom $x^6 - x^5 + 7x^4 + 13x_3 - 14x^2 + 14x - 20$ findet die Nullstellen:

```
Eingabe der Koeffizienten des Polynoms:  (leere Eingabe = Beenden)
a0: -20
a1: 14
a2: -14
a3: 13
a4: 7
a5: -1
a6: 1
a7:
Nullstelle 1:  1
Nullstelle 2:  -2
Nullstelle 3:  -2.37285e-021 + j1
Nullstelle 4:  -2.37285e-021 - j1
Nullstelle 5:  1 + j3
Nullstelle 6:  1 - j3
```

Wir erhalten hier also sehr gute Näherungen für die Nullstellen und sehen, dass der verschwindend kleine Realteil bei den Nullstellen 3 und 4 wohl auf Rechenungenauigkeiten zurückzuführen ist. Eine Verbesserung bezüglich höherer Genauigkeit durch Verkleinern der Schranke für ε bringt keine wesentliche Verbesserung. Deshalb ist es unumgänglich, solche Näherungen kritisch zu betrachten und bei Bedarf zu runden.

6.7 Rationale Funktionen

Rationale Funktionen sind per Definition der Quotient zweier Polynomfunktionen:

$$f(x) := \frac{\sum_i a_i x^i}{\sum_j b_j x^j} \quad resp. \quad f(x) := \frac{\sum_i (x - x_i)}{\sum_j (x - x_j)} \qquad (a_i, b_j \in \mathbb{R}, x_i, x_j \in \mathbb{C}) \qquad \textbf{Rationale Funktion} \tag{6.18}$$

Der Grad *i* des Zählerpolynoms kann durchaus unterschiedlich vom Grad *j* des Nennerpolynoms sein. Ist der Grad des Nennerpolynoms $j > 0$ spricht man von *gebrochen rationalen* Funktionen. Für uns sind nachfolgend nur gebrochen rationale Funktionen interessant.

Gebrochen rationale Funktionen eine Reihe spezieller Eigenschaften, die wir nachfolgend etwas näher untersuchen wollen. Wir halten dabei den Bezug zu elektrotechnischen Anwendungen, speziell zu den Übertragungsfunktionen von linearen Netzwerken.

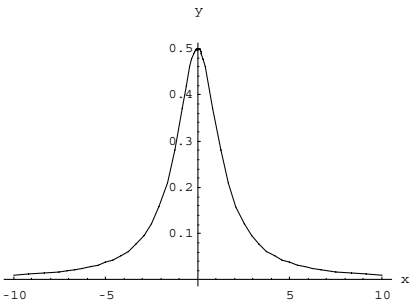
Anhand der Gradbetrachtung von Zähler- und Nennerpolynom sehen wir das asymptotische Verhalten:

$$i < j: \quad \lim_{x \to \infty} \frac{\sum_i a_i x^i}{\sum_j b_j x^j} = 0 \tag{6.19}$$

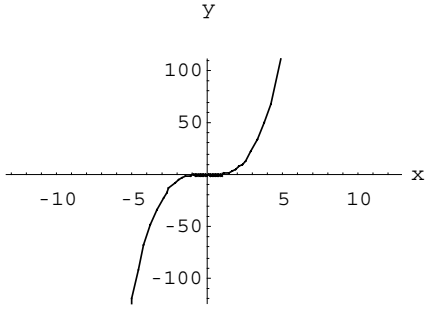
$$i > j: \quad \lim_{x \to \infty} \frac{\sum_i a_i x^i}{\sum_j b_j x^j} = \pm\infty \tag{6.20}$$

Beispiele:

1. Die Funktion $y(x)=\frac{1}{2+x^2}$ strebt für $x \to \pm\infty$ nach Null:

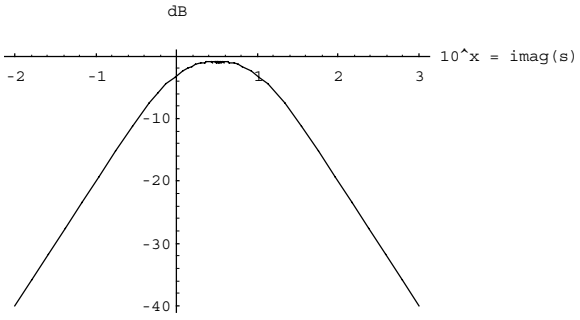


2. Die Funktion $y(x)=\frac{x^5}{1+x^2}$ strebt für $x \to \pm\infty$ nach einem uneigentlichen Wert:



3. In der Elektrotechnik werden die Übertragungsfunktionen und Amplitudengänge meist doppelt logarithmisch dargestellt. Eine Bandfilterfunktion mit Bandbreite mit ω_{g1}=1 und ω_{g2}=10 wäre hier eine rationale Funktion mit zwei Nullstellen im Nenner und einer Nullstelle im Zähler:

$$f(s)=\frac{10s}{(s+1)(s+10)} \qquad (s=\sigma+j\omega)$$



Mathematica:

```
f[s_]=10 s/((s + 1)(s + 10))
Plot[20 Log[10, Abs[f[I 10^x]]],{x,-1.5,2.5},
     AxesLabel->{"10^imag(s)","dB"}]
```

Für den Fall *i=j* erhalten wir einen unbestimmten Ausdruck, der nach den bekannten Regeln näher untersucht werden kann.

6.7.1 Pole und Nullstellen

Besondere Aussagekraft haben die **Nullstellen und Pole** einer rationalen Funktion:

Nullstellen: Nullstellen einer rationalen Funktion heissen diejenigen Werte x, die $f(x)=0$ liefern. Dies sind insbesondere die Nullstellen des Zählerpolynoms.

Pole: Pole heissen diejenigen Werte x, die $f(x)=\pm\infty$ liefern. Dies sind insbesondere die Nullstellen des Nennerpolynoms.

Besonders einfach kann man eine Pol-/Nullstellenbetrachtung bei rationalen Funktionen vornehmen, wenn die Polynome als Produkt von Linearfaktoren vorliegen. Ansonsten sind die Nullstellen des Zählers und Nenners mit geeigneten Methoden zu bestimmen.

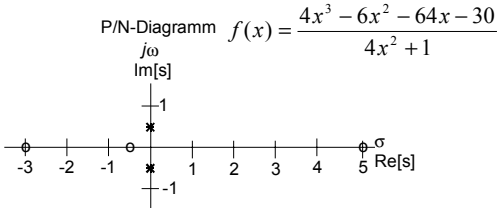
Mit Hilfe einer Pol-/Nullstellenbetrachtung der Übertragungsfunktion erhält man Aufschluss über die Stabilität eines Systems (stabil, instabil, schwingfähig).

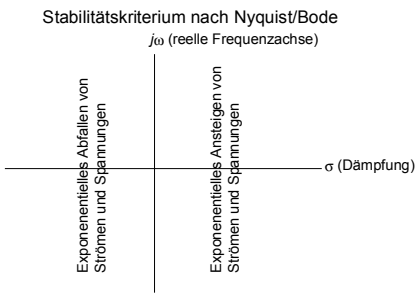
Beispiele:

$f(s)=\dfrac{10s}{(s+1)(s+10)}$ hat eine Nullstelle bei s=0 und Pole bei s=-1 und s=-10:

$f(x)=\dfrac{4x^3-6x^2-64x-30}{4x^2+1}$ hat Nullstellen bei s=-3, s=-0.5, s=5 und Pole bei $s=\pm 0.5j$.

Oft werden zur grafischen Beurteilung der Stabilität eines Systems die Pole und Nullstellen in einem Diagramm auf der Gausschen Zahlenebene aufgetragen (sog. Pol-/Nullstellendiagramm):





Zur Stabilität lassen sich demzufolge die Aussagen formulieren:

1. Hat ein System mehr Nullstellen als Pole, oder mehr als ein Pol im Ursprung, oder Pole rechts der $j\omega$-Achse, so ist es zwingend instabil.

2. Hat ein System konjugiert komplexe Pole, so ist es schwingfähig.

3. Ein System hat minimale Phasendrehung, wenn alle Nullstellen links oder auf der jω-Achse liegen.

   Bei einem Netzwerk mit minimaler Phasendrehung herrscht zwischen Amplitudengang und Phasengang eine integro-differenzielle Beziehung, d.h. die Phase entspricht, vom Verlauf her, der Ableitung des Amplitudenganges.

Liegt der Amplitudengang als stückweise stetige Funktion vor (Bode-Diagramm), dann gilt:

- Pro -20dB Amplitudensteigung dreht die Phase um -90°.
- Pro +20dB Amplitudensteigung dreht die Phase um +90°.

Netzwerke mit Leiterstruktur haben zwingend minimale Phasendrehung. Bei Netzwerken mit Brückenstruktur können Fälle von nicht minimaler Phasendrehung auftreten.

6.7.2 Konstruktion von Übertragungsfunktionen aus dem Amplitudengang

Ist der Amplitudengang eines linearen Netzwerkes mit minimaler Phasendrehung bekannt, so kann auf einfache Weise die zugehörige Übertragungsfunktion entwickelt werden. Das Verfahren ist eine Heuristik.

Zu Konstruktion werden folgende Aussagen benutzt:

Pro -20dB-Knick im Amplitudengang hat die Übertragungsfunktion einen Pol. Die Polfrequenz (Nullstelle im Nennerpolynom) ist der negative Wert der Grenzfrequenz.

Pro +20dB-Knick im Amplitudengang hat die Übertragungsfunktion eine Nullstelle. Die Nullstellenfrequenz (Nullstelle im Zählerpolynom) ist der negative Wert der Grenzfrequenz.

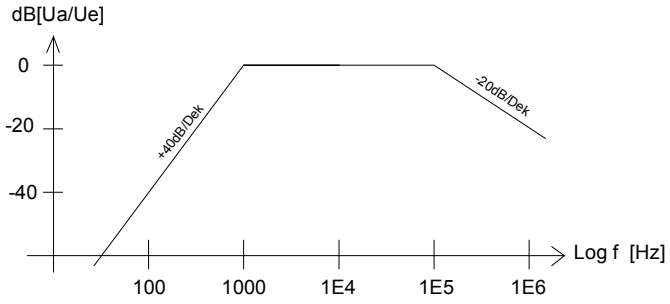
Per Definition beginnt die Kurve mit Steigung 0.

Somit lässt sich die Übertragungsfunktion *G(s)* formalisieren:

$$G(s) = k\frac{(s+s_{g11})(s+s_{g12})\ldots(s+s_{g1n})}{(s+s_{g21})(s+s_{g22})\ldots(s+s_{g2n)}} \quad (s, s_{gij} \in \mathbb{C}, k \in \mathbb{R}) \tag{6.21}$$

Die Zählernullstellen verkörpern hier die Frequenzen wo ein +20dB-Knick stattfindet. Die Nennernullstellen zeigen einen -20-dB-Knick an. *k* ist eine zu bestimmende Korrekturkonstante. Sie verschiebt die Kurve in vertikaler Richtung.

Beispiel: Konstruieren Sie den Amplitudengang *F(ω)* und den Frequenzgang *f(ω)*!



Da wir eine sinusförmige, konstante Anregung haben gilt σ=0 und wir erhalten $s=j\omega$.
Wir beurteilen zuerst den Kurvenverlauf bezüglich Pole/Nullstellen sowie Steigungen:

Frequenz		Steigung (nachfolgend)
0	Nullstelle (doppelt)	+40 dB/Dek
1000	Pol (doppelt)	0
10000	Pol	-20 dB/Dek

Die doppelte Nullstelle bei 0 entsteht aus der Aussage, dass die Kurve mit der Steigung 0 beginnt und wir mit +40 dB/Dek zum ersten Pol laufen.

Von der Struktur her wird die Übertragungsfunktion im Zähler ein (degeneriertes) Polynom 2. Grades und im Nenner ein Polynom 3. Grades haben:

$$G(s) = \frac{s^2}{(s - s_{g1})^2 (s - s_{g2})}$$

Durch Einsetzen von $s=j\omega$ und Überführen der Polfrequenzen in Polkreisfrequenzen können wir jetzt die Frequenzgangfunktion *f* bezüglich der Kreisfrequenz ω notieren:

$$f(j\omega) = \frac{(j\omega)^2}{(1000 \cdot 2\pi + j\omega)^2 (10000 \cdot 2\pi + j\omega)} k$$

Die Konstante *k* ist so zu bestimmen, dass die Kurve für einen Referenzwert stimmt.
Wir wählen hier den -3dB-Punkt bei 100kHz aus der Grafik und lösen die Gleichung auf:

$$\left| \frac{1}{1+j} \right| = \left| \frac{-2 \cdot 10^{10} \pi^2}{(2000\pi + j10^5 \cdot 2\pi)^2 (2 \cdot 10^5 \pi + j1 \cdot 10^5 \cdot 2\pi)} k \right|$$

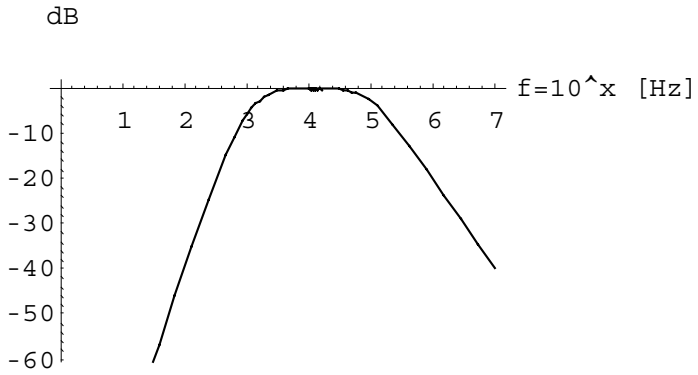
$$\frac{1}{\sqrt{2}} = \frac{4 \cdot 10^{10} \pi^2}{(1000^2 \cdot 4\pi^2 + 4 \cdot 10^{10} \pi^2)\sqrt{(4 \cdot 10^{10} \pi^2 + 4 \cdot 10^{10} \pi^2)}} k = \frac{4 \cdot 10^{10} \pi^2}{4\pi^2 (1000^2 + 10^{10}) 2\pi \sqrt{(10^{10} + 10^{10})}}$$

$$k = \frac{(1000^2 + 10^{10})\sqrt{2 \cdot 10^{10}}}{\sqrt{2} \cdot 10^{10}} = \frac{(1000^2 + 10^{10}) 2\pi}{10^5} = 200020\pi$$

Somit wird der Frequenzgang:

$$f(j\omega) = \frac{(j\omega)^2 200020\pi}{(2000\pi + j\omega)^2 (2 \cdot 10^5 \pi + j\omega)}$$

Ein Plot mit Kontrollrechnung für die Grenzfrequenzen zeigt uns die Richtigkeit der entwickelten Funktion:



Mathematica:

```
k=200020 Pi
f[w_]:=(I w)^2/((1000 * 2Pi + I w)^2 (10^5 2 Pi + I w))* k
Plot[20 Log[10, Abs[f[2 Pi 10^x]]],{x,0.5,7},
     AxesLabel->{"f=10^x [Hz]","dB"}]
```

Eine Kontrollrechnung liefert uns auch die (exakten) Beträge der Amplitude, 0.5 für 1kHz und 1/ sqrt(2) für 100kHz.
Etwas problematischer wird die Sache, wenn es sich um ein schwingfähiges System handelt. Bei Schwingfähigkeit würde mindestens ein Polpaar konjugiert komplex. Aus dem Amplitudengang kann aber nur schlecht auf Schwingfähigkeit geschlossen werden, so dass diese Methode für solche Fälle nicht sinnvoll ist.

6.7.3 Interpretation der komplexen Kreisfrequenz s

Bis anhin wurde die Kreisfrequenz mit jω allein definiert. Wieso braucht man jetzt eine neue Definition?
Grund: Wir betrachteten ausschliesslich Systeme, die mit einer konstanten Sinusschwingung mit der Frequenz *f* angesteuert wurden. Daraus resultiert die Kreisfrequenz jω=2jπ*f*

In der Regel werden aber Systeme nicht mit konstanten Spannungen angeregt, sondern sie verändern ihre Amplitude im Verlauf einer Zeitfunktion. Die anregende Spannung für ein Netzwerk besteht also nicht ausschliesslich aus einer Frequenzkomponente, sondern hat zusätzlich eine zeitabhängige Dämpfungskomponente.

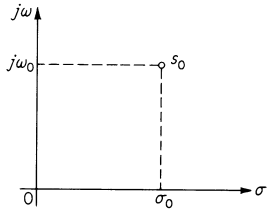
Aus der Fourieranalyse /synthese weiss man, dass jedes Signal *f(t)* aus Sinus- und Cosinusschwingungen zusammengesetzt werden kann. Mit der Eulerschen Darstellung kann man also schreiben:

$$f(t) = \sum_{i=0}^{n} k_i e^{s_i t} \tag{6.22}$$

Hierbei sind k_i und s_i geeignete komplexe Konstanten, welche durch komplexe Frequenzen dargestellt werden.

Wenn eine komplexe Frequenz s_0 hat also einen Realteil σ_0 und einen Imaginärteil ω_0 besitzt, dann gilt

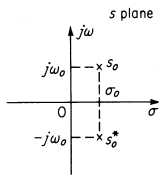
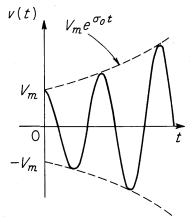
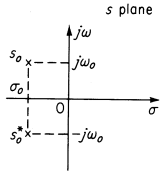
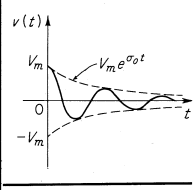
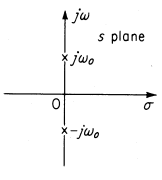
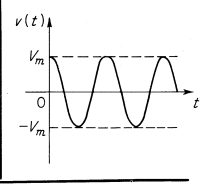
$$s_0 = \sigma_0 + j\omega_0 \tag{6.23}$$



und die komplexe Frequenz wird durch einen Punkt in der *s*-Ebene dargestellt:

Einen bildlichen Eindruck der Wirkung von verschiedenen Dämpfungen σ erhalten wir mit der Illustration:

Table 2.1.1. Locations of complex frequencies and the corresponding time functions

σ	Location of s_o	Equation for $v(t)$	Plot for $v(t)$
$\sigma > 0$ (s in the right half of the s plane)		$v(t) = V_m e^{\sigma_o t} \cos \omega_o t$	
$\sigma < 0$ (s in the left half of the s plane)		$v(t) = V_m e^{\sigma_o t} \cos \omega_o t$	
$\sigma = 0$ (s on the $j\omega$ axis)		$v(t) = V_m \cos \omega_o t$	

Quelle:
Ruston & Bordogna, Electric Networks:
Functions, Filters, Analysis
McGraw-Hill, 1966

6.8 Aufgaben

Polynome, Methode von Horner

1. Zerlegen Sie folgende Polynome in Linearfaktoren durch direkte Rechnung. Die Polynome vom Grad > 3 sind durch klassische Polynomdivision sukzessiv zur zerlegen :

   a.) x^2+9x+8
   b.) x^2+4x+5
   c.) x^3+x^2-6x
   d.) x^3+2x^2-5x-6
   e.) $x^4-10x^3+35x^2-50x+24$

2. Führen Sie folgende Polynomdivisionen mit Hilfe des Horner-Schemas durch:

   a.) $(x^2-14x+40):(x-10)$
   b.) $(2x^3-5x^2+18):(2x+3)$
   c.) $(8x^5-16x^4+30x^3+14x^2-28x+40):(2x^2-4x+8)$
   d.) $(2x^3-5x^2+18):(2x+3)$
   e.) $(x^9-x^8+x-1):(x-1)$
   f.) $(x^{10}-x^8+x^2-1):(x^2-1)$

3. Bestimmen Sie konkret *f(2)*, *f'(2)* und *f''(2)* der nachfolgenden Polynome:

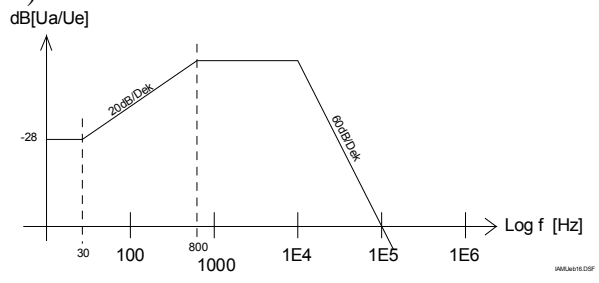
   a.) $3x^4-2x^2+3x-4$
   b.) $x^6-5x^5+6x^4+2x^3-3x^2-x-1$

Entwickeln Sie für die nachfolgenden grafischen Amplitudengänge die zugehörigen rationalen Funktionen $|F(\omega)|$ und F

\\Wendy\HTA-BE Dok\E98\IAM_Skript\Kapitel7\IAM_Kapitel7.doc
\\Wendy\HTA-BE Dok\E98\IAM_Skript\Kapitel8\IAM_Kapitel8.doc

4. *(jω):*

   a.)



   b.)