Numerische Methoden in der Physik (515.421)

Heinrich Sormann
Institut für Theoretische Physik - Computational Physics
TU Graz

Skriptum WS 2012/2013

Das Problem im Umgang mit Computern ist,
daß sie nur Antworten geben.

Pablo Picasso zugeschrieben...

Kapitel 1

Einführung

1.1 Grundbegriffe

Die Lehrveranstaltung *Numerische Methoden in der Physik* beschäftigt sich mit der Lösung physikalisch-technischer Probleme unter Verwendung von Methoden der Numerischen Mathematik, wobei die Komplexizität der Problemstellungen den Einsatz eines Computers erforderlich macht.

Eine Definition des Begriffes *Numerische Mathematik* findet sich z.B. in [1], S.8:

Die Numerische Mathematik befaßt sich mit der Lösung von mathematischen Problemen mit Hilfe zahlenmäßigen Rechnens. Darunter ist eine endliche Folge von Grundrechenoperationen, also Addition, Subtraktion, Multiplikation und Division zu verstehen, wobei Zahlen mit einer beschränkten Stellenanzahl verwendet werden.

Ebenfalls in [1] ist die grundsätzliche Methodik der Numerischen Mathematik zusammen mit den unvermeidlich auftretenden Fehlern schematisch dargestellt:

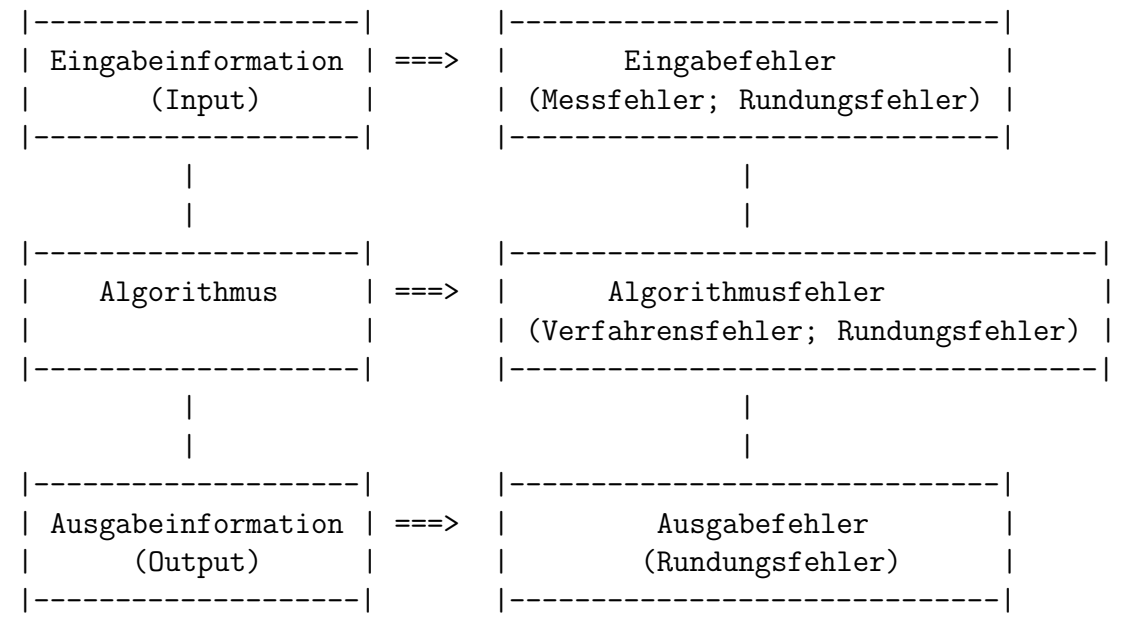


Diagramm zur Methodik der Numerischen Mathematik.

Ein zentraler Begriff der Numerischen Mathematik ist der Begriff *Algorithmus*. Darunter versteht man nach [2], S.9

eine endliche Menge von genau beschriebenen Anweisungen, die mit vorgegebenen Anfangsdaten in bestimmter Reihenfolge nacheinander auszuführen sind, um die Lösung zu ermitteln.

1.2 Allgemeines zum Thema: Fehler

Wie aus dem obigen Diagramm hervorgeht, tragen alle Aktivitäten bei der Durchführung einer numerischen Rechnung (Dateneingabe; eigentlicher Rechenvorgang; Datenausgabe) zum Fehler des Ergebnisses bei. Ursache dafür sind die auftretenden *Rundungsfehler* und *Verfahrensfehler*, auf die in den folgenden Kapiteln ausführlich eingegangen wird.

Ergebnisse von Computerberechnungen sind also (bis auf ganz wenige Ausnahmen) fehlerbehaftet, d.h. sie entsprechen nicht exakt den „wahren“ Ergebnissen des behandelten Problems.

Es ist daher von entscheidender Wichtigkeit, bei der Arbeit mit einem Computer die auftretenden Fehler unter Kontrolle (d.h. so klein wie irgend möglich) zu halten bzw. die Genauigkeit der erhaltenen Ergebnisse so gut als möglich beurteilen zu können.

Als Literatur zum Thema "Fehler bei der Anwendung numerischer Verfahrenßind die einschlägigen Kapitel aus [7] und [8] sehr zu empfehlen.

1.2.1 Absoluter und relativer Fehler. Maschinengenauigkeit.

Bevor auf die einzelnen Fehlertypen näher eingegangen wird, sollen im folgenden die Begriffe *absoluter* und *relativer* Fehler erläutert werden:

Ist x der (unbekannte) wahre Wert eines Rechenergebnisses und $\bar{x}$ der entsprechende Näherungswert, so nennt man

$$\epsilon_a = x - \bar{x} \tag{1.1}$$

den *absoluten* Fehler und

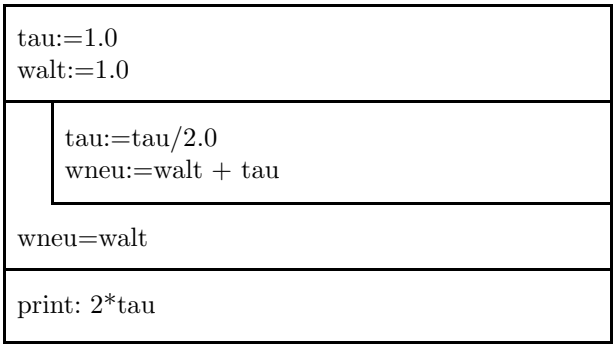
$$\epsilon_r = \frac{\epsilon_a}{\bar{x}} = \frac{x - \bar{x}}{\bar{x}} \tag{1.2}$$

den *relativen* Fehler von $\bar{x}$.

In der Praxis wird meistens der relativen Fehlerbetrachtung nach (1.2) der Vorzug gegeben. Der Grund dafür wird im folgenden einfachen Beispiel klar:

	x	$\bar{x}$	ϵ_a	ϵ_r
(a)	0.1	0.09	0.01	≈ 0.1
(b)	1000.0	999.99	0.01	≈ 0.00001

Struktogramm 1 — Ermittlung der Kenngröße τ



Es leuchtet sofort ein, daß die Näherung (b) weit befriedigender ist als die Näherung (a), obwohl der absolute Fehler in beiden Fällen gleich ist.

Ein weiterer wichtiger Begriff ist die **Maschinen-Genauigkeit**: Es ist dies die kleinste (positive) Zahl τ, für welche gilt:

$$1 + \tau > 1$$

Für einen (nicht-existierenden) Super-Rechner, der reelle Zahlen mit beliebiger Genauigkeit abspeichern kann, erfüllt natürlich jedes τ die obige Ungleichung. Für einen realen Computer ist τ jedoch eine wichtige Kenngröße, die mit dem Algorithmus im Struktogramm 1 leicht ermittelt werden kann.
Die folgenden Ergebnisse wurden auf einem IBM-PC erhalten (wie alle in diesem Skriptum enthaltenen Testergebnisse, soferne es nicht ausdrücklich anders angegeben ist):

	Bytes	τ	Anz. sign. Stellen
C-float	4	$1.19 \cdot 10^{-7}$	7
C-double	8	$2.22 \cdot 10^{-16}$	16
F90-real	4	$1.19 \cdot 10^{-7}$	7
F90-double prec.	8	$2.22 \cdot 10^{-16}$	16
Pascal-single	4	$1.19 \cdot 10^{-7}$	7
Pascal-real	6	$1.82 \cdot 10^{-12}$	12
Pascal-double	8	$2.22 \cdot 10^{-16}$	16
Pascal-extended	10	$1.08 \cdot 10^{-19}$	19

Realisierung des Struktogramms 1 in C und F90:

```
// C-PROGRAMM ZUR BERECHNUNG DER MASCHINEN-GENAUIGKEIT

#include <iostream.h>

void main()
{
  float tau,walt,wneu;

  tau=1.0;
  walt=1.0;
  do {
    tau/=2.0;
    wneu=walt+tau;
  } while (wneu != walt);
  cout<<"TAU = "<<2*tau<<"\n";
}
```

```
PROGRAM maschin

! F90-Programm zur Berechnung der Maschinen-Genauigkeit

IMPLICIT NONE

REAL tau,walt,wneu

tau=1.0
walt=1.0

DO
  tau=tau/2.0
  wneu=walt+tau
  IF(wneu .EQ. walt)EXIT
END DO

PRINT '(" tau = ",E12.6)',2.0*tau

END PROGRAM maschin
```

1.2.2 Eingabe- oder Input-Fehler. Schlecht konditionierte Probleme.

Eingabe- bzw. *Input*-Fehler (*inherent errors*) sind Unsicherheiten der eingegebenen Daten, mit welchen der Computer die Berechnung durchführt. Diese Unsicherheiten können verschiedene Ursachen haben, z.B. experimentelle Meßfehler oder *Rundungsfehler*.

Wenn die Ergebnisse einer Berechnung stark von diesen Eingabefehlern beeinflußt werden, spricht man von einem 'schlecht konditionierten Problem'.

Dazu als Beispiel ein schlecht konditioniertes lineares Gleichungssystem [7], S.97:

$$x + 5.0y = 17.0$$

$$1.5x + 7.501y = a$$

Angenommen, die Größe a sei der experimentell ermittelte Wert

$$a = 25.503 \pm 0.001$$

Wie man sich leicht überzeugen kann, lauten die exakten Lösungen dieses Systems für $a = 25.503$: $x = 2.0$ und $y = 3.0$.
Die folgende Tabelle zeigt nun, wie massiv sich die Ergebnisse des Gleichungssystems unter der Annahme ändern, daß die letzte Ziffer von a um eine Einheit unsicher ist:

a	x	y
25.503	2.	3.
25.502	7.	2.
25.504	-3.	4.

Das obige Gleichungssystem ist also extrem schlecht konditioniert, und die Ergebnisse sind (bei Annahme einer experimentell bedingten Unsicherheit von a) völlig sinnlos!

1.2.3 Algorithmusfehler

sind Fehler, die bei der Auswertung der Rechenvorschrift geschehen.
Die Ursache dafür sind entweder *Rundungsfehler* oder *Verfahrensfehler*.

1.2.4 Verfahrensfehler

Verfahrensfehler entstehen dadurch, daß das gegebene mathematische Problem durch ein vereinfachtes Problem ersetzt wird.
Ein Beispiel dafür ist die numerische Integration:

Das bestimmte Integral

$$I = \int_{x=1}^{2} \frac{dx}{x^2}$$

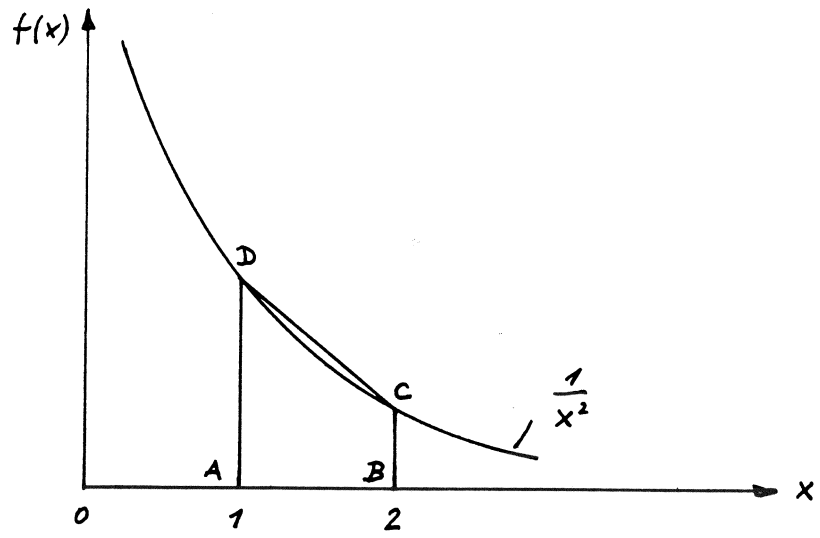


Abbildung 1.1: Prinzip einer numerischen Integration.

mit der exakten Lösung 0.5 kann z.B. numerisch dadurch gelöst werden, daß man die 'wahre' Fläche durch die Trapezfläche $ABCD$ ersetzt (Abb.1.1). Das 'numerische' Ergebnis lautet in diesem Fall 0.625, d.h. es tritt ein (absoluter) Verfahrensfehler

$$\epsilon_V = 0.5 - 0.625 = -0.125$$

auf. Weitere typische Verfahrensfehler ergeben sich z.B. aus dem vorzeitigen Abbrechen einer unendlichen Reihe (*Abbruchfehler*, engl. *truncation error*) usw.

1.2.5 Rundungsfehler

Die Tatsache, daß für die Abspeicherung einer reellen Zahl (genauer gesagt: für deren Mantisse) nur eine endliche Anzahl von Bits zur Verfügung steht, ist die Ursache für die in fast [1] jedem Computerprogramm auftretenden, unvermeidlichen [2] Rundungsfehler.

Ein simples Beispiel:

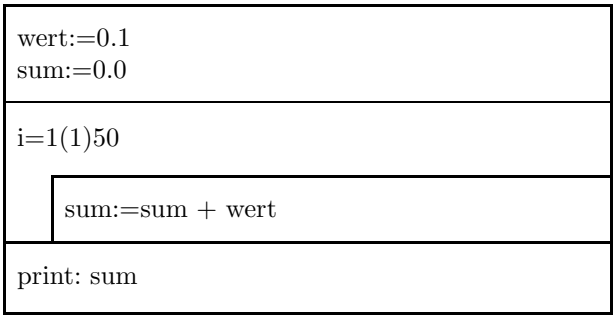
```
   Rechnung "von Hand":          Computer (7 sign. Stellen):

     123456.789                      0.1234568E+06
+      9.876543                    + 0.9876543E+01
---------------                    ----------------
     123466.665543                   0.1234667E+06
```

[1] Außer man kommt z.B. mit reiner INTEGER-Arithmetik aus, was sehr selten ist.

[2] 'Unvermeidlich' heißt, wie noch ausführlich demonstriert wird, keineswegs, daß man keine Möglichkeiten hat, Rundungsfehler zu reduzieren!

Struktogramm 2 — Demonstration Rundungsfehler-Effekt



Ein weiteres Beispiel: Die Auswertung des Struktogramms 2 zeigt ein auf den ersten Blick unverständliches Phänomen:
Auf dem PC lautet das Ergebnis 4.999998, d.h. man erhält einen Rundungsfehlereffekt, obwohl nur 50mal die Konstante 0.1 aufaddiert wird, die ja exakt abgespeichert werden kann.

Die Ursache für diesen Effekt ist darin begründet, daß die Zahl 0.1 im Dezimalsystem zwar ohne Schwierigkeiten hingeschrieben werden kann, nicht aber im vom Computer verwendeten Dualsystem. In diesem ist 0.1 nämlich periodisch:

$$0.1_{10} = 0.000110011001100\ldots_2$$

1.3 Verfahrens- und Rundungsfehler

Im folgenden soll das Fehlerverhalten von Computerprogrammen an Hand einiger typischer Beispiele demonstriert werden. Es sollen dabei Möglichkeiten gezeigt werden, wie man Verfälschungen von Resultaten auf Grund von Verfahrens- und Rundungsfehlern erkennen bzw. möglichst klein halten kann.

1.3.1 Zusammenhang zwischen Rundungsfehler und Algorithmus

Das erste Beispiel zeigt, wie die Rundungsfehler bei der Auswertung eines mathematischen Ausdrucks durch geeignete Umformulierung des Algorithmus reduziert werden können.

Es geht um die numerische Auswertung des Ausdruckes

$$F(x) = \frac{\sqrt{1+x} - \sqrt{1-x}}{x}$$

für $x << 1$. Da für $F(x)$ eine geschlossene Formel vorliegt, gibt es hier natürlich keinen Verfahrensfehler. Auch die Ein- und Ausgabefehler sollen bei diesem Beispiel vernachlässigt werden.

Programmiert man nun die Formel ohne weitere Umformung, also etwa

$$F := (SQRT(1. + X) - SQRT(1. - X))/X \quad ,$$

so erhält man für kontinuierlich kleiner werdende x-Werte das folgende Ergebnis:

Tab.1.1: Numerische Auswertung von F(x) bzw. F1(x).

x	F	F1
10^0	$0.1414214E+01$	$0.1414214E+01$
10^{-1}	$0.1001256E+01$	$0.1001256E+01$
10^{-2}	$0.1000013E+01$	$0.1000013E+01$
10^{-3}	$0.1000041E+01$	$0.1000000E+01$
10^{-4}	$0.1000153E+01$	$0.1000000E+01$
10^{-5}	$0.1001357E+01$	$0.1000000E+01$
10^{-6}	$0.1013279E+01$	$0.1000000E+01$
10^{-7}	$0.1192093E+01$	$0.1000000E+01$
10^{-8}	$0.0000000E+01$	$0.1000000E+01$

Bei den F-Werten in der obigen Tabelle fallen 2 Dinge auf:

- Die Funktionswerte streben nicht - wie zu erwarten - gegen 1, sondern nehmen ab ca. $x = 10^{-2}$ wieder zu.
- Ab einem minimalen x-Wert erhält man für F immer exakt Null.

Dieses Verhalten ist leicht erklärbar:

- Für kleine x repräsentieren die beiden Wurzeln $\sqrt{1+x}$ und $\sqrt{1-x}$ zwei *fast gleich große Zahlen*. Bei der Subtraktion dieser Zahlen und der anschließenden Division durch x wirken sich die (an sich kleinen) Rundungsfehler der einzelnen Terme nach dem Fehlerfortpflanzungsgesetz sehr stark aus ('subtractive cancellation').
- Für $x < \tau$ ergibt sich $1 + x = 1$ und $1 - x = 1$; somit wird der Zähler von $F(x)$ exakt Null.

Soweit die Fehlerdiagnose. Die Therapie ist in diesem Fall sehr einfach:
Formt man nämlich $F(x)$ in

$$F1(x) \equiv F(x) = \frac{(\sqrt{1+x} - \sqrt{1-x})}{x} \cdot \frac{(\sqrt{1+x} + \sqrt{1-x})}{(\sqrt{1+x} + \sqrt{1-x})} = \frac{2}{\sqrt{1+x} + \sqrt{1-x}}$$

um, so erhält man die korrekten Ergebnisse (s. Tabelle 1.1).

Ein weiteres Beispiel ([9], S.178) soll demonstrieren, daß man auch bei (scheinbar) eindeutigen Problemen in Schwierigkeiten geraten kann, wenn man bekannte Formeln 'einfach drauflos programmiert'.

Es geht um die numerische Auswertung quadratischer Gleichungen

$$ax^2 + bx + c = 0$$

mit den reellen Koeffizienten a, b und c.

Die wohlbekannten Lösungen lauten

$$(a) \quad x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad (b) \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Man sieht jedoch sofort, daß für sehr kleine Koeffizienten a und/oder c die Gefahr einer 'subtractive cancellation' im Zähler von x_1 (für $b > 0$) bzw. für x_2 (für $b < 0$) gegeben ist.
Es ist aber leicht möglich, die obigen Formeln äquivalent umzuformen zu

$$(c) \quad x_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}} \qquad (d) \quad x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \quad .$$

Aus den vorherigen Überlegungen ergibt sich, daß für $b > 0$ die Formeln (c) und (b) rundungsfehler-stabil sind, für $b < 0$ die Formeln (a) und (d).

Eine Zusammenfassung dieser Ausdrücke führt zum folgenden Algorithmus:

$$x_1 = \frac{q}{a} \qquad x_2 = \frac{c}{q}$$

mit

$$q \equiv -\frac{1}{2}\left[b + sgn(b)\sqrt{b^2 - 4ac}\right] \quad .$$

1.3.2 Rundungs- und Verfahrensfehler bei der numerischen Differentiation

Das Grundprinzip einer numerischen Differentiation besteht darin, daß man den gewünschten *Differentialquotienten* durch einen entsprechenden *Differenzenquotienten* ersetzt, also z.B.

$$\frac{d}{dx} f(x) \mid_{x=x_o} \approx \frac{f(x_o + h) - f(x_o)}{h} \tag{1.3}$$

schreibt.

Aus (1.3) geht hervor, daß der Differenzenquotient für die *Schrittweite* $h \to 0$ zur ersten Ableitung der Funktion $f(x)$ an der Stelle x_o konvergiert. Für eine *endliche* Schrittweite h ist natürlich mit einem *Verfahrensfehler* ϵ_V zu rechnen, wobei ϵ_V mit kleiner werdendem h abnehmen wird.
Theoretische Überlegungen zeigen, daß ϵ_V die Form

$$\epsilon_V = C_V(h) \cdot h \quad , \tag{1.4}$$

hat, wobei die Größe C in vielen Fällen nur schwach von h abhängt und in der Praxis durch eine Konstante ersetzt werden kann:

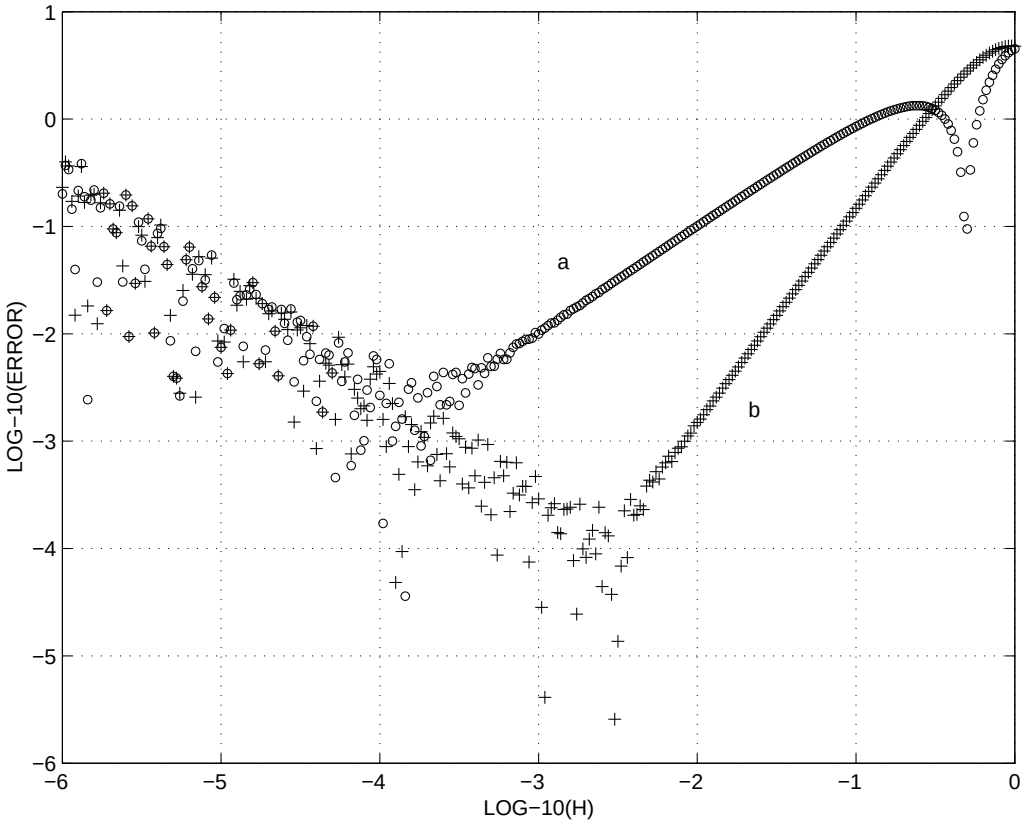


Abbildung 1.2: Fehlerdiagramm zur numerischen Differentiation. (a) Formel (1.3), (b) Formel (1.7)

$$C_V(h) \approx C_V \quad .$$

Trägt man nun ϵ_V in Abhängigkeit von h in einem *doppelt-logarithmischen* Diagramm auf, so ergibt sich als *Fehlerkurve* die lineare Beziehung

$$\lg \epsilon_V = \lg C_V + \lg h \tag{1.5}$$

mit der Steigung +1.
Diese Zusammenhänge sind in der Abb.1.2, Kurve (a) dargestellt, und zwar für das konkrete Beispiel

$$f(x) = \ln x \cdot \sin(5x)$$

$$\frac{d}{dx} f(x) \mid_{x_o=3} = \frac{\sin(15)}{3} + 5 \ln 3 \cos(15) \quad .$$

Offensichtlich ist der Zusammenhang (1.4) für den Bereich $10^{-3} < h < 10^{-1}$ sehr gut erfüllt. Für größere und für kleinere Schrittweiten zeigt jedoch das 'wahre' Fehlerverhalten starke Abweichungen vom theoretisch zu erwartenden Verfahrensfehler.
Die Ursachen dafür sind die folgenden:

- Die Abweichung im Bereich $h > 10^{-1}$ ist darin begründet, daß bei zu großen Schrittweiten die Konstanz von C_V nicht mehr erfüllt ist.

- Interessanter ist das Fehlerverhalten im Bereich $h < 10^{-3}$, wo die Abhängigkeit des Fehlers von h offensichtlich prinzipiell nicht mehr der Glg.(1.4) entspricht, sondern durch ein starkes Überhandnehmen des Rundungsfehlers geprägt ist. Die Ursache dafür ist wieder eine ßubtractive cancellationïm Zähler von (1.3).

Die Abhängigkeit des Rundungsfehlers von h entspricht keiner glatten Kurve, hat aber tendenziell die Form

$$\epsilon_R = \frac{C_R}{h} \quad . \tag{1.6}$$

Aus dem bisher Gesagten ergeben sich die folgenden wichtigen Konsequenzen:

- Bei der Verwendung einer *optimalen* Schrittweite h_{opt} erreicht man einen minimalen Gesamtfehler (im konkreten Beispiel beträgt dieser Fehler ca. $5 \cdot 10^{-3}$), der durch eine Variation von h *nicht weiter unterschritten werden kann*!

- Die Verwendung einer Schrittweite $h < h_{opt}$ führt zu keiner Verbesserung des numerischen Ergebnisses, sondern - im Gegenteil - zu einer Verschlechterung!

Eine Reduktion dieses minimalen Fehlers kann auf zwei Arten geschehen:

- Der Rundungsfehler kann durch eine Erhöhung der Abspeicherungs--Genauigkeit der reellen Zahlen (single $\rightarrow$ double; float $\rightarrow$ double etc.) verkleinert werden. Dadurch ist es möglich, mit kleineren Schrittweiten zu arbeiten.

- Der Verfahrensfehler kann durch die Verwendung eines *verbesserten Algorithmus* reduziert werden, etwa durch die Verwendung der leistungsfähigeren Formel

$$\frac{d}{dx} f(x) \mid_{x=x_o} \approx \frac{f(x_o + h) - f(x_o - h)}{2h} \tag{1.7}$$

für den Differenzenquotienten. Da bei dieser Formel ϵ_V mit abnehmenden h bedeutend schneller abnimmt als dies bei der Formel (1.3) der Fall war, nämlich

$$\epsilon_V = \bar{C}_V(h) \cdot h^2 \quad , \tag{1.8}$$

der Rundungsfehler jedoch im wesentlichen dieselbe h-Abhängigkeit hat wie vorher, kann der erreichbare Mindestfehler um etwa eine Größenordnung gesenkt werden (vgl. die Kurve (b) in der Abb.1.2).

Zusammenfassend kann gesagt werden:

Die Fehlerdiagnose in diesem Beispiel (und in vielen Bereichen der numerischen Mathematik) wird dadurch entscheidend erleichtert, daß unter bestimmten Voraussetzungen (hier: in einem bestimmten h-Bereich) *der Verfahrensfehler den Rundungsfehler stark dominiert.*

1.3.3 Fehlerdiagnose bei der numerischen Auswertung der Errorfunktion.

In vielen Anwendungen ist die numerische Auswertung der Funktion

$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$$

erforderlich, wobei $\mathrm{erf}(x)$ die *Gauss'sche Fehlerfunktion* darstellt.

Die Integraldarstellung bzw. die Darstellung in Form einer Taylorreihe lautet:

$$\mathrm{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_{z=0}^{x} dz e^{-z^2} = 1 - \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \quad .$$

Die numerische Auswertung von $\mathrm{erfc}(x)$ kann von dieser Taylorreihe ausgehen, deren Konvergenz im Prinzip für alle reellen Argumente x gesichert ist:

$$\mathrm{erfc}(x) \approx 1 - \frac{2}{\sqrt{\pi}} \sum_{n=0}^{nmax} a_n \quad mit \quad a_n = \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \quad .$$

EINSCHUB: Num. Auswertung einer Taylorreihe

- Vom Standpunkt der Rechenökonomie wäre es sehr ungünstig, jeden Term a_n der Reihe unabhängig von den anderen zu bestimmen. Viel besser ist es, jedes a_n aus dem vorherigen a_{n-1} zu berechnen. Im konkreten Fall findet man durch Berechnung des Quotienten a_n/a_{n-1} leicht den *rekursiven* Zusammenhang

  $$a_n = \left[-\frac{x^2(2n-1)}{n(2n+1)} \right] a_{n-1} \qquad \text{für} \quad n = 1, 2, \ldots$$

  mit

  $$a_0 = x \, .$$

- Durch das Abbrechen der Reihe bei $n = nmax$ wird ein Verfahrensfehler auftreten. Da diese Reihe aber aus monoton kleiner werdenden Gliedern [3] mit alternierenden Vorzeichen besteht, ist ϵ_V dem Betrage nach sicher nicht größer als der erste vernachlässigte Term a_{nmax+1}. Durch diese einfache Abschätzmöglichkeit für den Verfahrensfehler ist es möglich, die Reihe bis auf Maschinengenauigkeit auszuwerten, d.h. solange, bis der aufsummierte Wert durch weitere Terme nicht mehr verändert wird. D.h.: In den Grenzen der Maschinengenauigkeit tritt kein Verfahrensfehler auf!

[3] Zumindest ab einem bestimmten a_n!

Struktogramm — Auswertung einer Taylorreihe

x:= .....
n:=0 sumnew:=x a:=x
n:=n+1 taylor:=sumnew a:= -x*x*(2*n-1)/n/(2*n+1) * a sumnew:=sumnew+a
sumnew=taylor
print: x, taylor

Aus dem vorher Gesagten geht hervor, daß alle Abweichungen der numerischen von den exakten Ergebnissen nur durch Rundungsfehler-Akkumulationen zustande kommen. Dies kann man sofort erkennen, wenn man die Auswertung sowohl mit einfacher als auch mit doppelter Genauigkeit durchführt. Die Ergebnisse dieses Tests für eine Reihe von x sind in der Tabelle 1.2, Spalte 2, angegeben.

Aus dem Vergleich der Ergebnisse für die Auswertung der Taylorreihe ergibt sich, daß die numerische Berechung von erfc(x) für kleine Argumente x sehr gut funktioniert, für größere x jedoch total versagt.
In diesem Argumentbereich muß ein anderer Algorithmus verwendet werden.

Dafür bietet sich ein sogenannter *Kettenbruch* [11] an, der die Form

$$\mathrm{erfc}(x) = \frac{e^{-x^2}}{\sqrt{\pi}} \cdot \left(\frac{1}{x+} \frac{1/2}{x+} \frac{1}{x+} \frac{3/2}{x+} \cdots \right) \quad .$$

hat. Allerdings hat die übliche numerische Auswertung eines Kettenbruches den Nachteil, daß bereits vor Beginn der Auswertung feststehen muß, wieviele Terme des Bruches in die Rechnung miteinbezogen werden. Zeigt es sich, daß eine Erhöhung der Term-Anzahl nötig ist (um den Verfahrensfehler zu reduzieren), so muß die Auswertung ganz neu begonnen werden, und man kann nicht (wie etwa bei der Auswertung einer Taylorreihe) einfach zusätzliche Terme zum 'alten' Ergebnis hinzufügen [4].

Im konkreten Fall wurde der Kettenbruch jeweils für 31 und für 61 Terme ausgewertet, wieder sowohl mit einfacher als auch mit doppelter Genauigkeit. Aus dem Vergleich der Ergebnisse (Tabelle 1.2, dritte und vierte Spalte) können die folgenden Schlüsse gezogen werden:

- Unterschiede bei den Ergebnissen mit 31 bzw. 61 Termen → *Verfahrensfehler.*

[4] Eine solche Möglichkeit in Form einer besseren Auswertung eines Kettenbruches findet man z.B. in [9], S.135f).

- Unterschiede bei den Ergebnissen mit einfacher bzw. doppelter Genauigkeit → *Rundungsfehler*.

Tab.1.2: Auswertung der Funktion erfc(x) mittels der Taylorreihe bzw. des Kettenbruches. Alle Ergebnisse sind in halb-exponentieller Schreibweise mit 7 Mantissenstellen angegeben, wobei der Exponent E eine Berechnung in einfacher Genauigkeit bedeutet, der Exponent D eine Berechnung in doppelter Genauigkeit.

x	Taylorreihe $\epsilon_V = 0$	Kettenbruch 31 Terme	Kettenbruch 61 Terme
0.01	0.9887166E+00	0.1261318E+02	0.9042203E+01
	0.9887166D+00	0.1261318D+02	0.9042202D+01
0.1	0.8875371E+00	0.1401417E+01	0.1131721E+01
	0.8875371D+00	0.1401417D+01	0.1131721D+01
0.5	0.4795001E+00	0.4802107E+00	0.4795305E+00
	0.4795001D+00	0.4802107D+00	0.4795305D+00
1.	0.1572992E+00	0.1572995E+00	0.1572992E+00
	0.1572992D+00	0.1572995D+00	0.1572992D+00
2.	0.4677685E-02	0.4677735E-02	0.4677735E-02
	0.4677735D-02	0.4677735D-02	0.4677735D-02
3.	0.2991446E-04	0.2209050E-04	0.2209050E-04
	0.2209050D-04	0.2209050D-04	0.2209050D-04
4.	0.2364931E-02	0.1541726E-07	0.1541726E-07
	0.1544033D-07	0.1541726D-07	0.1541726D-07
5.	0.4645361E+02	0.1537460E-11	0.1537460E-11
	0.5458862D-07	0.1537460D-11	0.1537460D-11

Als Zusammenfassung der Ergebnisse kann folgendes festgestellt werden:

- Taylorreihe: Verfahrensfehler im gesamten x-Bereich Null, Rundungsfehler mit steigendem x stark ansteigend.

  Die Taylorreihen-Auswertung gibt bis ca. $x = 1$ Ergebnisse mit mind. 7 sicheren Ziffern.

- Kettenbruch: Verfahrensfehler nimmt mit steigendem x ab, der Rundungsfehler ist im gesamten untersuchten x-Bereich sehr klein.

  Die Kettenbruch-Auswertung mit 31 Termen gibt ab ca. $x = 2$ Ergebnisse mit mind. 7 sicheren Ziffern.

1.3.4 Beispiel für stabile und instabile Algorithmen

Viele numerische Verfahren beruhen auf *Rekursionsformeln* vom Typus

$$y_n = a \cdot y_{n-1} + b \cdot y_{n-2} \qquad n = 2, 3, \ldots \quad .$$

Bei der Anwendung solcher Formeln muß besonders auf die *Stabilität* des Algorithmus geachtet werden:

Ein Algorithmus heißt stabil oder instabil, je nachdem ein im n-ten Rechenschritt zugelassener Rechnungsfehler bei exakter Rechnung in den Folgeschritten abnimmt oder anwächst (aus [2], S.9).

Ein sehr instruktives Beispiel stellt in diesem Zusammenhang die numerische Auswertung von *sphärischen Besselfunktionen* mittels einer „Vorwärts-Rekursion“ dar:

$$j_o(x) = \frac{\sin x}{x} \qquad j_1(x) = \frac{\sin x}{x^2} - \frac{\cos x}{x}$$

$$j_l(x) = \frac{2l-1}{x} \cdot j_{l-1}(x) - j_{l-2}(x) \qquad l = 2, 3, \ldots, lmax$$

Bei der numerischen Auswertung dieser Formel gibt es natürlich überhaupt keine Verfahrensfehler; alle auftretenden Fehler können nur Rundungsfehler sein! Die Größe der Rundungsfehler kann wiederum durch den Vergleich von Ergebnissen abgeschätzt werden, die mit einfacher bzw. doppelter Genauigkeit berechnet wurden (s. Tabelle 1.3).

Tab.1.3: Numerische Berechnung der sphärischen Besselfunktionen der Ordnungen 0-9 mittels Vorwärts-Rekursion.

	einfache Gen.		doppelte Gen.
x = 0.3000			
L= 0	0.9850674E+00	0	0.9850674D+00
1	0.9910274E-01	1	0.9910289D-01
2	0.5960023E-02	2	0.5961525D-02
3	0.2309625E-03	3	0.2558598D-03
4	-.5708978E-03	4	0.8536426D-05
5	-.1735790E-01	5	0.2329701D-06
6	-.6358853E+00	6	0.5811086D-08
7	-.2753767E+02	7	0.1884359D-07
8	-.1376248E+04	8	0.9363682D-06
9	-.7795982E+05	9	0.5304202D-04
x = 1.0000			
L= 0	0.8414710E+00	0	0.8414710D+00
1	0.3011687E+00	1	0.3011687D+00
2	0.6203508E-01	2	0.6203505D-01
3	0.9006739E-02	3	0.9006581D-02
4	0.1012087E-02	4	0.1011016D-02
5	0.1020432E-03	5	0.9256116D-04
6	0.1103878E-03	6	0.7156936D-05
7	0.1332998E-02	7	0.4790142D-06
8	0.1988459E-01	8	0.2827691D-07
9	0.3367050E+00	9	0.1693277D-08
x = 10.0000			
L= 0	-.5440211E-01	0	-.5440211D-01
1	0.7846694E-01	1	0.7846694D-01
2	0.7794219E-01	2	0.7794219D-01
3	-.3949584E-01	3	-.3949584D-01
4	-.1055893E+00	4	-.1055893D+00
5	-.5553451E-01	5	-.5553451D-01
6	0.4450132E-01	6	0.4450132D-01
7	0.1133862E+00	7	0.1133862D+00
8	0.1255780E+00	8	0.1255780D+00
9	0.1000964E+00	9	0.1000964D+00
x = 20.0000			
L= 0	0.4564726E-01	0	0.4564726D-01
1	-.1812174E-01	1	-.1812174D-01
2	-.4836553E-01	2	-.4836552D-01
3	0.6030358E-02	3	0.6030359D-02
4	0.5047615E-01	4	0.5047615D-01
5	0.1668391E-01	5	0.1668391D-01
6	-.4130000E-01	6	-.4130000D-01
7	-.4352891E-01	7	-.4352891D-01
8	0.8653319E-02	8	0.8653319D-02
9	0.5088423E-01	9	0.5088423D-01

Die Ergebnisse von Tabelle 1.3 zeigen deutlich, daß - insbesondere für kleine Argumente x - das verwendete Verfahren stark instabil ist. Diese Instabilität führt mit steigender Ordnung der Besselfunktion zu einem völligen Versagen des Verfahrens.
Für größere Argumente von x wird die Stabilität des Verfahrens zunehmend besser!

Um auch im Bereich kleiner Argumente brauchbare numerische Ergebnisse zu erhalten, muß von der bisher verwendeten „Vorwärts-Rekursion“ auf eine „Rückwärts-Rekursion“ übergegangen werden:

$$j_{L+1}(x) = 0 \qquad j_L(x) = \delta$$

$$j_l(x) = \frac{2l+3}{x} \cdot j_{l+1}(x) - j_{l+2}(x) \qquad l = L-1, L-2, \ldots, lmax, lmax-1, \ldots, 0$$

Dabei ist δ, eine beliebige (kleine) Zahl, der Startwert der Rückwärts-Rekursion. L ist eine natürliche Zahl $>$ lmax. Es zeigt sich trotz der willkürlichen Startwerte der Rekursion, daß die Werte für kleiner werdende l sehr rasch gegen eine Zahlenfolge $\alpha \cdot j_l$ konvergieren. Die vorerst unbekannte Konstante α kann durch eine Normierung des Wertes für $l = 0$ auf die Größe $\sin x/x$ bestimmt werden.

Wegen der Willkür von δ sind die so erhaltenen Ergebnisse mit einem Verfahrensfehler behaftet, der umso kleiner ist, je größer der Startindex L gewählt wurde. Um eine Abschätzung von ϵ_V zu erhalten, wurde die Auswertung zweimal durchgeführt, und zwar mit $L = 18$ und mit $L = 27$. In beiden Fällen wurde die Rechnung sowohl mit einfacher als auch mit doppelter Genauigkeit durchgeführt.

Die Ergebnisse dieses Tests sind in der Tabelle 1.4 zusammengefaßt.
Es zeigt sich, daß die Rückwärts-Rekursion für den ganzen untersuchten x-Bereich einen sehr stabilen Algorithmus darstellt: Die Ergebnisse von einfacher und doppelter Genauigkeit unterscheiden sich erst in der siebenten Ziffer der Mantissen um maximal zwei Einheiten. Was den Verfahrensfehler betrifft, so ist mit steigendem Argument x eine steigende Tendenz dieses Fehlers zu beobachten.

Resumee aus den Tabellen 1.3 und 1.4:

- Für nicht zu große Argumente x (für $lmax = 9$ bis ca. $x = 10.$) ist wegen der größeren Stabilität unbedingt die Rückwärts-Rekursion zu verwenden.
- Erst für den Bereich großer x (für $lmax = 9$ ab ca. $x = 10.$) ist die Vorwärts-Rekursion wegen der nicht vorhandenen Verfahrensfehler vorzuziehen.

Tab.1.4: Numerische Berechnung der sphärischen Besselfunktionen der Ordnungen 0-9 mittels Rückwärts-Rekursion.

	einfache Genauigkeit		doppelte Genauigkeit	
	rueckwaerts(18)	rueckwaerts(27)	rueckwaerts(18)	rueckwaerts(27)
x = 0.3000				
L= 0	0.9850674E+00	0.9850674E+00	0.9850674D+00	0.9850674D+00
1	0.9910290E-01	0.9910290E-01	0.9910289D-01	0.9910289D-01
2	0.5961526E-02	0.5961525E-02	0.5961525D-02	0.5961525D-02
3	0.2558598E-03	0.2558598E-03	0.2558598D-03	0.2558598D-03
4	0.8536426E-05	0.8536425E-05	0.8536426D-05	0.8536426D-05
5	0.2329583E-06	0.2329583E-06	0.2329583D-06	0.2329583D-06
6	0.5378445E-08	0.5378444E-08	0.5378444D-08	0.5378444D-08
7	0.1076069E-09	0.1076069E-09	0.1076069D-09	0.1076069D-09
8	0.1899475E-11	0.1899474E-11	0.1899474D-11	0.1899474D-11
9	0.2999847E-13	0.2999847E-13	0.2999847D-13	0.2999847D-13
x = 1.0000				
L= 0	0.8414710E+00	0.8414710E+00	0.8414710D+00	0.8414710D+00
1	0.3011687E+00	0.3011687E+00	0.3011687D+00	0.3011687D+00
2	0.6203505E-01	0.6203505E-01	0.6203505D-01	0.6203505D-01
3	0.9006580E-02	0.9006579E-02	0.9006581D-02	0.9006581D-02
4	0.1011016E-02	0.1011016E-02	0.1011016D-02	0.1011016D-02
5	0.9256115E-04	0.9256114E-04	0.9256116D-04	0.9256116D-04
6	0.7156936E-05	0.7156935E-05	0.7156936D-05	0.7156936D-05
7	0.4790134E-06	0.4790133E-06	0.4790134D-06	0.4790134D-06
8	0.2826499E-07	0.2826498E-07	0.2826499D-07	0.2826499D-07
9	0.1491376E-08	0.1491376E-08	0.1491377D-08	0.1491377D-08
x = 10.0000				
L= 0	-.5440211E-01	-.5440211E-01	-.5440211D-01	-.5440211D-01
1	0.7846695E-01	0.7846695E-01	0.7846695D-01	0.7846694D-01
2	0.7794219E-01	0.7794220E-01	0.7794220D-01	0.7794219D-01
3	-.3949586E-01	-.3949586E-01	-.3949585D-01	-.3949584D-01
4	-.1055893E+00	-.1055893E+00	-.1055893D+00	-.1055893D+00
5	-.5553451E-01	-.5553451E-01	-.5553451D-01	-.5553451D-01
6	0.4450133E-01	0.4450133E-01	0.4450133D-01	0.4450132D-01
7	0.1133862E+00	0.1133862E+00	0.1133862D+00	0.1133862D+00
8	0.1255780E+00	0.1255780E+00	0.1255780D+00	0.1255780D+00
9	0.1000964E+00	0.1000964E+00	0.1000964D+00	0.1000964D+00
x = 20.0000				
L= 0	0.4564727E-01	0.4564726E-01	0.4564726D-01	0.4564726D-01
1	-.8801447E-01	-.1812270E-01	-.8801444D-01	-.1812269D-01
2	-.5884944E-01	-.4836567E-01	-.5884943D-01	-.4836567D-01
3	0.7330211E-01	0.6031280E-02	0.7330208D-01	0.6031273D-02
4	0.8450518E-01	0.5047661E-01	0.8450516D-01	0.5047661D-01
5	-.3527479E-01	0.1668320E-01	-.3527476D-01	0.1668320D-01
6	-.1039063E+00	-.4130086E-01	-.1039063D+00	-.4130085D-01
7	-.3226432E-01	-.4352875E-01	-.3226432D-01	-.4352875D-01
8	0.7970807E-01	0.8654292E-02	0.7970804D-01	0.8654284D-02
9	0.1000162E+00	0.5088490E-01	0.1000161D+00	0.5088490D-01