

## 5 Numerische Methoden der praktischen Mathematik

### 5.1 Einführung

Zwischen Mathematik und Praxis (Technik, Physik, Ökonomie, ...) ist über die Jahrhunderte hinweg eine befruchtende Wechselwirkung zu beobachten:

Mit den Rechenhilfsmitteln wurden zunächst nur formelmäßig lösbare Probleme mathematisch bearbeitet. Später, zur Zeit der mechanischen Rechenmaschinen, entwickelten die Mathematiker Näherungsverfahren, welche mittels sogenannten Rechenschemata gelöst wurden. Heute gestatten die modernen EDV-Anlagen neue numerische Lösungsverfahren, welche als Programme auf diesen Anlagen ausgeführt werden.

Ausgangspunkt der Lösung praktischer Probleme mittels der Mathematik sind häufig durch Messungen ermittelte Werte.

⇒ **Fehlerbehaftete Eingabewerte**, deren Fehlertoleranzen von der verwendeten Messtechnik abhängen.

Numerische Verfahren sind oft Näherungsverfahren.

⇒ **Verfahrensfehler**, welche sich bei der Berechnung fortpflanzen.

Mathematische Hilfsmittel, früher Rechenschieber, heute EDV-Anlagen, können nicht unbegrenzt genau arbeiten.

⇒ **Rechenfehler**, diese werden zwar durch die immer genauer arbeitenden Computer geringer, sind aber durch die Endlichkeit des Speichers und damit der Zahlendarstellung nicht zu beseitigen.

Die Lösung des Problems ist folglich mit Eingabefehlern, Verfahrensfehlern und Rechenfehlern behaftet. Die Korrektheit einer Lösung wird durch **Fehlerfortpflanzung** beeinflusst: Fehlerbehaftete Eingabewerten werden mittels fehlerbehafteten Verfahren und ungenauer Rechnung bearbeitet.

Die Numerik beschäftigt sich mit Verfahren zur Lösung mathematische Probleme, deren Fehlerfortpflanzung und Ergebnisauswertung. Einem numerische Verfahren liegt stets ein Algorithmus zugrunde, es enthält Kontrollen zur Fehlerfortpflanzung und entscheidet, in welchem Bereich die Lösungsmethode angewendet werden kann. Bei der Auswahl des Verfahrens sind Rechenzeit und Speicherökonomie stets zu beachten. Man beurteilt ein numerisches Verfahren streng nach ökonomischen Gesichtspunkten. Mathematisch elegante Lösungen sind häufig numerisch ungünstig.

**Prinzip: Ein verwendetes numerisches Verfahren muss die notwendige Genauigkeit gewähren, aber unnötigen Rechenaufwand vermeiden.**



## 5.2 Lösung linearer Gleichungssysteme

### Grundgedanke

Durch Umformen der Koeffizientenmatrix in eine geeignete Form (Dreiecks- oder Diagonalmatrix) werden Unbekannte eliminiert.

### Ausgangspunkt

Lineares inhomogenes oder homogenes Gleichungssystem mit einer Koeffizientendeterminante  $\neq 0$ .

### Das Eliminationsverfahren von Carl-Friedrich Gauß (1777-1855)

Durch Multiplikation und Addition von Gleichungen wird eine Koeffizientenmatrix in Dreiecksgestalt erzeugt, d.h. eine Gleichung mit einer Unbekannten, eine Gleichung mit zwei Unbekannten usw. usf. Danach wird aus der Gleichung mit einer Unbekannten diese berechnet, dann in die Gleichung mit zwei Unbekannten eingesetzt und die zweite Unbekannte berechnet usw. usf. bis alle Unbekannten aufgelöst sind.

**Betrachte drei lineare Gleichungen mit drei Unbekannten  $x_1$ ,  $x_2$  und  $x_3$ .**

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = a_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = a_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = a_3 \end{array} \quad \text{mit} \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \neq 0$$

**Rechenschema** für ein lineares Gleichungssystem mit 3 Unbekannten s. nächste Seite

### Beispiel

$$\begin{array}{l} x_1 - 2x_2 + x_3 = 0 \\ 2x_1 - 5x_2 + 2x_3 = -2 \\ 2x_1 - 2x_2 + 3x_3 = 7 \end{array} \quad \text{mit} \quad \begin{vmatrix} 1 & -2 & 1 \\ 2 & -5 & 2 \\ 2 & -2 & 3 \end{vmatrix} = 1 \cdot (-15 + 4) + 2 \cdot (6 - 4) + 1 \cdot (-4 + 10) = -1 \neq 0$$

### Rechenschema:

1	-2	1	0		$x_1 = 1$
2	-5	2	-2	$q = -2$	
-2	4	-2	0		
2	-2	3	7	$q = -2$	
-2	4	-2	0		
0	-1	0	-2		$x_2 = 2$
0	2	1	7	$q = 2$	
0	-2	0	-4		
0	0	1	3		$x_3 = 3$

### Verkürztes Rechenschema:

1	-2	1	0		1
2	-5	2	-2	-2	
2	-2	3	7	-2	
0	-1	0	-2		2
0	2	1	7	2	
0	0	1	3		3



**Rechenschema:**Elimination von  $x_1$ :

	$a_{11}$	$a_{12}$	$a_{13}$	$a_1$			$x_1 = \frac{a_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}$
	$a_{21}$	$a_{22}$	$a_{23}$	$a_2$	$q = -\frac{a_{21}}{a_{11}}$	*	
	$qa_{11}$	$qa_{12}$	$qa_{13}$	$qa_1$			
+	$a_{31}$	$a_{32}$	$a_{33}$	$a_3$	$q = -\frac{a_{31}}{a_{11}}$		
	$qa_{11}$	$qa_{12}$	$qa_{13}$	$qa_1$			
Elimination von $x_2$ :	0	$a'_{22}$	$a'_{23}$	$a'_2$			$x_2 = \frac{a'_2 - a'_{23}x_3}{a'_{22}}$
+	0	$a'_{32}$	$a'_{33}$	$a'_3$	$q = -\frac{a'_{32}}{a'_{22}}$		
	0	$qa'_{22}$	$qa'_{23}$	$qa'_2$			
+	0	0	$a''_{33}$	$a''_3$			$x_3 = \frac{a''_3}{a''_{33}}$

Voraussetzung:  $a_{11}, a'_{22}, a''_{33} \neq 0$



**Gauss.bas**

```
REM Gauss-Algorithmus
REM mit Unterprogrammtechnik und strukturierten Variablen
REM M. Meiler
REM 07.01.2002

REM Hauptprogramm
CLS : GOSUB eingabe: GOSUB gauss: GOSUB ausgabe: GOSUB ergebnis
END

REM Unterprogramme
eingabe:                                'zeilenweises Einlesen der Matrix
INPUT "Anzahl der Gleichungen: ", Laenge%
DIM a(Laenge%, Laenge% + 1)           'Vereinbarung einer Matrix als Feld
PRINT "Matrixeingabe"
FOR i% = 1 TO Laenge%
    FOR j% = 1 TO Laenge% + 1          '+1: rechte Seite des LGS
        INPUT ; " ", a(i%, j%)
    NEXT j%
    PRINT                               'Zeilenvorschub
NEXT i%
RETURN

gauss:                                  'Gauss-Algorithmus: Erzeugen der Dreiecksgestalt
FOR k% = 1 TO Laenge% - 1
    FOR i% = k% + 1 TO Laenge%
        q = -a(i%, k%) / a(k%, k%)
        ' PRINT "q="; q
        FOR j% = k% TO Laenge% + 1
            a(i%, j%) = a(i%, j%) + q * a(k%, j%)
        NEXT j%
        ' GOSUB ausgabe
    NEXT i%
NEXT k%
RETURN

ergebnis:                               'Berechnen der Lösung
DIM x(Laenge%)                          'Vereinbarung eines Lösungsvektors als Feld
FOR i% = Laenge% TO 1 STEP -1            'Berechnen der Lösung
    x(i%) = a(i%, Laenge% + 1)
    FOR j% = i% + 1 TO Laenge%
        x(i%) = x(i%) - a(i%, j%) * x(j%)
    NEXT j%
    x(i%) = x(i%) / a(i%, i%)
NEXT i%
PRINT "Lösung: ";                        'Ausgabe der Lösung
FOR i% = 1 TO Laenge%
    PRINT x(i%);
NEXT i%
RETURN

ausgabe:                                'zeilenweise Ausgabe einer Matrix
PRINT "Matrixausgabe"
FOR ii% = 1 TO Laenge%
    FOR jj% = 1 TO Laenge% + 1
        PRINT " "; a(ii%, jj%);
    NEXT jj%
    PRINT
NEXT ii%
INPUT "Weiter mit ENTER ", e$
RETURN
```



### 5.3 Numerische Integration

Die numerische Integration beschäftigt sich mit der angenäherten zahlenmäßigen Berechnung bestimmter Integrale.

$$F = \int_a^b f(x) dx$$

#### Grundgedanke

Die Funktion  $f(x)$  wird näherungsweise durch ein Interpolationspolynom  $P_r(x)$ ,  $r$  Grad des Polynoms, dargestellt.

$$F = \int_a^b f(x) dx \approx \int_a^b P_r(x) dx$$

#### Ausgangspunkt

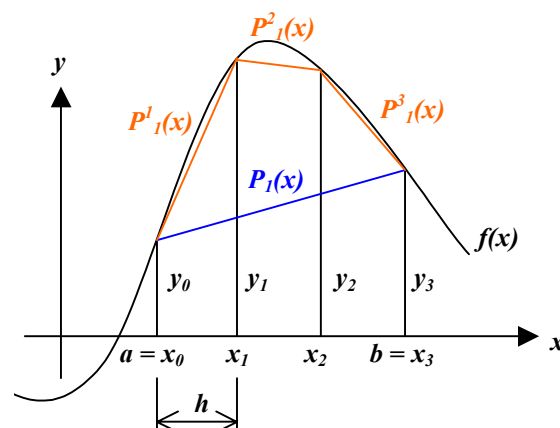
Gegeben sind  $n+1$  äquidistante (gleicher Abstand) Stützstellen  $x_0, x_1, \dots, x_n$  mit  $x_0 = a$ ,  $x_n = b$  und  $x_{i+1} = x_i + h$  (d.h.  $x_{i+1} = x_0 + (i+1)h$ ) für  $i = 0, 1, \dots, n-1$  und ihre Funktionswerte  $y_0 = f(x_0), y_1 = f(x_1), \dots, y_n = f(x_n)$ .

#### Prinzip

Durch die Punkte  $(x_i, y_i)$  werden Polynome gelegt, welche die Funktion angenähert darstellt.

#### 5.3.1 Trapezregel

Integration durch ein lineares Interpolationspolynom  $P_1(x) = a_1x + a_0$ :



⇒ Je mehr Stützstellen, desto genauer wird die Annäherung an die eigentliche Funktion.

$$F = \int_{a=x_0}^{b=x_2} f(x) dx \approx \int_{x_0}^{x_1} P_1(x) dx + \int_{x_1}^{x_2} P_2(x) dx + \int_{x_2}^{x_3} P_3(x) dx$$



Die Anwendung der Trapezflächenberechnung  $\int_{x_i}^{x_{i+1}} P_1^i(x) dx = \frac{h}{2}(y_i + y_{i+1})$  liefert dann

$$\int_{x_0}^{x_2} f(x) dx \approx \frac{h}{2}(y_0 + y_1) + \frac{h}{2}(y_1 + y_2) + \frac{h}{2}(y_2 + y_3) = \frac{h}{2}(y_0 + 2y_1 + 2y_2 + y_3).$$
**Allgemein**

$[a, b]$  wird in  $n$  Teilintervalle  $[x_0, x_1], [x_1, x_2], \dots, [x_{n-1}, x_n]$  mit  $x_0 = a$ ,  $x_n = b$  zerlegt. In jedem dieser Teilintervalle wird die Funktion  $f(x)$  durch ein lineares Interpolationspolynom angenähert. Das Integral über diese Polynome erhält man mittels der Flächenberechnung des entsprechenden Trapezes:

**Trapezregel:**  $F = \int_a^b f(x) dx \approx \frac{h}{2}(y_0 + 2y_1 + 2y_2 + \dots + 2y_{n-1} + y_n).$

**Beispiel**

Zu berechnen sei das Flächenintegral  $\int_0^1 \frac{dx}{1+x}$  durch lineare Interpolation. Es sollen dabei sieben Stützstellen verwendet werden  $\Rightarrow n = 6$ ,  $a = 0$ ,  $b = 1$ ,  $h = \frac{b-a}{n} = \frac{1}{6}$ :

$i$	0	1	2	3	4	5	6
$x_i$	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{5}{6}$	1
$y_i$	1	$\frac{6}{7}$	$\frac{3}{4}$	$\frac{2}{3}$	$\frac{3}{5}$	$\frac{6}{11}$	$\frac{1}{2}$
TR	1	0.85714	0.75	0.666667	0.6	0.54545	0.5

**Mathematische Lösung:**  $\int_0^1 \frac{dx}{1+x} = \int_1^2 \frac{dt}{t}$  mit  $t = 1+x \Rightarrow \frac{dt}{dx} = 1$   
 $= \ln|t| \Big|_1^2 = \ln 2 - \ln 1 = \ln 2 - 0 = \ln 2 \approx \underline{\underline{0.6931}}$  (TR).

**Trapezregel:**  $\int_0^1 \frac{dx}{1+x} \approx \frac{1}{12} \left( 1 + \frac{12}{7} + \frac{3}{2} + \frac{4}{3} + \frac{6}{5} + \frac{12}{11} + \frac{1}{2} \right) \approx \underline{\underline{0.6949}}$  (TR)

**Programm Trapez.bas :** **0.6948762** (einfache Genauigkeit)  
**0.6948761666666** (doppelte Genauigkeit)

Durch eine genauere Rechnung erhält man kein besseres Ergebnis. Die Verfahrensfehler kompensieren sich nicht durch die Erhöhung der Rechengenauigkeit.



**Trapez.bas**

```
REM Berechnung eines bestimmten Integrals
REM mittels Trapezregel
REM M. Meiler, 23.01.1998

REM Hauptprogramm
CLS : GOSUB eingabe: GOSUB trapez: GOSUB ausgabe
END

REM Eingabe der obere und unteren Grenze
REM und der Stuetzstellen
eingabe:
DO
  INPUT "Untere Integrationsgrenze a= ", a!
  INPUT "Obere Integrationsgrenze b(b>a)= ", b!
LOOP UNTIL (b! - a!) > 0

DO
  INPUT "Anzahl der Stuetzstellen (s > 1): ", s%
LOOP UNTIL s% >= 2

n% = s% - 1
h! = (b! - a!) / n%
DIM y!(n%)

FOR i% = 0 TO n%
  LOCATE , 5
  PRINT "x("; i%; ")="; a! + i% * h!;
  LOCATE , 30
  PRINT "y("; i%; ")=";
  INPUT " ", y!(i%)
NEXT i%
RETURN

REM Berechnung des Integrals mittels der Trapezregel
trapez:
y! = y!(0) + y!(n%)

FOR j% = 1 TO n% - 1
  y! = y! + 2 * y!(j%)
NEXT j%

y! = y! * h! / 2
RETURN

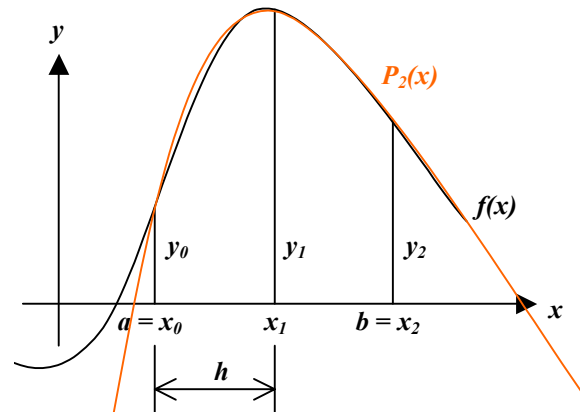
REM Ausgabe des Ergebnisses
ausgabe:
PRINT "Ergebnis: "; y!
PRINT
RETURN
```



### 5.3.2 Simpsonregel

Integration durch Interpolation mit Polynomen zweiten Grades (Parabeln)

$$P_2(x) = a_2x^2 + a_1x + a_0.$$



Gegeben sind drei Stützstellen, durch die eine Parabel zu legen ist.

Die Berechnung der Koeffizienten der Parabel erfolgt mittels Gaußschem Eliminationsverfahren unter Berücksichtigung, dass die Stützstellen der Funktion Punkte der Parabel sind:

$$\begin{array}{l} \downarrow \\ \begin{array}{l} a_0 + a_1x_0 + a_2x_0^2 = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 = y_1 \\ a_0 + a_1x_2 + a_2x_2^2 = y_2 \end{array} \end{array} \Rightarrow \begin{array}{l} a_0 = y_0 - \frac{(y_1 - y_0)x_0}{h} + \frac{(y_2 + y_0 - 2y_1)x_1x_0}{2h^2} \\ a_1 = \frac{y_1 - y_0}{h} - \frac{y_2 + y_0 - 2y_1}{2h^2}(x_1 + x_0) \\ a_2 = \frac{y_2 + y_0 - 2y_1}{2h^2} \end{array} \uparrow$$

$$\begin{aligned} P_2(x) &= a_2x^2 + a_1x + a_0 \\ &= \left( \frac{y_2 + y_0 - 2y_1}{2h^2} \right) x^2 \\ &\quad + \left( \frac{y_1 - y_0}{h} - \frac{y_2 + y_0 - 2y_1}{2h^2}(x_1 + x_0) \right) x \\ &\quad + \left( y_0 - \frac{(y_1 - y_0)x_0}{h} + \frac{(y_2 + y_0 - 2y_1)x_1x_0}{2h^2} \right) \\ &= y_0 + \frac{y_1 - y_0}{h}(x - x_0) + \frac{y_2 + y_0 - 2y_1}{2h^2}(x - x_0)(x - x_1) \\ &= y_0 + \Delta_1(x - x_0) + \Delta_2(x - x_0)(x - x_1) \end{aligned}$$

<p><b>Parabelgleichung:</b></p> $P_2(x) = a_2x^2 + a_1x + a_0 = y_0 + \Delta_1(x - x_0) + \Delta_2(x - x_0)(x - x_1)$ <p>mit <math>\Delta_1 = \frac{y_1 - y_0}{h}</math> und <math>\Delta_2 = \frac{y_2 + y_0 - 2y_1}{2h^2}</math></p>
--



**Integralberechnung**

$$\int_{x_0}^{x_2} P_2(x) dx = \int_{x_0}^{x_2} (y_0 + \Delta_1(x - x_0) + \Delta_2(x - x_0)(x - x_1)) dx$$

$$\int_{x_0}^{x_2} y_0 dx = y_0 x \Big|_{x_0}^{x_2} = (x_2 - x_0) y_0 = 2h y_0$$

$$\int_{x_0}^{x_2} \Delta_1(x - x_0) dx = \Delta_1 \left( \int_{x_0}^{x_2} x dx - \int_{x_0}^{x_2} x_0 dx \right) = 2h \Delta' \text{ mit } \Delta' = (y_1 - y_0)$$

$$\begin{aligned} \int_{x_0}^{x_2} \Delta_2(x - x_0)(x - x_1) dx &= \Delta_2 \left( \int_{x_0}^{x_2} x^2 dx + \int_{x_0}^{x_2} x_0 x_1 dx - \int_{x_0}^{x_2} (x_0 + x_1) x dx \right) \\ &= \frac{1}{3} h \Delta'' \text{ mit } \Delta'' = (y_2 + y_0 - 2y_1) \end{aligned}$$

$$\int_{x_0}^{x_2} P_2(x) dx = 2h(y_0 + \Delta' + \frac{1}{6} \Delta'') = 2h(y_0 + y_1 - y_0 + \frac{1}{6}(y_2 + y_0 - 2y_1)) = \frac{h}{3}(y_0 + 4y_1 + y_2)$$

**Keplersche Fassregel:**  $F = \int_{x_0}^{x_2} f(x) dx \approx \frac{h}{3}(y_0 + 4y_1 + y_2)$  .

**Allgemein**

$[a, b]$  wird in  $n$  Teilintervalle  $[x_0, x_1], [x_1, x_2], \dots, [x_{n-1}, x_n]$  mit  $x_0 = a$ ,  $x_n = b$  und  $n$  gerade zerlegt. In jeweils für zwei aufeinanderfolgenden Teilintervalle wird die Keplersche Fassregel angewendet und das Integral durch Summation der Ergebnisse näherungsweise berechnet:

$$F = \int_a^b f(x) dx \approx \frac{h}{3} [(y_0 + 4y_1 + y_2) + (y_2 + 4y_3 + y_4) + \dots + (y_{n-2} + 4y_{n-1} + y_n)]$$

**Simpsonregel:**  $F = \int_a^b f(x) dx \approx \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \dots + 2y_{n-2} + 4y_{n-1} + y_n)$

**Beispiel**

Berechnen des Integral  $\int_0^1 \frac{dx}{1+x}$  durch quadratische Interpolation (s.o.)  $\Rightarrow n=6$ ,

$$a=0, b=1, h=\frac{1}{6}:$$

**Simsonregel:**  $\int_0^1 \frac{dx}{1+x} \approx \frac{1}{18} \left( 1 + \frac{24}{7} + \frac{3}{2} + \frac{8}{3} + \frac{6}{5} + \frac{24}{11} + \frac{1}{2} \right) \approx \underline{\underline{0.6932}}$  (TR)

**Programm *Simpson.bas* :** **0.6931682** (einfache Genauigkeit)

Beide Ergebnisse sind besser als die der Trapezregel. Das mit der Simpsonregel entwickelte Interpolationspolynom liefert eine bessere Annäherung der Ausgangsfunktion.



***Simpson.bas***

```
REM Berechnung eines bestimmten Integrals
REM mittels Simpsonregel
REM M. Meiler, 23.01.1998

REM Hauptprogramm
CLS : GOSUB eingabe: GOSUB simpson: GOSUB ausgabe
END

REM Eingabe der obere und unteren Grenze
REM und der Stuetzstellen
eingabe:
DO
  INPUT "Untere Integrationsgrenze a= ", a!
  INPUT "Obere Integrationsgrenze b(b>a)= ", b!
LOOP UNTIL (b! - a!) > 0

DO
  INPUT "Anzahl der Stuetzstellen (s>1, ungerade) "; s%
LOOP UNTIL s% >= 2 AND s% MOD 2 = 1

n% = s% - 1
h! = (b! - a!) / n%
DIM y!(n%)

FOR i% = 0 TO n%
  LOCATE , 5
  PRINT "x("; i%; ")="; a! + i% * h!;
  LOCATE , 30
  PRINT "y("; i%; ")=";
  INPUT " ", y!(i%)
NEXT i%
RETURN

REM Berechnung des Integrals mittels der Simpsonregel
simpson:
y! = y!(0) + y!(n%)

FOR j% = 1 TO n% - 1 STEP 2
  y! = y! + 4 * y!(j%)
NEXT j%

FOR j% = 2 TO n% - 2 STEP 2
  y! = y! + 2 * y!(j%)
NEXT j%

y! = y! * h / 3
RETURN

REM Ausgabe des Ergebnisses
ausgabe:
PRINT " Ergebnis: "; y!
PRINT
RETURN
```



## 5.4 Polynomberechnung mittels Hornerschema

Polynome spielen in der Numerik eine besondere Rolle. Im vorherigen Abschnitt verwendete man für die Berechnung von Flächenintegralen Interpolationspolynome, welche die Funktion unter dem Integral angenähert darstellen und deren Integral man leicht berechnen kann. Oft kennt man die zu betrachteten Funktion überhaupt nicht und hat nur einige Funktionswerte (Messwerte) zur Verfügung. Auch dann wird diese Funktion mittels der diskret gegebenen Funktionswerte durch Interpolationspolynome approximiert, in dem man Polynome durch die vorhandenen diskreten Punkte legt.

### 5.4.1 Polynomberechnung

#### Grundgedanke

Durch geeignetes Ausklammern verhindert man das zeitaufwendige Berechnen der Potenzen in einem Polynom und spart damit Rechenzeit.

#### Ausgangspunkt

Polynom r-ten Grades 
$$P_r(x) = a_r x^r + a_{r-1} x^{r-1} + \dots + a_1 x + a_0$$

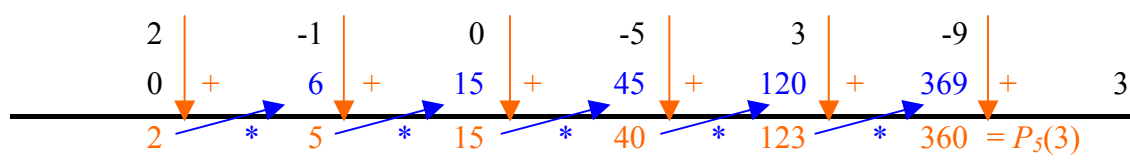
#### Beispiel

Gegeben sei das Polynom 5. Grades  $P_5(x) = 2x^5 - x^4 - 5x^2 + 3x - 9$ , gesucht sei der Wert des Polynoms an der Stelle  $x_0 = 3$ , also  $P_5(3)$ . Nun formt man das Polynom so um, dass keine Potenzen mehr vorhanden sind.

$$P_5(x) = 2x^5 - x^4 - 5x^2 + 3x - 9 = (((((2x - 1)x + 0)x - 5)x + 3)x - 9$$

Diese Darstellungsweise gestattet eine sehr schnelle Berechnung, auch mit dem Taschenrechner.

Das hierfür verwendete Rechenschema trägt den Namen Hornerschema, nach William Georg Horner (1786 - 1837), und sieht folgendermaßen aus:

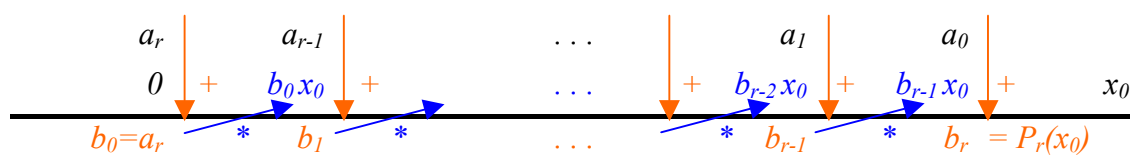


Die Probe zeigt, dass das Ergebnis korrekt ist.

#### Allgemein

$$P_r(x) = a_r x^r + a_{r-1} x^{r-1} + \dots + a_1 x + a_0 = (\dots (a_r x + a_{r-1}) x + \dots + a_1) x + a_0$$

**Horner'schema** für die Polynomwertbestimmung an der Stelle  $x_0$





mit  $b_0 = a_r$ ,  $b_{i+1} = a_{r-i-1} + b_i x_0$  für  $i = 1, \dots, r$  und  $b_r = P_r(x_0)$ .

Durch dieses Verfahren wird der Rechenaufwand wesentlich verringert:

### Polynomberechnung ohne Hornerschema:

$$\begin{aligned} \# \text{ Mult} &= \sum_{i=1}^{r+1} i = \frac{(r+1)(r+2)}{2} \\ \# \text{ Add} &= r \\ \# \text{ Mult} + \# \text{ Add} &= \frac{r^2 + 3r + 2 + 2r}{2} = \frac{r^2 + 5r + 2}{2} \end{aligned}$$

### Polynomberechnung mit Hornerschema:

$$\begin{aligned} \# \text{ Mult} &= r \\ \# \text{ Add} &= r \\ \# \text{ Mult} + \# \text{ Add} &= 2r \end{aligned}$$

### *Horner1.bas*

```
REM Polynomwertbestimmung mittels Hornerschema
REM M. Meiler, 21.01.2002

DO
  CLS
  GOSUB polynomeingabe: PRINT
  PRINT "Eingegebenes Polynom": GOSUB polynomausgabe: PRINT

  DO
    GOSUB argumenteingabe: PRINT: GOSUB horner1
    GOSUB polynomwertausgabe: PRINT
    PRINT "Berechnen des gleichen Polynoms ";
    INPUT "mit einem neuen Argument (j/n)? ", Weiter$
    LOOP WHILE Weiter$ = "j"

    INPUT "Berechnen eines neuen Polynoms(j/n)? ", Weiter$
    LOOP WHILE Weiter$ = "j"
  END

  REM Eingabe des Grades und der Koeffizienten des Polynoms
  polynomeingabe:
  DO
    INPUT "Grad des Polynoms(r > 0): ", r%
    LOOP UNTIL r% > 0

  REDIM a!(r%)

  PRINT "Koeffizienteneingabe"
  FOR i% = 0 TO r%
    PRINT "a!( "; i%; ")="; : INPUT " ", a!(i%)
  NEXT i%
  RETURN

  REM Eingabe des Arguments zur Berechnung des Polynomwertes
  argumenteingabe:
  INPUT "Argument x = ", x!
  RETURN
```



```

REM Berechnung des Polynomwertes mittels Horner
horner1:
b! = 0

FOR k% = r% TO 0 STEP -1
  b! = a!(k%) + b! * x! ' : PRINT "b("; k%; ")="; b!
NEXT k%

y! = b!
RETURN

REM Ausgabe des Polynoms in ueblicher Schreibweise
polynomausgabe:
FOR ii% = 0 TO r%
  PRINT a!(ii%); "x^"; ii%;: IF ii% <> r% THEN PRINT "+ ";
NEXT ii%
PRINT
RETURN

REM Ausgabe des berechneten Polynomwertes
polynomwertausgabe:
PRINT "P( "; x!; ")="; y!
RETURN

```

## 5.4.2 Berechnung der 1. Ableitung eines Polynoms

Aus dem Hornerschema geht hervor, dass

$$b_0 = a_r, b_{i+1} = a_{r-i-1} + b_i x_0 \text{ für } i = 1, \dots, r \text{ und } b_r = P_r(x_0).$$

Umgeformt ergibt sich

$$a_r = b_0 \text{ und } a_{r-i-1} = b_{i+1} - b_i x_0.$$

Setze  $j = r - i - 1$ , so ist

$$a_r = b_0 \text{ und } a_j = b_{r-j} - b_{r-j-1} x_0 \text{ und wegen } b_r = P_r(x_0) \text{ folgt}$$

$$\begin{aligned}
 P_r(x) &= a_r x^r + a_{r-1} x^{r-1} + a_{r-2} x^{r-2} + \dots + a_1 x + a_0 \\
 &= \underline{b_0 x^r} + \underline{(b_1 - b_0 x_0) x^{r-1}} + \underline{(b_2 - b_1 x_0) x^{r-2}} + \dots + \underline{(b_{r-1} - b_{r-2} x_0) x} + \underline{(b_r - b_{r-1} x_0)} \\
 &= \underline{(x - x_0) b_0 x^{r-1}} + \underline{(x - x_0) b_1 x^{r-2}} + \dots + \underline{(x - x_0) b_{r-2} x} + \underline{(x - x_0) b_{r-1}} + b_r \\
 &= (x - x_0) (b_0 x^{r-1} + b_1 x^{r-2} + \dots + b_{r-2} x + b_{r-1}) + \underline{b_r} \\
 \Rightarrow P_r(x) &= (x - x_0) (b_0 x^{r-1} + b_1 x^{r-2} + \dots + b_{r-2} x + b_{r-1}) + \underline{P_r(x_0)} \\
 \Rightarrow \frac{P_r(x) - P_r(x_0)}{x - x_0} &= p_r(x) \text{ mit } p_r(x) = b_0 x^{r-1} + b_1 x^{r-2} + \dots + b_{r-2} x + b_{r-1}
 \end{aligned}$$



$\Rightarrow p_r(x)$  kann zur Berechnung der 1. Ableitung an der Stelle  $x = x_0$  genutzt werden:

Differenzenquotient

$$\frac{dP(x)}{dx} = \lim_{x \rightarrow x_0} \frac{P(x) - P(x_0)}{x - x_0} = P'(x_0) \quad \text{mit} \quad P'(x) = b_0 x^{r-1} + b_1 x^{r-2} + \dots + b_{r-2} x + b_{r-1}$$

### Erweitertes Hornerschema

#### Beispiel

$P_5(x) = 2x^5 - x^4 - 5x^2 + 3x - 9$ , gesucht  $P'_5(3)$ .

2	-1	0	-5	3	-9							
0	+	6	+	15	+	45	+	120	+	369	+	3
2	*	5	*	15	*	40	*	123	*	360		
0	+	6	+	33	+	144	+	552	+			3
2	*	11	*	48	*	184	*	675				

#### Probe

$$P_5(x) = 2x^5 - x^4 - 5x^2 + 3x - 9, \quad P'_5(x) = 10x^4 - 4x^3 - 10x + 3,$$

$$P_5(3) = 486 - 81 - 45 + 9 - 9 = 360, \quad P'_5(3) = 810 - 108 - 30 + 3 = 675.$$

### Allgemein

**Erweitertes Hornerschema** für die Polynomwertbestimmung und der 1. Ableitung an der Stelle  $x_0$

The diagram illustrates the recursive calculation of polynomial coefficients for  $P_r(x_0)$  and  $P'_r(x_0)$ . It shows two horizontal lines representing the polynomials, with coefficients  $a_r, a_{r-1}, \dots, a_1, a_0$  and  $b_0, b_1, \dots, b_{r-1}, b_r$  for the top line, and  $c_0, c_1, \dots, c_{r-1}, c_r$  for the bottom line. The coefficients are calculated recursively using the recurrence relations:

$$b_i = a_i + b_{i-1}x_0 \quad \text{for } i = 1, 2, \dots, r$$

$$c_i = b_i + c_{i-1}x_0 \quad \text{for } i = 1, 2, \dots, r$$

The final result for the top line is  $P_r(x_0)$  and for the bottom line is  $P'_r(x_0)$ . The diagram uses blue arrows to indicate the flow of the recursive calculation, and orange arrows to indicate the flow of the coefficients. The coefficients  $a_r$  and  $c_0$  are both equal to  $a_r$ .

mit  $c_0 = b_0 = a_r$ ,

$b_{i+1} = a_{r-i-1} + b_i x_0$  für  $i = 1, \dots, r$ ,

$c_{i+1} = b_{i+1} + c_i x_0$  für  $i = 1, \dots, r-1$ ,

$b_r = P_r(x_0)$  und  $c_{r-1} = P'_r(x_0)$ .

#### Ein weiteres Beispiel:

$$P_4(x) = 4x^4 - 5x^3 + x - 2, \quad P_4(4) = 706, \quad P'_4(4) = 785$$



**Horner2.bas**

```
REM Polynomwertbestimmung mittels
REM erweiterten HORNER -Schema
REM M. Meiler, 21.01.2002

DO
  CLS
  GOSUB polynomeingabe: PRINT
  PRINT "Eingegebenes Polynom": GOSUB polynomausgabe: PRINT
  DO
    GOSUB argumenteingabe: PRINT: GOSUB horner2
    GOSUB polynomwertausgabe: PRINT

    PRINT "Berechnen des gleichen Polynoms ";
    INPUT "mit neuem Argument(j/n)? ", Weiter$
    LOOP WHILE Weiter$ = "j"
    INPUT "Berechnen eines neuen Polynoms(j/n)? ", Weiter$

  LOOP WHILE Weiter$ = "j"
END

REM Eingabe des Grads und der Koeffizienten des Polynoms
polynomeingabe:
DO
  INPUT "Grad des Polynoms (r > 0): ", r%
  LOOP UNTIL r% > 0

  REDIM a!(r%)

  PRINT "Koeffizienteneingabe"
  FOR i% = 0 TO r%
    PRINT "a!(;" i%; ")="; : INPUT " ", a!(i%)
  NEXT i%
  RETURN

REM Eingabe des Arguments zur Berechnung des Polynomwertes
argumenteingabe:
INPUT "Argument x = ", x!
RETURN

REM Berechnung des Polynomwertes und der ersten Ableitung
REM mittels erweitertem Hornerschema
horner2:
b! = 0: c! = 0

FOR k% = r% TO 0 STEP -1
  b! = a!(k%) + b! * x! : 'PRINT "b("; k%; ")="; b!;
  IF k% <> 0 THEN
    c! = b! + c! * x! : 'PRINT "c("; k%; ")="; c!
  END IF
NEXT k%

y! = b! : y1! = c!
RETURN

REM Ausgabe des Polynoms in ueblicher Schreibweise
polynomausgabe:
```



```

FOR ii% = 0 TO r%
    PRINT a!(ii%); "x^"; ii%;
    IF ii% <> r% THEN PRINT "+ ";
NEXT ii%
PRINT
RETURN

REM Ausgabe des berechneten Polynomwertes
polynomwertausgabe:
PRINT "P("; x!; ")="; y!: PRINT "P'("; x!; ")="; y1!
RETURN

```

### 5.4.3 Anwendungsbeispiel

Bei der numerischen Integration (s. Abschnitt 5.3) können die Funktionswerte der Stützstellen mittels *Hornerschema* berechnet werden. Damit wird deren Eingabe überflüssig und Eingabefehler vermieden:

#### *PTrapez.bas*

```

REM Integration von Polynomen mittels Trapezregel
REM Polynomwertbestimmung mittels HORNER-Schema
REM M. Meiler, 20.01.2003

REM Hauptprogramm
CLS : GOSUB eingabe: GOSUB trapez: GOSUB ausgabe
END

REM Eingabe der obere und unteren Grenze
REM und Berechnung der Stuetzstellen
eingabe:
GOSUB polynomeingabe: PRINT
GOSUB grenzeneingabe: PRINT
GOSUB stuetzstellen: PRINT
RETURN

REM Eingabe des Grades und der Koeffizienten des Polynoms
polynomeingabe:
DO
    INPUT "Grad des Polynoms(r > 0): ", r%
LOOP UNTIL r% > 0
REDIM a!(r%)
PRINT "Koeffizienteneingabe"
FOR i% = 0 TO r%
    PRINT "a!("; i%; ")="; : INPUT " ", a!(i%)
NEXT i%
RETURN

REM Eingabe der obere und unteren Grenze
grenzeneingabe:
DO
    INPUT "Untere Integrationsgrenze a= ", a!
    INPUT "Obere Integrationsgrenze b(b>a)= ", b!
LOOP UNTIL (b! - a!) > 0
RETURN

```



```
REM Berechnung der Stuetzstellen mittels HORNER
stuetzstellen:
DO
  INPUT "Anzahl der Stuetzstellen (s > 1): ", s%
LOOP UNTIL s% >= 2

n% = s% - 1
h! = (b! - a!) / n%
REDIM y!(n%)

FOR i% = 0 TO n%
  x! = a! + i% * h!
  GOSUB horner1: y!(i%) = y!
NEXT i%
RETURN

REM Berechnung des Polynomwertes mittels Horner
horner1:
b! = 0
FOR k% = r% TO 0 STEP -1
  b! = a!(k%) + b! * x!
NEXT k%
y! = b!
RETURN

REM Berechnung des Integrals mittels Trapezregel
trapez:
y! = y!(0) + y!(n%)

FOR j% = 1 TO n% - 1
  y! = y! + 2 * y!(j%)
NEXT j%

y! = y! * h! / 2
RETURN

REM Ausgabe des Ergebnisses
ausgabe:
PRINT "Ergebnis: "; y!
PRINT
RETURN
```



## 5.5 Iterationsverfahren

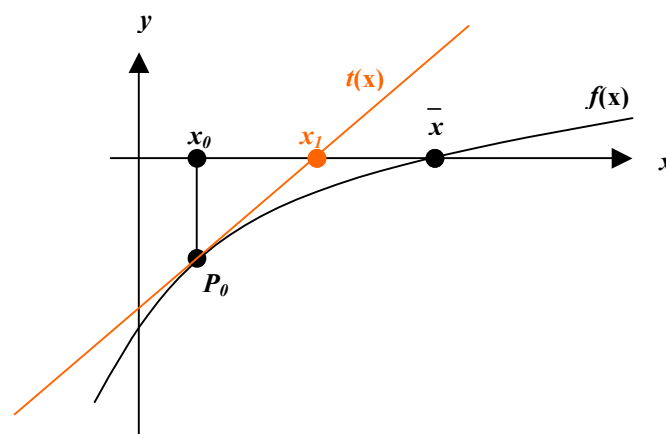
Iterationsverfahren berechnen ausgehend von einer Anfangsnäherungen eine bessere Näherung. Das wird solange wiederholt (iteriert), bis eine gewünschte Genauigkeit erreicht wurde. Sie haben mit der Entwicklung der Computertechnik durch Rechengeschwindigkeit und Rechengenauigkeit an Bedeutung gewonnen.

### 5.5.1 Newtonverfahren (Tangentenverfahren)

Das nach Isaac Newton (1643-1727) benannte Verfahren dient der Nullstellenbestimmung einer gegebenen Funktion.

#### Grundgedanke

Ausgehend von einer bereits vorhandenen Näherung  $x_0$  wird im Punkt  $P_0 = (x_0, f(x_0))$  an die Funktion  $f(x)$  eine Tangente  $t(x)$  gelegt. Der Schnittpunkt dieser Tangente mit der Abzisse liefert eine neue Näherung  $x_1$ . Der Vorgang wird so oft wiederholt, bis man eine geeignete Näherung einer Nullstelle  $\bar{x}$  gefunden hat.



#### Ausgangspunkt

Eine stetig differenzierbare Funktion  $f(x)$  und eine Näherung  $x_0$ .

Berechnung der Tangente  $t(x)$  in  $P_0$  an  $f(x)$ :

Die Tangente  $t(x)$  an die Funktion  $f(x)$  durch den Punkt  $P_0 = (x_0, y_0)$ , wobei  $y_0 = f(x_0)$ , wird mittels **Punktrichtungsgleichung** berechnet. Sei  $y'_0 = f'(x_0)$  die Steigung von  $f(x)$  im Punkt  $P_0$ , so ist

$$y - y_0 = y'_0(x - x_0)$$

und somit

$$t(x) = y = y_0 + y'_0(x - x_0)$$

Tangente in  $P_0$  an  $f(x)$ .



Berechnung der Nullstelle  $x_1$  von  $t(x)$ :

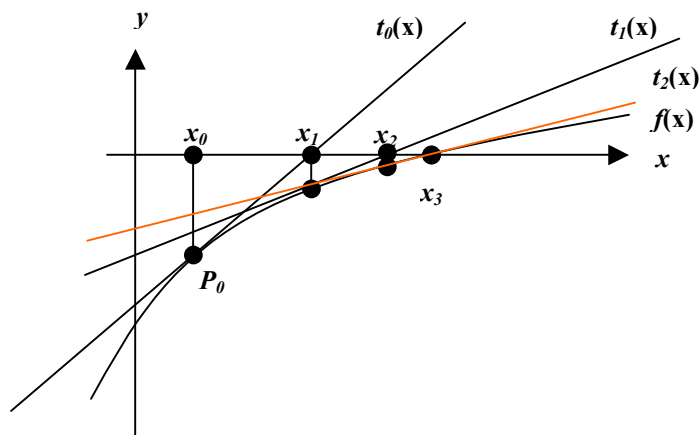
$$t(x_1) = 0 \Rightarrow y_0 + y'_0(x_1 - x_0) = 0 \Rightarrow x_1 - x_0 = -\frac{y_0}{y'_0} \Rightarrow x_1 = x_0 - \frac{y_0}{y'_0}$$

$\Rightarrow$

**Newtonverfahren:**  $\bar{x} \approx x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

### Allgemein

Ausgehend von einer Näherung  $x_0$  wird eine Folge von Näherungen  $x_1, x_2, \dots, x_n$  berechnet, die gegen  $\bar{x}$  konvergiert.



### 5.5.2 Anwendungsbeispiel 1

Berechnung der  $k$ -ten Wurzel aus  $a$  mit  $k \in \mathbb{N}$ ,  $k > 1$ ;  $a \in \mathbb{R}$ ,  $a > 0$

$$x = \sqrt[k]{a} \Rightarrow f(x) = x^k - a = 0$$

$\Rightarrow$  Das Problem reduziert sich auf die Nullstellenbestimmung von

$$f(x) = x^k - a \text{ mit } f'(x) = k \cdot x^{k-1}$$

mittels Newtonverfahren 
$$x_1 = x_0 - \frac{x_0^k - a}{k \cdot x_0^{k-1}}.$$

Speziell für  $a = 4$ ,  $k = 2 \Rightarrow f(x) = x^2 - a$   $f'(x) = 2 \cdot x$  und dem Anfangswert  $x_0 = 4$  ergibt sich mittels Taschenrechner aus dem Newtonverfahren:

$i$	$x_i$	$y_i = x_i^2 - a$	$y'_i = 2 \cdot x_i$	$x_{i+1} = x_i - \frac{y_i}{y'_i}$
0	4	12	8	$4 - 12/8 = 2.5$
1	2.5	2.25	5	$2.5 - 2.25/5 = 2.05$
2	2.05	0.203	4.1	$2.05 - 0.203/4.1 = 2$
3	2	0 (Abbruch)		

Das Verfahren bricht für  $x_3 = 2$  ab und konvergiert damit gegen den korrekten Wurzelwert.



***Newton.bas***

```

REM Newtonverfahren (Tangentenverfahren)
REM Berechnung der k. Wurzel aus a
REM M. Meiler, 26.01.2001

' k%    ... Wurzelexponent
' a#    ... Radikant
' EPS#  ... Genauigkeit fuer Abbruchbedingung
' x0#   ... Naehierung, anfangs a#
' y0#   ... Funktionswert an der Stelle x0#
' y1#   ... 1. Ableitung an der Stelle x0#

CLS
DO
  GOSUB eingabe: GOSUB newton: GOSUB ausgabe
  INPUT "Weiter (j/n)? ", Weiter$
LOOP WHILE Weiter$ = "j"
END

eingabe:
DO
  PRINT "Wurzelberechnug  - k. Wurzel aus a  - (k>1, a>0)": PRINT
  INPUT "k = ", k%
  INPUT "a = ", a#
LOOP UNTIL k% > 1 AND a# > 0
PRINT
DO
  PRINT "Genauigkeit (0<EPS<1)": PRINT
  INPUT "EPS = ", EPS#
LOOP UNTIL 0 < EPS# AND EPS# < 1
RETURN

newton:
x0# = a#: y0# = x0# ^ k% - a#: y1# = k% * x0# ^ (k% - 1)

DO
  'Testausgabe
  'LOCATE , 10: PRINT x0#

  'Testausgabe
  'LOCATE , 10: PRINT y0#; : LOCATE , 40: PRINT y1#: PRINT

  IF ABS(y0#) < EPS# THEN EXIT DO

  x0# = x0# - y0# / y1#
  y0# = x0# ^ k% - a#: y1# = k% * x0# ^ (k% - 1)
  ' Newton

LOOP
RETURN

ausgabe:
PRINT "Die "; k%; ". Wurzel von "; a#; " ist "; x0#; " !"
RETURN

```



**Protokoll** zum Programm *Wurzel.bas* mit den Eingabewerten:

**a# = 4, k% = 2, EPS# = .0001.**

k%	a#	ESP#	x0#	y0#	y1#	Bemerkungen
2	4	.0001				Eingabewerte
			4	12	8	Startwerte
			2.5	2.25	5	1. Schleifendurchlauf
			2.05	0.202 5	4.1	2. Schleifendurchlauf
			2.00061	0.002 44	4.001 22	3. Schleifendurchlauf
			<b>2.000 000 1</b>	0.000 000 37	4.000 000 4	4. Schleifendurchlauf <b>Abbruch</b>

### 5.5.3 Anwendungsbeispiel 2

Nullstellenberechnung von Polynomen.

Gegeben:  $f(x) = P_r(x) = a_r x^r + a_{r-1} x^{r-1} + \dots + a_1 x + a_0$

Näherung  $x_0$

Gesucht: Näherung  $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ ,

wobei  $f(x_0)$  und  $f'(x_0)$  mittels Hornerschema berechnet werden.

Programmschritte:

1. Eingabe des Polynoms (Unterprogramm **eingabe** im Programm *Horner2.bas*).
2. Eingabe der Genauigkeit *EPS* der Berechnung.
3. Berechnung einer Nullstellennäherung.
  - (a) Eingabe der Anfangsnäherung  $x_0$ .
  - (b) Berechnung des Funktionswertes  $f(x_0)$  und dessen Ableitung  $f'(x_0)$  mittels erweitertem Hornerschema (Unterprogramm **horner2** im Programm *Horner2.bas*).
  - (c) Ist  $|f(x_0)| \geq EPS$ :  
Berechnung einer neuen Näherung  $x_1$  mittels Newtonverfahren.  
Setze  $x_0 = x_1$  und weiter mit 3 (b).
  - (d) Näherung  $x_1$  für eine Nullstelle wurde gefunden.
4. Soll eine neue Nullstellennäherung berechnet werden, weiter bei 3.
5. Soll die Genauigkeit verändert werden, weiter bei 2.
6. Soll ein neues Polynom berechnet werden, weiter bei 1.
7. Fertig

#### *PNewton.bas*

```
REM Newtonverfahren (Tangentenverfahren)
REM Nullstellenberechnung in Polynomen
REM M. Meiler, 26.01.2001
```

```
' EPS#      ... Genauigkeit fuer Abbruchbedingung
' r%        ... Grad des Polynoms
' a#( i%)   ... i. Koeffizient des Polynoms
```



```

' a!      ... Startwert
' x0#     ... Naehierung, anfangs a!
' y0#     ... Funktionswert an der Stelle x0#
' y1#     ... 1. Ableitung an der Stelle x0#
' x#      ... Argument fuer Polynomberechnung nach Horner
' b#      ... Funktionswert an der Stelle x# nach Horner
' c#      ... 1. Ableitung an der Stelle x# nach Horner

DO
  CLS
  GOSUB polynomeingabe
  DO
    GOSUB epseingabe
    DO
      GOSUB startwerteingabe
      GOSUB newton
      GOSUB ausgabe
      INPUT "Neuer Startnaehierungswert (j/n)? ", Weiter$
      LOOP WHILE Weiter$ = "j"
      INPUT "Neue Genauigkeit (j/n)? ", Weiter$
      LOOP WHILE Weiter$ = "j"
      INPUT "Neues Polynom (j/n)? ", Weiter$
      LOOP WHILE Weiter$ = "j"
    END
  END

polynomeingabe:
DO
  INPUT "Grad des Polynoms: ", r%
LOOP UNTIL r% > 0
REDIM a#(r% + 1)
PRINT "Koeffizienteneingabe"
FOR i% = 0 TO r%
  PRINT "a("; i%; ")="; : INPUT " ", a#(i%)
NEXT i%
RETURN

epseingabe:
DO
  PRINT "Genauigkeit (0<EPS<1)": PRINT
  INPUT "EPS = ", EPS#
LOOP UNTIL 0 < EPS# AND EPS# < 1
RETURN

startwerteingabe:
INPUT "Startwert a= ", a!
x0# = a!
RETURN

newton:
x# = x0#: GOSUB horner: y0# = b#: y1# = c#
DO
  ' Testausgabe
  ' LOCATE , 10: PRINT x0#
  ' Testausgabe
  ' LOCATE , 10: PRINT y0#; : LOCATE , 40: PRINT y1#: PRINT

```



```

IF ABS(y0#) < EPS# THEN EXIT DO

x0# = x0# - y0# / y1#
x# = x0#: GOSUB horner: y0# = b#: y1# = c#
LOOP
RETURN

horner:
b# = 0: c# = 0
FOR k% = r% TO 0 STEP -1
  b# = a#(k%) + b# * x#
  IF k% <> 0 THEN c# = b# + c# * x#
  ' PRINT "b="; b#; "c="; c#
NEXT k%
RETURN

ausgabe:
PRINT "Das Newtonverfahren auf das Polynom mit den Koeffizienten"
FOR ii% = 0 TO r%
  PRINT "a("; ii%; ")="; a#(ii%)
NEXT ii%
PRINT "und den Startwert "; a!; "angewand,"
PRINT "liefert als Nullstelle den Wert "; x0#; "!"
PRINT
RETURN

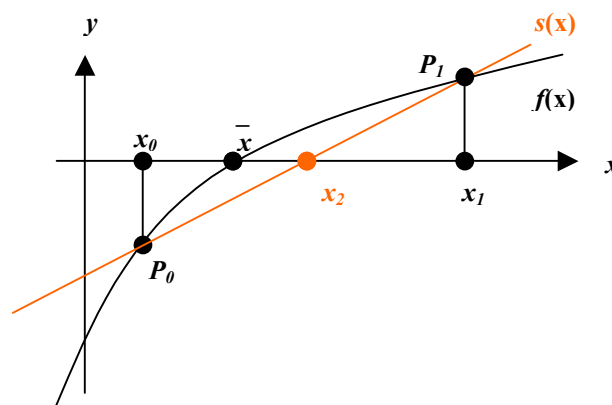
```

### 5.5.4 Regula falsi (Sekantenverfahren)

Auch hier sollen die Nullstellen einer Funktion  $f(x)$  berechnet werden.

#### Grundgedanke

Ausgehend von zwei bereits vorhandenen Näherung  $x_0$ ,  $x_1$  mit  $f(x_0) < 0$  und  $f(x_1) > 0$  wird durch die Punkte  $P_0 = (x_0, f(x_0))$  und  $P_1 = (x_1, f(x_1))$  eine Sekante  $s(x)$  gelegt. Der Schnittpunkt dieser Sekante mit der Abzisse liefert eine neue Näherung  $x_2$ . Der Vorgang wird so oft wiederholt, bis man eine geeignete Näherung einer Nullstelle  $\bar{x}$  gefunden hat.





**Ausgangspunkt**

Funktion  $f(x)$  und zwei Nullstellennäherung  $x_0$  und  $x_1$  mit  $f(x_0) < 0$  und  $f(x_1) > 0$ .

Berechnung der Sekante  $s(x)$  durch  $P_0$  und  $P_1$ :

Die Sekante  $s(x)$  durch die Punkte  $P_0 = (x_0, y_0)$  und  $P_1 = (x_1, y_1)$ , wobei  $y_0 = f(x_0)$ ,  $y_1 = f(x_1)$ , wird mittels **Zweipunktegleichung** berechnet.

$$(y - y_0)(x_1 - x_0) - (x - x_0)(y_1 - y_0) = 0$$

und somit

$$s(x) = y = y_0 + \frac{(x - x_0)(y_1 - y_0)}{(x_1 - x_0)}$$

Sekante durch  $P_0$  und  $P_1$ .

Berechnung der Nullstelle  $x_2$  von  $s(x)$ :

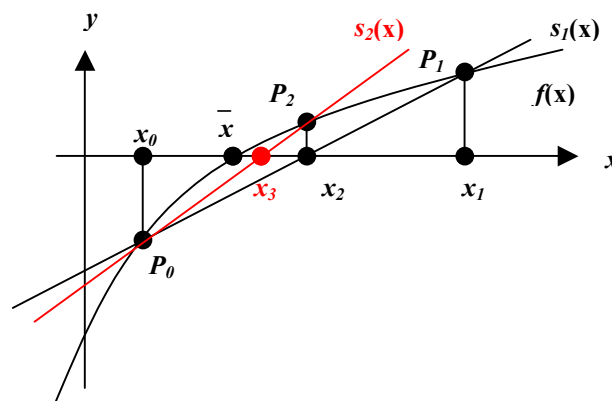
$$s(x_2) = 0 \Rightarrow y_0 + \frac{(x_2 - x_0)(y_1 - y_0)}{(x_1 - x_0)} = 0 \Rightarrow x_2 - x_0 = -y_0 \frac{(x_1 - x_0)}{(y_1 - y_0)}$$

$\Rightarrow$

**Regula falsi:**  $\bar{x} \approx x_2 = x_0 - y_0 \frac{(x_1 - x_0)}{(y_1 - y_0)}$

**Allgemein**

Ausgehend von zwei Näherung  $x_0, x_1$  wird eine Folge von Näherungen  $x_1, x_2, \dots, x_n$  berechnet, die gegen  $\bar{x}$  konvergiert.



Der Vorteil dieses Verfahrens gegenüber des Newtonverfahrens ist, dass man auf die Berechnung der ersten Ableitung verzichten kann. Der Nachteil besteht in der langsameren Konvergenz.

**5.5.5 Anwendungsbeispiel 3**

Berechnung der  $k$ -ten Wurzel aus  $a$  mit  $k \in \mathbb{N}$ ,  $k > 1$ ;  $a \in \mathbb{R}$ ,  $a > 0$  analog s. o.

$$x = \sqrt[k]{a} \Rightarrow f(x) = x^k - a = 0$$



⇒ Das Problem reduziert sich wiederum auf die Nullstellenbestimmung von

$$f(x) = x^k - a$$

Speziell für  $a=4$ ,  $k=2 \Rightarrow f(x) = x^2 - a$  und den Anfangswerten  $x_0=1$ ,  $x_1=4$  ergibt sich mittels Taschenrechner und Regula falsi:

$i$	$x_i$	$x_{i+1}$	$y_i = x_i^2 - a$	$y_{i+1} = x_{i+1}^2 - a$	$x_{i+2} = x_i - y_i \frac{x_{i+1} - x_i}{y_{i+1} - y_i}$	$y_{i+2} = x_{i+2}^2 - a$
0	1	4	-3	12	1.6	-1.44
1	1.6	4	-1.44	12	1.8571429	-0.55102
2	1.8571429	4	-0.55102	12	1.951295	-0.19244
3	1.951295	4	-0.19244	12	1.9836308	-0.0652087
4	1.9836308	4	-0.0652087	12	1.9943517	-0.0225613

Das Verfahren konvergiert gegen den korrekten Wurzelwert.

### **Regula.bas**

```
REM Regula falsi (Sekanten-Verfahren)
REM Berechnung der k. Wurzel aus a
REM M. Meiler, 26.01.2001
```

```
' k%    ... Wurzelexponent
' a#    ... Radikant
' EPS#  ... Genauigkeit fuer Abbruchbedingung
' x0#   ... 1. Naehierungswert
' x1#   ... 2. Naehierungswert
' x2#   ... neuer Naehierungswert
' y0#   ... Funktionswert an der Stelle x0#
' y1#   ... Funktionswert an der Stelle x1#
' y2#   ... Funktionswert an der Stelle x2#
```

```
CLS
DO
  GOSUB eingabe: GOSUB regula: GOSUB ausgabe
  INPUT "Weiter (j/n)? ", Weiter$
LOOP WHILE Weiter$ = "j"
END
```

```
eingabe:
DO
  PRINT "Wurzelberechnug - k. Wurzel aus a - (k>1, a>0)"
  INPUT "k = ", k%
  INPUT "a = ", a#
LOOP UNTIL k% > 1 AND a# > 0
PRINT
DO
  PRINT "Genauigkeit (0<EPS<1)": PRINT
  INPUT "EPS = ", EPS#
LOOP UNTIL 0 < EPS# AND EPS# < 1
DO
  PRINT "Startwerte x0 und x1 mit x0 < k. Wurzel aus a < x1"
  INPUT "x0 = ", x0#
  INPUT "x1 = ", x1#
```



```

    y0# = x0# ^ k% - a#; y1# = x1# ^ k% - a#
LOOP UNTIL y0# < 0 AND y1# > 0
RETURN

regula:
DO
    'Testausgabe
    LOCATE , 10: PRINT x0#; : LOCATE , 40: PRINT x1#
    'Testausgabe
    LOCATE , 10: PRINT y0#; : LOCATE , 40: PRINT y1#: PRINT
                                     ' Regula falsi
    x2# = x0# - y0# * (x1# - x0#) / (y1# - y0#)
    y2# = x2# ^ k% - a#

    IF ABS(y2#) < EPS# THEN EXIT DO

    IF SGN(y1#) = SGN(y2#) THEN
        x1# = x2#: y1# = y2#
    ELSE
        x0# = x2#: y0# = y2#
    END IF
LOOP
RETURN

ausgabe:
PRINT "Die "; k%; ". Wurzel von "; a#; " ist "; x2#; " !"
RETURN

```

## 5.6 Zusammenfassung

- **Eingabefehler** lassen sich in der Regel nur durch die Verbesserung der Messtechniken vermeiden. Die Toleranz, in der diese Fehler liegen, entscheiden über die zu verwendete Rechengenauigkeit und Genauigkeit der Verfahren.
- Mathematisch exakte Verfahren können durch **Rechenfehler** zu unexakten Ergebnissen führen. Rechenfehler können durch die Verwendung anderer Datentypen minimiert aber nicht beseitigt werden. Solche Fehler sind auch bei hochentwickelter Computertechnik nicht auszuschließen.  
**Gauss.bas.** Das Programm zum Gaußverfahren liefert in der Regel keine Dreiecksmatrix, obwohl eine solche mathematisch zu erwarten ist.
- Näherungsverfahren sind mathematisch unexakte Verfahren. Sie ersetzen komplizierte Verfahren durch einfachere und liefern nur Näherungen der Lösung. **Verfahrensfehler** können durch die Verwendung anderer Datentypen in der Regel nicht verbessert werden.  
**Trapez.bas, Simpson.bas.** Bei der Integration durch Polynominterpolation wurden die Funktion durch Polynome ersetzt, welche sich rechentechnisch besser integrieren lassen. Eine Verbesserung der Ergebnisse konnte durch mehr Stützstellen oder einen höheren Grad des Interpolationspolynoms, also der Verbesserung des verwendeten Verfahrens, erreicht werden.



*Wurzel.bas*, *Ragula.bas* berechnen Wurzeln als Nullstellen einer Funktion mit unterschiedlichen Näherungsverfahren. Beide Verfahren benötigen bereits Näherungslösungen. Das Newtonverfahren konvergiert sehr schnell, benötigt aber die erste Ableitung der betrachteten Funktion. Wendet man die Verfahren auf Funktionen mit mehreren Nullstellen an, so findet man u. U. nicht alle Nullstellen der Funktion. Die Anfangsnäherung ist für das Ergebnis ausschlaggebend.

- Durch die vorhandenen Datentypen und Rechenoperationen sind oft **Grenzen für die Berechnung** gesetzt, Ergebnisse sind deshalb stets bzgl. ihrer Korrektheit abzuschätzen.

*Fak.bas*. Die Fakultätsberechnung lieferte bei ganzen Zahlen bis 7!, bei langen ganzen Zahlen und rationalen Zahlen einfacher Genauigkeit bis 12! und bei rationalen Zahlen doppelter Genauigkeit bis 22! exakte Ergebnisse. Bei größeren Zahlen wurde die Berechnung durch Überlauf abgebrochen bzw. durch Runden falsch.

- Mathematisch elegante Verfahren sind rechentechnisch nicht immer die günstigsten. Rechentechnisch optimale Verfahren verbrauchen wenig Rechenzeit. Eine **Rechenzeitminimierung** kann man durch Minimierung der Anzahl der verwendeten Operationen erreichen.

*Horner1.bas*. Bei der Polynomwertberechnung eines Polynoms vom Grad  $r$  ohne Hornerschema benötigt man  $\frac{r^2 + 5r + 2}{2}$  Additionen und Multiplikationen, das

sind zum Beispiel bei  $r = 4$  insgesamt 19 Operationen. Durch die Verwendung des Hornerschemas konnte dieser Rechenaufwand auf  $2r$  verringert werden, bei  $r = 4$  sind das nur 8, also weniger als die Hälfte.

- Die **modulare Programmierung** gestatten die mehrfache Verwendung von schon bereits ausgetesteten Programmteilen, hier Unterprogrammen. In sich abgeschlossene Programmteile werden in einem Unterprogramm zusammengefasst. Diese können getrennt vom Hauptprogramm ausgetestet, jederzeit wieder aufgerufen und leicht durch andere ausgetauscht werden.

*Horner1.bas*, *Horner2.bas*, *Tabelle.bas*, *PTrapez.bas*, *PIntegr.bas*, *PNewton.bas*. Die Module zum Eingeben eines Polynoms, zur Berechnung eines Polynomwertes bzw. eines Polynomwertes und dessen Ableitung an einer gegebenen Stelle mittels Hornerschema können bei der Lösung verschiedener Probleme wiederverwendet werden, wie zum Beispiel beim Tabellieren eines Polynoms, bei der Integralberechnung eines Polynoms zum Berechnen der Funktionswerte an den Stützstellen oder wie im letzten Abschnitt, bei der Nullstellenbestimmung eines Polynoms zur Berechnung des Polynomwertes und deren Ableitung von einer Näherung.