Kapitel 5

Numerische Lösung von transzendenten Gleichungen

5.1 Das grundsätzliche Problem.

Gegeben sei eine reellwertige Funktion $F(x)$. Gesucht seien jene reellen Werte von x, für welche die Gleichung

$$F(x) = 0 \tag{5.1}$$

erfüllt ist. Die x-Werte, die diese Bedingung erfüllen, werden *Lösungen, Nullstellen, Wurzeln, ...* von (5.1) genannt.

Aus der Fülle der in der Literatur beschriebenen Methoden sollen in diesem Kapitel die folgenden Verfahren behandelt werden:

- Iterationsverfahren (Newton-Raphson-Verfahren; Regula Falsi).
- Das Intervallschachtelungsverfahren.
- Die numerische Behandlung nicht-linearer Gleichungssysteme.

Anmerkung: Die Betonung auf *transzendente* Gleichungen im Titel dieses Kapitels schließt natürlich die Behandlung *algebraischer* Gleichungen vom Typus

$$F(x) \equiv P_m(x) = \sum_{j=1}^{m} \alpha_j x^{j-1} = 0$$

nicht aus! In der hier verwendeten Terminologie sind algebraische Gleichungen lediglich Spezialfälle transzendenter Gleichungen.
Für die numerische Bestimmung von Nullstellen von algebraischen Gleichungen (Polynomen) werden in der Literatur eine Reihe von speziellen Verfahren beschrieben (s. z.B. das Verfahren von Lobatschewski und Graeffe [20], S. 60ff etc.), die im Rahmen dieser Lehrveranstaltung nicht behandelt werden.

5.2 Iterationsverfahren.

5.2.1 Allgemeines.

Die Gleichung $F(x) = 0$ kann immer (und fast immer auf mehrere Arten) auf die Form

$$x = f(x) \tag{5.2}$$

gebracht werden. Auf diese Weise erhält man einen typischen *Iterationsansatz* für die Ermittlung einer Nullstelle:

$$\begin{aligned} x_0 &\quad \text{Startwert} \\ x_1 &= f(x_0) \\ x_2 &= f(x_1) \\ &\vdots \\ x_{t+1} &= f(x_t) \qquad (t = 0, 1, 2, \ldots) \end{aligned} \tag{5.3}$$

Im Falle einer Konvergenz führt eine solche Iteration beliebig genau (bis auf Rundungsfehler) an die exakte Lösung des Problems heran. Es gilt

$$\lim_{t\to\infty} x_t \to x_{exakt}$$

Ein solches Konvergenzverhalten ist jedoch keineswegs selbstverständlich, sondern *hängt sehr von der Art der Umformung von (5.1) in (5.2) ab*, wie im folgenden Beispiel demonstriert werden soll.

Gesucht ist die Nullstelle der Funktion $F(x) = x^3 - x - 5$ *in der Umgebung von 2*. Es soll nun gezeigt werden, wie verschieden das Konvergenzverhalten ist, wenn die Umformung auf (5.2) auf verschiedene Weise erfolgt:

$$a)\quad x = x^3 - 5 \qquad b)\quad x = \frac{5}{x^2 - 1} \qquad c)\quad x = \sqrt[3]{x + 5}$$

x0 = 2.0

t	(a)	(b)	(c)
0	2	2	2
1	3	1.6667	1.9129
2	22	2.8125	1.9050
3	10643	0.7236	1.9042
4	.	-10.4944	1.9042
.	.	.	.
.	.	.	.
	DIV	DIV	KONV

(b) zeigt ein interessantes Verhalten: s. Demo in der Vorlesung!

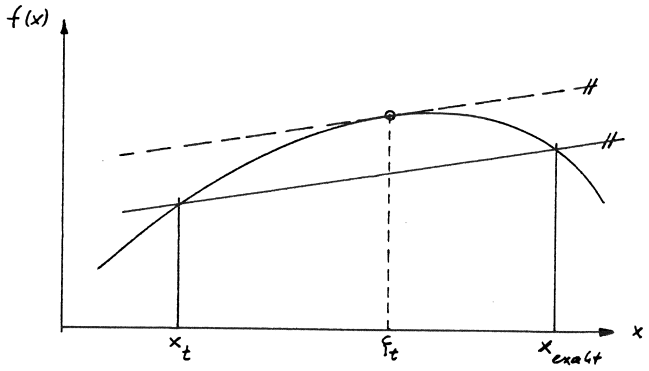


Abbildung 5.1: Zur Fehlerabschätzung bei Iterationen.

5.2.2 Konvergenzkriterien und Fehlerabschätzungen.

Um die Bedingung zu erhalten, unter der eine Konvergenz der Iteration gewährleistet ist, geht man von der Gleichung (5.2) für die exakte Lösung

$$x_{exakt} = f(x_{exakt}) \tag{5.4}$$

aus. Beim $(t+1)$-ten Iterationsschritt erhält man hingegen

$$x_{t+1} = f(x_t) - \delta_t \quad , \tag{5.5}$$

wobei δ_t den Rundungsfehler darstellt, der bei der numerischen Auswertung der Funktion f für $x = x_t$ auftritt. Subtraktion von (5.5) von (5.6) ergibt

$$x_{exakt} - x_{t+1} = f(x_{exakt}) - f(x_t) + \delta_t \, .$$

Unter Anwendung des Mittelwertsatzes (s. Abb.5.1) kann man schreiben:

$$\frac{f(x_{exakt}) - f(x_t)}{x_{exakt} - x_t} = f'(\xi_t) \quad \text{mit} \quad \xi_t \in [x_t, x_{exakt}] \quad .$$

Somit ergibt sich weiters

$$x_{exakt} - x_{t+1} = f'(\xi_t) \cdot (x_{exakt} - x_t) + \delta_t \tag{5.6}$$

Diese Gleichung kann Schritt für Schritt reduziert werden zu

$$\begin{aligned} x_{exakt} - x_{t+1} \quad = \quad & f'(\xi_t) f'(\xi_{t-1}) \cdots f'(\xi_0) \cdot (x_{exakt} - x_0) + \\ & + \delta_t + f'(\xi_t)\delta_{t-1} + f'(\xi_t) f'(\xi_{t-1})\delta_{t-2} + \\ & + \cdots + f'(\xi_t) f'(\xi_{t-1}) \cdots f'(\xi_1) \cdot \delta_0 \end{aligned}$$

Für eine *Fehlerabschätzung* soll nun angenommen werden, daß

1. die Funktion im Intervall I als Maximalbetrag ihrer ersten Ableitung den Wert m habe, wobei gilt, daß $x_0, x_1, \ldots, x_{t+1}, \ldots, x_{exakt} \in I$:

$$m = \max_I \mid f'(x) \mid \tag{5.7}$$

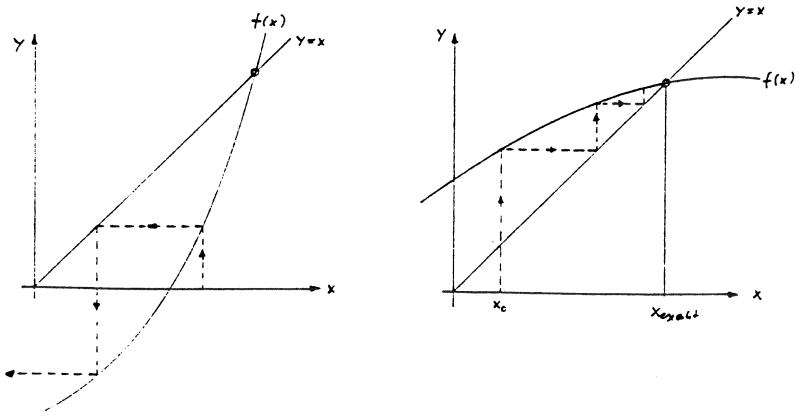


Abbildung 5.2: Konvergenzverhalten der Iteration (5.3).

2. *δ nicht vom Iterationsindex abhängt*, d.h. daß für alle $t = 0, 1, \ldots$ gilt:

$$\delta_t \approx \delta \quad .$$

Dann ergibt sich

$$\mid x_{exakt} - x_{t+1} \mid \leq m^{t+1} \mid x_{exakt} - x_0 \mid + \mid \delta \mid (1 + m + m^2 + \cdots m^t)$$

Der Grenzwert dieses Ausdruckes lautet:

$$\lim_{t\to\infty} \mid x_{exakt} - x_{t+1} \mid \leq \underbrace{\mid x_{exakt} - x_0 \mid \cdot \lim_{t\to\infty} m^{t+1}}_{\text{Verfahrensfehler}} + \underbrace{\frac{\mid \delta \mid}{1-m}}_{\text{Rundungsfehler}} \quad .$$

Beide Terme divergieren für $m \geq 1$, d.h. das *Konvergenzkriterium* für die Iteration lautet:

$$0 \leq m < 1 \quad . \tag{5.8}$$

Ist diese Bedingung erfüllt, ergibt sich

$$\lim_{t\to\infty} \mid x_{exakt} - x_{t+1} \mid \leq \frac{\mid \delta \mid}{1-m} \quad . \tag{5.9}$$

Dieses Ergebnis ist von entscheidender Bedeutung! Es zeigt die *numerische Stabilität* des Iterationsverfahrens: Die Größe des auftretenden Rundungsfehlers konvergiert ebenfalls zu einem Grenzwert. All diese Zusammenhänge sind in den Abbildungen 5.2 und 5.3 dargestellt.

Was nun die Möglichkeiten betrifft, die auftretenden Fehler *abzuschätzen*, kann man wieder von der Gleichung (5.6) ausgehen:

$$\begin{aligned}
x_{exakt} - x_{t+1} &= f'(\xi_t) \cdot (x_{exakt} - x_t) + \delta_t \\
&= f'(\xi_t) \cdot (x_{exakt} - x_t + x_{t+1} - x_{t+1}) + \delta_t \\
(x_{exakt} - x_{t+1}) \cdot (1 - f'(\xi_t)) &= f'(\xi_t) \cdot (x_{t+1} - x_t) + \delta_t \\
(x_{exakt} - x_{t+1}) &= \frac{f'(\xi_t)}{1 - f'(\xi_t)}(x_{t+1} - x_t) + \frac{\delta_t}{1 - f'(\xi_t)} \quad .
\end{aligned}$$

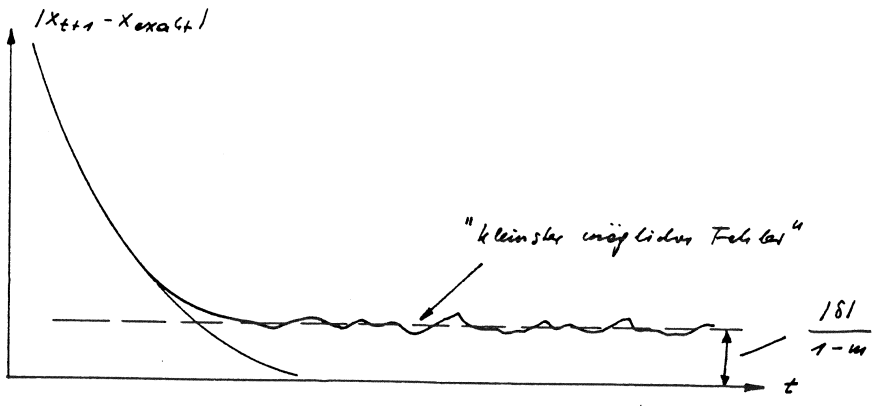


Abbildung 5.3: Fehlerentwicklung der Iteration (5.3).

Im Falle einer Konvergenz gilt nun die Fehlerabschätzung

$$\mid x_{exakt} - x_{t+1} \mid \leq \frac{m}{1-m} \mid x_{t+1} - x_t \mid + \frac{\mid \delta \mid}{1-m} \quad . \tag{5.10}$$

Wenn es möglich ist, die maximale erste Ableitung der Funktion $f(x)$ im Intervall I zu bestimmen bzw. die Größe δ abzuschätzen, kann (5.10) als Basis für eine *Fehlerkontrolle* der Iteration genommen werden: Man iteriert solange, bis der Ausdruck rechts in (5.10) kleiner geworden ist als eine gewünschte Genauigkeitsschranke ϵ, wobei jedoch zu berücksichtigen ist, daß ϵ nicht kleiner sein darf als die durch $\mid \delta \mid /(1-m)$ gegebene *Grenzgenauigkeit*!

In der Praxis ist jedoch die Bestimmung von m und δ meistens nicht möglich oder zu kompliziert. Dann behilft man sich mit der stark vereinfachten Fehlerabschätzung

$$\mid x_{exakt} - x_{t+1} \mid \leq \mid x_{t+1} - x_t \mid \quad , \tag{5.11}$$

woraus sich diese

Abbruchbedingung — .

|x(t+1)-x(t)| < EPS	
Y	N
Ende der Iteration	naechster Iterationsschritt

für die Iteration ergibt.
Es muß aber ausdrücklich darauf hingewiesen werden, daß (5.11) nur für $m \leq 1/2$ gilt, hingegen für $1/2 < m < 1$ grob falsch sein kann! Außerdem ist (5.11) nur solange nützlich, *als der Verfahrensfehler größer ist als der Rundungsfehler!*

Das folgende Beispiel soll einige der eben besprochenen Zusammenhänge illustrieren:

Die Funktion, deren Nullstelle numerisch ermittelt werden soll, lautet

$$F(x) = a + (1-a)x^2 - x \quad .$$

Wie man sich leicht überzeugen kann, hat diese quadratische Funktion die beiden Nullstellen

$$x_1 = 1 \qquad \text{und} \qquad x_2 = \frac{a}{1-a} \quad .$$

Die Nullstelle x_1 soll nun durch Ausführung der Iteration

$$x_{t+1} = a + (1-a)x_t^2$$

approximiert werden.

Die erste Ableitung von $f(x)$ hat an der Stelle $x = 1$ den Wert $2(1-a)$. Dies ist gleichzeitig der Maximalwert der Ableitung im Iterationsbereich, d. h. es gilt

$$m = |2(1-a)| \, .$$

Es ist auf Grund der Ergebnisse (5.8) und (5.10) zu erwarten, daß

- die Iteration für $m \geq 1$ d. h. für $a \leq 0.5$ bzw. $a \geq 1.5$ divergiert;
- die vereinfachte Fehlerabfrage (5.11) für $1/2 < m < 1$, d.h. für $a < 0.75$ versagt.

Ergebnisse dieser Testrechnung:

```
.
Parameter  a = 2.500  > 1.5  d.h. Divergenz ist zu erwarten:

   t        x_{t}      x_{ex}-x_{t+1}    x_{t+1}-x_{t}

   0    .1200000E+01
   1    .3400000E+00    .6600000E+00     -.8600000E+00
   2    .2326600E+01   -.1326600E+01      .1986600E+01
   3   -.5619601E+01    .6619601E+01     -.7946201E+01
   4   -.4486988E+02    .4586988E+02     -.3925028E+02
   5   -.3017459E+04    .3018459E+04     -.2972589E+04
   6   -.1365759E+08    .1365759E+08     -.1365457E+08
   7   -.2797945E+15    .2797945E+15     -.2797945E+15
   8   -.1174274E+30    .1174274E+30     -.1174274E+30
   9   -.2068380E+59    .2068380E+59     -.2068380E+59
  10   -.6417295+117    .6417295+117     -.6417295+117
```

```
.
Parameter  a = 0.600  ===> m=0.8  d.h.: Konvergenz, aber
                                  Versagen von (5.11):

   t        x_{t}      x_{ex}-x_{t+1}    x_{t+1}-x_{t}

   0    .6000000E+00
   1    .7440000E+00    .2560000E+00      .1440000E+00
   2    .8214144E+00    .1785856E+00      .7741440E-01
   3    .8698886E+00    .1301114E+00      .4847425E-01
   4    .9026825E+00    .9731750E-01      .3279386E-01
   5    .9259343E+00    .7406572E-01      .2325178E-01
   6    .9429417E+00    .5705828E-01      .1700744E-01
   7    .9556556E+00    .4434437E-01      .1271392E-01
   8    .9653111E+00    .3468892E-01      .9655443E-02
   9    .9727302E+00    .2726981E-01      .7419114E-02
  10    .9784816E+00    .2151839E-01      .5751419E-02


Parameter  a = 1.200  ===> m=0.4  d.h.: Konvergenz und
                                  Gueltigkeit von (5.11):


   0    .6000000E+00
   1    .1128000E+01   -.1280000E+00      .5280000E+00
   2    .9455232E+00    .5447680E-01     -.1824768E+00
   3    .1021197E+01   -.2119718E-01      .7567398E-01
   4    .9914313E+00    .8568734E-02     -.2976591E-01
   5    .1003413E+01   -.3412809E-02      .1198154E-01
   6    .9986325E+00    .1367453E-02     -.4780262E-02
   7    .1000547E+01   -.5466072E-03      .1914060E-02
   8    .9997813E+00    .2187027E-03     -.7653099E-03
   9    .1000087E+01   -.8747150E-04      .3061742E-03
  10    .9999650E+00    .3499013E-04     -.1224616E-03
```

5.3 Das Newton-Raphson-Verfahren.

Im Abschnitt 5.2.1 wurde darauf hingewiesen, daß es gewöhnlich mehrere Arten gibt, die Gleichung $F(x) = 0$ auf die äquivalente Form $f(x) = x$ zu bringen. Eine sehr wichtige Umformung ist gegeben durch

$$f(x) = x - \frac{F(x)}{F'(x)} \quad , \tag{5.12}$$

woraus sich aus (5.4) die bekannte *Newton-Raphson-Iteration* mit der Vorschrift

$$x_{t+1} = x_t - \frac{F(x_t)}{F'(x_t)} \tag{5.13}$$

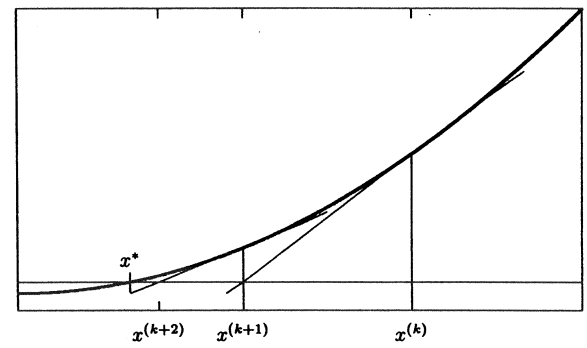


Abbildung 5.4: Grafische Interpretation der Newton-Raphson-Iteration.

ergibt. Die grafische Interpretation dieser Formel ('Tangentenmethode') ist hinlänglich bekannt (s. Abb.5.4).

Für die Fehlerabschätzung gilt wieder (5.10), wobei aber m gemäß (5.9, 5.12) die konkrete Form

$$m = \max_I \frac{d}{dx}\left(x - \frac{F(x)}{F'(x)}\right) = \max_{[x_0, x_{exakt}]}\left(\frac{F(x)F''(x)}{[F'(x)]^2}\right)$$

hat. Wegen der Konvergenzbedingung (5.8), nach der m echt kleiner als Eins sein muß, kann also gesagt werden:
Bei der Newton-Raphson-Iteration ist mit Schwierigkeiten zu rechnen, wenn in der Umgebung der gesuchten Nullstelle die Steigung von $F(x)$ klein wird, wie dies z.B. bei nahe beieinanderliegenden Nullstellen vorkommen kann.

Wenn aber die Newton-Raphson-Methode funktioniert, dann konvergiert sie sehr schnell, wie die folgende Rechnung demonstrieren soll:

Entwickelt man die Funktion $F(x)$ an der Stelle x_t in eine Taylorreihe bis zum (einschließlich) dritten Term, so ergibt sich

$$F(x) = F(x_t) + F'(x_t)(x - x_t) + \frac{1}{2}F''(x_t)(x - x_t)^2 .$$

Setzt man für x die exakte Nullstelle, so erhält man nach Division durch $F'(x_t)$

$$0 = -\frac{F(x_t)}{F'(x_t)} - (x_{exakt} - x_t) - \frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{exakt} - x_t)^2 .$$

Der erste Term rechts entspricht dem Korrekturterm in der Newton-Raphson-Formel, d.h. man erhält weiters

$$0 = x_{t+1} - x_t - x_{exakt} + x_t - \frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{exakt} - x_t)^2$$

bzw.

$$(x_{exakt} - x_{t+1}) = -\frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{exakt} - x_t)^2 . \tag{5.14}$$

Aus diesem Ergebnis geht hervor, daß der absolute Fehler der Nullstelle zum $(t+1)$-ten Iterationsschritt proportional zum *Quadrat* des absoluten Fehlers

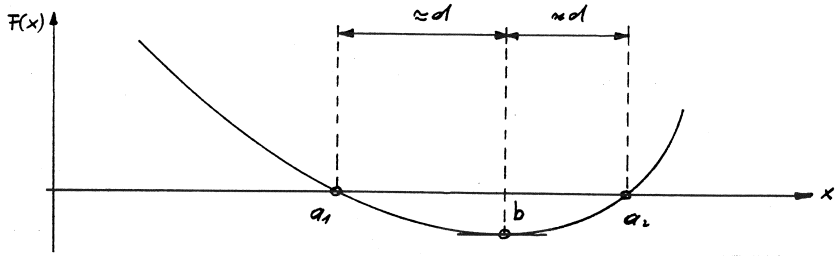


Abbildung 5.5: Zur Methode von Macon.

zum $t-$ten Iterationsschritt ist.
Man sagt: *die Newton-Raphson-Methode ist quadratisch konvergent.*

Eine Möglichkeit, mit eng nebeneinander liegenden Nullstellen-Paaren fertigzuwerden, ist die

5.3.1 Methode von Macon.

Im allgemeinen bereitet es keine Schwierigkeiten, das Minimum oder Maximum zwischen den beiden Nullstellen a_1 und a_2 zu lokalisieren. Nimmt man an, der Abszissenwert des Extremums, b, sei hinlänglich genau bekannt, so kann man $F(x)$ an der Stelle b in eine Taylorreihe entwickeln:

$$F(x) = F(b) + (x-b)F'(b) + \frac{1}{2}(x-b)^2F''(b) + \cdots \quad . \tag{5.15}$$

Man kann nun in erster Näherung annehmen, daß die beiden Nullstellen a_1 und a_2 *symmetrisch* um b verteilt sind und jeweils den Abstand d vom Extremum haben. Dann gilt

$$F(b+d) \approx 0 \qquad \text{und} \qquad F(b-d) \approx 0 \quad .$$

Eingesetzt in (5.15) ergibt das

$$0 \approx F(b) + \frac{1}{2}(+d)^2F''(b) \qquad 0 \approx F(b) + \frac{1}{2}(-d)^2F''(b) \quad .$$

Diese beiden Gleichungen sind äquivalent und erlauben die näherungsweise Berechnung von

$$d = \pm\sqrt{-\frac{2F(b)}{F''(b)}} \quad . \tag{5.16}$$

Startet man nun die Newton-Raphson-Iteration mit $b-d$ bzw. $b+d$ als Anfangswerte, gibt es meist keine Konvergenzprobleme.

5.3.2 Das Programm RTNEWT.

Quelle: [9],S. 257f mit Änderungen.

Das Funktions-Unterprogramm RTNEWT (RooT NEWTon) berechnet innerhalb eines gegebenen Intervalls eine reelle Nullstelle mittels der Newton-Raphson-Iteration.

INPUT-Parameter:

X1,X2: Anfang und Ende eines Intervalls, in welchem die Nullstelle liegt.

JMAX: maximale Anzahl von Iterationsschritten.

XACC: relative Genauigkeitsschranke gemäß Glg.(5.11).

OUTPUT-Parameter:

RTNEWT: Näherungswert für die Nullstelle.

FEHLER: für die Fehlerdiagnostik:

FEHLER = 0:	Newton-Iteration ok.
FEHLER = 1:	Kein Vorzeichenwechsel in [X1,X2]
FEHLER = 2:	Keine Konvergenz innerhalb JMAX Iterationen
FEHLER = 3:	'jumped out of brackets' während Newton.

Interne Variable:

FUNC,DF: $F(x)$ bzw. $F'(x)$. Die Prozedur, in der diese Größen berechnet werden, muß den Namen **FUNC** haben.

DX: Iterations-Korrektur.

Anmerkungen:

- An Beginn des Programms wird abgefragt, ob (mindestens) eine Nullstelle im Suchintervall [X1,X2] liegt. Dies wird daran erkannt, ob die Funktion $F(x)$ bzgl. der Intervallgrenzen einen Vorzeichenwechsel hat.
- Die im Programm verwendete *relative* Fehlerabfrage führt zu Konvergenzproblemen, wenn die gesuchte Nullstelle den Wert 0.0 hat.
- Der erste 'Notausgang' des Programms RTNEWT ('jumped out of brackets') wird verwendet, wenn während der Iteration das vom aufrufenden Programm vorgegebene Iterations-Intervall verlassen wird (s. Abb.5.6, links).
- Der zweite 'Notausgang' ('exceeding maximum iterations') ist notwendig im Falle einer Divergenz oder auch einer zu langsamen Konvergenz, aber auch in speziellen Fällen, wo sich die Iteration in einem Loop verfängt (s. Abb.5.6, rechts).

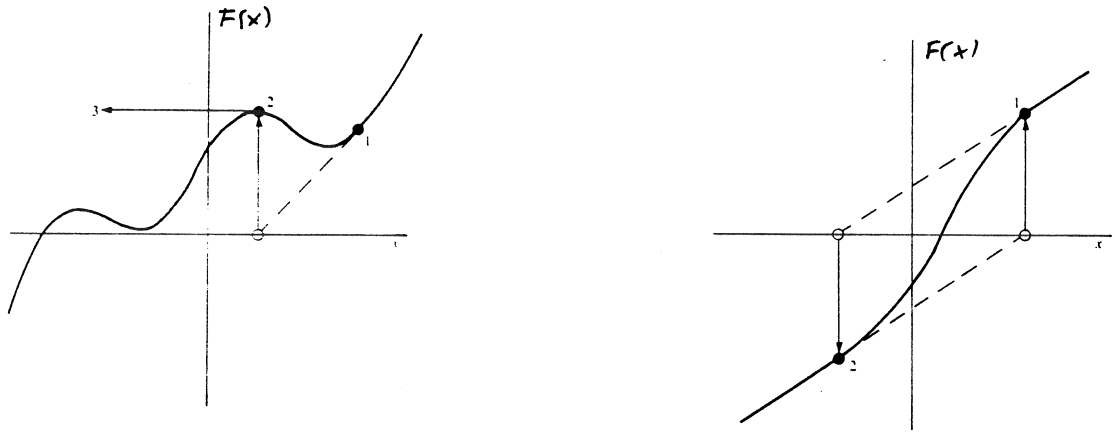


Abbildung 5.6: Probleme bei der Newton-Raphson-Iteration.

Struktogramm 14 — FUNCTION RTNEWT(X1,X2,JMAX, XACC,FEHLER)

<table>
<tr><td colspan="3">X:=0.5*(X1+X2)</td></tr>
<tr><td colspan="3">Y (FUNC(X1,DF)*FUNC(X2,DF)) > 0.0 N</td></tr>
<tr><td colspan="2">FEHLER:=1
RTNEWT:=X
print:ERR(1) 'no zero in interval'
(return)</td><td>.....</td></tr>
<tr><td colspan="3">FEHLER:=2
J:=0</td></tr>
<tr><td colspan="3">DX:=FUNC(X,DF)/DF
X:=X-DX</td></tr>
<tr><td colspan="3">Y (X1-X)*(X-X2) < 0.0 N</td></tr>
<tr><td>FEHLER:=3</td><td colspan="2">Y |DX/X| < XACC N</td></tr>
<tr><td></td><td>FEHLER:=0</td><td>.....</td></tr>
<tr><td colspan="3">J:=J+1</td></tr>
<tr><td colspan="3">J>JMAX .or. FEHLER≠2</td></tr>
<tr><td colspan="3">Y FEHLER=2 N</td></tr>
<tr><td colspan="2">print: 'ERR(2): no convergence'</td><td>.....</td></tr>
<tr><td colspan="3">Y FEHLER=3 N</td></tr>
<tr><td colspan="2">print: 'ERR(3): jumped out of brackets'</td><td>.....</td></tr>
<tr><td colspan="3">RTNEWT:=X</td></tr>
<tr><td colspan="3">(return)</td></tr>
</table>

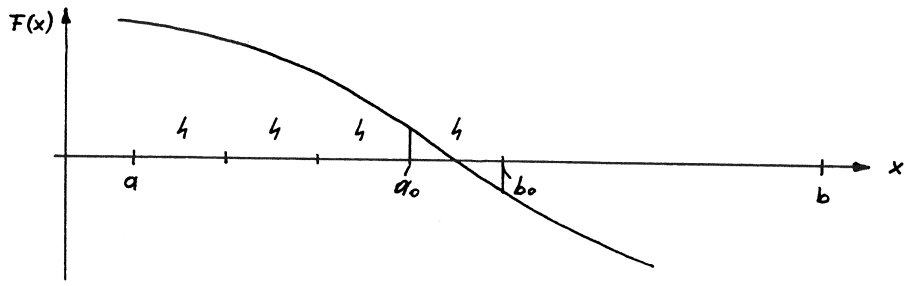


Abbildung 5.7: Prinzip einer Nullstellen-Grobsuche.

5.3.3 Ein Testprogramm für RTNEWT.

Im allgemeinen hat die gegebene Funktion $F(x)$ mehr als eine reelle Nullstelle. In solchen Fällen ist es vernünftig, die Newton-Raphson-Iteration mit einer sogenannten 'Grob-Suche' zu kombinieren.
Das Prinzip dieser einfachen Strategie ist in der Abb. 5.7 dargestellt: die x-Achse wird in Schritten der Breite h auf einen Vorzeichenwechsel der Funktion $F(x)$ untersucht. Wenn ein Intervall mit der Eigenschaft $F(a_0) \cdot F(b_0)$ auftritt, so befindet sich in $[a_0, b_0]$ (mindestens) eine Nullstelle, und die Newton-Iteration kann beginnen.
Solch eine 'Grob-Suche' ist auch Teil des in Abschnitt 5.5 behandelten *Intervallschachtelungsverfahrens.*

Es folgt nun ein Anwendungsbeispiel in C:

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

double func(double x,double *df)
// Diese Funktion berechnet die Funktionswerte sowie
// die Werte der ersten Ableitung fuer das Testbeispiel 5.3.3
{
  *df=4*pow(x,3)-27*pow(x,2)-4*x+120;
  return pow(x,4)-9*pow(x,3)-2*pow(x,2)+120*x-130;
}

double rtnewt(double x1, double x2, int jmax, double xacc, int *fehler)
// Diese Funktion bestimmt die reelle Nullstelle der Funktion 'func'
// im Intervall [x1,x2] mit der relativen Genauigkeit 'xacc'.
// Nach Beendigung der Rechnung kann die Variable 'fehler' einen der
// folgenden Werte haben:

//         fehler=0       Die Newton-Raphson Iteration ist ok.
//         fehler=1       Kein Vorzeichenwechsel der Funktion 'func'
//                        im Intervall [x1,x2].
//         fehler=2       Keine Konvergenz waehrend 'jmax' Iter.schritten.
//         fehler=3       Waehrend der Iteration springt der x Wert
//                        aus dem Intervall [x1,x2].
```

```
{
  int j,pause;
  double x,df,dx;

  x=0.5*(x1+x2);

  if((func(x1,&df)*func(x2,&df)) > 0.0) {
    *fehler=1;
    printf("ERROR(1): kein Vorzeichenwechsel in [x1,x2]\n");
    return x;
  }

  *fehler=2;
  j=0;

  do {
    dx=func(x,&df)/df;
    x=x-dx;

    if(((x1-x)*(x-x2))<0.0) *fehler=3;
    else {
      if(fabs(dx/x) < xacc) *fehler=0;
    }
    j++;
  } while ((j<=jmax) && (*fehler==2));

  if(*fehler==2)printf("ERROR(2):  keine Konvergenz\n");
  if(*fehler==3)printf("ERROR(3):  x aus [x1,x2] gesprungen\n");

  return x;
}


//             ****** main program ******
void main()
{
  int jmax,fehler,pause;
  double a,b,hgrob,eps,xl,xr,yl,yr,df,zero;

// Parameter fuer die Grobsuche:
  a=-10.0;  b=10.0;  hgrob=0.5;

// Parameter fuer die Newton-Raphson-Iteration:
  jmax=20;
  eps=0.0000001;

  printf("TEST:  NEWTON-RAPHSON-ITERATION MIT GROBSUCHE\n");
  printf("\n   Grobsuch-Bereich:       %7.3f  bis  %7.3f\n",a,b);
```

```
  printf("   Schrittweite Grobsuche: %7.3f\n",hgrob);
  printf("       Par. fuer Newton:  rel. Genauigkeit   = %12.10f\n", eps);
  printf("                          max. Iter.schritte = %4i\n\n", jmax);

// Grobsuche:
  xl=a;
  yl=func(xl,&df);
  xr=a+hgrob;
  yr=func(xr,&df);

  do {
    if(yl*yr < 0.0) {
// Grobsuche hat Vorzeichenwechsel gefunden; Newton-Iteration wird gestartet:
      zero=rtnewt(xl,xr,jmax,eps,&fehler);

      if(fehler==0)
        printf("  Nullstelle = %9.6f\n",zero);
      else printf("   fehler(%1i) in RTNEWT\n",fehler);
    }
    xl=xr;
    yl=yr;
    xr=xl+hgrob;
    yr=func(xr,&df);
  } while(xl <= b);
  printf("\nRechnung beendet.\n");
}
```

```
*********************************************
TEST:  NEWTON-RAPHSON-ITERATION MIT GROBSUCHE
*********************************************

   Grobsuch-Bereich:       -10.000  bis   10.000
   Schrittweite Grobsuche:   0.500
       Par. fuer Newton:  rel. Genauigkeit   = 0.0000001000
                          max. Iter.schritte =   20

  Nullstelle = -3.600135
  Nullstelle =  1.228589
  Nullstelle =  3.972068
  Nullstelle =  7.399477

Rechnung beendet.
```

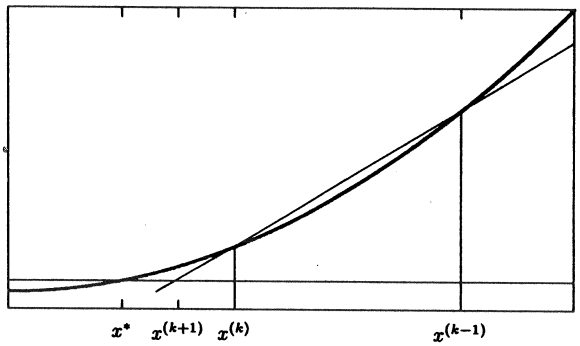


Abbildung 5.8: Grafische Interpretation der Regula Falsi.

5.4 Die Regula Falsi.

Diese mit dem Newton-Raphson-Verfahren nahe verwandte Methode wird häufig dann verwendet, wenn die Berechnung der Ableitung der Funktion $F(x)$ aus irgendwelchen Gründen Schwierigkeiten bereitet. In solchen Fällen kann der *Differentialquotient* in (5.13) durch den *Differenzenquotienten* ersetzt werden:

$$x_{t+1} = x_t - F(x_t) \cdot \frac{x_t - x_{t-1}}{F(x_t) - F(x_{t-1})} \tag{5.17}$$

Grafisch bedeutet das, daß die Newton'sche *Tangentenmethode* durch eine *Sekantenmethode* ersetzt wird (s. Abb.5.8).

5.5 Das Intervallschachtelungsverfahren.

Um die rellen Nullstellen der Funktion $F(x)$ in einem vorgegebenen Intervall $[a, b]$ zu ermitteln, wird - ausgehend von a - mit einer vorgegebenen Schrittweite h eine *grobe Lokalisierung* der Nullstellen vorgenommen.

Angenommen, die erste Nullstelle befinde sich im Intervall $[a_0, b_0]$ (s. Abb.5.7). Dies wird daran erkannt, daß innerhalb dieses Intervalls die Funktion $F(x)$ einen *Vorzeichenwechsel* hat, daß also gilt:

$$F(a_0) \cdot F(b_0) < 0\,.$$

Ist ein Intervall mit der obigen Eigenschaft gefunden worden, folgt die eigentliche Intervallschachtelung. Diese beginnt mit der Mittelung des Intervalls $[a_0, b_0]$:

$$x_0 = \frac{a_0 + b_0}{2}\,.$$

Es gibt nun drei Möglichkeiten:

1. $F(x_0) = 0$: x_0 (genau) die gesuchte Nullstelle.
   (Dieser Fall wird wegen der Rundungsfehler bei der Funktionsauswertung selten auftreten).

2. $F(a_0) \cdot F(x_0) < 0$ In diesem Fall liegt die gesuchte Nullstelle im linken Halbintervall $[a_0, x_0]$.

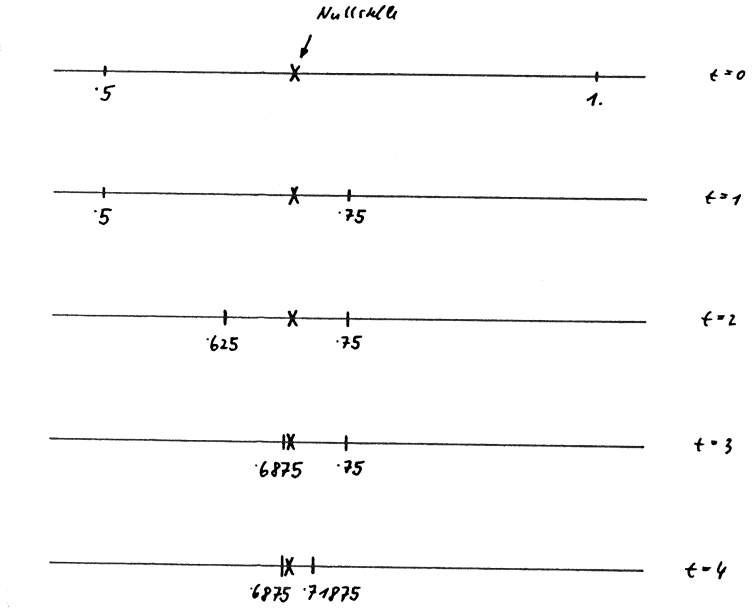
3. $F(a_0)\cdot F(x_0) > 0$ In diesem Fall liegt die gesuchte Nullstelle im rechten Halbintervall $[a_0, x_0]$.

Im Fall (1) ist die Nullstelle gefunden, in den Fällen (2) und (3) wird das linke bzw. rechte Halbintervall weiter geteilt. Dieser Prozess wird solange fortgesetzt, bis die Nullstelle mit der gewünschten Genauigkeit ermittelt ist, d. h. bis die aktuelle Intervallbreite nach t Teilungen ($[a_t, b_t]$) kleiner ist als die geforderte Genauigkeit der Nullstelle.

Dieses einfache Algorithmus wird durch das folgende Beispiel verdeutlicht: gesucht sei die Nullstelle von

$$F(x) = \mathrm{e}^{-x} - \frac{1}{2}\,,$$

welche den Wert $x_{exakt} = \ln 2 = 0.6931472\ldots$ hat. Das mittels einer Grobsuche gefundene Anfangs-Intervall $[a_0, b_0]$ sei [0.5,1.0]. Die Intervallschachtelung läuft nun wie folgt ab:



Es soll noch erwähnt werden, daß die Intervallschachtelungsmethode zu jenen numerischen Verfahren gehört, bei denen – zumindest für den absoluten Fehler – eine *a priori* Fehlerabschätzung in der Form

$$|x_t - x_{exakt}| \leq \frac{1}{2}(b_t - a_t) = \frac{b_0 - a_0}{2^{t+1}}\,, \tag{5.18}$$

möglich ist, wobei t die Anzahl der erfolgten Intervallhalbierungen ist. Diese Formel zeigt einen Nachteil dieses Verfahrens auf, nämlich seine relativ langsame Konvergenz: man braucht 3.3 Rechenschritte, um die absolute Genauigkeit des Ergebnisses um eine Zehnerpotenz zu erhöhen (s. Abschnitt 5.5.2).

5.5.1 Probleme beim Intervallschachtelungsverfahren.

Im Prinzip ist diese Methode sehr sicher. Ist eine Nullstelle erst einmal im Intervall $[a_0, b_0]$ lokalisiert, so ist ihre Berechnung (abgesehen von Rundungs-

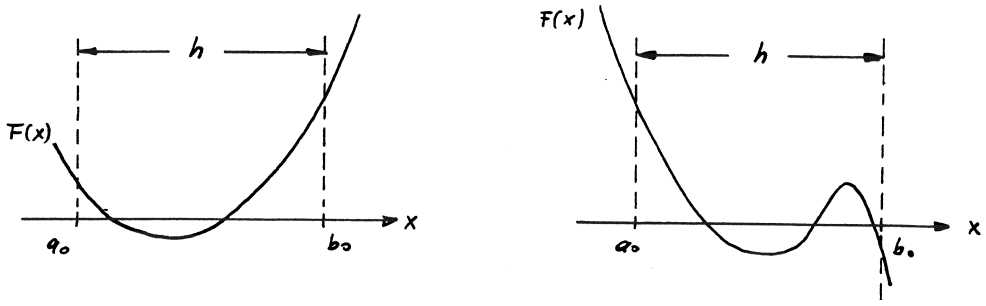


Abbildung 5.9: Probleme beim Intervallschachtelungsverfahren.

fehlern) mit beliebiger Genauigkeit möglich. Es gibt also keinerlei Konvergenzprobleme.
Die einzige Unsicherheitsquelle liegt in der *Wahl der Schrittweite h für die Grobsuche*. Das Kriterium: Vorzeichenwechsel = Nullstelle ist nämlich keinesfalls unproblematisch (s. Abb. 5.9):

- Im Fall Abb. 5.9, links findet *kein Vorzeichenwechsel statt, obwohl sich innerhalb des Intervalls Nullstellen befinden.* Dies trifft offenbar immer dann zu, wenn sich im Intervall $[x_1, x_2]$ eine *gerade* Anzahl von Nullstellen befindet.
- Im Fall Abb. 5.9, rechts *resultiert der Vorzeichenwechsel aus einer ungeraden Anzahl von Nullstellen.*

In beiden Fällen ist also die Anfangs-Schrittweite h zu groß gewählt worden! h stellt also die *Auflösungsgrenze* des Verfahrens dar:

Es können (im allgemeinen) nur solche Nullstellen gefunden werden, deren Abstand nicht kleiner ist als die Anfangs-Schrittweite h.

5.5.2 Das Programm INTSCH.

Quelle: [2],S.281f mit einigen Änderungen und Vereinfachungen.

INPUT-Parameter:

ANF,AEND: Beginn und Ende des Grobsuchbereiches.

H: Schrittweite der Grobsuche.

GEN: Fehlergrenze (i. a. relativer Fehler; s. Anmerkung).

ANZMAX: maximale Zahl der Nullstellen gemäß dem Speicherplatz im Ausgabefeld NULLST.

OUTPUT-Parameter:

NULLST( ): Feld mit den berechneten reellen Nullstellen.

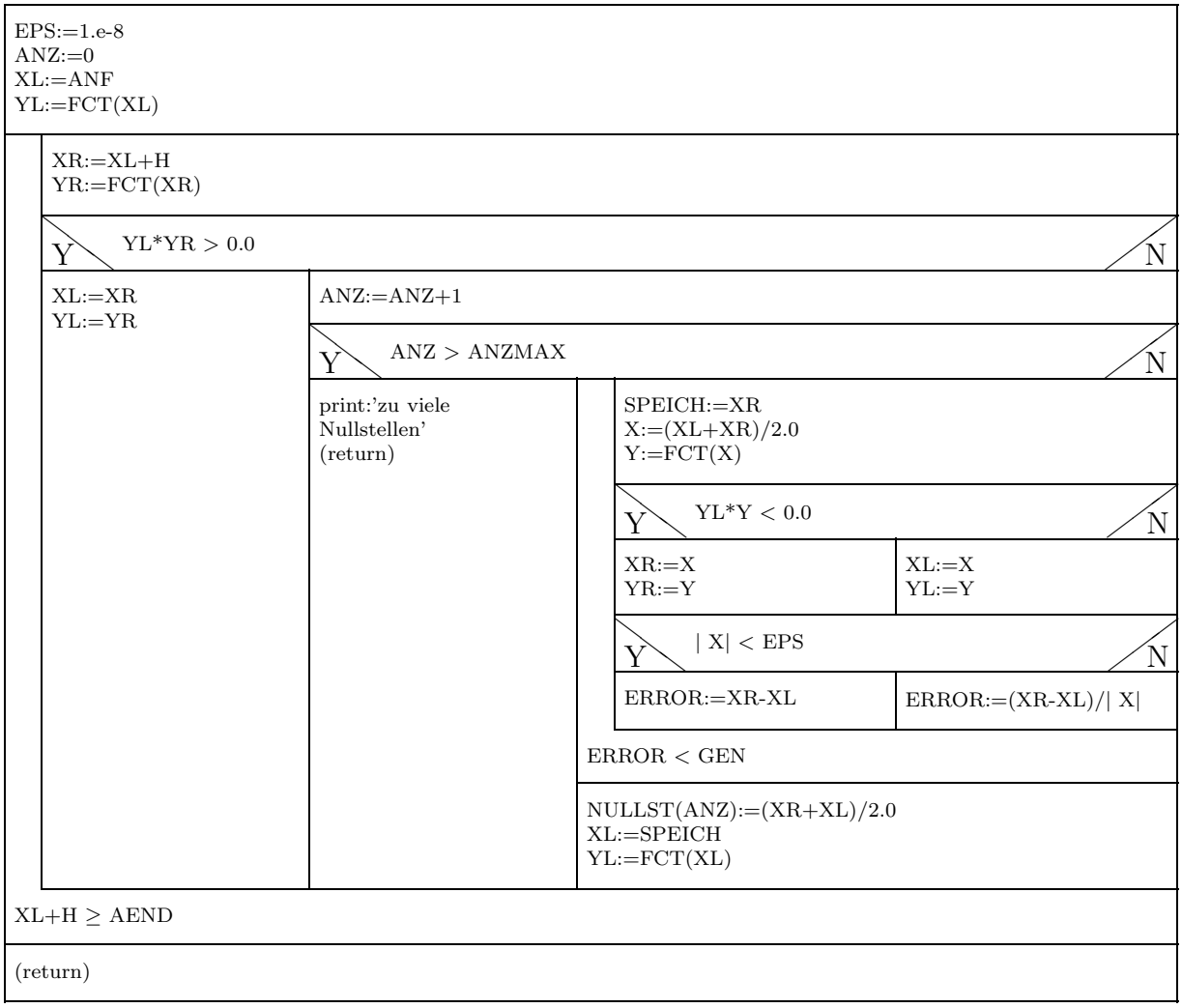
ANZ: Anzahl der von INTSCH berechneten Nullstellen.

Fehlerdiagnostik:
ANZ=0: es wurden keine Nullstellen gefunden.
ANZ=ANZMAX+1: es wurden mehr als ANZMAX Nullstellen gefunden, aber nur die ANZMAX ersten Nullstellen wurden im Feld NULLST abgespeichert.

allgemeine Anmerkung:
Das im folgenden präsentierte Programm ist gegenüber der Quelle deutlich vereinfacht; insbesondere ist der Fall nicht berücksichtigt, daß eine Nullstelle *exakt* auf einer Intervallgrenze liegt.

Anmerkung zur Fehlerabfrage:
Im Programm INTSCH wird standardmäßig eine relative Fehlerabfrage durchgeführt. Nur wenn die Nullstelle sehr klein ist (dem Betrage nach kleiner als 10^{-8}), wird auf eine absolute Fehlerabfrage umgeschaltet.

Struktogramm 15 — INTSCH(ANF,AEND,H,GEN,ANZMAX,NULLST,ANZ)



Eine Testbeispiel für INTSCH

Die reellen Nullstellen der algebraischen Gleichung

$$F(x) = x^4 - 9x^3 - 2x^2 + 120x - 130 = 0$$

sind im Bereich von –10.0 bis +10.0 mit einer relativen Genauigkeit von GEN=10^{-7} zu bestimmen; die Intervallbreite bei der Grobsuche sei 0.5.

INTSCH liefert die folgenden vier Nullstellen:

```
.
      1       -3.600135
      2        1.228589
      3        3.972068
      4        7.399477
```

Dieses Ergebnis ist natürlich exakt dasselbe wie bei der Newton-Iteration (vgl. Abschnitt 5.3.3). Es ist hier interessant, den Aufwand zu untersuchen, den beide Methoden benötigen, um ein Ergebnis dieser Genauigkeit zu erzielen.
Wenn man die Anzahl der Funktionsaufrufe von 'func' (im Newton-Raphson-Programm) abzählt, kommt man auf 66 Aufrufe; die gleiche Analyse für das Intervallschachtelungsprogramm ergibt 131 Aufrufe von 'fct'. Da aber beim Newton-Verfahren bei jedem Aufruf von 'func' 2 Funktionen berechnet werden, nämlich der Funktionswert sowie der Wert der ersten Ableitung, ist der numerische Aufwand in beiden Programmen etwa gleich.

Diese Analyse gibt allerdings ein falsches Bild von der Leistungsfähigkeit der Newton-Raphson-Iteration. Ein großer Teil der Funktionsaufrufe von 'func' geschieht bei der Grobsuche, bei der die Berechnung der Ableitungen unnötig ist! Es wäre daher ökonomischer, beim Newton-Raphson-Verfahren zwei unabhängige Funktionen (z.B. 'func' und 'dfunc') zu definieren, wobei 'func' nur die Funktion und 'dfunc' nur die Ableitung berechnet.
Geht man so vor, ergeben sich bei der Anwendung des Programms RTNEWT auf das Testproblem nur mehr 66 Aufrufe von 'func' und 15 Aufrufe von 'dfunc', also insgesamt 81 Funktionsaufrufe.

D.h.: Je ökonomischer man die Grobsuche gestaltet, desto stärker wird die Überlegenheit der Newton-Iteration gegenüber der Intervallschachtelung deutlich. Betrachten Sie etwa die erste Nullstelle: sie liegt im 'Grobsuch-Intervall' [-4.0, -3.5]. Im folgenden sehen Sie die weitere Annäherung an die Nullstelle, links mittels Newton-Raphson und rechts mittels Intervallschachtelung:

```
Newton-Raphson    Int.schachtelung
-3.750000            -3.750000
-3.609011            -3.625000
-3.600169            -3.562500
-3.600135            -3.593750
-3.600135            -3.609375
                     -3.601562
                     -3.597656
                     -3.599609
                     -3.600586
                     -3.600098
                     -3.600342
                     -3.600220
                     -3.600159
                     -3.600128
                     -3.600143
                     -3.600136
                     -3.600132
                     -3.600134
                     -3.600135
                     -3.600135
                     -3.600135
                     -3.600135
```

Ein Vorteil der Intervallschachtelungsmethode gegenüber Newton-Raphson besteht natürlich darin, daß erstere keine Ableitungen der Funktion $F(x)$ benötigt! Diesen Vorteil teilt sie mit der im Abschnitt 5.4 kurz beschriebenen Regula Falsi.

5.5.3 Ein Anwendungsbeispiel aus der Quantenmechanik.

Im folgenden Beispiel geht es um die Energie-Eigenwerte eines eindimensionalen Potentialtopfes:

Gesucht sind jene Energiewerte E, die ein Teilchen der Masse m unter dem Einfluß des *Kastenpotentials* (s.Abb.5.10)

$$V(x) = \begin{array}{lll} 0 & \text{für} & -\infty < x \leq 0 \quad \text{Bereich I} \\ -V_0 & \text{für} & 0 < x \leq a \quad \text{Bereich II} \\ 0 & \text{für} & a < x < \infty \quad \text{Bereich III} \end{array}$$

annehmen kann, wobei nur *gebundene Zustände* berücksichtigt werden sollen:

$$-V_0 < E < 0$$

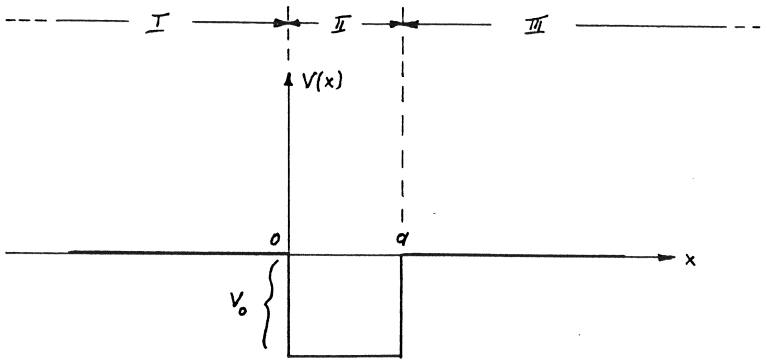


Abbildung 5.10: Der eindimensionale Potentialtopf.

Zu lösen ist also die eindimensionale Schrödingergleichung

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + V(x)\psi(x) = E\psi(x) \quad .$$

In atomaren Einheiten, d.h. Längen in Bohr, Energien in Rydberg ergibt sich $\hbar^2/2m \equiv 1$, und man erhält die Differentialgleichung

$$\psi''(x) + [E - V(x)]\psi(x) = 0$$

mit den Randbedingungen

$$\psi(+\infty) \to 0 \qquad \text{und} \qquad \psi(-\infty) \to 0 \quad .$$

Lösungsansätze:

- Für die Bereiche I und III:

$$\psi''(x) + E \cdot \psi(x) = 0 \quad \to \qquad \begin{array}{c} \psi_I(x) = A_I e^{\sqrt{-E}x} + B_I e^{-\sqrt{-E}x} \\ \psi_{III}(x) = A_{III} e^{\sqrt{-E}x} + B_{III} e^{-\sqrt{-E}x} \end{array}$$

  Die Randbedingungen sind nur für $B_I = A_{III} = 0$ erfüllt, und man erhält:

$$\psi_I(x) = A_I e^{\sqrt{-E}x} \qquad \text{und} \qquad \psi_{III}(x) = B_{III} e^{-\sqrt{-E}x} \tag{5.19}$$

- Bereich II:

$$\psi''(x) + [E + V_0]\psi(x) = 0 \quad \to$$

$$\psi_{II}(x) = A_{II} \sin\left(x\sqrt{E + V_0}\right) + B_{II} \cos\left(x\sqrt{E + V_0}\right) \tag{5.20}$$

Zusätzlich sind noch die 4 *Anschlußbedingungen*

$$\begin{array}{rcl} \psi_I(0) & = & \psi_{II}(0) \\ \psi_I'(0) & = & \psi_{II}'(0) \\ \psi_{II}(a) & = & \psi_{III}(a) \\ \psi_{II}'(a) & = & \psi_{III}'(a) \end{array} \tag{5.21}$$

zu erfüllen. Setzt man nun die Ansätze (5.19) und (5.20) in die Gleichungen (5.21) ein, so erhält man 4 lineare, homogene Gleichungen für die noch unbekannten Koeffizienten A_I, A_{II}, B_{II} und B_{III}:

(Abkürzung: $\sqrt{E+V_0} \equiv \kappa$).

$$\begin{aligned} A_I - B_{II} &= 0 \\ A_I\sqrt{-E} - A_{II}\kappa &= 0 \\ A_{II}\sin\kappa a + B_{II}\cos\kappa a - B_{III}e^{-\sqrt{-E}a} &= 0 \\ A_{II}\kappa\cos\kappa a - B_{II}\kappa\sin\kappa a + B_{III}\sqrt{-E}e^{-\sqrt{-E}a} &= 0 \end{aligned}$$

In Matrix-Schreibweise hat dieses System die Form

$$\underbrace{\begin{pmatrix} 1 & 0 & -1 & 0 \\ \sqrt{-E} & -\kappa & 0 & 0 \\ 0 & \sin\kappa a & \cos\kappa a & -e^{-\sqrt{-E}a} \\ 0 & \kappa\cos\kappa a & -\kappa\sin\kappa a & \sqrt{-E}e^{-\sqrt{-E}a} \end{pmatrix}}_{M(E)} \begin{pmatrix} A_I \\ A_{II} \\ B_{II} \\ B_{III} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (5.22)$$

Der Lösungsvektor des Systems (5.22) ist nur dann vom Nullvektor verschieden, wenn die Determinante der Koeffizientenmatrix $M(E)$ verschwindet, d.h. das Teilchen kann nur jene Energiewerte haben, für welche gilt:

$$\det(M(E)) = 0$$

Die analytische Auswertung der Determinante ist elementar, und man erhält den Ausdruck

$$\det(M(E)) = -e^{-a\sqrt{-E}}\left[(V_0+2E)\sin\left(a\sqrt{E+V_0}\right) - 2\sqrt{-E(E+V_0)}\cos\left(a\sqrt{E+V_0}\right)\right]$$

Unter Berücksichtigung der Tatsache, daß $\exp(-a\sqrt{-E})$ stets > 0 ist, ergibt sich: *Die Energie-Eigenwerte des Problems sind die reellen Lösungen der Gleichung*

$$F(E) = (V_0+2E)\sin\left(a\sqrt{E+V_0}\right) - 2\sqrt{-E(E+V_0)}\cos\left(a\sqrt{E+V_0}\right)$$

Die Nullstellensuche für beliebige a und V_0 soll mittels der Intervallschachtelungs-Methode durchgeführt werden. Diese ist im konkreten Fall der Newton-Raphson-Iteration vorzuziehen, weil man sich dadurch die Berechnung der Ableitung der Funktion $F(E)$ erspart.

Ergebnis für einen Potentialtopf mit $a = 2$ Bohr und $V_0 = 225$ Rydberg:

```
.
Es wurden 10 Energiewerte mit der rel. Genauigkeit von 0.000001
ermittelt:

     1   Energie [Ry] = -222.83185
     2   Energie [Ry] = -216.33258
     3   Energie [Ry] = -205.51910
     4   Energie [Ry] = -190.42145
     5   Energie [Ry] = -171.08820
     6   Energie [Ry] = -147.59515
     7   Energie [Ry] = -120.06418
     8   Energie [Ry] =  -88.70779
     9   Energie [Ry] =  -53.96208
    10   Energie [Ry] =  -17.15278
```

5.6 Nichtlineare Gleichungssysteme.

Probleme dieser Art sollen hier nur prinzipiell erläutert werden, weil sie sich - wie sogleich gezeigt wird - mittels des bereits ausführlich behandelten Gauss-Newton-Marquardt-Verfahrens numerisch lösen lassen.

Hat man nämlich anstatt einer einzelnen Gleichung $F(x) = 0$ ein *System von n nicht-linearen Gleichungen mit den n Unbekannten* x_1, x_2,..., x_n vorliegen, also

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ F_k(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ F_n(x_1, x_2, \ldots, x_n) &= 0 \quad , \end{aligned} \tag{5.23}$$

so werden diese Gleichungen nur dann simultan erfüllt, wenn gilt:

$$S = \sum_{k=1}^{n} [F_k(x_1, x_2, \ldots, x_n)]^2 = 0$$

In diesem Fall ist also Null das anzustrebende Minimum der Summe S, und gesucht wird jener Vektor (werden jene Vektoren) $\mathbf{x}$, der (die) dies bewerkstelligt (bewerkstelligen)!

$$S = \sum_{k=1}^{n} [F_k(\mathbf{x})]^2 \rightarrow \text{Minimum} \tag{5.24}$$

Damit ist das Problem (5.23) auf ein *spezielles nicht-lineares Least-Squares-Problem zurückgeführt*, und man kann das Gauss-Newton-Marquardt-Verfahren (s. Kap. 4) anwenden:

- Linearisierung des Modells durch Taylorreihen-Entwicklungen der Funktionen $F_k(\mathbf{x})$ an der Stelle $\mathbf{x} = \mathbf{x}^o$ ('Startvektor'):

$$F_k(\mathbf{x}) = F_k(\mathbf{x}^o) + \sum_{j=1}^{n} \left(\frac{\partial F_k(\mathbf{x})}{\partial x_j} \right)_{\mathbf{x}^o} \cdot (x_j - x_j^o) + \cdots \quad . \tag{5.25}$$

  Im weiteren werden wieder die Abkürzungen

$$dF_{k,j} \equiv \left(\frac{\partial F_k(\mathbf{x})}{\partial x_j} \right)_{\mathbf{x}^o} \qquad \text{und} \qquad F_k \equiv F_k(\mathbf{x}^o)$$

  verwendet.

- Einsetzen der Entwicklungen in (5.24) und partielle Ableitung der Summe nach den gesuchten Parametern x_1 bis x_n:

$$S \approx \sum_{k=1}^{n} \left(F_k + \sum_{j=1}^{n} dF_{k,j} \cdot (x_j - x_j^o) \right)^2$$

$$\frac{\partial S}{\partial x_l} = 2 \sum_{k=1}^{n} \left(F_k + \sum_{j=1}^{n} dF_{k,j} \cdot (x_j - x_j^o) \right) \cdot F_{k,l} = 0$$

  für alle $l = 1, \ldots, n$.

- Daraus resultiert das folgende lineare Gleichungssystem für eine *iterative Verbesserung* der Lösungen:

$$A \cdot (\mathbf{x} - \mathbf{x}^o) = \beta \qquad \text{mit}$$

$$A = [\alpha_{ij}] \qquad \alpha_{ij} = \sum_{k=1}^{n} dF_{k,i} \cdot dF_{k,j} \qquad \text{und} \qquad \beta_i = -\sum_{k=1}^{n} F_k \cdot dF_{k,i} \tag{5.26}$$

  Vergleicht man nun das System (5.26) mit dem System (4.20), so ist die nahe Verwandtschaft sofort evident. Auch zur Lösung von (5.26) ist es sehr zu empfehlen, die 'Marquardt-Variation' zu verwenden, um eine hohe Konvergenz-Sicherheit der Iteration zu erhalten.

5.6.1 Ein Testbeispiel.

Man löse das Gleichungssystem[1]

$$x^3 - 3xy^2 - 1 = 0 \qquad y^3 - 3x^2y = 0$$

nach der im Kap. 5.6. angegebenen Methode.

Exakte Lösungen:
$(x = 1,\, y = 0)$ $(x = -1/2,\, y = \sqrt{3}/2)$ $(x = -1/2,\, y = -\sqrt{3}/2)$

Numerische Lösung mittels der Gauss-Newton-Marquardt-Methode:

```
.
abs. Gen. der Fitwerte = .100000E-06

tmax =    50

guessed values:   x(1) = -.2000000E+00
                  x(2) = -.5000000E+00

  0    .740389E+00     -.200000E+00     -.500000E+00
*
*
*
*
*
  1    .100543E+00     -.595187E+00     -.823118E+00
  2    .374490E-02     -.513359E+00     -.850427E+00
  3    .478015E-05     -.500104E+00     -.865304E+00
  4    .227994E-09     -.500000E+00     -.866020E+00
  5    .543610E-15     -.500000E+00     -.866025E+00
  6    .543610E-15     -.500000E+00     -.866025E+00

Ergebnisse mittels MRQMIN und MRQCOF:
====================================
  x(1) = -.50000E+00      (exakt: -0.5000000)
  x(2) = -.86603E+00      (exakt: -0.8660254)
```

Anmerkung: Wie bei iterativen Methoden in der Anwendung auf nicht-lineare Probleme üblich, hängt es von den Startwerten des GNM-Prozesses ab, welche der drei Lösungen man erhält. Mit den oben gewählten Startwerten bekommt man offenbar die dritte Lösung.

Die erste Lösung würde man z.B. für die Startwerte $x(1)$=–1.0 und $x(2)$=0.0 erhalten, die zweite Lösung für die Startwerte $x(1)$=–1.0 und $x(2)$=1.0.

[1] aus: F. Stummel, K. Hainer, *Praktische Mathematik*, Teubner Studienbücher, Stuttgart 1971.

5.7 Software-Angebot

TOMS (www.netlib.org) bietet eine Implementierung des Newton-Verfahrens in Verbindung mit Intervallschachtelung unter TOMS-681 sowie
unter TOMS-378 eine Programmbeschreibung in ALGOL: *Discretized Newton-like method for solving a system of simultaneous nonlinear equations.*

NAG: *c05ajf* und *c05axf*
verwenden die *Regula falsi* zur Lösung nicht-linearer Gleichungen.

Numerical Recipes: Abgesehen vom Newton-Raphson-Programm 'rtnewt', das im Abschnitt 5.3.2 dieses Skriptums vorgestellt wird, bieten die NumRec-Bücher eine Reihe anderer Routinen zur Auffindung reeller Nullstellen reellwertiger Funktionen: hervorzuheben ist hier die Routine 'zbrent', welche auf Wijngaarden, Dekker und Brent zurückgeht (1960). Für die iterative Annäherung an komplexwertige Nullstellen von Polynomen mit komplexwertigen Koeffizienten sollten Sie das Programm 'laguer' probieren (Methode von Laguerre).

Numerical Algorithms with C or FORTRAN: Im Kapitel 2 dieses Buches (bzw. im entsprechenden Unterverzeichnis des Linux-PC) finden Sie die folgenden Programme:

`fnewton.c`	`newpsz.f90`	`Newtonverfahren fuer reelle Funktionen`
`fpegasus.c`	`pegasu.f90`	`Pegasusverfahren fuer reelle Funktionen`
`froots.c`		`Pegasus-, Pegasus-King-, Anderson-Bjoerck-und Anderson-Bjoerck-King-Verfahren`
`fzeroin.c`	`zeroin.f90`	`Zeroin-Verfahren fuer reelle Funktionen`
`fmueller.c`	`muller.f90`	`Verfahren von Mueller zur Berechnung aller Nullstellen eines reellen Polynoms`
`fbauhube.c`	`baupol.f90`	`Verfahren von Bauhuber zur Berechnung aller Nullstellen eines komplexen Polynoms`
`flaguer.c`	`laguer.f90`	`Verfahren von Laguerre zur Berechnung aller reellen Nullstellen eines reellen Polynoms`

MATLAB bietet neben anderen Routinen die Programme

`roots.m`	`Berechnung der Nullstellen eines Polynoms, wobei das Funktionsargument die Koeffizienten des Polynmos sind.`
`fzero.m`	`Berechnung der Nullstelle einer Funktion F(x) in der Naehe eines Anfangswertes x0.`