Grundlagen der elektronischen Datenverarbeitung II

Numerische Methoden in der Physik

M.Faber, E. Jericha, G. Kahl, H. Leeb,
Ch. Lemell, M. Mantler, H. Müller, W. Werner

Fachbereich Physik

Technische Universität Wien

Sommersemester 2004

21. Juni 2004

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Lehrziel und Struktur

Die Beschreibung von beobachteten Phänomenen und Zusammenhängen mittels mathematischer Methoden ist ein wichtiges Anliegen der Physik und stellt auch die Basis für deren Anwendung in der Technik dar. Der quantitative Auswertung der entsprechenden Theorien erfordert numerische Rechnungen, die in der ersten Hälfte des letzten Jahrhunderts durch die beschränkte Schnelligkeit des Menschen auf wenige schematische Beispiele bzw. grobe Näherungen beschränkt waren. Die Entwicklung von Computern in den letzten Jahrzehnten hat die Lösungskapazität wesentlich erhöht, sodass wir heute auch komplexere Probleme quantitativ erfassen können. Die erzielten Leistungssteigerungen der Computer sind nicht nur auf die in den letzten Jahren erfolgte Steigerung der Prozessgeschwindigkeiten zurückzuführen, sondern auch auf signifikante Verbesserungen im Speicherbereich sowie im Einsatz von massivem Parallelismus. Diese Entwicklungen auf dem Rechnersektor haben große Auswirkungen auf die Gesellschaft im Allgemeinen. Im Bereich der Physik hat der Einsatz von Computern zum Teil methodische Veränderungen bewirkt. Insbesondere die Möglichkeit der Simulation von komplexen Systemen führte zur Etablierung eines neuen Zweiges der Physik, den man als *Numerische Physik* bezeichnet.

Numerische Methoden zur Lösung der ein physikalisches Problem beschreibenden mathematischen Zusammenhänge sind die Grundvoraussetzung für den Einsatz von Rechnern in der Physik. Die Lehrveranstaltung *Grundlagen der elektronischen Datenverarbeitung II* soll die wichtigsten numerischen Techniken vermitteln. Lehrziel der Lehrveranstaltung ist das Verständnis der grundlegenden Algorithmen wie sie in der Physik zum Einsatz kommen, die Verwendung von Programmbibliotheken und die konkrete Umsetzung physikalischer Problemstellungen. Der pragmatische Zugang steht dabei im Vordergrund, während die Behandlung im Sinne der *Numerischen Mathematik* nur auf das Notwendigste beschränkt wird. In der Lehrveranstaltung können nur sehr wenige Methoden behandelt werden. Für einen umfassenderen Überblick wird auf die Serie der *Numerical Recipes* [1, 2, 3, 4] verwiesen, die nicht nur eine Beschreibung der mathematischen Grundlagen geben, sondern auch entsprechende Programme in

FORTRAN, C oder C++.

In der vorliegenden Lehrveranstaltung werden die numerischen Methoden unabhängig von der verwendeten Programmiersprache behandelt. Für die Demonstrationsbeispiele wird meist die Programmiersprache FORTRAN in der Version FORTRAN77 verwendet, die für numerische Rechnungen bestens geeignet ist. Weiters kommen die Programmiersprachen C und C++, welche bereits eingehend in der vorausgegangenen Vorlesung Grundlagen der elektronischen Datenverarbeitung I behandelt wurden, zum Einsatz. In der Lehrveranstaltung werden zwar Hinweise auf Grundelemente und Besonderheiten dieser Programmiersprachen gegeben, für eine detaillierte Behandlung wird aber auf die einschlägigen Referenzhandbücher verwiesen (Diese finden Sie z.B. im Directory /home/EDV2/Unterlagen/Manuals/).

Der Aufbau der Lehrveranstaltung sieht einen möglichst großen Eigenanteil vor, in dem die in einem kompakten Vorlesungsmodul besprochenen Algorithmen selbständig in ein Programm umgesetzt und auf konkrete physikalische Probleme angewendet werden sollen. Folgende Algorithmen sind Gegenstand der Lehrveranstaltung:

- **Einheit 1:** Umsetzen kleinerer Probleme in ein FORTRAN-Programm.
- **Einheit 2:** Berechnung spezieller Funktionen
- **Einheit 3:** Statistik und Zufallszahlen
- **Einheit 4:** Lösung linearer Gleichungen
- **Einheit 5:** Interpolation, Differentiation und Integration
- **Einheit 6:** Nullstellenbestimmung und Anpassungsprobleme
- **Einheit 7:** Gewöhnliche Differentialgleichungen
- **Einheit 8:** Fourier Transformationen
- **Einheit 9:** Differenzenverfahren für die Laplace-Gleichung
- **Einheit 10:** Lösung der radialen Schrödingergleichung

Durch die Anwendung auf konkrete physikalische Fragestellungen wird gleichzeitig auch die Eigenschaft physikalischer Systeme demonstriert, die in den Grundlagenvorlesungen bisher nur an einfachen Beispielen behandelt werden konnten.

1.2 Erstellung eines FORTRAN-Programms

FORTRAN ist eine weitgehend standardisierte Programmiersprache, welche sich für die Behandlung von numerischen Problemen bestens eignet. Aufgrund der frühen Standardisierung von FORTRAN war bereits in den Anfängen der numerischen Physik die Portabilität von FORTRAN-Programmen weitgehend gewährleistet. Eine große Zahl

von Standardprogrammen steht heute in den verschiedenen Datenzentren, wie z.B. der Kerndatenbank der *International Atomic Energy Agency* (IAEA), zur Verfügung.

Im folgenden soll am Beispiel einer Matrixmultiplikation die Struktur von FORTRAN und einige wichtige Befehle besprochen werden. Das nachstehend angeführte Programm `MatMul.f` liest die Matrix

$$\mathcal{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{1.1}$$

von der Datei `matA.d` zeilenweise ein und berechnet das Produkt mit der in einem *DATA Statement* vorgegebenen Matrix

$$\mathcal{B} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} . \tag{1.2}$$

Das Ergebnis $\mathcal{C} = \mathcal{A}\,\mathcal{B}$ sowie das Quadrat der Matrix $\mathcal{A}$ werden berechnet. Die einzelnen Programmschritte sind in den Kommentarzeilen beschrieben.

```
C     ============================================
C     I  FORTRAN Demonstrationsprogramm          I
C     I  M A T R I X M U L T I P L I K A T I O N  I
C     ============================================
C
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
C
      PARAMETER (N=3)
C
      DIMENSION A(1:N,1:N),B(1:N,1:N)
      DIMENSION C(1:N,1:N),D(1:N,1:N)
C
      DATA B /1.,2.,3.,4.,5.,6.,7.,8.,9./
C
C     Oeffnen der Datei 'matA.d' mit der logischen Nummer 1
      open(1,file='matA.d',form='formatted')
C
C     Einlesen der Matrix A von der Datei 'matA.d' ein
C     Die Matrix wird Zeile fuer Zeile eingelesen, wobei die
C     Zahlen durch Abstaende oder Beistriche getrennt sein
C     koennen
      do i=1,N
         read(1,*) (A(i,j),j=1,N)
      end do
C
C     Schliessen der Datei 1
```

```
      close(1)
C
C     Schreiben eines Kopfes fuer die Ausgabe
      write(6,6000)
C     zeilenweise Ausgabe der Matrizen A und B
      do j=1,N
         write(6,6010) (A(j,i),i=1,N),(B(j,i),I=1,N)
      end do
C
C     format Vereinbarungen fuer die Ausgabe
 6000 format(/4x,59(1h=)/4x,'Ergebnisse des Demonstrationsprograms:',
     1   ' MATRIXMULTIPLIKATION'/4x,59(1h=)//25x,'Eingabe Matrizen'/
     2    10x,8hMatrix A,30X,8hMatrix B)
 6010 format(3(2x,f6.2),16x,3(f6.2,2x))
C
C     Durchfuehren der Matrixmultiplikation
      call mult(N,A,B,C)
C
C     Ausgabe der Matrix C als Ergebnis der Matrixmultiplikation
C              C= A * B
C     Ausgabe eines Kopfes
      write(6,6020)
C     zeilenweise Ausgabe der Matrix C
      do j=1,N
         write(6,6010) (C(j,i),i=1,N)
      end do
 6020 format(//'Ergebnis der Matrixmultiplikation C = A * B'/
     1    /10x,8hMatrix C)
C
C     Berechnen des Quadrats der Matrix A
      call mult(N,A,A,D)
C
C     Ausgabe der Matrix D als Ergebnis der Matrixmultiplikation
C              D= A * A
C     Ausgabe eines Kopfes
      write(6,6030)
C     zeilenweise Ausgabe der Matrix D
      do j=1,N
         write(6,6010) (D(j,i),i=1,N)
      end do
 6030 format(//'Ergebnis der Matrixmultiplikation D = A * A'/
     1    /10x,'Matrix D = A**2')

C
C     stop der Programmausfuehrung
      stop
```

```
C
C     Ende der Programmbefehle
      end
```

Das Unterprogramm (SUBROUTINE oder FUNCTION) ist in diesem Demonstrationsbeispiel in das Programm `MatMul.f` inkludiert. Es kann aber auch als eigenständige Datei erstellt und dann in das auszuführende Programm eingebunden werden.

```
      subroutine mult(N,A,B,C)
C     ========================
C
C     subroutine  m u l t  fuehrt die Multiplikation von
C     zwei quadratischen Matrizen der Dimension N durch und
C     speichert das Resultat auf das Feld C, d.h. C=A*B
C
C        N   Dimension der Matrizen A,B,C
C        A,B zweidimensionale arrays mit den Matrizen A und B
C        C   zweidimensionales array mit dem Resultat C
C
C     -------------------------------------------------------
C
      IMPLICIT REAL*8(A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
C
      DIMENSION A(1:N,1:N),B(1:N,1:N),C(1:N,1:N)
C
C     Durchfuehren der Matrixmultiplikation C=A*B
C     Man sollte versuchen den ersten Index stets schneller
C     variieren zu lassen als den zweiten Index
      do i=1,N
         do j=1,N
            C(j,i)=0.
            do k=1,N
               C(j,i)=C(j,i)+A(j,k)*B(k,i)
            end do
         end do
      end do
C
C     gehe zum rufenden Programm zurueck
      return
C
C     Ende der Subroutine
      end
```

In FORTRAN werden zwei- und höherdimensionale Felder (*Arrays*) stets in der eindimensionalen Form gespeichert, wobei der erste Index am schnellsten variiert. Bei

Operationen mit großen mehrdimensionalen Feldern sollte man diese Reihenfolge soweit als möglich beachten, um hohe Geschwindigkeit bei der Durchführung zu erreichen.

1.3 Aufgaben

Als erste Übung sollen FORTRAN-Programme für einige kleinere Problemstellungen erstellt werden. Man schreibe FORTRAN-Programme zur Lösung der folgenden Aufgaben:

- **Aufgabe 1.1: Untersuchung von Dreiecken**
  Schreiben Sie ein FORTRAN-Programm, das entscheidet, ob die Werte von drei `REAL`-Variablen (Eingabedaten) als Kanten ein Dreieck bilden können. Dabei übernimmt das Hauptprogramm die Ein- und die Ausgabe, während eine `SUBROUTINE` entscheidet, welcher der folgenden Fälle vorliegt:

  – kein Dreieck
  – gleichseitiges Dreieck
  – rechtwinkeliges Dreieck
  – gleichschenkeliges Dreieck
  – allgemeines Dreieck

  Die Ausgabe soll sowohl die drei Zahlenwerte als auch einen Hinweis enthalten, welcher der obigen Fälle realisiert wurde. Das Programm soll mit selbstgewählten Beispielen getestet werden, wobei jeder der fünf Fälle mindestens einmal vorkommen soll. Beachten Sie, dass auf Grund der internen Darstellung der `REAL`-Variablen Abfragen nur bis zu einer gewisse Genauigkeitsgrenze durchgeführt werden können (z.B. 10^{-6}).

- **Aufgabe 1.2: Berechnung des Mittelwertes und der Varianz**
  Schreiben Sie ein FORTRAN-Programm, das den Mittelwert $\bar{a}$ und die Varianz σ von n Zahlen a_i, $i = 1, \cdots, n$, berechnet. Dabei werden die a_i als `array` in ein Unterprogramm übergeben, das die tatsächliche Berechnung von $\bar{a}$ und σ durchführt. Eventuell kann diese Übergabe über ein Feld mit anpassbarer Dimension erfolgen.

  $\bar{a}$ und σ sind folgendermaßen definiert:

$$\bar{a} = \frac{1}{n} \sum_{i=1}^{n} a_i$$

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (a_i - \bar{a})^2.$$

Im Hauptprogramm werden n und die a_i eingelesen; nach dem Aufruf des Unterprogramms sollen die a_i und die Ergebnisse für $\bar{a}$ und σ ausgegeben werden.

Testen Sie das Programm mit selbstgewählten Datensätzen (n mindestens 7).

- **Aufgabe 1.3: Quadratwurzel mittels iterativem Algorithmus**
  Die Quadratwurzel einer reelen Zahl kann mittels folgendem iterativen Algorithmus berechnet werden:

$$x_0 = 1 \quad x_{i+1} = \frac{1}{2}(x_i + a/x_i)$$

und

$$\lim_{i \to \infty} x_i = \sqrt{a} \quad .$$

Der Algorithmus startet von einem Anfangswert x_0 und wird beendet, wenn sich zwei aufeinanderfolgende x_i-Werte um weniger als den Betrag δ unterscheiden.

Man schreibe eine `SUBROUTINE`, welche den oben gegebenen Algorithmus zur Berechnung der Quadratwurzel verwendet. Die formalen Parameter der `SUBROUTINE` sind a, δ, das Endergebnis und die Zahl der erforderlichen Iterationen. Im rufenden `PROGRAM` soll die Genauigkeit des Algorithmus an fünf selbstgewählten Beispielen überprüft werden. Ein Vergleich mit dem "exakten" Resultat, und zwar mit dem Ergebnis der `SQRT`-Funktion, soll durchgeführt werden.

Kapitel 2

Spezielle Funktionen

2.1 Einleitung

In diesem Kapitel über *Spezielle Funktionen* wollen wir einige für die mathematische Behandlung physikalischer Probleme nützliche Funktionen näher betrachten. Speziell werden wir auf die Generierung der Gamma-Funktion, der Legendre-Polynome und der sphärischen Bessel- und Neumannfunktionen eingehen. Weitere *Spezielle Funktionen* findet man in den kommerziell verfügbaren Programmbibliotheken wie z.B. die NAG- oder die IMSL-Library. Die Programme dieser Bibliotheken sind meist sehr umfangreich, da sie als *Black Boxes* verwendet werden und eine sorgfältige Kontrolle der Genauigkeit, auch für Sonderfälle, garantieren müssen. Bei den nachstehend angeführten Funktionen skizzieren wir nur die mathematischen Grundlagen, welche für die numerische Darstellung verwendet werden können.

2.2 Gamma-Funktion

In vielen Problemen der Physik und der Statistik wird die Gamma-Funktion $\Gamma(z)$ bzw. die Faktorielle benötigt. So tritt sie z.B. in quantenmechanischen Problemen der Atom- und Kernphysik in Normierungskonstanten auf. Die Gamma-Funktion ist über das Integral

$$\Gamma(z) = \int_0^\infty dt\ t^{z-1} e^{-t} \tag{2.1}$$

definiert. Ist das Argument eine natürliche Zahl, $z = n$ ($n \in \mathcal{N}$), so entspricht die Gamma-Funktion der Faktoriellen,

$$n! = \Gamma(n+1)\,. \tag{2.2}$$

Für allgemeines z erfüllt die Gamma-Funktion die Rekursionsbeziehung

$$\Gamma(z+1) = z\ \Gamma(z) \tag{2.3}$$

Kennt man die Gamma-Funktion für z-Werte in der gesamten Halbebene $\mathrm{Re}z > 1$, so lassen sich über die Reflexionsformel,

$$\Gamma(1-z) = \frac{\pi}{\Gamma(z)\sin(\pi z)} = \frac{\pi z}{\Gamma(1+z)\sin(\pi z)}\,, \tag{2.4}$$

auch die Werte der Gamma-Funktion für $\mathrm{Re}z < 1$ bestimmen. Die Gamma-Funktion hat Pole bei $z = 0$ und bei allen negativen ganzen Zahlen.

Für die numerische Berechnung von $\Gamma(z)$ kann man von einer der speziellen Darstellungen für bestimmte Intervalle ausgehen (siehe z.B. [5], Kapitel 6) und durch Anwendung der Rekursionsbeziehung (2.3) und der Reflexionsformel (2.4) die gesamte komplexe z-Ebene abdecken. Eine der Näherungen für reelle Argumentwerte x aus dem Intervall $[1, 2[$ ist von der Form

$$\Gamma(x) \approx \sum_{k=0}^{8} a_k (x-1)^k\,, \tag{2.5}$$

mit den Konstanten

a_0	=	1.000000	a_3	=	−0.897057	a_6	=	0.482199
a_1	=	−0.577192	a_4	=	0.918207	a_7	=	−0.193528
a_2	=	0.988206	a_5	=	−0.756704	a_8	=	0.035868

Im allgemeinen ist es günstiger den Logarithmus der Funktion, $\ln\Gamma(z)$, numerisch zu implementieren. Dadurch lässt sich ein *floating point overflow* des Rechners vermeiden. Die Rekursionsbeziehung (2.3) reduziert sich dann zu einer Summe

$$\ln\Gamma(z+N) = \sum_{n=0}^{N-1} \ln(z+n) + \ln\Gamma(z)\,. \tag{2.6}$$

Wählt man nun die Zahl N hinreichend groß, so kann man die folgende asymptotische Form zur Berechnung von $\ln\Gamma(z+N)$ verwenden (siehe [5])

$$\ln\Gamma(\bar{z}) \approx (\bar{z}-\tfrac{1}{2})\ln\bar{z} - \bar{z} + \tfrac{1}{2}\ln(2\pi) + \frac{1}{12\bar{z}} - \frac{1}{360\bar{z}^2} + \frac{1}{1260\bar{z}^5} - \frac{1}{1680\bar{z}^7} + \cdots \tag{2.7}$$

with $\bar{z} = z + N \to \infty$ in $|\arg\bar{z}| < \pi$. Für $\mathrm{Re}z < 1$ muss zusätzlich noch die Reflexionsformel (2.4) eingesetzt werden.

Zum Abschluss sei noch die spezielle Näherung von Lanczos genannt [5],

$$\begin{aligned}\Gamma(1+z) &= (z+\gamma+\tfrac{1}{2})^{z+0.5} e^{-(z+\gamma+0.5)} \\ &\times \sqrt{2\pi}\left[1 + \frac{c_1}{z+1} + \frac{c_2}{z+2} + \frac{c_5}{z+5} + \epsilon\right] \quad \text{für } \mathrm{Re}z > 0\end{aligned} \tag{2.8}$$

mit $\gamma = 5$ und den Konstanten

c_1	=	76.18009173	c_2	=	−86.50532033	c_3	=	24.01409822
c_4	=	−1.231739516	c_5	=	0.00120858003	c_6	=	0.000053682

Diese Formel kann in der gesamten komplexen Halbebene $\mathrm{Re}z > 0$ angewendet werden, wobei der Fehler $|\epsilon| < 2 \times 10^{-10}$ ist.

2.3 Legendre Polynome und Kugelflächenfunktionen

Kugelflächenfunktionen, $Y_{\ell m}(\theta, \varphi)$ werden in einer Vielzahl physikalischer Probleme verwendet. Ihre Verwendung ist besonders dann vorteilhaft, wenn das betrachtete Problem eine sphärische Symmetrie aufweist, wie dies z.B. in vielen Fragestellungen der Atom- und Kernphysik der Fall ist.

Die Kugelflächenfunktionen sind auf der Einheitskugel definiert und stellen einen orthonormierten Satz von Basisfunktionen dar,

$$\int_0^{2\pi} d\varphi \int_{-1}^{+1} d(\cos\theta) Y^*_{\ell',m'}(\theta,\varphi) Y_{\ell m}(\theta,\varphi) = \delta_{\ell'\ell}\delta_{mm'} \,, \tag{2.9}$$

wobei (*) komplexe Konjungation bedeutet.

Die Kugelflächenfunktionen haben die Form

$$Y_{\ell m}(\theta,\varphi) = \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}\; P_\ell^m(\cos\theta) e^{im\varphi} \tag{2.10}$$

und es gilt die Beziehung

$$Y_{\ell,-m}(\theta,\varphi) = (-1)^m Y^*_{\ell m}(\theta,\varphi) \,. \tag{2.11}$$

Die Abhängigkeit der Kugelflächenfunktionen vom Winkel θ ist durch die assoziierten Legendre Polynome, $P_\ell^m(\cos\theta)$, gegeben. Diese hängen über die Beziehung

$$P_\ell^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^m}{dx^m} P_\ell(x) \tag{2.12}$$

mit den gewöhnlichen Legendre Polynomen, $P_\ell(x)$, zusammen. An dieser Stelle soll bemerkt werden, dass die Position der magnetischen Quantenzahl m als oberer oder unterer Index beachtet werden muss. Es gilt

$$P_{\ell m}(x) = (-1)^m P_\ell^m(x) \,. \tag{2.13}$$

Die assoziierten Legendre Polynome niedriger Ordnung und die entsprechenden Kugelflächenfunktionen haben eine einfache Form und sind in der Tabelle 2.3 angegeben.

Bei der Berechnung der assoziierten Legendre Polynome sollte man nicht versuchen, die Polynomdarstellung numerisch zu implementieren, da dies zu Auslöschungsfehler bei der Addition benachbarter Terme mit entgegengesetztem Vorzeichen führt. Außerdem werden bei höheren Legendre Polynomen die Einzelterme wesentlich größer als die Summe, sodass zusätzlich Genauigkeit verloren wird. Auf diese Art können bei doppelt genauer Rechnung (REAL*8) nur Polynome bis etwa $\ell = 18$ berechnet werden.

$P_{00}(x) =$	1	$Y_{00} =$	$\sqrt{\frac{1}{4\pi}}$
$P_{11}(x) =$	$(1-x^2)^{1/2}$	$Y_{11} =$	$-\sqrt{\frac{3}{8\pi}}\sin\theta e^{i\varphi}$
$P_{10}(x) =$	x	$Y_{10} =$	$\sqrt{\frac{3}{4\pi}}\cos\theta$
$P_{22}(x) =$	$3(1-x^2)$	$Y_{22} =$	$\frac{1}{4}\sqrt{\frac{15}{2\pi}}\sin^2\theta e^{2i\varphi}$
$P_{21}(x) =$	$3x(1-x^2)^{1/2}$	$Y_{21} =$	$-\sqrt{\frac{15}{8\pi}}\sin\theta e^{i\varphi}$
$P_{20}(x) =$	$\frac{1}{2}(3x^2-1)$	$Y_{20} =$	$\sqrt{\frac{5}{4\pi}}(\frac{3}{2}\cos^2\theta-\frac{1}{2})$

Tabelle 2.1: Assoziierte Legendre Polynome und Kugelflächenfunktionen niedrigster Ordnung

Ein in Hinblick auf die numerische Genauigkeit robusterer Algorithmus kann auf der Basis der Rekursionsbeziehung

$$(\ell-m)P_{\ell m}(x) = (2\ell-1)xP_{\ell-1,m}(x) - (\ell+m-1)P_{\ell-2,m}(x) \tag{2.14}$$

aufgebaut werden. Startpunkt der Rekursion sind die Polynome $P_{\ell\ell}(x)$, die durch

$$P_{\ell\ell}(x) = (2\ell-1)!!(1-x^2)^{\ell/2} \tag{2.15}$$

gegeben sind. Setzt man außerdem $P_{\ell,m=\ell+1} = 0$, so lässt sich die Rekursion bis zu höchsten ℓ-Werten stabil durchführen. Man geht dabei in folgender Reihenfolge vor:

m	Berechnung nach (2.15)	Berechnung über Rekursionsformel (2.14)
0	P_{00}	$P_{10}, P_{20}, P_{30}, P_{40}, \ldots$
1	P_{11}	$P_{21}, P_{31}, P_{41}, \ldots$
2	P_{22}	$P_{32}, P_{42}, \ldots$
3	P_{33}	$P_{43}, \ldots$
$\vdots$	$\vdots$	$\vdots$

Abschließend soll noch darauf aufmerksam gemacht werden, dass nicht jede Rekursionsformel zu einem stabilen Algorithmus führt. Wir werden diesen Punkt noch genauer bei den sphärischen Bessel Funktionen behandeln. Im Fall der Legendre Polynome gibt

es eine Reihe anderer Rekursionsbeziehungen, die meist jedoch nicht zu einem stabilen Algorithmus führen und daher für numerische Berechnung der assoziierten Legendre Polynome nicht geeignet sind.

2.4 Sphärische Bessel Funktionen

Ein wichtiges Funktionensystem bei der Lösung der Schrödingergleichung stellen die sphärischen Bessel- und Neumannfunktionen dar. Diese sind Lösungen der gewöhnlichen Differentialgleichung

$$\left\{ \frac{d^2}{d\rho^2} + \frac{2}{\rho}\frac{d}{d\rho} + 1 - \frac{\ell(\ell+1)}{\rho^2} \right\} R_\ell(\rho) = 0\,, \tag{2.16}$$

und entspricht der radialen Schrödingergleichung für ein freies Teilchen. Die Differentialgleichung (2.16) hat zwei linear unabhängige Lösungen. Diese unterscheiden sich durch ihr Verhalten bei $\rho = 0$,

- reguläre Lösung bei $\rho = 0$: spärische Besselfunktion $j_\ell(\rho)$
- irreguläre Lösung bei $\rho = 0$: sphärische Neumannfunktion $n_\ell(\rho)$

Die mathematischen Eigenschaften der sphärischen Bessel- und Neumannfunktionen sind gut untersucht. Wichtige mathematische Zusammenhänge sind z.B. im Tabellenwerk von Abramowitz und Stegun [5] zusammengefasst. Die Funktionen zeigen unter anderem das folgende Verhalten am Ursprung

$$j_\ell(\rho) \overset{\rho\to 0}{\longrightarrow} \frac{\rho^\ell}{(2\ell+1)!!}\left(1 - \frac{\rho^2}{2(2\ell+3)} + \mathcal{O}(\rho^4)\right)\,, \tag{2.17}$$

$$n_\ell(\rho) \overset{\rho\to 0}{\longrightarrow} \frac{(2\ell-1)!!}{\rho^{\ell+1}}\left(1 + \frac{\rho^2}{2(2\ell-1)} + \mathcal{O}(\rho^4)\right)\,. \tag{2.18}$$

Das asymptotische Verhalten ist durch

$$j_\ell(\rho) \overset{\rho\to\infty}{\longrightarrow} \frac{\sin(\rho - \ell\frac{\pi}{2})}{\rho}\,, \tag{2.19}$$

$$n_\ell(\rho) \overset{\rho\to\infty}{\longrightarrow} -\frac{\cos(\rho - \ell\frac{\pi}{2})}{\rho} \tag{2.20}$$

gegeben. Für die Bessel- und Neumannfunktionen können geschlossene Ausdrücke angegeben werden,

$$\ell = 0 \quad j_0(\rho) = +\tfrac{\sin\rho}{\rho} \quad , \qquad n_0(\rho) = -\tfrac{\cos\rho}{\rho} \quad ,$$

$$\ell = 1 \quad j_1(\rho) = -\tfrac{\cos\rho}{\rho} + \tfrac{\sin(\rho)}{\rho^2} \quad , \qquad n_1(\rho) = -\tfrac{\sin\rho}{\rho} - \tfrac{\cos(\rho)}{\rho^2} \quad ,$$

$$\ell = 2 \quad j_2(\rho) = -3\tfrac{\cos\rho}{\rho^2} + \left(\tfrac{3}{\rho^3} - \tfrac{1}{\rho}\right)\sin\rho \quad , \quad n_2(\rho) = -3\tfrac{\sin\rho}{\rho^2} - \left(\tfrac{3}{\rho^3} - \tfrac{1}{\rho}\right)\cos\rho \quad .$$

Die numerische Berechnung der spärischen Bessel- und Neumannfunktionen erfolgt am Besten über die Rekursionsformeln

$$g_{\ell+1}(\rho) + g_{\ell-1}(\rho) = \frac{2\ell+1}{\rho} g_\ell(\rho)\,, \tag{2.21}$$

wobei $g_\ell(\rho)$ für eine sphärische Bessel- oder Neumannfunktion steht. Analog zu der Berechnung der Legendrefunktionen führt man auch für die sphärischen Neumannfunktionen eine Rekursion durch, wobei man von $n_0(\rho)$ und $n_1(\rho)$ ausgeht. Bei den sphärischen Neumannfunktionen ist die Vorwärtsrekursion stabil und führt für alle Argumentwerte auf zuverlässige Ergebnisse.

Für die Bestimmung von $j_\ell(\rho)$ ist die Vorwärtsrekursion leider nicht ausreichend stabil für $\ell > \rho$. Die Ursache für die Probleme bei der Vorwärtsrekursion lässt sich aus der folgenden Umformung der Rekursionsvorschrift erkennen,

$$j_{\ell+1}(\rho) - 2j_\ell(\rho) + j_{\ell-1}(\rho) = \frac{2\ell+1-2\rho}{\rho} j_\ell(\rho)\,. \tag{2.22}$$

Wie wir in Kapitel 4 sehen werden, kann die Gleichung (2.22) als Differenzendarstellung der Differentialgleichung

$$\frac{d^2}{d\lambda^2} j_\lambda(\rho) = \frac{2\lambda+1-2\rho}{\rho} j_\lambda(\rho) \tag{2.23}$$

aufgefasst werden. Eine solche Differentialgleichung hat zwei linear unabhängige Lösungen. Ist $(2\lambda + 1 - 2\rho) < 0$, so erhält man 2 in λ oszillierende Lösungen. Bei $(2\lambda + 1 - 2\rho) > 0$ erhält man eine exponentiell in λ ansteigende und eine abfallende Lösung. Für die Verlässlichkeit des Rekursionsverfahrens ist es nun maßgebend, dass die divergierende Lösung unterdrückt wird. Dies ist bei der Vorwärtsrekursion der sphärischen Neumannfunktionen der Fall, nicht aber bei der Rekursion der sphärischen Besselfunktionen, d.h. bei ansteigenden $\lambda > \rho$ wird der divergierende Term ab einem gewissen Punkt dominant werden.

Diese prinzipielle Eigenschaft der Rekursion kann man zur Berechnung der sphärischen Besselfunktion ausnützen, indem man eine sogenannte Rückwertsrekursion durchführt. Man startet die Rekursion bei einem sehr hohen ℓ-Wert, $\ell = L > \rho$. Für die numerische Rechnung kann man aufrund des oben Gesagten annehmen, dass die divergierende Lösung sicherlich den numerischen Ansatz dominiert. Man kann daher beliebige Werte für $j_L(\rho)$ und $j_{L+1}(\rho)$ ansetzen. Führt man nun die Rekursion rückwärts durch, d.h. man berechnet sukzessive $j_{L-1}, j_{L-2}, \ldots j_1, j_0$ über die Rekursionsformel (2.21), so verschwindet der divergierende Anteil sehr rasch und es verbleibt nur die für die sphärischen Besselfunktionen relevante Lösung. Die erhaltenen Werte sind allerdings noch Vielfaches der entsprechenden j_ℓ-Werte und müssen durch Vergleich mit $j_0(\rho)$ normiert werden. Mehr Details über die Rückwärtsrekursion wurden von Gillman und Fiebig [6] angegeben.

2.5 Aufgaben

Im folgenden sollen FORTRAN Unterprogramme zur Berechnung der besprochenen Funktionen erstellt und graphisch mittels `gnuplot` dargestellt werden.

- **Aufgabe 2.1: Berechnung der Gamma-Funktion**
  Man schreibe ein FUNCTION Unterprogramm zur Berechnung von $\Gamma(x)$ für reele Argumentwerte x, wobei die Näherungsformel (2.5) angewendet wird. Man verwende einen rekursiven Algorithmus um die Gamma Funktion auch für reele x-Werte außerhalb des Intervalls $[1, 2[$ zu bestimmen. Die Konstanten a_k sollen als Konstantenarray gespeichert werden.

  Mit diesem Unterprogramm soll die Gamma Funktion im Intervall $x \in [0.4, 4]$ berechnet und graphisch mit dem Programm *gnuplot* dargestellt werden.

- **Aufgabe 2.2: Berechnung des Logharitmus der Gamma-Funktion**
  Man erstelle ein Unterprogramm `COMPLEX*16 FUNCTION LNGAM(z)` zur Berechnung des Logarithmus der Gamma-Funktion für komplexe z über die asymptotische Form (2.7).

  Man teste das Unterprogramm durch Verifizierung von $\Gamma(n+1) = n!$ und durch den Vergleich mit den Ergebnissen von Aufgabe 2.2. Die Funktion soll mittels `gnuplot` graphisch dargestellt. Insbesondere sollen Sie die Pole von $\Gamma(z)$ sichtbar machen.

- **Aufgabe 2.3: Berechnung und Test der modifizierten Legendrepolynome**
  Man erstelle ein Unterprogramm `SUBROUTINE LGNDR(X,LL,MM,PLM,NDIM1)` zur Berechnung aller assoziierten Legendrepolynome $P_{\ell m}(x)$ mit $\ell \leq LL$ und $m = 0, 1, \ldots, MM$ für $x = \cos\theta$. Die positive ganze Zahl `NDIM1` gibt die erste Dimension des Feldes `PLN(0:NDIM1,0:MM)` an.

  Man berechne alle assoziierten Legendrepolynome bis $\ell = LL = 3$ und stelle die Funktionen im Bereich $-1 \leq x \leq +1$ graphisch mittels `gnuplot` dar.

- **Aufgabe 2.4: Berechnung der spärischen Neumannfunktionen**
  Man erstelle ein Unterprogramm `SUBROUTINE SPHNEU(X,LL,FNEU,IFLAG)` zur Berechnung der sphärischen Neumannfunktionen $n_\ell(x)$ unter Verwendung der Vorwärtsrekursion von (2.21). Nach Durchlaufen des Unterprogramms `SPHNEU` sollen im Feld `FNE` die Werte $n_\ell(x)$ für alle $\ell = 0, 1, ..., LL$ gespeichert sein. `IFLAG` ist dabei ein Fehlerparameter, der bei Auftreten einer Singularität nach dem Aufruf von `SPHNEU` auf 1 gesetzt, sonst aber 0 ist.

  Man berechne die sphärischen Neumannfunktionen $n_\ell(x)$ im Intervall $0 < x \leq 50$ und stelle sie graphisch mittels `gnuplot` dar.

- **Aufgabe 2.5: Berechnung der spärischen Besselfunktionen**
  Man erstelle ein Unterprogramm `SUBROUTINE SPHBES(X,LL,FBES)` zur Berechnung der sphärischen Besselfunktionen $j_\ell(x)$ unter Verwendung der Rückwärtsrekursion von (2.21). Nach Durchlaufen des Unterprogramms `SPHBES` sollen im Feld `BES` die Werte $j_\ell(x)$ für alle $\ell = 0, 1, ..., LL$ gespeichert sein.

  Man berechne die sphärischen Besselfunktionen $j_\ell(x)$ im Intervall $0 \leq x \leq 20$ und stelle sie graphisch mittels `gnuplot` dar.

Kapitel 3

Statistik und Zufallszahlen

3.1 Einführung: Definition des Begriffes 'Wahrscheinlichkeit'

Die Wahrscheinlichkeitsrechnung erlaubt es, den *Erwartungswert* eines Ereignisses oder Ergebnisses eines Experiments vorherzusagen. Sei zum Beispiel A ein mögliches Ergebnis eines Experiments. Wenn N identische Experimente zur Bestimmung der interessierenden Größe durchgeführt werden, *erwartet* man, dass bei $NP(A)$ dieser Experimente A gemessen wird, wobei $P(A)$ die Wahrscheinlichkeit für das Eintreten von A als Ergebnis angibt. Wenn die Anzahl solcher identischer Experimente groß gewählt wird ($N \to \infty$), *erwartet* man also, dass ein Bruchteil $P(A)$ all dieser Experimente das Ergebnis A liefert. Wenn insgesamt n mögliche Ergebnisse auftreten können, die alle *gleich wahrscheinlich* sind, und m dieser Ergebnisse zum Ereignis A führen, dann ist $P(A) = m/n$. Ein einfaches Beispiel: die Wahrscheinlichkeit, mit einem (ungezinkten) Würfel eine Zahl kleiner oder gleich drei zu werfen, ist $P(\leq 3) = 3/6 = 1/2$.

3.2 Permutationen und Kombinationen

Die Definition der Wahrscheinlichkeit im vorigen Abschnitt zeigt, dass es zur Berechnung einer Wahrscheinlichkeit notwendig ist, Ereignisse zu zählen. In der Praxis wird man dabei oft mit sehr komplexen Zählvorgängen konfrontiert. Um solche Berechnungen zu erleichtern, stehen einige formale Regeln zur Verfügung. Diese beruhen allesamt auf zwei wichtigen Prinzipien:

- Additionsprinzip:
  Wenn sich zwei Ereignisse A und B gegenseitig ausschließen (in der Terminologie der Mengenlehre nennt man dies *unvereinbare Ereignisse*), und es gibt n Möglichkeiten, A zu realisieren, während m Möglichkeiten für B zur Verfügung stehen, dann kann das Ereignis bei dem A *oder* B auftritt, auf $m+n$ verschiedene Weisen zustande kommen.

- Multiplikationsprinzip:
  Wenn ein Ereignis auf n Weisen erzielt werden kann, und, nachdem es auf eine dieser Weisen realisiert wurde, ein zweites Ereignis auftritt, für das m Möglichkeiten zur Verfügung stehen, so kann ein Ereignis, in dem sowohl A als auch B auftreten, auf $m \times n$ Weisen zustandekommen.

Diese Prinzipien erlauben es, wichtige Zusammenhänge zwischen Wahrscheinlichkeiten für verschiedene Ereignisse herzustellen, wie wir im weiteren sehen werden.

Wenn die Anzahl der zu zählenden Objekte groß ist, ist es oft notwendig, die Anzahl der Permutationen oder Kombinationen solcher Objekte zu bestimmen. Eine Permutation (Vertauschung) N unterscheidbarer Objekte ist eine ihrer möglichen Anordnungen in einer bestimmten Reihenfolge. Wenn man also versucht, N Objekte in N dafür zur Verfügung stehende Plätze einzureihen, hat man N Möglichkeiten, den ersten Platz zu besetzen. Für den zweiten Platz bleiben dann noch $N-1$ Möglichkeiten übrig, usw. Die Gesamtanzahl der Permutationen ist also:

$$N \times (N-1) \times (N-2) \times ... \times 1 \equiv N! \tag{3.1}$$

Die Anzahl der Permutationen von $R \leq N$ unterscheidbaren Objekten aus der Menge N kann auf ähnliche Art gewonnen werden. Es stehen nun R Plätze zur Reihung zur Verfügung. Für den ersten Platz hat man wieder N Auswahlmöglichkeiten, für den zweiten $(N-1)$ und für den R-ten hat man $N-R+1$ Möglichkeiten. Insgesamt gibt es also

$$N \times (N-1) \times (N-2) \times ... \times (N-R+1) = \frac{N!}{(N-R)!} \tag{3.2}$$

Permutationen. Da die Anzahl der Permutationen von R Objekten $R!$ ist, erhält man für die Anzahl der *Kombinationen* von R Objekten aus der Menge N (eine Kombination ist eine Auswahl von R Objekten aus einer Menge N ohne Berücksichtigung der Reihenfolge):

$$\frac{N!}{R!(N-R)!} \tag{3.3}$$

.

Wenn sich die Menge N aus n_1 ununterscheidbaren Objekten eines Typs 1, n_2 ununterscheidbaren Objekten eines Typs 2, usw. zusammensetzt, läßt sich leicht beweisen, daß die Anzahl der Permutationen durch

$$\frac{N!}{n_1!n_2!n_3!...n_k!} \tag{3.4}$$

gegeben ist, wo k die Anzahl der verschiedenen unterscheidbaren Typen darstellt.

3.3 Zusammenhänge zwischen Wahrscheinlichkeiten

Um Zusammenhänge zwischen verschiedenen Wahrscheinlichkeiten zu finden, bedient man sich einiger Begriffe der Mengenlehre. Die *Vereinigung* zweier Mengen A und B

wird als $A \cup B$ geschrieben und enthält alle Elemente von A und von B. Insbesondere schließt sie alle Elemente ein, die sowohl zu A als auch zu B gehören. Letztere Teilmenge der zu beiden Mengen A und B gehörenden Elementen wird als *Durchschnitt* bezeichnet und mit $A \cap B$ angeschrieben. Zwei unvereinbare Mengen besitzen kein gemeinsames Element ($A \cap B = \oslash$) und schließen sich daher gegenseitig aus.

Anhand dieser Definitionen ist es einfach, die Wahrscheinlichkeit anzugeben, mit der bei einem Experiment entweder das Ereignis A oder B auftritt. Wir wollen diese mit $P(A \cup B)$ bezeichnen. Wenn die Einzelwahrscheinlichkeiten wieder mit $P(A)$ und $P(B)$ angegeben werden, lautet das Ergebnis:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{3.5}$$

Der letzte Term ist notwendig, weil $P(A)+P(B)$ den Bereich $A \cap B$ zweimal einschließt. Für zwei unvereinbare Ereignisse ist $P(A \cap B) = 0$ und man findet

$$P(A \cup B) = P(A) + P(B) \tag{3.6}$$

als direkte Konsequenz des Additionsprinzips.

Ein weiteres nützliches Konzept ist das der unabhängigen Ereignisse. Zwei Ereignisse sind dann und nur dann voneinander unabhängig, wenn gilt:

$$P(A \cap B) = P(A)P(B) \tag{3.7}$$

Hier gibt $P(A \cap B)$ die Wahrscheinlichkeit an, daß sowohl das Ereignis A als auch B auftreten. Wie man sieht, stellt (3.7) eine Form des Multiplikationsprinzips für Wahrscheinlichkeiten dar. Da für unvereinbare Ereignisse $P(A \cap B) = 0$ gilt, erkennt man, daß unvereinbare Ereignisse und unabhängige Ereignisse zwei völlig verschiedene Konzepte darstellen und nicht miteinander verwechselt werden dürfen.

Die bedingte Wahrscheinlichkeit $P(A|B)$, das ist die Wahrscheinlichkeit des Auftretens eines Ereignisses A, unter der Bedingung daß auch ein Ereignis B auftritt, wird definiert durch die Formel:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{3.8}$$

Da die Wahrscheinlichkeit für Ereignis A *und* B kommutativ ist, $P(A \cap B) = P(B \cap A)$, folgt aus (3.8) die Multiplikationsregel der Wahrscheinlichkeitsrechnung:

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B) \tag{3.9}$$

Wenn A und B unabhängige Ereignisse sind, folgt aus (3.7), daß

$$P(A|B) = P(A), \tag{3.10}$$

wie man es für unabhängige Ereignisse erwartet.

3.4 Wahrscheinlichkeitsverteilungen und Zufallszahlen

Eine Zufallszahl (Englisch: 'random variable' oder 'stochastic variable') wird in der Statistik verwendet, um Prozesse durch Wahrscheinlichkeiten, oder besser Wahrscheinlichkeitsverteilungen zu beschreiben. Dies ist am besten anhand eines Beispiels zu erklären. Für das klassische Beispiel des Würfelspiels kann man die Anzahl der gewürfelten Augen als stochastische Variable ('X') verwenden. Die möglichen Werte die diese Zufallszahl annehmen kann, sind natürlich durch $x_j = \{1, 2, 3, 4, 5, 6\}$ gegeben. Nachdem die Wahrscheinlichkeit für alle x_j gleich ist, ist die der Zufallszahl zugeordnete Verteilungsfunktion $f_X(x_j)$ unabhängig von x_j:

$$f_X(x_j) = const. \tag{3.11}$$

Wie weiter unten gezeigt wird, beschreibt die Verteilungsfunktion den Prozeß vollständig.

In diesem einfachen Beispiel ist die Verwendung einer Verteilungsfunktion nicht unbedingt notwendig, man kann genausogut von den Einzelwahrscheinlichkeiten ausgehen, um die Ereignisse eines beliebigen Würfelspiels vorherzusagen. Das geht aber nur, wenn der betrachtete Prozeß so einfach ist, daß man die Einzelwahrscheinlichkeiten aus theoretischen Überlegungen herleiten kann. In vielen Bereichen der Wissenschaft sind die Einzelwahrscheinlichkeiten für die interessierenden Ereignisse oft weitgehend unbekannt. Die Wahrscheinlichkeitsverteilung kann aber durch Einzelmessungen solcher Ereignisse zumindest teilweise (nie vollständig!) rekonstruiert werden. Mithilfe der Wahrscheinlichkeitsverteilung können Aussagen über das Auftreten von Einzelereignissen gemacht werden, da die Wahrscheinlichkeitsverteilung den Prozeß vollständig beschreibt. Dies ist sogar dann möglich, wenn die Wahrscheinlichkeitsverteilung nicht vollständig bekannt ist.

Es dürfte Anhand dieser Beschreibung klar sein, daß Ergebnisse (und Fehler) von Meinungsumfragen (zum Beispiel bei Wahlen) auf die Verwendung unvollständiger Verteilungsfunktionen zurückgeführt werden können: Anhand einer gewissen (kleinen) Anzahl von Stichproben gewinnt man (unvollständige) Information über die dem Wahlausgang zugrundeliegende Wahrscheinlichkeitsverteilung. Anhand der Verteilungsfunktion kann man im Mittel voraussagen, wer (mit einer gewissen Wahrscheinlichkeit) die Wahlen gewinnen sollte. Die Tatsache, daß die Wahrscheinlichkeitsverteilung durch das Nehmen einer endlichen Stichprobenzahl nicht exakt bekannt ist, führt zu einer gewissen Schwankungsbreite oder Unsicherheit, die offensichtlich zur Stichprobenzahl invers proportional ist (aber i.A. nicht linear!).

Da man der Verteilungsfunktion Wahrscheinlichkeiten zuordnen möchte, muß diese im gesamten Wertebereich positiv sein, und sollte außerdem auf eins normiert sein:

$$f_X(x_j) \geq 0 \tag{3.12}$$

$$\sum_j f_X(x_j) = 1 \tag{3.13}$$

Wenn eine unnormierte Verteilung $f'_X(x_j)$ vorliegt, kann durch die Vorschrift

$$f_X(x_j) = f'_X(x_j) \Big/ \sum_j f'_X(x_j), \tag{3.14}$$

immer eine normierte Verteilung $f_X(x_j)$ gebildet werden, wobei sich die Summe über den gesamten Wertebereich der stochastischen Variablen erstreckt. Man verifiziert leicht, daß die Funktion $f_X(x_j)$ der Normierungsbedingung (3.13) genügt. Für das eingangs angeführte Beispiel des Würfelspiels findet man so *const.* $= 1/6$. Wenn die Wahrscheinlichkeitsverteilung die Kriterien (3.12) und (3.13) erfüllt, kann die Wahrscheinlichkeit für das Auftreten eines Ereignisses in einem gewissen Wertebereich $S = \{x_1, x_2, .., x_k\}$ durch Aufsummieren der Verteilungsfunktion bestimmt werden:

$$P(X \in S) = \sum_{x_j=x_1}^{x_j=x_k} f_X(x_j). \tag{3.15}$$

Es ist zu beachten, daß (3.15) direkt aus dem Additionsprinzip für (auf eins normierte) Wahrscheinlichkeiten folgt.

Neben den diskreten Zufallszahlen und den ihnen zugeordneten Verteilungsfunktionen kennt man auch sogenannte kontinuierliche Zufallszahlen, die beliebige Werte in einem kontinuierlichen Interval annehmen können (z.B. den Bereich $x \in [x_1, x_2]$ auf der reellen Achse). Für eine stückweise kontinuierliche, positiv definite und auf das Intervall $[x_1, x_2]$ normierte Funktion $f_X(x)$ findet man die Wahrscheinlichkeit, daß sich X im Intervall $[a, b]$ befindet ($x_1 \leq a \leq b \leq x_2$) durch Integration:

$$P(a \leq X \leq b) = \int_{x=a}^{x=b} f_X(x)dx\,. \tag{3.16}$$

Analog zu (3.12) und (3.13) lauten die Kriterien für die kontinuierliche Wahrscheinlichkeitsverteilung neben der Bedingung der stückweisen Kontinuität:

$$f_X(x) \geq 0 \tag{3.17}$$

$$\int f_X(x)dx = 1\,, \tag{3.18}$$

wobei sich das Integral über den gesamten Definitionsbereich erstreckt.

Wir wollen uns als nächstes mit der Frage beschäftigen, wie man aus einer Reihe von Stichproben (Meßwerten) Information über die Verteilungsfunktion erhält. Dazu führt man die algebraischen Momente einer Zufallsgröße X (bzw. ihrer Verteilung) ein. Das n-te algebraische Moment $\langle X^n \rangle$ einer diskreten Zufallsgröße ist definiert als:

$$\langle X^n \rangle = \sum_j x_j^n f_X(x_j)\,. \tag{3.19}$$

Für eine kontinuierliche Verteilungsfunktion definiert man analog:

$$\langle X^n \rangle = \int x^n f_X(x) dx \,. \tag{3.20}$$

Wie üblich erstrecken sich hier Summation und Integration über den Definitionsbereich der Verteilung.

Man erkennt das erste Moment $\langle X \rangle$ als den Mittelwert (Erwartungswert) der Zufallsgröße, die Kombination $(\langle X^2 \rangle - \langle X \rangle^2)$ als Varianz. Ihre Quadratwurzel definiert die Standardabweichung σ_X (Schwankungsbreite):

$$\sigma_X \equiv \sqrt{\langle X^2 \rangle - \langle X \rangle^2} \tag{3.21}$$

Mit anderen Worten, die Momente einer Verteilung enthalten wesentliche Information über die Verteilung selbst. Die Momente niederer Ordnung sind hierbei am wichtigsten, da sie das globale Verhalten der Verteilungsfunktion beschreiben, während die höheren Ordnungen die Details der Verteilung beinhalten: Das erste Moment gibt den Mittelwert an, die Standardabweichung ist ein Maß für die Breite der Verteilung. Z.B. impliziert eine kleine Standardabweichung eine relativ scharfe Spitze der Verteilungsfunktion und gibt an, daß es ziemlich sicher ist, bei einer Messung der Variablen X einen Meßwert X_i in der Nähe von $\langle X \rangle$ zu finden.

Eine Stichprobe ist im wesentlichen eine Messung der Verteilungsfunktion. Man führt wiederholt eine Messung der stochastischen Variablen X durch, und erstellt aus den auftretenden Werten eine Häufigkeitsverteilung. Normiert man diese laut der Vorschrift (3.13) oder (3.18), ergibt sich daraus empirische Information über die Verteilungsfunktion. Die so gewonnene Verteilung ist aufgrund der endlichen Stichprobenanzahl immer mit einem experimentellen Fehler behaftet. Dieser kann durch wiederholtes Messen verringert werden. Die Momente niederer Ordnung können aber im Allgemeinen durch eine relativ kleine Stichprobenzahl genau bestimmt werden, während eine genaue Bestimmung der höheren Momente eine große Stichprobenzahl voraussetzt.

Schließlich soll noch gezeigt werden, daß eine Verteilungsfunktion vollständig durch ihre Momente spezifiert wird. Dazu führen wir die sogenannte charakteristische Funktion $\phi_X(k)$ ein, die als die Fouriertransformierte der Verteilungsfunktion definiert ist:

$$\phi_X(k) = \int e^{ikx} f_X(x) dx = \sum_{n=0}^{\infty} \frac{(ik)^n \langle X^n \rangle}{n!} = \left\langle e^{ikX} \right\rangle \tag{3.22}$$

Die letzten Schritte in obiger Gleichung erhält man durch die Taylorreihenentwicklung des Exponenten. In der Praxis ist eine solche Reihenentwicklung nur sinnvoll, wenn sie schnell genug konvergiert. Wenn alle Momente bekannt sind, kann man über die Reihenentwicklung (3.22) die Verteilungsfunktion $f_X(x)$ durch Rücktransformation bestimmen:

$$f_X(x) = \frac{1}{2\pi} \int e^{-ikx} \phi_X(k) dk \tag{3.23}$$

Obwohl die Gleichungen (3.22) und (3.23) zeigen, daß ihre Momente eine Verteilung vollständig beschreiben, ist die Konvergenz der Reihenentwicklung nach den Momenten in der Praxis problematisch. Viel bessere Konvergenzeigenschaften bietet eine sogenannte Kumulantenentwicklung. Diese ist definiert als:

$$\phi_X(k) = \exp\left[\sum_{n=1}^{\infty} \frac{(ik)^n}{n!} C_n(X)\right] \tag{3.24}$$

wobei $C_n(X)$ der Kumulant n–ter Ordnung ist. Durch Entwicklung der Gleichungen (3.22) und (3.24) nach k und Gleichsetzen der gleichen Ordnungen findet man für die ersten drei Kumulanten:

$$\begin{aligned} C_1(X) &= \langle X \rangle \\ C_2(X) &= \langle X^2 \rangle - \langle X \rangle^2 \\ C_3(X) &= \langle X^3 \rangle - 3\langle X \rangle \langle X^2 \rangle + 2\langle X \rangle^3 \end{aligned} \tag{3.25}$$

Man erkennt im ersten Kumulanten wieder den Mittelwert, während der zweite die Varianz darstellt. Der dritte Kumulant ist ein Maß für die Asymmetrie der Verteilung und wird in der Literatur oft als Kurtosis bezeichnet. Der wichtige Punkt an der Kumulantenentwicklung ist, daß man oft eine gute Abschätzung der Verteilungsfunktion bekommt, wenn die Kumulanten niedrigster Ordnung bekannt sind. Dies gilt z.B. für die δ-Verteilung, für die nur $C_1 \neq 0$ ist, und insbesondere für die in vielen Bereichen auftretende Gaußverteilung (siehe Abschnitt 3.5.3), deren Momente der Ordnung 3 und höher verschwinden. Mit anderen Worten, eine Gaußverteilung ist durch Mittelwert und Standardabweichung vollständig bestimmt.

3.5 Wichtige Verteilungsfunktionen

Im vorigen Abschnitt wurde gezeigt, daß die Verteilungsfunktion eines stochastischen Prozeßes diesen vollständig beschreibt und daß die entsprechende Wahrscheinlichkeitsdichte durch wiederholte Messung einer stochastischen Variablen empirisch bestimmt werden kann. Manchmal ist letzteres aber unnötig, da die Verteilungsfunktion aus den dem Prozeß zugrundeliegenden Gesetzen theoretisch hergeleitet werden kann. Drei wichtige Beispiele von in der Natur häufig auftretenden Verteilungsfunktionen sollen im weiteren vorgestellt werden: die Binomialverteilung, die Poissonverteilung und die Gaußverteilung.

Die Binomialverteilung beschreibt ein Experiment, das genau zwei mögliche Ergebnisse zur Folge haben kann, und folgt einfach aus dem Newtonschen Binomialsatz, daher der Name.

Die Poissonverteilung ist von großer Bedeutung, da sie die Statistik des Zählprozesses an sich beschreibt. Wie in Abschnitt (9.13) gezeigt wird, ist diese Verteilung vollständig durch das erste Moment, also ihren Mittelwert, bestimmt, und die Standardabweichung ergibt sich einfach als Wurzel aus dem Mittelwert. Diese Tatsache ist

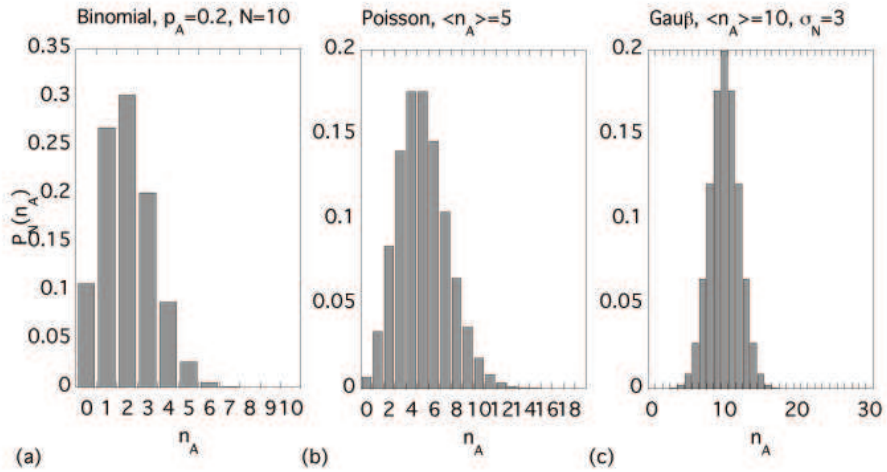

Abbildung 3.1: (a.) Binomialverteilung für $p_A = 0.2, N = 10$; (b.) Poissonverteilung für $\langle n_A \rangle = 5$; (c.) Gaußverteilung für $\langle n_A \rangle = 10.0, \sigma_N = 3.0$.

von hervorragender Bedeutung in der Experimentalphysik, da sie es erlaubt, den experimentellen Fehler in einer gemessenen Anzahl von Teilchen, die bei einem *beliebigen* Experiment detektiert werden, einfach durch ziehen der Quadratwurzel abzuschätzen.

Die Gaußverteilung ist insbesondere aufgrund ihres universalen Charakters von allergrößter Bedeutung. Die Universalität der Gaußverteilung steht im direkten Zusammenhang mit dem Begriff 'Wahrscheinlichkeit' selbst. Dies folgt aus dem sogenannten Gesetz der großen Zahlen, das im wesentlichen besagt, daß der Mittelwert einer Anzahl unabhängiger Experimente dem Mittelwert der dem Experiment zugrundeliegenden *Verteilungsfunktion* entspricht, wenn die Anzahl der Experimente genügend groß gewählt wird. Dies erscheint beim ersten Hinsehen als trivial, die Bedeutung dieser Aussage wird aber klar, wenn man sich vergegenwärtigt, daß dadurch die Grundlage für die empirische Bestimmung der Eigenschaften einer Verteilungsfunktion über das Nehmen von Stichproben gelegt wird. Eine spezielle Form des Gesetzes der großen Zahlen ist der zentrale Grenzwertsatz, der die Universalität der Gaußverteilung erklärt. Der zentrale Grenzwertsatz ist der Beweis für die Tatsache, daß bei einer großen Anzahl von Messungen einer stochastischen Variablen X, der *Mittelwert* dieser Messungen, also die stochastische Variable Y definiert durch ihre Werte y_N

$$y_N = \frac{x_1 + x_2 + ... + x_N}{N} \tag{3.26}$$

einer Gaußverteilung folgt. Der springende Punkt ist, daß dies für eine *beliebige* Form der Verteilungsfunktion $f_X(x)$ für die Variable X gilt. Den Beweis müssen wir aus Platzgründen weglassen. Wie bereits oben erwähnt, ist die Gaußverteilung dadurch

gekennzeichnet, daß nur das erste und zweite Moment von Null verschieden sind, die Momente höherer Ordnung verschwinden.

3.5.1 Die Binomialverteilung

Wir betrachten ein Experiment, das zwei mögliche Ergebnisse A und B hat. Die ihnen zugeordneten Wahrscheinlichkeiten seien p_A und p_B. Da die Summe dieser Wahrscheinlichkeiten eins sein muß, $p_A + p_B \equiv 1$, wird die Anzahl der gemessenen Ergebnisse nach N unabhängigen Messungen durch $N = n_A + n_B$ gegeben. Die Wahrscheinlichkeit für eine gewisse Permutation für das n_A-fache Auftreten des Ergebnisses A und das n_B-fache Zustandekommen des Ergebnisses B ist also $p_A^{n_A} p_B^{n_B}$. Wenn die Reihenfolge des Auftretens der Ereignisse A und B nicht von Bedeutung ist, beschreibt die Verteilung der möglichen *Kombinationen* eine Reihe solcher Experimente. Deren Verteilungsfunktion ist die Binomialverteilung. Aufgrund der Tatsache daß es genau $N!/n_A!n_B!$ Möglichkeiten gibt, eine Kombination von n_A und n_B Ereignissen zu realisieren, ist die Binomialverteilung gegeben durch:

$$P_N(n_A) = \frac{N!}{n_A! n_B!} p_A^{n_A} p_B^{n_B} = \frac{N!}{n_A!(N-n_A)!} p_A^{n_A} (1-p_A)^{(N-n_A)} \tag{3.27}$$

Die Binomialverteilung ist in Abbildung (3.1a) gezeigt für $N = 10$ und $p_A = 0.2$. Die Binomialverteilung (3.27) ist auf eins normiert, wie man unter Verwendung des Binomialsatzes leicht sieht:

$$\sum_{n_A=0}^{N} P_N(n_A) = \sum_{n_A=0}^{N} \frac{N!}{n_A!(N-n_A)!} p_A^{n_A} (1-p_A)^{(N-n_A)} = (p_A + p_B)^N = 1 \tag{3.28}$$

Das erste Moment ist:

$$\begin{aligned} \langle n_A \rangle &= \sum_{n_A=0}^{N} P_N(n_A) n_A = \sum_{n_A=0}^{N} \frac{N!}{n_A!(N-n_A)!} p_A^{n_A} (1-p_A)^{(N-n_A)}\, n_A \\ &= p_A \frac{\partial}{\partial p_A} \sum_{n_A=0}^{N} P_N(n_A) = p_A \frac{\partial}{\partial p_A} (p_A + p_B)^N = N p_A. \end{aligned} \tag{3.29}$$

Auf ähnliche Art und Weise findet man für das zweite Moment:

$$\langle n_A^2 \rangle = \sum_{n_A=0}^{N} P_N(n_A) n_A^2 = p_A^2 N(N-1) + N p_A \tag{3.30}$$

Die Standardabweichung σ_N ist somit:

$$\sigma_N = \sqrt{\langle n_A^2 \rangle - \langle n_A \rangle^2} = \sqrt{N p_A (1-p_A)} = \sqrt{N p_A p_B} \tag{3.31}$$

3.5.2 Die Poissonverteilung

Nimmt in Gl. (3.27) N immer mehr zu, während p_A abnimmt, bleibt gleichzeitig aber das Produkt Np_A konstant, geht die Binomialverteilung in die Poissonverteilung über:

$$P_N(n_A) = \langle n_A \rangle^{n_A} \frac{e^{-\langle n_A \rangle}}{n_A!} \tag{3.32}$$

Man verifiziert leicht, daß die Poissonverteilung auf eins normiert ist und daß das erste Moment dieser Verteilung durch $\langle n_A \rangle$ in Gl. (3.32) gegeben wird. Der Mittelwert $\langle n_A \rangle = Np_A$ charakterisiert die Poissonverteilung also vollständig. Abbildung (3.1b) zeigt die Poissonverteilung für $\langle n_A \rangle = 5$.

3.5.3 Die Gaußverteilung

Für große Werte von Np_A geht die Binomialverteilung in eine Gaußverteilung über. Für den Beweis wird auf die Literatur [**?**] verwiesen. Die Gaußverteilung ist gegeben durch:

$$P_N(n_A) = \frac{1}{\sigma_N \sqrt{2\pi}} \exp\Big(-\frac{1}{2} \frac{(n_A - \langle n_A \rangle)^2}{\sigma_N^2} \Big) \tag{3.33}$$

Mithilfe des Integrals

$$\int_{-\infty}^{\infty} e^{-a^2 x^2} dx = \frac{\sqrt{\pi}}{a} \tag{3.34}$$

zeigt man wieder leicht, daß die Verteilung (3.33) auf eins normiert ist. Das erste Moment ist durch $\langle n_A \rangle$ gegeben und kann durch eine entsprechende Variablentransformation berechnet werden. Die Standardabweichung σ_N ergibt sich durch Anwendung des Integrals

$$\int_{-\infty}^{\infty} x^2 e^{-a^2 x^2} dx = \frac{\sqrt{\pi}}{2a^3} \tag{3.35}$$

Man sieht aus der Symmetrie der Verteilung (3.33) um $\langle n_A \rangle$, daß sie durch ihre ersten zwei Momente vollständig beschrieben wird. Abbildung (3.1c) zeigt die Gaußverteilung für $\langle n_A \rangle = 10$ und $\sigma_N = 3$.

3.6 Zufallszahlen

Die moderne Physik hat ein grundlegendes Überdenken elementarer Stützen der klassischen Philosophie bewirkt, insbesondere des Kausalitätsprinzips. Als Physiker akzeptieren wir gewisse quantenmechanische Phänomene als statistisch im Sinne der prinzipiellen Unvorhersagbarkeit des Einzelereignisses. Ein Beispiel ist die Ungewißheit des genauen Zeitpunkts der Relaxation eines angeregten Atoms oder eines radioaktiven Zerfallsakts, die sich im Experiment mit einem einfachen Geigerzähler anschaulich

zeigen läßt: die Zeitabstände aufeinanderfolgender *ticks* sind im Einzelfall prinzipiell unvorhersehbar und zufällig, sie gehorchen aber statistischen Gesetzmäßigkeiten. Wir definieren **wahre Zufallszahlen** als Zahlenfolgen, deren Einzelwerte im physikalischen Sinn prinzipiell unvorhersehbar sind, deren Häufigkeitsverteilung und statistische Erwartungswerte aber physikalischen Gesetzmäßigkeiten folgen, oder aus mathematischer Sicht: *ein k-Tupel von Zahlen, das mit der statistischen Hypothese in Einklang steht, eine Realisierung eines zufälliges Vektors mit unabhängigen, identisch nach einer Verteilungsfunktion v verteilten Komponenten zu sein, nennt man ein k-Tupel von nach v verteilten Zufallszahlen (Überhuber).*
„Zufälle nennt man in der Natur, was beim Menschen Freiheit heißen würde“ (Goethe, 1811).

Viele Größen, die landläufig als statistisch angesehen werden, lassen sich auf prinzipiell determinierte Vorgänge zurückführen, die jedoch entweder aus praktischen Gründen nicht im Detail einzeln verfolgt werden oder aber grundsätzlich unzugänglich sind, wie etwa in chaotische Vorgängen, wo die Anfangsbedingungen einer Zustandsänderung nicht vollständig und ausreichend genau bestimmbar sind. Beispiele sind viele Meßfehler oder die Trajektorien eines Einzelmoleküls im Gas. Wenn daraus resultierende Zahlenfolgen den Gesetzmäßigkeiten der Folge wahrer Zufallszahlen gehorchen, heißen sie (experimentelle) Pseudozufallszahlen. Analog sind (algorithmische) Pseudozufallszahlen Zahlenfolgen, die aus sorgfältig entwickelten Algorithmen gewonnen werden. Algorithmen für Pseudozufallszahlen und ihre Anwendung sind Gegenstand der folgenden Ausführungen.

3.7 Generierung von Zufallszahlen

3.7.1 Algorithmen

Lineare Kongruenz Methode nach D.H. Lehmer, 1949 (Linear Congruential Generator). Sie ist die vermutlich häufigste Methode für Routineanwendungen und Basis für viele Bibliotheksfunktionen. Ihr Kern ist die Rekursionsvorschrift

$$a_{i+1} = (a_i \cdot b + c) \,\mathrm{mod} m\,. \tag{3.36}$$

Sie liefert eine Folge von natürlichen Pseudozufallszahlen $a_1, a_2, a_3, \ldots$

Die Parameter b, c sind geeignet gewählte natürliche Zahlen. Der Startwert a_0 ist eine beliebige natürliche Zahl im Bereich $[0, m-1]$. Oft wird (mit passenden Werten für b und m) $c = 0$ verwendet. Beispiel: $m = 32$, $b = 7$, $c = 1$, $a_0 = 0$ (das sind einfache Demo-Parameter. Empfohlene Parameter für reale Anwendungen: siehe Tabelle weiter unten. In C ist % der Modulo-Operator):

$$\begin{aligned} a_1 &= (07+1) \mod 32 &&= 1\%32 = 1 \\ a_2 &= (17+1) \mod 32 &&= 8\%32 = 8 \\ a_3 &= (87+1) \mod 32 &&= 57\%32 = 25... \end{aligned} \tag{3.37}$$

Wie man sieht, liegen die so gewonnenen Zufallszahlen im Bereich $(0, m-1)$. Will man große Zufallszahlen erzeugen, muß m groß sein, was nur bei entsprechend großen Werten von $(a_i b + c)$ möglich ist; hier liegt das Problem, weil dieser Zwischenwert maschinenintern durch die größte darstellbare Integerzahl begrenzt ist und die Gefahr eines Speicherüberlaufs besteht (overflow). Dieser ist schwer erkennbar, weil etwa in C dadurch keine Fehlermeldungen oder Warnungen erzeugt werden und mit den niedrigstelligen (least significant) Bits weitergerechnet wird. Abhilfe durch geeignete Algorithmen ist möglich (wird hier nicht näher behandelt, siehe aber die Funktion `myrandom2()` in Beispiel `rand1a.c`).

Die von der linearen Kongruenzmethode gelieferten Zahlenfolgen sind grundsätzlich periodisch. Bei schlechter Parameterwahl ist die Periode sehr kurz und das Verfahren mit diesen Parametern unbrauchbar, z.B. $a_0 = 0, b = 19, c = 1, m = 381 \Longrightarrow 0, 1, 20, 0, 1, 20, ..$ Manche, meist ältere Algorithmen liefern für die hochsignifikanten Bits (high bytes) bessere Zufallsqualitäten als für die wenig signifikanten Bits (low bytes). Es ist daher nicht sinnvoll, Zufallszahlen (etwa ein long integer) zu zerlegen und die Einzelbytes einzeln zu verwenden oder neu zu kombinieren. Beispielsweise erzeugt ein Generator mit $a_0 = 10^8, a = 31415821, b = 1, m = 1234567$ die Folge 35884508, 80001069, 63512650, 43635651, 1034472, 87181513, 6917174, 209855, 67115956, 59939877, bei der die letzte Stelle jeweils um 1 wächst. Die Ermittlung guter Parametersätze ist aufwendig, und es sollten nur erprobte Algorithmen und Parameterkombinationen eingesetzt werden.

3.7.2 Andere Methoden.

Kombinationsmethoden verwenden Kombinationen von Zufallszahlen verschiedener Generatoren. Mitunter können damit sehr große Periodenlängen erzielt werden. Physikalische Generatoren erzeugen wahre Zufallszahlen, mitunter durch miniaturisierte Systeme mit radioaktiven Quellen und Detektoren. Zum unten angeführten non-linear additive feedback random number generator siehe die Dokumentation in Linux. Ältere, einfache Generatoren sind der Fibonacci-Generator $(a_{n+1} = (a_n + a_{n-1}) \mod m)$ und der von Neumann Generator (Quadratmittengenerator: Man quadriert eine Zahl, schneidet aus der Mitte der Ziffernfolge eine Zahl heraus, die man wieder quadriert, usw.). Bitgeneratoren erzeugen eine (endlose) Folge von Einzelbits, die direkt verwendet oder aus denen Zahlen zusammengesetzt werden können.

3.7.3 Bibliotheksfunktionen.

In ANSI-C werden die Funktionen `int rand()` [in Linux/GNU oft `long rand()`] und `srand()` verwendet (`include math.h`). Sie beruhen häufig auf der linearen Kongruenzmethode. Der maximale Wert der darstellbaren Zufallszahl ist durch die systemdefinierte Makro-Variable `RAND_MAX` gegeben; sie ist oft gleich `INT_MAX` (oder `LONG_MAX`, für `long rand()`). Die ANSI-Norm verlangt ein `RAND_MAX` von lediglich 32767, was für technisch-wissenschaftliche Anwendungen völlig unzureichend ist. Wird die linea-

overflow bei	m	b	c	overflow bei	m	b	c
2^{20}	6075	106	1283	2^{27}	121500	1021	25673
2^{22}	7875	211	1663	2^{27}	259200	421	54773
2^{22}	7875	421	1663				
2^{23}	6075	1366	1283	2^{28}	117128	1277	24749
2^{23}	6655	936	1399	2^{28}	121500	2041	25673
2^{23}	11979	430	2531	2^{28}	312500	741	66037
2^{24}	14406	967	3041	2^{29}	145800	3661	30809
2^{24}	29282	419	6173	2^{29}	175000	2661	36979
2^{24}	53125	171	11213	2^{29}	233280	1861	49297
				2^{29}	244944	1597	51749
2^{25}	12960	1741	2731				
2^{25}	14000	1541	2957	2^{30}	139968	3877	29573
2^{25}	21870	1291	4621	2^{30}	214326	3613	45289
2^{25}	31104	625	6571	2^{30}	714025	1366	150889
2^{25}	139968	205	29573				
				2^{31}	134456	8121	28411
2^{26}	29282	1255	6173	2^{31}	259200	7141	54773
2^{26}	81000	421	17117				
2^{26}	134456	281	28411	2^{32}	233280	9301	49297
				2^{32}	714025	4096	150889
2^{27}	86436	1093	18257				

Tabelle 3.1: Parameter für die lineare Kongruenzmethode (aus: Numerical Recipies)

re Kongruenzmethode verwendet, entspricht `RAND_MAX` dem Parameter m. Die größte erreichbare Zufallszahl ist daher `RAND_MAX-1`, die kleinste Null.

Die Funktion `srand(int/long Anfangswert)` ist die seed-Funktion, die den Anfangswert der Zufallsfolge über den Parameter Anfangswert (`int` oder `long` je nach Implementierung) festlegt, also das a_0 in der oben verwendeten Bezeichnungsweise. Wenn im Code keine Vorkehrungen getroffen worden sind, läuft ein Programm immer mit der gleichen Zufallszahlenfolge ab. Dieses Verhalten ist während der Programmentwicklung im Allgemeinen wünschenswert und wird im Teststadium meist beibehalten. Wird ein zufälliger Anfangswert gewünscht, bietet sich die Funktion `time(NULL)` an, die als Rückgabewert die seit dem 1.1.1970, 0.00 Uhr ('Unix-Urknall') vergangene Sekundenanzahl liefert. Keineswegs sollte `srand(time(0))` mehrmals innerhalb einer Sekunde aufgerufen werden, weil damit der Generator immer auf den gleichen Anfangswert zurückgesetzt wird. `srand()` setzt eine globale Systemvariable und wirkt daher auf alle `rand()`-Aufrufe im Programm unabhängig von ihrer Position im Code. Meist wird `srand()` nur einmal, nämlich im Rahmen der Initialisierungen (Programmstart), aufgerufen.

Die oben erwähnte Unzulänglichkeit des Mindeststandards in der ANSI-Norm hat in der Unix-/Linux-Welt zur Implementierung von alternativen Funktionen (z.B. dem *non-linear additive feedback random number generator*) geführt. Statt `ran()` und `srand()` werden `random()` und `srandom()` eingesetzt (relativ große Periode von $(16 * (2^{31} - 1))$, sowie die Hilfsfunktionen `initstate()` und `setstate()`. Hier besteht in der Qualität der Zufallszahlen kein Unterschied zwischen low bytes und high bytes.

In Fortran 77 sieht der Standard keinen intrinsischen Zufallszahlengenerator vor (in Fortran 90 hingegen schon). Allerdings gibt es eine Vielzahl von frei erhältlichen Bibliotheken mit Zufallszahlengeneratoren. Unter Linux sind - compilerabhängig - für Fortran 77 oft die gleichen Generatoren wie unter C verfügbar. Allerdings sind für komplexere Anwendungen, z.B. Monte-Carlo Methoden, mitunter auch gute Standard-Bibliotheksfunktionen nicht ausreichend, und eine sorgfältige Vorarbeit ist unabhängig von der gewählten Sprache unerläßlich.

3.8 Häufigkeitsverteilungen

3.8.1 Gleichverteilte und ungleichverteilte Zufallszahlen.

Die meisten Standardgeneratoren (und Bibliotheksfunktionen) liefern gleichverteilte Zufallszahlen, d.h. jede Integerzahl (oder jedes beliebige Sub-Intervall bei floating point Zahlen) tritt mit gleicher Wahrscheinlichkeit und damit näherungsweise gleich häufig auf. Manchmal will man ungleichverteilte Zahlenfolgen generieren, deren Werte mit einer bestimmten Häufigkeitsverteilung auftreten, z.B. einer Gaußverteilung, Exponentialverteilung, logarithmischen Verteilung, dem Planck'schen Strahlungsgesetz, einer eigenen Vorschrift folgend, usw.

3.8.2 Zurückweisungsmethode

(Verwerfungsmethode, rejection method). Diese Methode ist sehr allgemein anwendbar und leicht zu implementieren, aber mitunter ineffizient. Die vorgegebene Verteilungsfunktion $y = v(x)$ sei im Intervall $A \leq x \leq B$ anzuwenden und habe darin einen maximalen Funktionswert y_{max}. Nun werden zwei unabhängige, gleichverteilte Zufallszahlen R_1 und R_2 im Intervall $(0, R_{max})$ generiert.

- Die erste wird auf das Intervall (A, B) abgebildet und wählt darin einen zufälligen x-Wert aus: $x_{R1} = A + R_1(B - A)/(R_{max})$. Nun wird der Funktionswert an dieser Stelle bestimmt: $v(x_{R_1})$.

- Dann wird R_2 auf den Bereich der Funktionswerte von $v(x)$, also $(0, y_{max})$ abgebildet und geprüft, ob $R_2/R_{max} < v(x_{R_1})/y_{max}$ ist. Bildlich entspricht das der Frage, ob der (auf y_{max} normierte) Zufallswert unterhalb des Funktionsgraphen, also unter $v(x_{R_1})$ liegt oder darüber.

- Gegebenenfalls wird R_1 als gültiger Wert akzeptiert. Andernfalls, also wenn $R_2/R_{max} > v(x_{R_1})/y_{max}$ ist, wird der gesamte Schritt verworfen.

Es führt also nicht jeder Schritt zum Erfolg, insbesondere ist die Methode dann äußerst ineffizient, wenn $v(x)$ nur eine lokale Spitze hat und überall sonst sehr klein ist, weil dann fast jeder Schritt verworfen werden muß. Eine mögliche Verbesserung für solche Fälle besteht darin, eine Funktion $V(x)$ zu suchen, deren Werte im Intervall $A \leq x \leq B$ leicht berechenbar sind und die eine möglichst eng anliegende Hülle um $v(x)$ bildet: an jeder Stelle muß aber $V(x) \geq v(x)$ sein. An die Stelle des Vergleichs $R_2/R_{max} < v(x_{R_1})/y_{max}$ tritt der Vergleich $R_2/R_{max} < v(x_{R_1})/V(x_{R_1})$. Die Funktion $V(x)$ kann auch stückweise zusammengesetzt werden und braucht lediglich eindeutig berechenbar zu sein; sie muß insbesondere nicht stetig sein.

3.8.3 Transformationsmethode.

Hier wird versucht, jene Funktion $f(R)$ von Zufallszahlen R explizit zu finden, mit der die Werte $x_i = f(R_i)$ einer vorgegebenen Wahrscheinlichkeitsdichte (Häufigkeitsverteilung) $v(x)$ auftreten. Beispielsweise liefern gaußverteilte Zufallsfolgen mit

$$v(x) = \frac{1}{\sqrt{2\pi}} \exp\Big(- \frac{(x - x_0)^2}{2} \Big) \tag{3.38}$$

bevorzugt Werte x nahe x_0. Es ist $v(x)dx$ die Wahrscheinlichkeit für das Auftreten einer Zufallszahl im Intervall $(x, x + dx)$ und (Normierung der Gesamtwahrscheinlichkeit auf 1):

$$\int_{x=-\infty}^{x=+\infty} v(x)dx = \frac{1}{\sqrt{2\pi}} \int_{x=-\infty}^{x=+\infty} \exp\Big(- \frac{(x - x_0)^2}{2} \Big) = 1 \tag{3.39}$$

Analoges gilt für eine Wahrscheinlichkeitsdichte (Häufigkeitsverteilung) $Z(R)$ von gleichverteilten Zufallszahlen R, die von R unabhängig ist, für die also $Z(R) = const = 1$. Nach einer Transformation $Z(R) \rightarrow v(x)$ mit Hilfe von $x = f(R)$ muß jede Wahrscheinlichkeit $Z(R)dR$ einer Wahrscheinlichkeit $v(x)dx$ entsprechen, und es muss in jedem Intervall

$$\left|Z(R)dR\right| = \left|v(x)dx\right| \rightarrow v(x) = Z(R)\left|\frac{dR}{dx}\right| = f(R) \tag{3.40}$$

sein. Mit $Z(R) = 1$ wird

$$\frac{dR}{dx} = v(x) \rightarrow R = \int v(x)dx = W(x) \rightarrow x = W^{-1}(R) \tag{3.41}$$

Die praktische Anwendbarkeit der Transformationsmethode hängt davon ab, wie leicht $W^{-1}(R)$ berechenbar oder in einem Computeralgorithmus (eventuell näherungsweise) darstellbar ist.

3.8.4 Poissonverteilte Zufallszahlen und Exponentialverteilung.

Die Poissonverteilung $P(n)$ beschreibt die Wahrscheinlichkeit für das Auftreten von n zufallsbedingten Ereignissen in einem dimensionslosen Intervall (interpretierbar als Zeit, Strecke, Raum, auch das Auftreten von x Fehlern in einer Produktionscharge, usw.). Sie hat einen einzigen Parameter, λ, der gleichzeitig Mittelwert und Standardabweichung darstellt und eine Rate (mittlere Anzahl von Ereignissen in einem Intervall) angibt:

$$P_n(n) = \frac{\lambda^n}{n!}e^{-\lambda} \tag{3.42}$$

n sind ganze Zahlen. Für große Werte (>10) von λ wird die Poissonverteilung näherungsweise zur Gaußverteilung mit $\sigma^2 = \lambda$. Ersetzt man λ für physikalische Anwendungen durch ein Produkt aus (Mittelwert/Einheitsintervall) $\times$ Intervallgröße, z.B. λ_t $[s^{-1}]$ (Ereignisse pro Zeiteinheit, Rate) $\times$ Zeitintervallgröße t [s], ist $\lambda_t t$ wieder eine dimensionslose Zahl und man erhält

$$P_x(x) = \frac{(\lambda_t t)^x}{x!}e^{-\lambda_t t} \tag{3.43}$$

Mit der Poissonverteilung eng verknüpft ist die Exponentialverteilung, welche die dimensionslose Intervallänge (beispielsweise als Wartezeit interpretierbar) bis zum Eintreten des nächsten (ersten) Ereignisses angibt. Wählt man wieder Zeitintervalle, muß man dazu die obige Gleichung nach der Zeit differenzieren und erhält

$$P_t(t) = \lambda_t e^{-\lambda_t t} \tag{3.44}$$

Der Erwartungswert und die Standardabweichung sind beide gleich $1/\lambda$ [s]. Für die Exponentialverteilung lassen sich das Integral $W(t)$ und die Umkehrfunktion W^{-1} leicht berechnen (Transformationsmethode):

$$W(t) = R = \int \lambda_t e^{-\lambda_t t} dt = 1 - e^{-\lambda_t t}$$
$$t = W^{-1}(1-R) = -\frac{1}{\lambda_t}\ln(1-R) \rightarrow -\frac{1}{\lambda_t}\ln R \tag{3.45}$$

Im letzten Schritt wurde die Tatsache verwendet, daß R und $1-R$ äquivalente gleichverteilte, auf 1 normierte Zufallszahlenfolgen sind. Gibt man eine mittlere Ereignisrate λ_t vor, z.B. 100 radioaktive Zerfälle pro Sekunde, bilden die Werte $t_i = -0.01 \cdot \ln(R_i)$ eine Folge von exponentialverteilten Wartezeiten, die aus gleichverteilten Zufallszahlen R_i berechnet werden kann. Siehe Beispielprogramme.

3.8.5 Gaußverteilte Zufallzahlen

Gaußverteilte Zufallzahlen folgen der Verteilungsfunktion

$$v(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{3.46}$$

Es ist keine analytische Darstellung der in der Transformationsmethode definierten Funktionen W und W^{-1} bekannt, aber es gibt einige effiziente Algorithmen:
Box-Muller Algorithmus: (ohne Beweis)

- Generiere zwei gleichverteilte, auf 1 normierte Zufallszahlen R_1 und R_2.
- Berechne $\omega = 2\pi R_1$ und $\varrho = -2\ln(R_2)$
- Berechne $X_1 = \sqrt{\varrho}\cos\omega$ und $X_2 = \sqrt{\varrho}\sin\omega$

X_1 und X_2 sind die gesuchten gaußverteilten Zufallszahlen. Sie benötigen also Paare von gleichverteilten Zufallszahlen, um Paare von gaußverteilten Zufallszahlen zu erhalten.
Polarmethode von Marsaglia: (ohne Beweis)

- Generiere zwei gleichverteilte, auf 1 normierte Zufallszahlen R_1 und R_2.
- Setze $S = 2R_1 - 1$ und $T = 2R_2 - 1$
- Berechne $W^2 = S^2 + T^2 > 1$. Abbrechen/Neubeginn, wenn $W^2 > 1$
- Berechne $X_1 = (S/W)\sqrt{(-2\ln(W^2))}$ und $X_1 = (T/W)\sqrt{(-2\ln(W^2))}$.

X_1 und X_2 sind wiederum die gesuchten gaußverteilten Zufallszahlen. Sie benötigen auch hier Paare von gleichverteilten Zufallszahlen, um Paare von gaußverteilten Zufallszahlen zu erhalten. Allerdings führt nicht jede Kombination von Werten R_1 und R_2 zum Erfolg, es handelt sich um eine Version der Verwerfungsmethode. Sie ist gegenüber dem Box-Muller-Algorithmus vorteilhaft, wenn die Berechnung von Winkelfunktionen zeitaufwendig ist.

3.8.6 Auf benutzerdefinierte Intervalle beschränkte (ganzzahlige) Zufallszahlen.

Die mit Bibliotheksfunktionen generierten, ganzzahligen Zufallszahlen liegen im Bereich von 0 und einem Maximalwert, der in C für `rand()` durch den Makro `RAND_MAX` gegeben ist. Will man den Bereich einschränken, etwa auf die Werte $(1, 2, 3, 4, 5, 6)$ zur Simulation eines Würfels, liegt eine Modulo-Division nahe: $1 + \texttt{rand}()\%m$ mit $m = 6$. Die Methode hat zwei Nachteile: erstens beruht das Ergebnis auf den least significant bits, was, wie erwähnt, bei manchen Generatoren unvorteilhaft ist. Zweitens ist im allgemeinen der interessierende Bereich (mit Startwert $0 : 0, 1, 2, 3, 4, 5$) nicht ganzzahlig in `RAND_MAX` enthalten. Es gibt also einige Zahlen, die ein Mal öfter vorkommen als andere, nämlich die zwischen $m * (\texttt{int})(\texttt{RAND\_MAX}/m)$ und `RAND_MAX`. Im vorliegenden Fall ist das nicht kritisch, weil $\texttt{RAND\_MAX}/5$ sehr groß im Vergleich zum gewählten Bereich der Zufallszahlen ist. Ist aber entweder `RAND_MAX` klein oder der gewünschte Bereich groß, kann das Probleme ergeben. Man muß korrekterweise alle Zufallszahlen $R \geq m * (\texttt{int})(\texttt{RAND\_MAX}/m)$ verwerfen.

3.8.7 Beliebig normierte (Gleitkomma-) Zufallszahlen

Auf 1 normierte Zufallszahlen.

Die ganzzahligen Zufallszahlen liegen im Bereich von 0 und `RAND_MAX`. Die Normierung auf 1 erfolgt durch `(double) rand()/(RAND_MAX+1.0)` oder `(double) rand()/(double)(RAND_MAX))` je nachdem, ob `RAND_MAX` die größtmögliche Zufallszahl miteinschließt oder nicht und ob im Ergebnis der Wert 1 inkludiert sein soll (letzteres meist nicht). Beachten Sie den Dezimalpunkt nach `1.` in `(RAND_MAX+1.)`, der den Ausdruck zu einer `double` Größe macht. Weglassen erzeugt möglicherweise eine kleine Katastrophe, weil `RAND_MAX` oft gleich `LONG_MAX` ist und `RAND_MAX+1` dann durch overflow -2147483648 ergibt!

Andere Normierungen.

Grundsätzlich gilt das oben Gesagte. Die Linux-Dokumentation empfiehlt für den nonlinear additive feedback random number generator mit Bezug auf die Numerical recipies als Beispiel für die Normierung auf 10 `j=1+(int) (10.0*rand()/(RAND_MAX+1.0));` Beachten Sie, daß sowohl der Token 1.0 als auch der Token 10.0 die Berechnung des inneren Ausdrucks (Klammer) mit `double` Präzision bewirken. Das double-Zwischenergebnis wird in ein `int` konvertiert.

3.9 Anwendungsbeispiel von Zufallszahlen: Monte Carlo Methoden.

Monte-Carlo Methoden werden für Simulationen verwendet, wenn das Verhalten des zugrundeliegenden Modells nicht oder nur schwer analytisch oder durch einen geschlossenen Algorithmus beschrieben werden kann. Anschauliche Beispiele sind strömungstechnische Probleme oder die so genannten Life-Modelle, in denen die Entwicklung biologischer Populationen modelliert wird. Das Monte-Carlo Verfahren beruht darauf, daß einzelne Individuen (Moleküle, Sandkörner, biologische Einheiten) beobachtet werden. Jede Wechselwirkung oder jeder Verhaltensschritt erfolgt mit einer vorgegebenen Wahrscheinlichkeit und wird mit einem Zufallszahlengenerator ausgewählt. Für gesicherte Aussagen über das Verhalten des Gesamtsystems müssen sehr viele Einzelsequenzen (histories) mit jeweils vielen Wechselwirkungen und Auswahlmöglichkeiten verfolgt und die ermittelten Kenngrößen gesammelt werden.

Monte Carlo Methoden sind zeitaufwendig und erfordern daher sorgfältig ausgearbeitete Algorithmen. „Die Software wird schneller langsam, als die Hardware schneller wird “(Niklaus Wirth, ETH Zürich, Erfinder von Pascal).

Siehe Demoprogramme, insbesondere life.c und die dazugehörige Übungsaufgaben.

3.9.1 Anwendungsbeispiel Absorptionsgesetz.

Strahlung dringe senkrecht zur Oberfläche in Materie ein. Die Anzahl der Photonen, die eine Tiefe t erreichen, ist durch das Absorptionsgesetz gegeben:

$$N_t = N_0 e^{-\mu t} \tag{3.47}$$

Dabei ist: N_0 die Photonenzahl vor dem Absorber, N_t die Photonenzahl im Absorber nach der Weglänge t, t ist Weglänge im Absorber und μ eine Materialkonstante (Absorptionskoeffizient). Nun wollen wir würfeln (zunächst mit $m = 6$ Möglichkeiten), wie weit ein Photon fliegen soll, das heißt, in welcher Tiefe t es absorbiert wird. Dadurch werden Photonenzahlen mit Zufallszahlen korreliert: jeder Zufallszahl p_n entspricht ein Tiefenbereich $t_n - t_{n-1}$. Wenn jede Zufallszahl gleich wahrscheinlich ist, muß jeder Tiefenbereich die gleiche Anzahl absorbierter Photonen enthalten, also $N_0/6$, und offenbar mit steigender Tiefe anwachsen.
Bereich 1:

$$\Delta N = N_0 e^{-\mu t_0} - N_0 e^{-\mu t_1} = N_0(1 - e^{-\mu t_1})$$

Bereich 2:

$$\Delta N = N_0 e^{-\mu t_1} - N_0 e^{-\mu t_2} = N_0(1 - e^{-\mu t_2}) - \Delta N$$

Bereich n:

$$\Delta N = N_0 e^{-\mu t_{n-1}} - N_0 e^{-\mu t_n} = N_0(1 - e^{-\mu t_n}) - (n-1)\Delta N$$

Daraus folgt:

$$e^{-\mu t_n} = 1 - \frac{n\Delta N}{N_0} \Longrightarrow t_n = -\frac{1}{\mu} \ln\left(1 - \frac{n\Delta N}{N_0}\right) = -\Lambda \ln\left(1 - \frac{n}{m}\right) \tag{3.48}$$

$\Lambda = 1/\mu$ ist die mittlere freie Weglänge; $m = N_0/\Delta N$ ist die Anzahl der Intervalle und gleichzeitig die Anzahl der Würfelmöglichkeiten.

Für $\mu = 1$ cm erhält man $t_0 = 0, t_1 \approx 0.18$ cm, $t_2 \approx 0.40$ cm, $t_3 \approx 0.69$ cm, $t_4 \approx 1.09$ cm, $t_5 \approx 1.79$ cm, und $t_6 = \infty$; würfelt man 4, wäre die Interpretation eine Absorption des Photons irgendwo im Tiefenbereich 4 (0.69cm bis 1.09cm). Natürlich ist diese Intervallteilung sehr grob. Erlaubt man eine große Anzahl von (unterschiedlichen) Zufallszahlen, erhält man feinere Abstufungen, z.B. mit $m =$ `RAND_MAX`. Man kann $R = p/m$ auch als eine auf 1 normierte Zufallszahl R ansehen. Da die Zufallsfolge R äquivalent mit der Zufallsfolge der Werte von $1 - R$ ist, ist obige Gleichung für t_n äquivalent mit

$$t_R = \Lambda \ln R \tag{3.49}$$

Achtung: Wenn Sie R aus (`float`)`rand`()/(`float`)`RAND_MAX` berechnen, ist die Anzahl der möglichen unterschiedlichen Zufallszahlen R auch nur `RAND_MAX`. Siehe Demoprogramme, insbesondere `rand2.c` und `rand2a.c` und die Übungsaufgaben.

3.10 Demoprogramme und Aufgaben

- **Aufgabe 3.1: Erzeugung von Zufallszahlen mit Generator** `rand1a.c`
  `rand1a.c` mit Typ 4 liefert offenbar unbrauchbare Ergebnisse (rufen Sie `rand1a` mit Parametern $(-500, 4)$, dann mit $(-5000, 4)$, dann mit $(-50000, 4)$, usw. auf). Was ist daran unbrauchbar? Programmieren Sie einen einfachen Test, der diesen Fall erkennt.

```
/*********************************************************************/
/* rand1.c */
/* */
/* Aufruf: */
/* rand1 Anzahl */
/* */
/* Berechnet gleichverteilte Zufallszahlen und stellt sie */
/* grafisch dar. */
/* Anzahl = (int), Anzahl der Datenpunkte (Zufallszahlen[paare]) */
/* Beispiel: rand1 50000 */
/* Die Grafik wird durch Anklicken geloescht */
/* */
/* Es werden auf RAND_MAX normierte Zufallszahlen und die */
/* Bibliotheksfunktion rand() verwendet */
```

```
/* Fuer die Grafiken wird die GNU plotlib mit integer-Funktionen */
/* verwendet */
/* */
/* Compilieren: cc rand1.c -o rand1 -lplot */
/* */
/*****************************************************************/



/*****************************************************************/
/* rand1a.c */
/* */
/* Aufruf: */
/* rand1a Anzahl Typ */
/* */
/* Berechnet gleichverteilte Zufallszahlen und stellt sie */
/* grafisch dar. Statt rand() sollen hier versuchsweise andere/eigene */
/* Algorithmen verwendet werden. */
/* Anzahl = (int), Anzahl der Datenpunkte (Zufallszahlen[paare]) */
/* Negativer Wert: zeichnet Kreise statt Punkte */
/* (manchmal besser sichtbar) */
/* Typ = (int), einer der folgenden Generatoren: */
/* 0 ... Quotient des least significant bytes von rand() und 255 */
/* 1 ... Quotient der least significant 10 bits von rand() und 1023 */
/* 2 ... Methode von Park und Miller (Numerical Recipies) */
/* 3 ... Lehmer Methode mit a=ran(), b=16807, m=RAND_MAX */
/* [Overflow-Gefahr!!] */
/* 4 ... Lehmer Methode mit a=211, b=1663, m=7865 */
/* Beispiel: rand1a 50000 1 */
/* */
/* Compilieren: cc rand1a.c -o rand1a -lplot */
/* */
/*****************************************************************/
```

- **Aufgabe 3.2:**
  Gehen Sie von Programm `rand2.c` und der zugehörigen Beschreibung im Skriptum aus, modifizieren Sie den Zufallszahlengenerator so, daß er würfelt (1,2,3,4,5 oder 6 Augen) und stellen Sie die Tiefenverteilung der Photonen dafür grafisch ansprechend dar. Bauen Sie Photonenzähler für jedes Intervall ein, und geben Sie die Histogrammdaten an (Ausgabe der Zahlenwerte am Bildschirm).

```
/*****************************************************************/
/* rand2.c */
/* */
```

```
/* Aufruf: */
/* rand2 Anzahl Absorptionskoeffizient */
/* */
/* Berechnet log-verteilte Zufallszahlen und stellt sie */
/* grafisch dar. */
/* Es entspricht das entstehende Bild dem von Photonen, die von */
/* oben auf eine Probe auftreffen und verschieden weit eindringen */
/* Die Erklaerung des zugrundeliegenden Absorptionsgesetzes im */
/* Zusammenhang mit desem Beispiel finde Sie im Skriptum */
/* */
/* Anzahl = (int), Anzahl der Datenpunkte (Zufallszahlen[paare]) */
/* Beispiel: rand2 50000 10. */
/* */
/* Fuer die Grafiken wird die GNU plotlib mit float-Funktionen */
/* verwendet */
/* */
/* Compilieren: cc rand2.c -o rand2 -lplot */
/* */
/*********************************************************************/


/*********************************************************************/
/* rand2a.c */
/* */
/* Unterschied zu rand1.c: grafische Darstellung durch Striche */
/* (Photonenpfade) und zu exponentialverteilten Zeitpunkten */
/* (Simulation der statistischen Zeitintervalle der Photonenemission) */
/* Fuer die Wartezeit wird die Funktion usleep() verwendet */
/* */
/* Aufruf: */
/* rand2a Anzahl Absorptionskoeffizient */
/* */
/* Berechnet log-verteilte Zufallszahlen und stellt sie */
/* grafisch dar. */
/* Es entspricht das entstehende Bild dem von Photonen, die von */
/* oben auf eine Probe auftreffen und verschieden weit eindringen */
/* Die Erklaerung des zugrundeliegenden Absorptionsgesetzes im */
/* Zusammenhang mit desem Beispiel finde Sie im Skriptum */
/* */
/* Anzahl = (int), Anzahl der Datenpunkte (Zufallszahlen[paare]) */
/* Beispiel: rand2 50000 10. */
/* */
/* Fuer die Grafiken wird die GNU plotlib mit float-Funktionen */
/* verwendet */
```

```
/* */
/* Compilieren: cc rand2a.c -o rand2a -lplot */
/* */
/*************************************************************************/
```

- **Aufgabe 3.3:**
  Gehen Sie von Demoprogramm rand2.c aus, und ermitteln Sie (durch programmiertes Probieren) die Tiefe $d_{1/2}$, bis zu der die Hälfte der Photonen absorbiert wird (Halbwertsdicke). Geben Sie diese Tiefe $d_{1/2}$ als Zahlenwert (Ausgabeparameter Ihres Programms) für mehrere (auch kleine) Gesamtphotonenzahlen an. Stellen Sie die Verteilung grafisch dar (wie in `rand2.c`) und markieren Sie darin die ermittelte Tiefe $t_{1/2}$.

```
/*************************************************************************/
/* rand3.c */
/* */
/* Aufruf: */
/* rand3 Anzahl Typ */
/* */
/* Berechnet verschieden verteilte Zufallszahlen und stellt sie */
/* grafisch dar (gleichverteilt, linear, Gauss, zweidimensional-Gauss */
/* exponential(time) */
/* Anzahl = (int), Anzahl der Datenpunkte (Zufallszahlen[paare]) */
/* Typ = (char*), Auswahl: const lin gauss gaussgauss time */
/* Beispiel: rand1 50000 gaussgauss */
/* Die Zeitkonstante fuer die Option time ist 0.7s */
/* */
/* Compilieren: cc rand3.c -o rand3 -lplot */
/* */
/*************************************************************************/
```

- **Aufgabe 3.4:**
  Fügen Sie zu `rand3.c` einen Fibonacci-Generator und einen einfachen Quadratmittengenerator (mit 2 Ziffernstellen) hinzu. Der Code braucht nicht effizient zu sein.

```
/*************************************************************************/
/* life.c */
/* */
/* Monte-Carlo Demo */
/* Die Population des Modells besteht aus Bauern(gelb,40%), */
/* Handwerkern(rot, 20%), Jaegern (gelb,30%) und Kriegern(blau, 10%). */
/* Je 50% sind maennlich und weiblich. Ihr Lebensraum besteht aus */
```

```
/* 100x100 Feldern. Anfangs werden 1000 Individuen statistisch */
/* generiert und auf die Felder aufgeteilt. */
/* Die Lebensregeln sind: */
/* Krieger wandern (springen) weit und toeten eventuelle Bewohner des */
/* neuen Felds Sie haben keine direkten Nachkommen. Jaeger wandern */
/* etwas weniger weit, Handwerker und Bauern noch weniger. */
/* Wenn sie auf Felder des anderen Geschlechts kommen, */
/* entstehen neue Individuen nach obiger Wahrscheinlichkeit, Bauern */
/* haben viele Nachkommen. Jede Wanderung ist mit Altern verbunden, */
/* hohes Alter=Tod. Im Sueden ist die Lebenswerwartung hoeher als */
/* im Norden. */
/* */
/* Durch Variation der Parameter entstehen Szenarios, in denen die */
/* Population waechst, in anderen stirbt sie aus (Balkengrafik) */
/* (Schwarze Balken=Gesamtpopulation, bunte Balken=Anteil der Typen) */
/* Mit den eingestellten Parametern waechst sie. */
/*****************************************************************/
```

- **Aufgabe 3.5:**
  Stellen Sie in `life.c` die Gesamtpopulation und die Einzelpopulationen als Funktion der Zeit dar (natürlich müssen Sie statt der Endlosschleife eine begrenzte Gesamtzahl von Durchläufen angeben und vermutlich jeweils über größere Zeitintervalle mitteln, um die Anzahl der Datenpunkte in Grenzen zu halten). Ist die Entwicklung im Norden anders als im Süden?
  Aufgabe 3.5.a: Bauen Sie eine beliebige neue Lebensregel ein, die das Verhalten merklich ändert.

```
/*****************************************************************/
/* kanone.c */
/* */
/* Eine Kanone schiesst Kugeln ab, die nach den Gesetzen des schiefen */
/* Wurf nach parabolischer Bahn in einem Zielfeld auftreffen. */
/* Vertikaler und horizontaler Abschusswinkel sowie die Anfangs- */
/* geschwindigkeit variieren statistisch. Gezeigt sind grafisch die */
/* Auftreffpunkte im Zielfeld. */
/* */
/* Dieses Programm eignet sich als Startpunkt (und schon fast */
/* Loesung) des klassischen Problems der Bestimmung von PI aus der */
/* Verteilung der Auftreffpunkte (unter der Annahme von */
/* Gleichverteilungen, also eines Auftreffquadrats, ist */
/* PI/4 = Anzahl der Auftreffpunkte im Inkreis des Quadrats / Anzahl */
/* der Auftreffpunkte im Quadrat). */
/*****************************************************************/
```

- **Aufgabe 3.6:**
  Die Abschätzung von π mit der in `kanone.c` erwähnten Methode funktioniert natürlich nur gut mit gleichverteilten Auftreffpunkten. Ändern Sie das Programm und die Parameter und wählen Sie ein Quadrat so, daß die Auftreffpunkte in Ihrem Quadrat näherungsweise gleichverteilt sind.

  – Welche Näherungswerte von π erhalten Sie?

  – Wie groß sind Mittelwert und Standardabweichung aus mehreren Durchläufen?

  – Wie ändert sich das Ergebnis, wenn Sie wieder Gaußverteilungen statt der Gleichverteilung verwenden?

Kapitel 4

Lineare Gleichungssysteme

4.1 Allgemeines

In der numerischen Mathematik werden viele Probleme, z.B. die Interpolation von Funktionen, die Lösung von Differentialgleichungen, die Lösung von Integralgleichungen, etc., auf die Lösung eines linearen Gleichungssystem zurückgeführt. Lineare Gleichungssysteme sind mathematisch bestens untersucht und können unter vorgegebenen (meist schwachen) Bedingungen gelöst werden. Die Eigenschaften linearer Gleichungssysteme finden auch in der formalen Mathematik in vielen Beweisen ihre Anwendung.

Ein lineares Gleichungssystem für die Unbekannten $x_1, x_2, \ldots, x_N$ ist von der Form

$$\mathcal{A}\mathbf{x} = \mathbf{b} \quad \Longleftrightarrow \quad \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ A_{M1} & A_{M2} & \ldots & A_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}, \tag{4.1}$$

wobei $\mathbf{x}$ und $\mathbf{b}$ Vektoren der Dimension N sind und $\mathcal{A}$ eine $M \times N$ Matrix ist. Die Matrix $\mathcal{A}$ und der Vektor $\mathbf{b}$ werden als bekannt vorausgesetzt. Man muss nun die folgenden Fälle unterscheiden:

- $M < N$ oder $M = N, \det \mathcal{A} = 0 \rightarrow$ Das Gleichungssystem ist unterbestimmt,
- $M = N, \det \mathcal{A} \neq 0 \rightarrow$ Das Gleichungssystem ist eindeutig lösbar,
- $M > N \rightarrow$ Das Gleichungssystem ist überbestimmt, ausgenommen für den Fall, dass $M - N$ Gleichungen über lineare Abhängigkeiten reduziert werden können.

Im Folgenden beschränken wir uns auf den Fall $M = N$, sodass $\mathcal{A}$ eine quadratische Matrix ist. Ist nun $\det \mathcal{A} = 0$ so spricht man von einem singulären Gleichungssystem, welches nicht eindeutig nach $\mathbf{x}$ aufgelöst werden kann. Nichtsinguläre Gleichungssysteme, d.h. $\det \mathcal{A} \neq 0$, können im Prinzip eindeutig gelöst werden. Bei der numerischen Rechnung hat man aber Schwierigkeiten mit der Genauigkeiten zu erwarten, wenn

$\det \mathcal{A}$ sehr klein wird, bzw. wenn die Eigenwerte von $\mathcal{A}$ eine hohe *Dynamik* aufweisen, d.h. über viele Größenordnungen gehen.

Zur Lösung von linearen Gleichungssystemen gibt es je nach Form von $\mathcal{A}$ eine Vielzahl spezieller Verfahren, welche in zwei Gruppen unterteilt werden können,

a Direkte Verfahren,

b Iterative Verfahren.

Direkte Verfahren liefern in endlich vielen Schritten das exakte Ergebnis, wenn man von Rundungsfehlern absieht. Bei iterativen Verfahren erreicht man das exakte Ergebnis erst nach unendlich vielen Schritten. Im vorliegenden Kapitel beschränken wir uns auf die Darstellung von zwei direkten Verfahren, und zwar (a) den Gaußschen Eliminationsalgorithmus und (b) eine spezielle Methode für tridiagonale Matrizen. Des weiteren wird das Grundkonzept iterativer Verfahren besprochen. Letztere eignen sich vor allem für die Lösung von hochdimensionalen linearen Gleichungssystemen. Eine konkrete Anwendung einer solchen iterativer Methoden erfolgt im Kapitel **??**.

Abschließend sei noch erwähnt, dass diese Verfahren auch für die Bestimmung der inversen Matrix $\mathcal{A}^{-1}$ verwendet werden können, wenn man die N Gleichungssysteme $(i = 1, 2, \ldots, N)$

$$\begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ A_{N1} & A_{N2} & \ldots & A_{NN} \end{pmatrix} \begin{pmatrix} (A^{-1})_{1i} \\ (A^{-1})_{2i} \\ \vdots \\ (A^{-1})_{Ni} \end{pmatrix} = \begin{pmatrix} \delta_{1i} \\ \delta_{2i} \\ \vdots \\ \delta_{Ni} \end{pmatrix} \tag{4.2}$$

löst.

4.2 Das Eliminationsverfahren von Gauß

4.2.1 Der Algorithmus

Beim Gaußschen Eliminationsverfahren zur Lösung eines linearen Gleichungssystems wird durch geeignete Vertauschung und Linearkombination von Zeilen des Gleichungssystem (4.2) die Matrix $\mathcal{A}$ schrittweise auf Dreiecksgestalt gebracht:

$$\begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1N} \\ 0 & r_{22} & \ldots & r_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & r_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} . \tag{4.3}$$

Die Bestimmung der Lösung $\mathbf{x}$ aus einem Gleichungssystem mit Dreiecksgestalt ist sehr einfach. Man erhält:

$$x_i = \frac{1}{r_{ii}} \left(c_i - \sum_{\ell=i+1}^{N} r_{il} x_l \right) \quad , i = N, N-1, \ldots, 1 . \tag{4.4}$$

Die Erzeugung eines äquivalenten linearen Gleichungssystems mit Dreiecksgestalt ist ein elementarer Prozess, welcher in $N-1$ Schritten leicht erreicht werden kann. Zur Illustration des Prozesses zur Umformung des Gleichungssystems betrachten wir den i-ten Schritt,

$$\mathcal{A}^{(i)}\mathbf{x} = \mathbf{c}^{(i)} \quad \Longrightarrow \qquad \mathcal{A}^{(i+1)}\mathbf{x} = \mathbf{c}^{(i+1)}\,, \tag{4.5}$$

wobei $\mathbf{c}^{(i)}$ der aus $\mathbf{b}$ nach i Schritten abgeleitete Vektor der rechten Seite von (4.2) und $\mathcal{A}^{(i)}$ bis zur Zeile i bereits eine obere Dreiecksmatrix ist,

$$\mathcal{A}^{(i)} = \begin{pmatrix} a_{11}^{(i)} & a_{12}^{(i)} & \dots & a_{1i-1}^{(i)} & a_{1i}^{(i)} & a_{1i+1}^{(i)} & \dots & a_{1N}^{(i)} \\ 0 & a_{22}^{(i)} & \dots & a_{2i-1}^{(i)} & a_{2i}^{(i)} & a_{2i+1}^{(i)} & \dots & a_{2N}^{(i)} \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & \dots & a_{i-1i-1}^{(i)} & a_{i-1i}^{(i)} & a_{i-1i+1}^{(i)} & \dots & a_{i-1N}^{(i)} \\ 0 & 0 & \dots & 0 & a_{ii}^{(i)} & a_{ii+1}^{(i)} & \dots & a_{iN}^{(i)} \\ 0 & 0 & \dots & 0 & a_{i+1i}^{(i)} & a_{i+1i+1}^{(i)} & \dots & a_{i+1N}^{(i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{Ni}^{(i)} & a_{Ni+1}^{(i)} & \dots & a_{NN}^{(i)}\,. \end{pmatrix} \tag{4.6}$$

Dabei sind $\mathcal{A}^{(1)} = \mathcal{A}$ und $\mathbf{c}^{(1)} = \mathbf{b}$. Um die entsprechenden Umformungen des i-Schrittes (4.5) durchzuführen, sind drei Teilschritte erforderlich:

a) `Suche eines geeigneten Pivotelementes` r_{ii}.
Hier wollen wir uns auf die Spalten-Pivotsuche beschränken, d.h. man sucht das Element mit dem größten Absolutwert aus der i-ten Teilspalte $a_{ji}^{(i)}, j = i, \dots, N$. Sei dieses das Element der r-ten Zeile, dann setzt man den Pivot $r_{ii} = a_{ri}^{(i)}$. Existiert kein nichtverschwindendes Element in der i-ten Teilspalte, d.h. $a_{ji}^{(i)} = 0\ \forall j, j = i, \dots, N$, so ist die Matrix $\mathcal{A}$ singulär.

b) `Vertauschen der` r`-ten mit der` i`-ten Zeile in` $\mathcal{A}^{(i)}$ `und` $\mathbf{c}^{(i)}$.
Bezeichnet man mit $\bar{\mathcal{A}}^{(i)}$ und $\bar{\mathbf{c}}^{(i)}$ die Matrix $\mathcal{A}^{(i)}$ und den Vektor $\mathbf{c}^{(i)}$ nach Vertauschen der Zeilen, so lautet nun das Gleichungssystem $\bar{\mathcal{A}}^{(i)}\mathbf{x} = \bar{\mathbf{c}}^{(i)}$.

c) `Berechnen der Elemente von` $\mathcal{A}^{(i+1)}$ `und` $\mathbf{c}^{(i+1)}$.
Die unteren $N-i$ Zeilen sind von der Neuberechnung betroffen. Man zieht dabei von der m-ten Zeile ein Vielfaches der i-ten Zeile der Matrix $\bar{\mathcal{A}}^{(i)}$ ab, sodass das Matrixelement $a_{mi}^{(i+1)} = 0$ wird. Dies wird für alle $m = i+1, i+2, \dots, N$ durchgeführt.

- Die Matrixelemente der ersten i Zeilen bleiben unverändert, d.h. $a_{mn}^{(i+1)} = \bar{a}_{mn}^{(i)} = r_{mn}$ und $c_m^{(i+1)} = \bar{c}^{(i)}$ für alle $m = 1, 2, \dots, i$ und $n = 1, 2, \dots, N$.
- Die Transformation der Matrixelemente der übrigen Zeilen $m = i+1, i+2, \dots, N$ erfolgt nach der Vorschrift

$$a_{mn}^{(i+1)} = \bar{a}_{mn}^{(i)} - \lambda_{mi}\bar{a}_{in}^{(i)}\,, \quad n = 1, 2, \dots, N \tag{4.7}$$

mit

$$\lambda_{mi} = \frac{\bar{a}_{mi}^{(i)}}{r_{ii}} \quad \text{für alle } m = i+1, i+2, \ldots, N\,. \tag{4.8}$$

- Die Transformation der übrigen Zeilen des Vektors $\mathbf{c}^{(i)}$ erfolgt nach der Vorschrift

$$c_m^{(i+1)} = \bar{c}_m^{(i)} - \lambda_{mi}\bar{c}_i^{(i)}\,, \quad \text{für alle } m = i+1, i+2, \ldots, N\,. \tag{4.9}$$

Nach $N-1$ Schritten ergibt sich für $\mathcal{A}^{(N)}$ die gewünschte obere Dreiecksmatrix und der entsprechende Vektor $\mathbf{c}^{(N)}$, sodass man mittels (4.4) den Lösungsvektor $\mathbf{x}$ des linearen Gleichungssystems (4.2) erhält. Wie man leicht durch Abzählen der erforderlichen Operationen in diesem Algorithmus sieht, erhöht sich mit steigender Zahl N der Unbekannten der Aufwand mit N^3.

Betrachtet man nun den Algorithmus in seiner Gesamtheit, so stellt man fest, dass die Multiplikationsfaktoren λ_{mn} mit $\lambda_{mm} = 1$ eine untere Dreiecksmatrix $\mathcal{L}$ bilden. Nimmt man keine Zeilenvertauschungen im Rahmen der Pivotsuche vor, dann gilt der Zusammenhang

$$\mathcal{L}\mathcal{R} = \mathcal{A}\,, \tag{4.10}$$

wobei die Bezeichnung $\mathcal{R} = \mathcal{A}^{(N)}$ verwendet wurde. Im Gaußschen Eliminationsverfahren wird also eine Darstellung der Matrix $\mathcal{A}$ als Produkt einer unteren mit einer oberen Dreiecksmatrix realisiert. Ohne Pivotsuche lässt sich daher der Eliminationsalgorithmus kompakt darstellen

$$r_{ik} = a_{ik} - \sum_{j=1}^{i-1} \lambda_{ij} r_{jk} \quad k = i, i+1, \ldots, N\,, \tag{4.11}$$

$$\lambda_{ki} = \frac{1}{r_{ii}} \left(a_{ki} - \sum_{j=1}^{i-1} \lambda_{kj} r_{ji} \right) \quad k = i+1, i+2, \ldots, N\,. \tag{4.12}$$

Zur Berechnung der Matrixelemente r_{ik}, λ_{ik} können die Indices in unterschiedlicher Reihenfolgen durchlaufen werden. Dementsprechend lassen sich zwei unterschiedliche Algorithmen formulieren und zwar jenen nach Banachiewicz und das Verfahren nach Crout. In der Literatur (z.B. NAG Programmbibliothek) gibt es daher stets Hinweise auf den aktuell verwendeten Algorithmus.

Abschließend soll noch festgehalten werden, dass im Gaußschen Eliminationsalgorithmus durch die Zerlegung in Dreiecksmatrizen auch die Determinante der Matrix $\mathcal{A}$ gegeben ist,

$$\det \mathcal{A} = \prod_{i=1}^{N} r_{ii}\,. \tag{4.13}$$

4.2.2 Pivotsuche und Rundungsfehlereinfluss

In Hinblick auf die numerische Genauigkeit, d.h. die Minimierung von Rundungsfehlern, ist die Wahl des Pivot ausschlaggebend. Dies lässt sich leicht an einem kleinem Beispiel demonstrieren.

Abhängigkeit der Genauigkeit von Pivotwahl

Wir betrachten in zweistelliger Gleitpunktrechnung das Gleichungssystem

$$\begin{pmatrix} 0.5\times 10^{-2} & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1.0 \end{pmatrix} \quad \Longrightarrow \text{ exakte Lösung } \begin{matrix} x = \frac{5000}{9950} = 0.503 \\ y = \frac{4950}{9950} = 0.497 \end{matrix}$$

$r_{11} = 0.005$

$$\begin{pmatrix} 0.005 & 1 \\ 0 & -200 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.5 \\ -99.0 \end{pmatrix} \quad \Longrightarrow \text{ exakte Lösung } \begin{matrix} x = \frac{0}{200} = 0.00 \\ y = \frac{99}{200} = 0.50 \end{matrix}$$

$r_{11} = 1$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix} \quad \Longrightarrow \text{ exakte Lösung } \begin{matrix} x = \frac{5}{10} = 0.50 \\ y = \frac{5}{10} = 0.50 \end{matrix}$$

Aus dem Beispiel könnte der falsche Eindruck entstehen, dass die Wahl des größten Elements als Pivot (wie es auch im oben ausgeführten Algorithmus enthalten ist) zu dem geringsten Rundungsfehler führt. Dies ist aber im allgemeinen nicht der Fall, da es auf die Gesamtheit der involvierten Restmatrix ankommt.

Neben der *Teilpivotsuche* oder *Spaltenpivotsuche* wie sie im oben besprochenen Algorithmus verwendet wurde, gibt es noch die *Totalpivotsuche*. Bei letzterer wird das absolut größte Matrixelement der gesamten Restmatrix (gebildet aus i-ter bis N-ter Zeile und Spalte) als Pivotelement r_{ii} verwendet. Die Totalpivotsuche ist allerdings programmtechnisch wesentlich aufwändiger und erfordert nicht nur ein Vertauschen der Zeilen, sondern auch der Spalten. Dies impliziert einen wesentlich höheren Verwaltungsaufwand, da man auch die Reihenfolge der Elemente in den Vektoren $\mathbf{x}$ und $\mathbf{b}$ umstellen und registrieren muss.

Beim Verfahren von *Householder und Schmidt* werden unitäre Matrizen $\mathcal{P}^{(i)}$, $\quad i = 1, 2, \ldots, N-1$ bestimmt, welche die folgenden Transformationen vermitteln

$$\mathcal{P}^{(i)}\begin{pmatrix} a_{ii}^{(i)} \\ a_{i+1i}^{(i)} \\ \vdots \\ a_{Ni}^{(i)} \end{pmatrix} = \begin{pmatrix} r_{ii} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{mit} \quad |r_{ii}| = \sqrt{\sum_{j=i}^{N}\left(a_{ji}^{(i)}\right)^2}. \tag{4.14}$$

In der Householder Transformation sind alle Zeilen der jeweiligen Restmatrix gleichermaßen involviert. Durch die Unitarität der Transformation bleiben die Längen jedes Spaltenvektors konstant und man erhält eine niedrige Konditionszahl des Algorithmus,

d.h. die Rundungsfehler bleiben minimal. Für Details der Transformation von Householder und Schmidt wird auf die Literatur in Numerischer Mathematik verwiesen (siehe z.B. Stoer [11]).

4.3 Lineare Gleichungssysteme mit Tridiagonaler Matrix

Viele Probleme werden auf die Lösung von Gleichungssystemen mit tridiagonaler Matrix

$$\mathcal{A} = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_{N-1N-2} & a_{N-1N-1} & a_{N-1N} \\ 0 & \dots & 0 & 0 & a_{NN-1} & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} \tag{4.15}$$

zurückgeführt. Die spezielle Strukur der Matrix erlaubt eine wesentliche Reduktion der Rechenzeit. Zunächst machen wir den Ansatz einer linearen Rekursionsvorschrift zwischen den Unbekannten

$$x_{i+1} = \alpha_i x_i + \beta_i \,, \tag{4.16}$$

wobei α_i und β_i noch zu bestimmende Größen sind. Einsetzen des Ansatzes (4.16) in das lineare Gleichungssystem (4.15) liefert für die i-te Zeile

$$a_{ii-1}x_{i-1} + a_{ii}x_i + a_{ii+1}x_{i+1} = a_{ii-1}x_{i-1} + a_{ii}x_i + a_{ii+1}(\alpha_i x_i + \beta_i) = b_i. \tag{4.17}$$

Separation von x_i und Vergleich der Koeffizienten liefert die Rekursionsvorschrift für die Koeffizienten α_i und β_i

$$\alpha_{i-1} = -\frac{a_{ii-1}}{a_{ii} + a_{ii+1}\alpha_i} \,, \quad \beta_{i-1} = -\frac{\beta_i a_{ii+1} - b_i}{a_{ii} + a_{ii+1}\alpha_i} \,. \tag{4.18}$$

Den Rekursionsbeginn erhält man aus einer näheren Betrachtung der letzten Zeile

$$a_{NN-1}x_{N-1} + a_{NN} = b_N \quad \Longrightarrow x_N = -\frac{a_{NN-1}}{a_{NN}}x_{N-1} + \frac{b_N}{a_{NN}} = \alpha_{N-1}x_{N-1} + \beta_{N-1} \,. \tag{4.19}$$

Vergleich mit der Rekursionsvorschrift (4.18) zeigt, dass man diese Werte am einfachsten erhält, wenn man $\alpha_N = \beta_N = 0$ verwendet.

Die Lösung des Gleichungssystems mit tridiagonaler Matrix erfordert $2N$ Schritte. Beginnend mit dem Ansatz $\alpha_N = \beta_N = 0$ berechnet man nun α_i und β_i für $i = N-1, N-2, \dots, 1$ über die Rekursionsvorschrift (4.18). Mit der Kenntnis von α_1 und β_1 lässt sich die Unbekannte x_1 berechnen,

$$x_1 = \frac{b_1 - a_{12}\beta_1}{a_{11} + a_{12}\alpha_1} \,. \tag{4.20}$$

Anschließend kann man über die Rekursionsvorschrift (4.16) die Unbekannten $x_2, x_3, \ldots, x_N$ berechnen.

4.4 Iterative Methoden

Das Konzept der iterativen Methoden lässt sich an der einfachen linearen Gleichung $(1-a)x = b$ demonstrieren. Wir versuchen diese Gleichung mit der Iterationsvorschrift

$$x^{(r+1)} = ax^{(r)} + b \quad \text{mit} \quad r = 0, 1, \ldots \tag{4.21}$$

zu lösen, wobei $x^{(r)}$ der Wert der Lösung nach dem r-ten Schritt ist. Geht man von einem beliebigen Startwert $x^{(0)}$ aus, so erhält man durch Einsetzen die Iterationsfolge

$$\begin{aligned} x^{(1)} &= ax^{(0)} + b\,, \\ x^{(2)} &= ax^{(1)} + b = a^2x^{(0)} + ab + b\,, \\ \ldots &= \ldots\ldots\ldots \\ x^{(r+1)} &= ax^{(r)} + b = a^{r+1}x^{(0)} + (a^r + a^{r-1} + \cdots + a + 1)b\,. \end{aligned} \tag{4.22}$$

Ist $|a| < 1$ so gelten die Grenzwerte

$$\lim_{r\to\infty} a^{r+1} = 0 \quad \text{und} \quad \sum_{k=0}^{\infty} a^k = \frac{1}{1-a}\,, \tag{4.23}$$

sodass bei fortlaufender Iteration die Größe $x^{(r)}$ gegen die Lösung der linearen Gleichung

$$(1-a)x = b \quad \Longrightarrow \quad x = \frac{b}{1-a} \tag{4.24}$$

strebt. Allerdings konvergiert das iterative Verfahren nur für $|a| < 1$. Dies erkennt man auch aus der Berechnung des Fehlers des Algorithmus nach r Schritten

$$e^{(r)} = x - x^{(r)} = a^r(x - x^{(0)}) = a^r e^{(0)}\,, \tag{4.25}$$

welcher nur für $|a| < 1$ mit fortlaufender Iteration abnimmt. Bei $|a| \geq 1$ divergiert das über die Vorschrift (4.21) gegebene iterative Verfahren.

Um den Konvergenzbereich der iterativen Verfahren zu vergrößern, modifizieren wir die Iterationsvorschrift (4.21) indem wir eine Gewichtung mit dem jeweils vorhergehenden Iterationsschritt vornehmen

$$x^{(r+1)} = \omega(ax^{(r)} + b) + (1-\omega)x^{(r)} = (1-\omega(1-a))x^{(r)} + \omega b \quad \text{mit} \quad r = 0, 1, \ldots\,. \tag{4.26}$$

Geht man nun von einem beliebigen Startwert $x^{(0)}$ aus, so konvergiert nun diese Iterationsfolge zu

$$x = \frac{\omega b}{1 - [1 - \omega(1-a)]} = \frac{b}{1-a}\,, \tag{4.27}$$

falls $|1-\omega(1-a)|<1$, d.h. wir haben ein konvergentes Iterationsverfahren falls $0<(1-a)\omega<2$ gilt. Die beste Konvergenz ergibt sich, wenn wir für den Relaxationsfaktor ω den Wert

$$\omega=\frac{1}{1-a} \tag{4.28}$$

wählen, denn dann ist der die Konvergenz bestimmende Faktor $c=1-\omega(1-a)=0$ und der erste Iterationsschritt führt bereits zum exakten Wert der Lösung. Es gibt also einen optimalen ω-Wert, bei dem die Anzahl der erforderlichen Iterationsschritte zur Erreichung einer vorgegebenen Genauigkeit minimal wird.

Die Übertragung des Iterationskonzeptes auf lineare Gleichungssysteme ist einfach und soll am Beispiel eines Systems mit drei Unbekannten demonstriert werden,

$$\begin{aligned} a_{11}x_1+a_{12}x_2+a_{13}x_3 &= b_1\\ a_{21}x_1+a_{22}x_2+a_{23}x_3 &= b_2\\ a_{31}x_1+a_{32}x_2+a_{33}x_3 &= b_3 \quad . \end{aligned} \tag{4.29}$$

Um zu einer Iterationsvorschrift zu kommen, separiert man in jeder Gleichung von (4.29) eine Unbekannte, d.h. man erhält ein Gleichungssystem von der Form

$$\begin{aligned} x_1 &= \phantom{c_{21}x_1}+c_{12}x_2+c_{13}x_3+d_1\\ x_2 &= c_{21}x_1\phantom{+c_{12}x_2}+c_{23}x_3+d_2\\ x_3 &= c_{31}x_1+c_{32}x_2\phantom{+c_{23}x_3}+d_3 \end{aligned} \tag{4.30}$$

wobei $c_{ii}=0$, $c_{ij}=-a_{ij}/a_{ii}$ für $i\neq j$ und $d_i=b_i/a_{ii}$ für $i,j=1,2,3$ sind. Wie man aus (4.30) sieht, zerfällt das Gleichungssystem in zwei Teile, einen mit einer linken unteren Matrix $\mathcal{L}$ und einen mit einer rechten oberen Matrix $\mathcal{R}$

$$\mathcal{L}=\begin{pmatrix} 0 & 0 & 0\\ c_{21} & 0 & 0\\ c_{31} & c_{32} & 0 \end{pmatrix} \quad \text{und} \quad \mathcal{R}=\begin{pmatrix} 0 & c_{12} & c_{13}\\ 0 & 0 & c_{23}\\ 0 & 0 & 0 \end{pmatrix} . \tag{4.31}$$

Die Unbekannten x_1,x_2,x_3 fassen wir im Vektor $\mathbf{x}$ und die Komponenten d_1,d_2,d_3 im Vektor $\mathbf{d}$ zusammen.

Mit diesen Definitionen sind wir nun in der Lage vier verschiedene Iterationsverfahren zur Lösung eines linearen Gleichungssystems anzugeben:

a) Das *Gesamtschrittverfahren*, oft als *Methode von Jacobi* oder J-Methode bezeichnet,

$$\mathbf{x}^{(r+1)}=\mathcal{R}\mathbf{x}^{(r)}+\mathcal{L}\mathbf{x}^{(r)}+\mathbf{d}\,, \tag{4.32}$$

b) das *Einzelschrittverfahren*, oft als *Methode von Gauss und Seidel* oder GS-Methode bezeichnet,

$$\mathbf{x}^{(r+1)}=\mathcal{R}\mathbf{x}^{(r)}+\mathcal{L}\mathbf{x}^{(r+1)}+\mathbf{d}\,, \tag{4.33}$$

c) das *Gesamtschrittverfahren mit Überrelaxation*, oft als JOR-Methode bezeichnet, ist im Englischen als *simultaneous overrelaxation method* bekannt,

$$\mathbf{x}^{(r+1)} = \omega \left(\mathcal{R}\mathbf{x}^{(r)} + \mathcal{L}\mathbf{x}^{(r)} + \mathbf{d} \right) + (1-\omega)\mathbf{x}^{(r)} , \tag{4.34}$$

d) das *Einzelschrittverfahren mit Überrelaxation*, oft als SOR-Methode bezeichnet, ist im Englischen als *successive overrelaxation method* bekannt,

$$\mathbf{x}^{(r+1)} = \omega \left(\mathcal{R}\mathbf{x}^{(r)} + \mathcal{L}\mathbf{x}^{(r+1)} + \mathbf{d} \right) + (1-\omega)\mathbf{x}^{(r)} . \tag{4.35}$$

Für die numerische Implementierung sind die GS- und die SOR-Methode angenehmer, da sie keine Zwischenspeicherung der Lösung benötigen; allerdings ist dies meist kein entscheidendes Kriterium für die Auswahl eines bestimmten Verfahrens. Für die Diskussion der Konvergenzeigenschaften wird auf die Standardliteratur aus Numerischer Mathematik verwiesen.

4.5 Einbinden von Bibliotheksprogrammen

Die Lösung von linearen Gleichungssystemen ist eine der wichtigsten Problemstellungen im Zusammenhang mit der numerischen Behandlung physikalischer Fragen. Die verschiedenen Facetten solcher Gleichungssysteme erfordern zum Teil spezielle Algorithmen, um auch die geforderten Genauigkeiten zu erreichen. Es ist daher im Allgemeinen sinnvoll für die Lösung von Problemen der linearen Algebra auf sogenannte Programmbibliotheken zurückzugreifen, die spezielle und ausgereifte Algorithmen enthalten. Beispiel für eine allgemein zugängliche Programmbibliothek ist die CERNLIB, welche insbesondere die für die numerische Physik relevanten Algorithmen enthält.

4.6 Aufgaben

In diesem Kapitel geben wir nur Aufgaben zur direkten Lösung von linearen Gleichungssystemen, welche eine Überprüfung der entsprechenden Subroutinen zu den Algorithmen der Unterkapitel 4.2 und 4.3 an konkreten Beispielen ermöglichen.

- **Aufgabe 4.1:** Man schreibe ein Unterprogramm `SUBROUTINE GAUSS(N,A,N1,B,X)` zur Lösung eines linearen Gleichungssystems $\mathcal{A}\mathbf{x} = \mathbf{b}$ mit N komplexwertigen Unbekannten mittels des Gaußschen Eliminationsalgorithmus und Spaltenpivotsuche. `N1` ist die erste Dimensions des Feldes A(N1,N)

  a) Überprüfen Sie die ordnugsgemäße Funktion des Programmes.

  b) Lösen Sie das Gleichungssystem mit einem geeigneten Bibliotheksprogramm.

- **Aufgabe 4.2:** Man schreibe ein Unterprogramm `SUBROUTINE TRIDI(N,ABAND,N1,B,X)` zur Lösung eines linearen Gleichungssystems mit tridiagonaler Matrix. Die Werte der Bandmatrix sind in dem zweidimensionalen Feld `ABAND(N1,-1:1)` in folgender Weise gespeichert: a_{ii-1} =`ABAND(I,-1)`, a_{ii} =`ABAND(I,0)` und a_{ii+1} =`ABAND(I,+1)`.

  a) Überprüfen Sie die ordnugsgemäße Funktion des Programmes.

  b) Lösen Sie das Gleichungssystem mit einem geeigneten Bibliotheksprogramm.

Kapitel 5

Interpolation, Differentiation und Integration

5.1 Interpolation

5.1.1 Motivation und Klassifikation

Ein zentrales Thema bei der numerischen Behandlung von physikalischen Problemen stellt die Wahl einer geeigneten Interpolation zur Darstellung von Funktionen im Computer dar. Die Motivation für die Verwendung von Interpolationen ist vielfältig, z.B.

- die Zahl der Messpunkte ist gering,
- der Aufwand zur Berechnung eines Datenpunktes ist sehr hoch, d.h. man versucht möglichst wenige Punkte zu berechnen,
- die numerische Methode verlangt ein bestimmtes Gitter, die Mess- oder Rechenpunkte liegen aber auf einem anderen Gitter vor.

Diese Problem treten sowohl in der Auswertung von Messreihen als auch bei numerischen Rechnungen im Rahmen einer Theorie oder vorgegebenen Modells auf. Insbesondere im Zusammenhang mit graphischen Darstellungen kommt der Interpolation eine besondere Bedeutung zu.

Der Ausgangspunkt für eine Interpolation ist stets ähnlich:

- Gegeben ist eine unabhängige Variable x und für eine Menge von Werten $\{x_i\}$ aus einem Intervall $\mathcal{I}$ sind Funktionswerte $f(x_i) = f_i$ bekannt. Gesucht ist eine Funktion $\bar{f}(x)$ (eventuell geschlossener Ausdruck), sodass für beliebiges x aus $\mathcal{I}$ mit $\bar{f}(x)$ eine sinnvolle Näherung von $f(x)$ berechnet werden kann.
- Ziel der Interpolation ist nicht nur die Verwendung von $\bar{f}(x)$ zur Berechnung von Funktionswerten $f(x)$ mit $x_0 \in \mathcal{I}$ (Interpolation), sondern $\bar{f}(x)$ dient auch als Basis für die näherungsweise Integration, Differentiation und in einigen Fällen zur Extrapolation, d.h. für die Näherungen von Funktionswerten außerhalb des Interpolationsintervalls $\mathcal{I}$.

Die entsprechende mathematische Problemstellung lässt sich kompakt wie folgt formulieren

Interpolationsproblem:

Eine Funktion $f(x)$ sei an $N+1$ unterschiedlichen Argumentwerten $x_i,\ i = 0, 1, \ldots, N$, $x_i \neq x_k$ gegeben, $f_i = f(x_i)$. Ein Interpolationsproblem liegt dann vor, wenn für eine vorgegebene Funktionenklasse $\phi(x; a_0, a_1, \ldots, a_N)$ die Parameter a_i so bestimmt werden sollen, dass

$$\phi(x_i; a_0, a_1, \ldots, a_N) = f_i \quad \forall\ i = 0, 1, \ldots, N.$$

Die Paare (x_i, f_i) werden als Stützstellen bezeichnet.

Eine besondere Klasse sind lineare Interpolationsprobleme. Bei diesen hängt die Interpolationsfunktion $\phi(x; a_0, a_1, \ldots, a_N)$ linear von den Parametern a_i ab,

$$\phi(x; a_0, a_1, \ldots, a_N) = \bar{\phi}(x) + a_0\bar{\phi}_0(x) + a_1\bar{\phi}_1(x) + \cdots + a_N\bar{\phi}_N(x)\,. \tag{5.1}$$

Die in dieser Lehrveranstaltung behandelten Interpolationstechniken, (a) die Polynominterpolation und (b) die Spline-Interpolation, gehören beide der Klasse der linearen Interpolationsprobleme an. Beispiele für nichtlineare Interpolationsprobleme sind die Interpolation mit rationalen Funktionen sowie die Interpolation mit Exponentialsummen.

5.1.2 Polynominterpolation

Bei der Polynominterpolation wählt man die Interpolationsfunktion aus Π_N, der Menge der reellen oder komplexen Polynome $p(x)$ vom Grad $p \leq N$,

$$\phi(x; a_0, a_1, \ldots, a_N) = p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N\,. \tag{5.2}$$

Das Interpolationsproblem besteht nun darin, dass man zu $(N+1)$ beliebig vorgegebenen Stützpunkten $(x_i, f_i),\ i = 0, 1, 2, \ldots, N$ und $x_i \neq x_k$ für $i \neq k$ ein Polynom $p(x) \in \Pi_N$ bestimmt, welches an den $(N+1)$ Stützstellen mit der vorgegebenen Funktion übereinstimmt, d.h. $p(x_i) = f_i$ für $i = 0, 1, 2, \ldots, N$.

Es lässt sich nun zeigen, dass ein eindeutig bestimmtes Polynom $p(x)$ existiert, welches dieses Interpolationsproblem löst. Die Eindeutigkeit erhält man einfach über einen indirekten Beweis.

Beweis der Eindeutigkeit

Annahme es existieren zwei Polynome $p_1, p_2 \in \Pi_N$ für die gilt:
$p_1(x_i) = p_2(x_i) = f_i$ für alle $i = 1, 2, \ldots, N$
daraus folgt, dass das Polynom $p(x) = p_1(x) - p_2(x) \in \Pi_N$ zumindest $N + 1$ Nullstellen [$p(x_i) = 0,\ i = 0, 1, \ldots, N$] hat. Da $p \in \Pi_N$ nur maximal N Nullstellen haben kann, folgt $p(x) \equiv 0$ und $p_1(x) \equiv p_2(x)$.

Die Existenz eines Polynoms $p(x)$, welches das Interpolationsproblem löst, ist durch die *Lagrange Interpolationsformel*

$$p(x) = \sum_{i=0}^{N} f_i L_i(x) \tag{5.3}$$

gegeben, wobei $L_i(x)$ die *Lagrangeschen Interpolationspolynome* sind

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{N} \frac{x - x_k}{x_i - x_k} \quad \text{für} \quad i = 0, 1, \ldots, N\,. \tag{5.4}$$

Es ist offensichtlich, dass alle $L_i(x) \in \Pi_N$ sind und der Eigenschaft genügen $L_i(x_k) = \delta_{ik}$. Die Lagrangesche Interpolationsformel liefert also das gewünschte Interpolationspolynom, ein darauf aufgebauter Algorithmus benötigt allerdings relativ viele arithmetische Operationen, wie sich aus dem nachstehenden Beispiel vermuten lässt.

Beispiel einer Polynominterpolation:

Vier Datenpunkte $\{x_i, f(x_i)\}$, $i = 0, \ldots 3$, sind gegeben. Gesucht ist ein Polynom dritten Grades, das durch diese Datenpunkte geht. Gemäß der Lagrangeschen Interpolationsformel lautet das gesuchte Polynom

$$p_3(x) = \frac{(x-x_1)\ (x-x_2)\ (x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} f(x_0) + \frac{(x-x_0)\ (x-x_2)\ (x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} f(x_1)$$

$$+ \frac{(x-x_0)\ (x-x_1)\ (x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} f(x_2) + \frac{(x-x_0)\ (x-x_1)\ (x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} f(x_3).$$

Da die zu interpolierenden Funktionen selbst meist keine Polynome sind, wird bei der Interpolation ein Fehler gemacht. Sei $p_N(x)$ das Interpolationspolynom vom Grad N, dann kann man für den Fehler, $E(x) = f(x) - p_N(x)$, folgenden Ausdruck herleiten

[12]:

$$E(x) = f(x) - p_N(x) = \prod_{i=0}^{N}(x - x_i)\frac{f^{(N+1)}(\xi)}{(N+1)!} \tag{5.5}$$

mit $\xi \in [x_0, x_N]$.

Wie bereits oben erwähnt, ist die Lagrangesche Interpolationsformel für die praktische Rechnung ungeeignet. Da relativ viele arithmetische Operationen erforderlich sind, ist sie außerdem noch anfällig für Rundungsfehler. Je nachdem ob man die Funktion an einem oder mehreren Argumentwerten interpolieren will, bieten sich verschiedene Algorithmen an.

Der Algorithmus von Neville

Will man die interpolierte Funktion nur an einem zusätzliche Argumentwert, so bietet sich der Algorithmus von Neville zur Lösung des Interpolationsproblems an. Der Algorithmus ist flexibler bezüglich der Zahl der Datenpunkte und aufgrund der rekursiven Formulierung leichter implementierbar als die Lagrangesche Interpolationsformel. Natürlich erhält man aufgrund der Eindeutigkeit auch die *gleichen* Polynome wie bei der Anwendung der Lagrangeschen Interpolationsformel.

Wir nehmen an, dass die Funktion an $(N+1)$ Stützstellen $(x_i, f_i),\ i = 0, 1, \ldots, N$ gegeben ist. Wir definieren nun Polynome $P_{i_0 i_1 \ldots i_k} \in \Pi_k$ mit der Eigenschaft

$$P_{i_0 i_1 \ldots i_k}(x_{i_j}) = f_{i_j}\,, \quad j = 0, 1, \ldots, k\,. \tag{5.6}$$

Ausgehend von den Polynomen nullten Grades

$$P_i(x) = f_i \quad \forall\, x\,, \tag{5.7}$$

können wir mit der Rekursionsvorschrift

$$P_{i_0 i_1 \ldots i_k}(x) = \frac{(x - x_{i_0})P_{i_1 i_2 \ldots i_k}(x) - (x - x_{i_k})P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}} \tag{5.8}$$

den interpolierten Wert der Funktion an der Stelle x ermitteln,

$$p_N(x) = P_{i_0 i_1 \ldots i_N}(x)\,. \tag{5.9}$$

Der Algorithmus von Neville lässt sich in einem Schema übersichtlich darstellen und numerisch leicht implementieren. Als Beispiel betrachten wir das Schema für $N = 3$

	$k=0$	$k=1$	$k=2$	$k=3$
x_0	$f_0 = P_0(x)$			
		$P_{01}(x)$		
x_1	$f_1 = P_1(x)$		$P_{012}(x)$	
		$P_{12}(x)$		$P_{0123}(x)$
x_2	$f_2 = P_2(x)$		$P_{123}(x)$	
		$P_{23}(x)$		
x_3	$f_3 = P_3(x)$			

Man rechnet dabei spaltenweise und schreitet dann zur höheren Ordnung. Der Algorithmus von Aitken ist sehr ähnlich zu dem Algorithmus von Neville, führt aber auf ein unsymmetrisches Schema.

Newtonsche Interpolationsformel

Das Verfahren von Neville (und Aitken) sind nicht geeignet, wenn man eine Interpolation der Funktion $f(x)$ an mehreren Punkten benötigt, bzw. wenn man das Polynom selbst bestimmen will. Für diese Problemstellung eignet sich der Algorithmus von Newton, welcher auf der Basis "Dividierter Differenzen" arbeitet.

Gegeben sind die Stützstellen $(x_i, f(x_i))$, $i = 0, \ldots N$, mit $f(x_i) = f_i$. Die Dividierte Differenz (*divided difference*) erster Ordnung zwischen zwei Stützstellen x_i und x_j ($i \neq j$) ist nun folgend definiert

$$f[x_i, x_j] = \frac{f_i - f_j}{x_i - x_j} = f[x_j, x_i]. \tag{5.10}$$

Ausgehend von der Dividierten Differenz nullter Ordnung, $f[x_i] = f_i$, können nun rekursiv Dividierte Differenzen höherer Ordnung definiert werden,

$$f[x_0, x_1, \ldots, x_i] = \frac{f[x_1, \ldots, x_i] - f[x_0, x_1, \ldots, x_{i-1}]}{x_i - x_0}. \tag{5.11}$$

Das gesuchte Interpolationspolynom $p_N(x)$, welches durch die $(N + 1)$ Stützstellen $(x_i, f(x_i))$ geht, ist gegeben durch

$$p_N(x) = a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + \cdots + (x - x_0)(x - x_1)\ldots(x - x_{N-1})a_N, \tag{5.12}$$

wobei man zeigen kann [12], dass die Koeffizienten a_i durch

$$a_i = f[x_0, x_1, \ldots, x_i]. \tag{5.13}$$

gegeben sind. Die rekursive Bestimmung der $f[x_0, x_1, \ldots, x_i]$ gemäß der Vorschrift (5.11) und somit der a_i, macht eine sehr einfache numerische Bestimmung der Polynomkoeffizienten möglich.

Im Programmbeispiel (`PROGRAM div_dif`) sind folgende fünf Datenpunkte $\{x_i, f(x_i)\}$, $i = 1, \ldots, 5$ gegeben (und durch eine `DATA`-Anweisung am Beginn des Hauptprogrammes initialisiert; sie können dort leicht durch andere Datensätze ausgetauscht werden):

x_i	f_i
3.2	22.0
2.7	17.8
1.0	14.2
4.8	38.3
5.6	51.7

Der rekursive Algorithmus (5.11) kann leicht in ein symmetrisches Schema gebracht werden:

i	x_i	f_i	$f[x_i, x_{i+1}]$	$f[x_i, \ldots, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
1	3.2	22.0				
			8.400			
2	2.7	17.8		2.856		
			2.118		-0.5280	
3	1.0	14.2		2.012		0.256
			6.342		0.0865	
4	4.8	38.3		2.263		
			16.750			
5	5.6	51.7				

Man nennt dies ein Differenzenschema nach Newton, bei dem die Dividierten Differenzen spaltenweise und dann von links nach rechts fortschreitend bestimmt werden. Haben die Stützstellen x_i gleichen Abstand, so ergeben sich wesentliche Vereinfachungen.

5.1.3 'Spline'-Interpolation

Motivation

Die Polynominterpolation kann bei großer Dynamik der zu interpolierenden Funktion zu starken Oszillationen neigen. Wir wollen diesen Effekt an einer einfachen Stufenfunktion $f(x)$ demonstrieren,

$$f(x) = \frac{1}{1 + \exp(x/a)} \quad \text{mit} \quad a = 0.02\,, \tag{5.14}$$

welche durch fünf Stützstellen gegeben ist.

i	1	2	3	4	5
x_i	-1.0	-0.1	0.0	0.1	1.0
f_i	1.000000	0.993307	0.500000	0.006693	0.000000

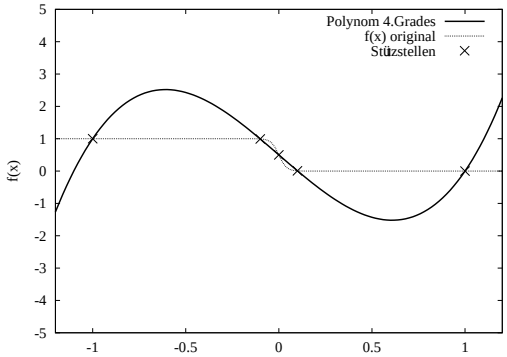

Abbildung 5.1: Interpolation der durch 5 Stützstellen (Sterne) gegebenen Stufenfunktion (5.14) mit einem Polynom 4. Grades (dicke Linie). Die Stufenfunktion ist als dünne punktierte Linie eingezeichnet.

Führen wir nun eine Polynominterpolation durch, so erhalten wir die in Abb. 5.1 dargestellte Interpolationskurve, welche beträchtliche Überschwingungen in dem an und für sich glatten Funktionsverlauf aufweist.

Erhöht man nun die Anzahl der Stützstellen um zwei (vier) weitere Punkte bei $x = \pm 0.3$ (und $x = \pm 0.6$) und führt die Interpolation mit einem Polynom 6. bzw. 8. Grades durch, so verbessert sich zwar das Gesamtbild (Abb. 5.2), die Oszillationen bleiben aber nach wie vor erhalten. Überdies erkennt man die rasch ansteigenden Fehler bei Extrapolation.

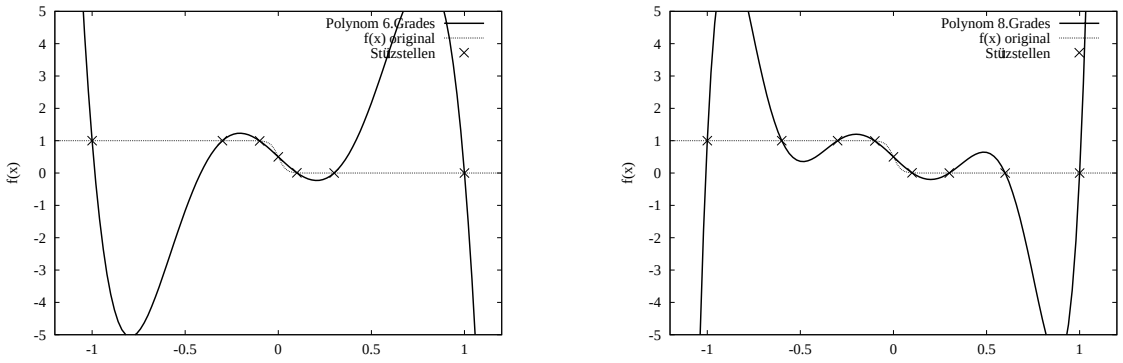

Abbildung 5.2: Interpolationspolynome (jeweils dicke Linie) vom Grad 6 (links) und vom Grad 8 (rechts) durch die mit Kreuzen gekennzeichneten Stützstellen. Die Stufenfunktion ist als dünne punktierte Linie eingezeichnet.

Führt man allerdings die Interpolation in den Teilintervallen zwischen den Stützstellen, $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, N-1$ durch Polynome niedrigen Grades unter der Forderung der Stetigkeit der niedrigsten Ableitungen durch ('Spline'-Interpolation), so erhält man bereits bei der Verwendung von nur 7 Stützstellen eine akzeptable Interpolation (siehe Abb. 5.3).

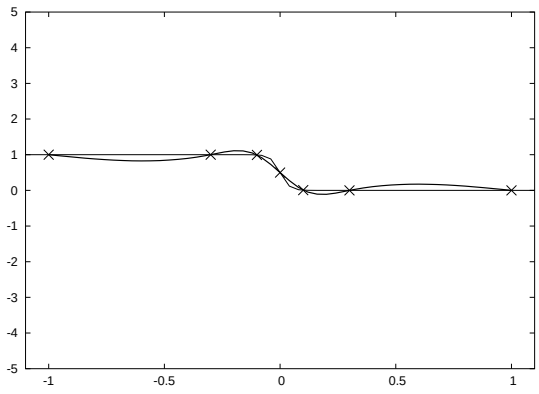

Abbildung 5.3: Interpolation der durch 7 Stützstellen (Kreuze) gegebenen Stufenfunktion (5.14) mit einer kubischen 'Spline'-Funktion (dicke Linie). Die Stufenfunktion ist als dünne punktierte Linie eingezeichnet.

Formalismus

Bei der Interpolation einer durch $N+1$ Stützstellen $(x_i, f_i),\ x_{i+1} > x_i, i = 0, 1, \ldots, N$ vorgegebenen Funktion möchte man im Allgemeinen eine möglichst glatte Funktion durch die Stützstellen legen. Wie man aus dem obigen Beispiel sieht, kann die Polynominterpolation aufgrund der starren Eigenschaften der Polynome zu unerwünschten Überschwingungen führen. Die Grundidee der 'Spline'-Interpolation besteht nun in der Suche einer glatten Funktion $s(x)$ durch die Stützstellen, d.h. das Integral

$$||s(x)||^2 \equiv \int_{x_0}^{x_N} dx|s''(x)|^2 \tag{5.15}$$

sollte für die interpolierende Funktion $s(x)$ minimal werden. Die Forderung (5.15) nimmt Anleihe in der Statik, wo die Deformation von elastischen Trägern unter Belastung berechnet werden. Die 'Spline'-Interpolation kann daher als jene Kurve angesehen werden, die gleichmäßig belastete und an den Stützpunkten aufgelegte elastische Latten (*Splines*) durch Verformung annehmen.

Für kubische 'Spline'-Funktionen $s(x)$ gilt nun die *Minimum-Norm* Eigenschaft

$$||f - s||^2 = ||f||^2 - ||s||^2 \geq 0\,, \tag{5.16}$$

falls **eine** der folgenden zusätzlichen Bedingungen erfüllt ist

$$\text{(a)} \quad s''(x_0) = s''(x_N) = 0\,, \tag{5.17}$$

$$\text{(b)} \quad s(x), f(x) \text{ sind periodisch, d.h. } s(x_0) = s(x_N) \text{ und } s'(x_0) = s'(x_N)\,, \tag{5.18}$$

$$\text{(c)} \quad f'(x_0) = s'(x_0) \quad \text{und} \quad f'(x_N) = s'(x_N)\,. \tag{5.19}$$

In jedem dieser Fälle ist die 'Spline'-Funktion eindeutig gegeben. Der Beweis dieser Aussage ist direkt aus der Definition der Norm einer 'Spline'-Funktion (5.15) über partielle Integration ableitbar. Unter der Annahme, dass die 'Spline'-Funktion $s(x)$ das Interpolationsproblem löst und zweimal stetig differenzierbar auf dem Intervall $[x_0, x_N]$ ist, erhält man die Beziehung,

$$||f - s||^2 = ||f||^2 - ||s||^2 - 2\left\{\left[f'(x_N) - s'(x_N)\right] s''(x_N) - \left[f'(x_0) - s'(x_0)\right] s''(x_0)\right\}\,. \tag{5.20}$$

Aus dieser lassen sich sofort die alternativen Bedingungen (5.17)-(5.19) ablesen.

Beweis der Eindeutigkeit der 'Spline'-Funktion

Annahme es existieren zwei kubische 'Spline'-Funktionen $s_1(x), s_2(x)$, welche das Interpolationsproblem lösen, d.h. $||s_1 - s_2||^2 = ||s_2 - s_1||^2 \geq 0 \Longrightarrow ||s_1 - s_2||^2 = 0$
Aus der Stetigkeit von $s_1(x)$ und $s_2(x)$ folgt
$||s_1 - s_2||^2 = \int_{x_0}^{x_N} dx(s_1'' - s_2'')^2 = 0 \Longrightarrow s_1''(x) = s_2''(x)$
d.h. $s_1(x) = s_2(x) + cx + d$. Da $s_1(x)$ und $s_2(x)$ das gleiche Interpolationsproblem lösen, folgt aus $s_1(x_0) = s_2(x_0) = f(x_0)$ und $s_1(x_N) = s_2(x_N) = f(x_N)$,
dass $c = d = 0$ und damit $s_1(x) \equiv s_2(x)$.

Für die Durchführung der 'Spline'-Interpolation muss man zunächst annehmen, dass die 'Spline'-Funktion auf dem Interval $[x_0, x_N]$ zweimal stetig differenzierbar ist. Um diese Bedingung unter Beibehaltung großer Flexibilität zu erfüllen, wird die (Standard) 'Spline'-Funktion in den Teilintervallen $[x_i, x_{i+1}]$ als kubisches Polynom definiert, dessen Funktionswerte, die erste und zweite Ableitung an den Intervallgrenzen mit den entsprechenden Werte der Polynome in den benachbarter Intervallen übereinstimmen.

Aufstellung der Gleichungen für die Spline Interpolation

Für $x \in [x_i, x_{i+1}]$ wird für die interpolierende Funktion $s(x)$ ein kubisches Polynom mit unbekannten Koeffizienten a_i, b_i, c_i und d_i angenommen,

$$s(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i\,, \tag{5.21}$$

das an den Intervallgrenzen die Funktionswerte f_i und f_{i+1} annimmt, d.h.

$$\begin{aligned} f_i &= d_i \\ f_{i+1} &= a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i. \end{aligned} \tag{5.22}$$

Dabei wurde die Intervallbreite $h_i = x_{i+1} - x_i$, $i = 0, 1, \ldots, N-1$ eingeführt. Für die N Teilintervalle hat man also $4N$ Koeffizienten festzulegen. Die Reproduktion der Funktionswerte (5.22) liefert $2N$ Gleichungen. Die verbleibenden $2N$ Koeffizienten müssen aus der Forderung der Stetigkeit der ersten und der zweiten Ableitung an den Intervallgrenzen abgeleitet werden.

Bezeichnen also M_i und M_{i+1} die zweiten Ableitungen des Polynoms an den Stellen x_i und x_{i+1}, so ergibt sich aus (5.21)

$$M_i = 2b_i \quad \text{und} \quad M_{i+1} = 6a_i h_i + 2b_i\,. \tag{5.23}$$

Mit diesen Ausdrücken lassen sich folgende Gleichungen für die unbekannten Koeffizienten a_i und b_i aufstellen,

$$b_i = \frac{1}{2}M_i \quad \text{und} \quad a_i = \frac{1}{6h_i}(M_{i+1} - M_i)\,. \tag{5.24}$$

Die letzten noch unbestimmten Koeffizienten c_i ergeben sich aus Gleichung (5.22)

$$c_i = \frac{f_{i+1} - f_i}{h_i} - h_i\frac{2M_i + M_{i+1}}{6}\,. \tag{5.25}$$

Der letzte Schritt besteht in der Bestimmung der Werte der zweiten Ableitungen von $s(x)$ an den Stützstellen. Die dazu erforderlichen Gleichungen erhält man aus der Stetigkeit der ersten Ableitung an den Intervallgrenzen. Setzt man den links- und rechtsseitigen Grenzwert für die erste Ableitung an den Intervallgrenzen gleich, so ergeben sich $N-1$ Gleichungen der Form

$$y_i' = c_i = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}\,. \tag{5.26}$$

Substitution der Ausdrücke für a_i, b_i, c_i und d_i führt zu dem linearen Gleichungssystem für die unbekannten Werte M_i, $i = 1, 2, \ldots, N-1$,

$$\frac{f_{i+1} - f_i}{h_i} - h_i \frac{2M_i + M_{i+1}}{6} = 3\left(\frac{M_i - M_{i-1}}{6h_{i-1}}\right) h_{i-1}^2 + 2\left(\frac{M_{i-1}}{2}\right) h_{i-1} \tag{5.27}$$

$$+\frac{f_i - f_{i-1}}{h_{i-1}} - h_{i-1}\frac{2M_{i-1} + M_i}{6}\,. \tag{5.28}$$

Zusammenfassung der Terme in M_i führt auf die einfache Form

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = \delta_i \quad \text{für} \quad i = 1, 2, \ldots, N-1\,, \tag{5.29}$$

mit

$$\lambda_i = \frac{h_i}{h_{i-1} + h_i}\,, \quad \mu_i = \frac{h_{i-1}}{h_{i-1} + h_i} = 1 - \lambda_i \tag{5.30}$$

und

$$\delta_i = \frac{6}{h_{i-1} + h_i}\left\{\frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}\right\}\,. \tag{5.31}$$

Insgesamt gibt es $(N-1)$ derartiger Gleichungen für die $(N+1)$ Unbekannten M_i. Es werden also noch zwei weitere Gleichungen über M_0 und M_N benötigt. Diese erhält man aus einer der Bedingungen (5.17)-(5.19).

ad a) *Natürlicher 'Spline':* $M_0 = 0$ und $M_N = 0$,
Damit reduzieren sich die Gleichungen für $i = 1$ und $i = N-1$ auf

$$2M_1 + \lambda_1 M_2 = \delta_1 \quad \text{und} \quad \mu_{N-1} M_{N-2} + 2M_{N-1} = \delta_{N-1}\,, \tag{5.32}$$

sodass ein Gleichungssystem mit tridiagonaler Matrix für den Vektor $M_1, M_2, \ldots, M_{N-1}$ resultiert.

ad b) *Periodische 'Spline'-Funktion:* Die Periodizität von $s(x)$ liefert die Beziehungen $M_0 = M_N$ und aus $s'(x_0) = s'(x_N)$ folgt

$$\mu_N M_{N-1} + 2M_N + \lambda_N M_1 = \delta_N \quad \text{und} \quad \mu_1 M_N + 2M_1 + \lambda_1 M_2 = \delta_1 \tag{5.33}$$

mit

$$\lambda_N = \frac{h_0}{h_{N-1} + h_0} \quad \text{und} \quad \delta_N = \frac{6}{h_{N-1} + h_0}\left[\frac{f_1 - f_0}{h_0} - \frac{f_N - f_{N-1}}{h_{N-1}}\right]\,. \tag{5.34}$$

ad c) *Ableitungen am Rand gegeben:* Für die Randwerte ergeben sich die beiden zusätzlichen Gleichungen,

$$2M_0 + \lambda_0 M_1 = \delta_0 \quad \text{und} \quad \mu_N M_{N-1} + 2M_N = \delta_N \tag{5.35}$$

mit

$$\delta_0 = \frac{6}{h_0}\left[\frac{f_1 - f_0}{h_0} - f'(x_0)\right]\,, \quad \lambda_0 = 1\,, \tag{5.36}$$

$$\delta_N = \frac{6}{h_{N-1}}\left[f'(x_N) - \frac{f_N - f_{N-1}}{h_{N-1}}\right]\,, \quad \mu_N = 1\,. \tag{5.37}$$

$$\tag{5.38}$$

Man erhält also wieder ein Gleichungssystem mit tridiagonaler Matrix allerdings für den Vektor $M_0, M_1, \ldots, M_N$.

Liegt die Lösung des Gleichungssystems vor, sind also alle M_i bekannt, so werden die unbekannten Koeffizienten der Interpolations Polynome über die Beziehungen (5.22), (5.24) und (5.25) berechnet,

$$a_i = \frac{1}{6h_i}(M_{i+1} - M_i) \qquad b_i = \frac{1}{2}M_i\,, \qquad d_i = f_i\,, \tag{5.39}$$

$$c_i = \frac{f_{i+1} - f_i}{h_i} - h_i\frac{2M_i + M_{i+1}}{6}\,. \tag{5.40}$$

Die Durchführung der 'Spline'-Interpolation erfordert also die Lösung des linearen Gleichungssystems für die M_i , $i = 0, 1, \ldots, N$. Soll die 'Spline'-Interpolation unter der Bedingung (5.17) oder (5.19) erfolgen, so handelt es sich um ein tridiagonales System, welches mit dem in Kapitel 4.3 besprochenen Verfahren gelöst werden kann. Die entsprechenden Programme enthalten allerdings noch weitere Möglichkeiten für die Bestimmung von M_0 und M_N, die im Programmbeispiel durch den Eingabeparameter `iend` unterschieden werden.

- `iend=1`: $M_0 = 0$ und $M_N = 0$ (Natürlicher 'Spline').
- `iend=2`: $M_0 = M_1$ und $M_N = M_{N-1}$.
- `iend=3`: M_0 ist ein aus M_1 und M_2 linear extrapolierter Wert; ebenso ist M_N ein aus M_{N-1} und M_{N-2} linear extrapolierter Wert.
- `iend=4`: Schätzwerte für die ersten Ableitungen an den beiden Enden des Interpolationsintervalls werden vorgegeben.

Die geeignete Annahme muss für jedes Interpolationsproblem speziell betrachtet werden. Man muss allerdings noch festhalten, dass die Fälle `iend=2` und `iend=3` nicht zur Minimum-Norm der 'Spline'-Interpolation führen.

Im Fall einer periodischen 'Spline'-Funktion (5.18) nimmt das zu lösende Gleichungssystem die Form

$$\begin{pmatrix} 2 & \lambda_1 & 0 & \ldots & 0 & 0 & \mu_1 \\ \mu_2 & 2 & \lambda_2 & \ldots & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & & \mu_{N-1} & 2 & \lambda_{N-1} \\ \lambda_N & 0 & 0 & \ldots & 0 & \mu_N & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ \vdots \\ M_{N-1} \\ M_N \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \vdots \\ \vdots \\ \delta_{N-1} \\ \delta_N \end{pmatrix}. \tag{5.41}$$

Abschließend soll noch festgehalten werden, dass die Werte f_i der zu interpolierenden Funktion nur in die Größen d_i eingehen, die Matrix des Gleichungssystems aber nur von den gewählten Teilintervallen abhängt.

5.2 Differentiation

5.2.1 Motivation

Die Aufgabe besteht in der numerischen Bestimmung der (höheren) Ableitungen einer (unbekannten) Funktion, die durch einen Satz von gegebenen Datenpunkten $\{x_i, f(x_i)\}$ beschrieben ist; dies ist z.B. bei der Lösung von Differentialgleichungen von zentraler Bedeutung.

5.2.2 Ableitungen über Interpolationspolynome

Unter der Annahme, dass eine Funktion hinreichend gut durch ihr Interpolationspolynom approximiert wird (siehe Abschnitt 5.1), kann man hoffen, dass diese Näherung auch für die erste Ableitung angewendet werden kann. Es wird sich allerdings zeigen, dass der Fehler in der ersten Ableitung größer ist, als bei den Funktionswerten.

Ist das Interpolationspolynom z.B. durch die Newtonsche Interpolationsformel bestimmt [vergleiche Unterkapitel 5.1.2], also

$$\begin{aligned} f(x) &= p_N(x) + E(x) \\ &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots \\ &\quad + f[x_0, x_1, \ldots, x_N](x - x_0)(x - x_1) \cdots (x - x_{N-1}) + E(x) \end{aligned} \tag{5.42}$$

so ergibt sich für $f'(x)$

$$\begin{aligned} f'(x)) &= f[x_0, x_1] + f[x_0, x_1, x_2](2x - x_0 - x_1) + \cdots \\ &\quad + f[x_0, x_1, \ldots, x_N] \sum_{k=0}^{N-1} \frac{(x - x_0) \cdots (x - x_{N-1})}{(x - x_k)} + \tilde{E}(x). \end{aligned} \tag{5.43}$$

Die Abschätzung für $\tilde{E}(x)$, den Fehler in $f'(x)$, ist komplizierter als jene von $E(x)$. Unter gewissen Annahmen lassen sich geschlossene Ausdrücke für diese Fehler angeben [12] aus denen ersichtlich ist, dass $E(x)$ größer ist als $\tilde{E}(x)$.

Höhere Ableitungen von $f(x)$ lassen sich formal leicht aus den obigen Ausdrücken herleiten.

5.2.3 Ableitungen bei gleichmäßig verteilten Datenpunkten

Sind die Datenpunkte auf einem regelmäßigen Gitter vorgegeben (äquidistante Datenpunkte), also $x_i = x_0 + ih$, $i = 1, \ldots, N$, so können folgende Ausdrücke für die ersten und zweiten Ableitungen von $f(x)$ angegeben werden:

erste Ableitungen:

$$f'(x_0) = \frac{f_1 - f_0}{h} + O(h) \tag{5.44}$$

$$f'(x_0) = \frac{f_1 - f_{-1}}{2h} + O(h^2) \tag{5.45}$$

$$f'(x_0) = \frac{-f_2 + 4f_1 - 3f_0}{2h} + O(h^3) \tag{5.46}$$

$$f'(x_0) = \frac{-f_2 + 8f_1 - 8f_{-1} + f_{-2}}{12h} + O(h^4); \tag{5.47}$$

zweite Ableitungen:

$$f''(x_0) = \frac{f_2 - 2f_1 + f_0}{h^2} + O(h) \tag{5.48}$$

$$f''(x_0) = \frac{f_1 - 2f_0 + f_{-1}}{h^2} + O(h^2) \tag{5.49}$$

$$f''(x_0) = \frac{-f_3 + 4f_2 - 5f_1 + 2f_0}{h^2} + O(h^3) \tag{5.50}$$

$$f''(x_0) = \frac{-f_2 + 16f_1 - 30f_0 + 16f_{-1} - f_{-2}}{12h^2} + O(h^4)\,. \tag{5.51}$$

Durch Indextransformation in den obigen Ausdrücken können Ableitungen für jeden der N Datenpunkte angegeben werden; zu beachten ist, dass bei Randpunkten einige der Ausdrücke mangels vorhandener Funktionswerte nicht verwendet werden können. Genauere Angaben für die Fehler sind in [13] zu finden.

5.3 Integration

5.3.1 Newton-Côtes Integrationsalgorithmen

In diesem Teilabschnitt werden nur numerische Integrationsalgorithmen, sogenannte Quadraturen, vorgestellt, die von Sätzen von Datenpunkten $\{x_i, f(x_i)\}$ mit äquidistanten Gitterpunkten (x_i-Werten) ausgehen. Die grundlegende Idee dieser Quadraturalgorithmen ist der Ersatz des an den Gitterpunkten gegebenen Integranden durch ein Interpolationspolynom, das dann exakt integriert werden kann. Auf diese Weise erhält man einen Näherungswert für das Integral über die durch die Datenpunkte dargestellte Funktion.

Man kann die numerischen Integrationslagorithmen auch vom Standpunkt aus sehen, dass zwar die explizite Form der zu integrierenden Funktion bekannt ist, aber keine Stammfunktion angegeben werden kann und somit die (bestimmte) Integration numerisch durchgeführt werden muss. Abgesehen von der numerischen Integration sind Quadraturalgorithmen auch im Zusammenhang mit der Lösung von Differentialgleichungen von Bedeutung.

Gegeben ist eine Funktion $f(x)$, die über ein Intervall $\mathcal{I} = [a, b]$ integriert werden soll, also

$$I = \int_a^b dx\ f(x)\,. \tag{5.52}$$

Sei die Funktion im Intervall $\mathcal{I} = [a, b]$ an $N + 1$ äquidistanten Gitterpunkten, $x_i = a + i(b - a)/N$, $i = 0, 1, \ldots, N$ gegeben, so lässt sich mittels der Lagrangeschen

Interpolationspolynome die Funktion darstellen und integrieren,

$$I = \int_a^b dx\ f(x) \approx \sum_{i=0}^{N} f_i \int_a^b dx L_i(x) = \sum_{i=0}^{N} f_i \alpha_i \,. \tag{5.53}$$

Die Koeffizienten α_i sind Integrale über die Lagrangeschen Interpolationspolynome, welche analytisch durchgeführt werden können und nur von der Intervallteilung und nicht von der Funktion abhängen. Je nach Größe von N erhält man auf diese Art die Grundformeln für die Trapezregel, die Simpsonregel, etc. Obwohl man Ausdrücke für beliebig großes N ableiten kann, ist die Verwendung der Newton-Côtes Formeln aus Genauigkeitsgründen nur bis $N = 6$ sinnvoll. Um ein Integral über einen größeren Bereich zu integrieren, führt man eine Summe über mehrere Teilintervalle durch. Man nimmt wieder einen Satz von äquidistanten Gitterpunkten, $x_0(= a), \ldots, x_n(= b)$ mit $x_i = x_0 + ih$ an, die durch eine konstante Schrittweite h voneinander getrennt sind. Die Funktionswerte an den Gitterpunkten (Stützstellen) werden mit $f_i = f(x_i)$ bezeichnet. Je nachdem, ob man die Funktion $f(x)$ zwischen den Stützstellen durch Polynome ersten, zweiten oder dritten Grades ersetzt, nimmt man verschiedene Teilintervalle an und erhält folgende Summenausdrücke

$$\int_a^b dx\ f(x) = \frac{h}{2}\left(f_0 + 2f_1 + 2f_2 + \cdots 2f_{n-1} + f_n\right) - \frac{b-a}{12} h^2 f''(\xi), \tag{5.54}$$

$$\int_a^b dx\ f(x) = \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 \cdots 2f_{n-2} + 4f_{n-1} + f_n\right) - \frac{b-a}{180} h^4 f^{IV}(\xi), \tag{5.55}$$

$$\int_a^b dx\ f(x) = \frac{3h}{8}\left(f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + 3f_5 \cdots 3f_{n-1} + f_n\right) - \frac{b-a}{80} h^4 f^{IV}(\xi). \tag{5.56}$$

Die jeweils letzten Terme in den obigen Gleichungen erlauben eine Fehlerabschätzung; es gilt jeweils $\xi \in [a, b]$.

Die erste Integrationsregel, Gleichung (5.54), entspricht der summierten Trapezregel; die Zahl der Stützstellen, $n + 1$, ist beliebig. Bei der zweiten Regel, der Simpson-Regel, muss n durch zwei teilbar sein, bei der dritten Regel, der Simpson 3/8 Regel, muss n durch drei teilbar sein.

5.3.2 Gauß Integration

Formeln (5.54), (5.55) und (5.56) zeigen, daß sich bei Wahl äquidistanter Stützstellen die Näherungsformeln für die Integrale durch Summen über die Funktionswerte an diesen Stützstellen, multipliziert mit geeigneten Vorfaktoren (Gewichtsfaktoren) schreiben lassen. Die Gauß Integration erweitert diese Idee folgendermaßen: geht man von den äquidistanten Stützstellen ab, läßt man also die Wahl der n Stützstellen offen, so hat man insgesamt $2n$ freie Parameter (n Stützstellen und die n dazugehörigen Gewichtsfaktoren), die man im Sinne einer Optimierung der Näherungsformeln wählen kann. Man könnte somit z.B. ein Polynom vom Grad $(2n - 1)$ durch die zu integrierende

Funktion legen. Die Näherungsformeln, die auf dieser Idee basieren werden Gauß Integrationsformeln genannt; sie können allerdings nur dann verwendet werden, wenn die zu integrierende Funktion $f(x)$ explizit bekannt ist.

Das Prinzip der Gauß Integration wird im folgenden für $n = 2$ veranschaulicht. Gegeben sei das Integral $\int_a^b dx\ f(x)$, wobei a und b endliche Werte haben sollen. Nach einer einfachen Variablensubstitution,

$$x = \frac{b-a}{2}t + \frac{b+a}{2}, \tag{5.57}$$

erfolgt die Integration über das symmetrische Intervall [-1,+1]. Der Ansatz von Gauß lautet also

$$\int_{-1}^{1} dt\ f(t) = a_1 f(t_1) + a_2 f(t_2) \tag{5.58}$$

mit unbekannten Gewichtsfaktoren a_1 und a_2 und unbekannten Stützstellen t_1 und t_2. Da diese Formel für Polynome bis zum Grad $(2n-1) = 3$ gelten muß, können diese vier unbekannten Parameter aus folgenden Gleichungen bestimmt werden:

$$\begin{aligned}
f(t) = t^3 \quad &: \quad \int_{-1}^{1} dt\ t^3 = 0 = a_1 t_1^3 + a_2 t_2^3, \\
f(t) = t^2 \quad &: \quad \int_{-1}^{1} dt\ t^2 = 2/3 = a_1 t_1^2 + a_2 t_2^2, \\
f(t) = t^1 \quad &: \quad \int_{-1}^{1} dt\ t = 0 = a_1 t_1 + a_2 t_2, \\
f(t) = 1 \quad &: \quad \int_{-1}^{1} dt = 2 = a_1 + a_2\,.
\end{aligned} \tag{5.59, 5.60}$$

Lösung dieser vier Gleichungen führt auf

$$a_1 = a_2 = 1 \qquad t_2 = -t_1 = \sqrt{\frac{1}{3}} = 0.5773 \tag{5.61}$$

und somit

$$\int_{-1}^{1} dt\ f(t) = f(-0.5773) + f(0.5773). \tag{5.62}$$

Ist $f(t)$ ein beliebiges Polynom dritten Grades, so ist diese Formel – konstruktionsgemäß – exakt.

Die Formeln für die Gauß-Integration lassen sich für beliebiges n erweitern: sie lauten dann

$$\int_{-1}^{1} dt f(t) = \sum_{i=1}^{n} w_i f(t_i)\,, \tag{5.63}$$

wobei die $2n$ Unbekannten w_i und t_i, $i = 1, \ldots, n$, aus den Gleichungen

$$\sum_{i=1}^{n} w_i t_i^k = \begin{cases} 0 & k = 1, 3, \ldots, 2n-1 \\ \frac{2}{k+1} & k = 0, 2, \ldots, 2n-2 \end{cases} \tag{5.64}$$

bestimmt werden. Es zeigt sich, dass die gesuchten t_i, $i = 1, \ldots, n$, die Nullstellen des Legendre-Polynoms $L_n(t)$ vom Grad n sind und dass sich die w_i, $i = 1, \ldots, n$, mit Hilfe von

$$w_i = \frac{\langle L_{n-1} | L_{n-1} \rangle}{L_{n-1}(t_i) L_n'(t_i)} \tag{5.65}$$

berechnet werden. Die Legendre-Polynome lassen sich am geeignetsten durch folgende rekursive Relation definieren,

$$(n+1)L_{n+1}(t) - (2n+1)tL_n(t) + nL_{n-1}(t) = 0 \tag{5.66}$$

mit

$$L_0(t) = 1 \qquad L_1(t) = t. \tag{5.67}$$

Die Legendre-Polynome sind auf $[-1, 1]$ orthogonal, d.h. es gilt

$$\int_{-1}^{1} L_n(t) L_m(t) dt = \begin{cases} 0 & n \neq m \\ > 0 & n = m \end{cases} . \tag{5.68}$$

Konkret ergibt sich $L_2(t)$ aus (5.66) und (5.67) zu

$$L_2(t) = \frac{3}{2}t^2 - \frac{1}{2} \tag{5.69}$$

und

$$w_i = [L_1(t_i) L_2'(t_i)]^{-1} = [t_i \; 3t_i]^{-1} . \tag{5.70}$$

Aus (5.69) sieht man leicht, dass die Nullstellen von $L_2(t)$ tatsächlich durch $t_i = \pm\sqrt{1/3}$ gegeben sind; setzt man diese Werte in (5.70) ein, so erhält man $w(t_i) = 1$ – vgl(5.61).

In der nachfolgenden Tabelle sind die Legendre-Polynome $L_3(t)$ und $L_4(t)$, ihre jeweiligen Nullstellen t_i und die Werte der Gewichtsfaktoren, $w(t_i)$, angegeben.

n	$L_n(t)$	Nullstellen t_i	Gewichtsfaktoren $w(t_i)$
2	$\frac{3}{2}t^2 - \frac{1}{2}$	-0.5773502692	1.0000000000
		0.5773502692	1.0000000000
3	$\frac{1}{2}(5t^3 - 3t)$	-0.7745966692	0.5555555556
		0.0000000000	0.8888888889
		0.7745966692	0.5555555556
4	$\frac{1}{8}(35t^4 - 30t^2 + 3)$	-0.8611363116	0.3478548451
		-0.3399810436	0.6521451549
		0.3399810436	0.6521451549
		0.8611363116	0.3478548451

Ganz allgemein kann die Gauß Integration für Integrale vom Typ $\int_a^b dx\ W(x)f(x)$ verwendet werden. Die Näherungsformel lautet dann

$$\int_a^b dx\ W(x)f(x) \sim \sum_{i=1}^{n} w_i f(x_i). \tag{5.71}$$

Dieser Ausdruck ist exakt, wenn $f(x)$ ein Polynom ist, dessen Grad kleiner oder gleich $(2n-1)$ ist. $W(x)$ heißt in der Literatur Gewichtsfunktion.

Die Berechnung der Gewichte w_i erfolgt mit Hilfe der zu $W(x)$ assoziierten orthogonalen Polynome $p_i(x)$. Ist $W(x)$ und ein Intervall $\mathcal{I} = [a, b]$ gegeben, so heißen Polynome $p_i(x)$, $i = 0, 1, 2 \ldots$, orthonormiert bezüglich $W(x)$ auf $\mathcal{I}$, wenn gilt

$$\int_a^b dx\ p_i(x)p_j(x)W(x) = \delta_{ij}. \tag{5.72}$$

Beginnend mit $p_0(x) = 1$ lassen sich diese Polynome mit Hilfe eines Gram-Schmidt Verfahrens konstruktiv bestimmen, was numerisch nicht immer ganz einfach ist. Für spezielle Gewichtsfunktionen $W(x)$ lassen sich die orthogonalen Polynome explizit angeben: z.B. sind für $W(x) = 1$ und $\mathcal{I} = [-1, 1]$ die assoziierten orthogonalen Polynome eben die oben erwähnten Legendre-Polynome $L_i(x)$.

Man kann nun folgendes ableiten: ist $\int_a^b dx\ W(x)f(x)$ näherungsweise zu berechnen, so gilt:

$$\int_a^b dx W(x)f(x) \sim \sum_{i=1}^{n} w_i f(x_i) \tag{5.73}$$

wobei die x_i die Nullstellen des zu $W(x)$ assoziierten Polynoms $p_n(x)$ vom Grad n sind und die Gewichte gegeben sind durch

$$w_i = \frac{\langle p_{n-1}|p_{n-1}\rangle}{p_{n-1}(x_i)p_n'(x_i)}\,. \tag{5.74}$$

Die Gauß-Integration erfolgt also bei vorgegebener Gewichtsfunktion $W(x)$ und Intervall $\mathcal{I}$ in zwei Stufen:

(i) Bestimmung der zu $W(x)$ assoziierten orthonormalen Polynome $p_i(x)$;

(ii) Wahl von n, Bestimmung der Nullstellen von $p_n(x)$ und Berechnung der Gewichte w_i.

5.4 Aufgaben

- **Aufgabe 5.1: 'Spline'-Interpolation**
  Ausgehend vom 'Spline'-Interpolationsprogramm, das Ihnen zur Verfügung gestellt wird, implementieren Sie jenen Programmteil, der eine periodische 'Spline'-Funktion [vgl. Formel (5.18)] $s(x)$ berechnet. Nehmen Sie als Beispiel die Funk-

tion $f(x) = x^3 - 8$; definieren Sie fünf Datenpunkte als Stützstellen (sh. `DATA`-Anweisung) und berechnen Sie die 'Spline'-Funktion für die drei im Skriptum diskutierten Möglichkeiten. Stellen Sie die Datenpunkte und die 'Spline'-Funktionen mithilfe von `gnuplot` graphisch dar.

- **Aufgabe 5.2: Ableitungen**
  Betrachten Sie die Funktion $\ln x$, berechnen Sie analytisch die erste und zweite Ableitung. Schreiben Sie ein `FORTRAN`-Programm, in dem Sie die Formeln (5.44) – (5.47) und (5.48) – (5.51) implementieren. Betrachten Sie den x-Bereich]0,3] berechnen Sie dort mit diesem Programm die erste und zweite Ableitung und vergleichen Sie das Ergebnis mit den analytischen Ausdrücken (graphische Darstellung mit `gnuplot`). Untersuchen Sie die Ergebnisse, wenn Sie die Schrittweite h variieren.

- **Aufgabe 5.3: Newton-Côtes-Quadratur**
  Schreiben Sie ein `FORTRAN`-Programm, das die Integrationsformeln nach Newton-Côtes, (5.54) – (5.56) implementiert. Im jeweiligen Programmsegment soll vor der expliziten Berechnung der Formel abgefragt werden, ob die Auflagen an die Stützstellenzahl erfüllt sind. Testen Sie das Programm am Integral $\int_0^\pi dx\ \mathrm{e}^x \cos x$ und vergleichen Sie die Ergebnisse (für verschiedene Werte von h) mit dem exakten Ergebnis.

- **Aufgabe 5.4: Gauß-Quadratur**
  Implementieren Sie die Integrationsformeln nach Gauß unter Verwendung der im Skriptum angegebenen Formeln für $n = 2, 3, 4$. Testen Sie die Ergebnisse am Integral $\int_0^\pi dx\ \mathrm{e}^x \cos x$ durch Vergleich mit dem exakten Ergebnis. Zeigen Sie mit Hilfe selbstgewählter Polynome, dass die Gauß-Integration exakte Ergebnisse für Polynome bis zum Grad (2n-1) liefert.

Kapitel 6

Nullstellenbestimmung und Anpassungsprobleme

6.1 Nullstellenbestimmung

Im vorangegangenen Kapitel wurde die Frage der numerischen Auswertung von Integralen behandelt. Dabei hat sich im Falle der Gauss-Quadratur

$$\int_{x_1}^{x_2} f(x)\,dx = \sum_{j=1}^{N} w_j f(x_j) \tag{6.1}$$

die Notwendigkeit der Bestimmung der Nullstellen der den Berechnungen zugrundeliegenden Polynome ergeben.

Wir werden uns daher diesem numerischen Problem zuwenden und die gängigsten Methoden vorstellen. Die Vorgangsweise wird in jedem Fall sehr ähnlich sein und kann wie folgt zusammengefasst werden:

1. Man finde einen Bereich, der (zumindest) eine Nullstelle enthält
2. Man wähle eine geeignete Methode der Nullstellenbestimmung.

6.1.1 'Bracketing' und Bisektionsverfahren

Die überwiegende Zahl von Nullstellen lässt sich durch den Vorzeichenwechsel der Funktion $f(x)$ an x_0 charakterisieren. Ausnahmen bilden komplexe Nullstellen (z.B. $x^2 + a = 0$), doppelt zu zählende Nullstellen (z.B. $x^2 = 0$) oder Singularitäten der Form $f(x) = 1/g(x)$ mit ungerader Funktion $g(x)$. *Es gibt keine Methode, die für eine unbekannte Funktion f(x) mit Sicherheit eine Nullstelle finden könnte.* Es ist daher ratsam, sich vor Beginn der Erstellung eines Programms mit Form und Art der zu untersuchenden Funktionen vertraut zu machen.

Den Vorzeichenwechsel von $f(x)$ macht man sich beim 'Bracketing' zunutze, indem man, ausgehend von einem vorgegebenen Ausgangspunkt x_i, durch sukzessive

Vergrößerung des Intervalls $[x_i, x_f]$ das erste x_f bestimmt, für das die Bedingung

$$f(x_i) \cdot f(x_f) \leq 0 \tag{6.2}$$

erfüllt ist. Anschließend kann man unter Wahrung der Bedingung (6.2) das Intervall wieder verkleinern und dadurch die (eine der) eingeschlossene(n) Nullstelle(n) auf beliebige Genauigkeit bestimmen (Bisektionsverfahren).

6.1.2 Newton-Raphson-Verfahren

Ist die analytische Form der ersten Ableitung von $f(x)$ bekannt, so empfiehlt sich die Verwendung des Newton-Raphson-Verfahrens. Es basiert auf der Taylorreihenentwicklung von $f(x)$ in der Nähe der Nullstelle $x_0 = x + \Delta x$

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{f''(x)}{2}\Delta x^2 + \dots . \tag{6.3}$$

Wegen $f(x + \Delta x) = 0$ folgt (unter Vernachlässigung aller Terme $O(\Delta x^2)$) $\Delta x = -f(x)/f'(x)$. Die Iterationsvorschrift für x_i lautet daher

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{6.4}$$

Das Newton-Raphson-Verfahren konvergiert quadratisch gegen die Nullstelle, d.h. mit jedem Iterationsschritt verdoppelt sich die Zahl der signifikanten Stellen. Ist die analytische Form der ersten Ableitung von $f(x)$ nicht bekannt und muss durch ein Differenzenverfahren bestimmt werden, so verschlechtert sich die Effizienz dramatisch. In diesem Fall empfiehlt sich die Verwendung anderer Verfahren, die ohne Berechnung der Ableitung auskommen.

Nachteil des Newton-Raphson-Verfahrens: Da in Gleichung (6.3) alle Terme $O(\Delta x^2)$ vernachlässigt werden, kann es bei Startwerten, die weit von der Nullstelle entfernt sind, zu unvorhersagbaren Ergebnissen kommen. Liegen Extrema von $f(x)$ zwischen dem Startwert x_i und der Nullstelle x_0 vor, so können numerische Instabilitäten (wegen $f'(x) \to 0$) auftreten.

6.1.3 Sekanten-Verfahren

Drei eng miteinander verwandte Verfahren begnügen sich mit der Auswertung der Funktion an zwei bzw. drei Werten für x, um eine iterative Annäherung an x_0 zu finden.

Das Sekanten-Verfahren und die *regula-falsi*-Methode nähern das Verhalten von $f(x)$ in der Umgebung der Nullstelle durch die Sekante, welche durch die Funktionswerte an den Grenzen von $[x_{i-1}, x_i]$ definiert ist. Der Schnittpunkt der Sekante mit der x-Achse definiert den neuen Näherungswert x_{i+1}, der entweder den Grenzwert x_{i-1} (Sekantenverfahren) ersetzt oder das Suchintervall so umdefiniert, dass Bedingung (6.2) erfüllt bleibt (*regula-falsi*).

Die Konvergenz von *regula-falsi* ist oft besser als linear, führt aber sicher zum Erfolg. Im Sekantenverfahren muss die Möglichkeit der Divergenz aufgrund des lokalen Verhaltens der Funktion $f(x)$ gegen die bessere Konvergenz ($\propto \epsilon^{1.618}$) abgewogen werden.

Eine interssante Variante der *regula-falsi*-Methode, die Sicherheit und Schnelligkeit ($\propto \epsilon^{\sqrt{2}}$) verbindet, ist die *Nullstellenbestimmung nach Ridder*. Zunächst wird f nicht nur an den Stellen x_{i-1} und x_i, sondern auch in der Mitte des Intervalls $x_m = (x_{i-1} + x_i)/2$ berechnet. Nach Abspalten eines Exponentialfaktors (für Details der Ableitung siehe z.B. [15]) erhält man für den neuen Wert x_{i+1}

$$x_{i+1} = x_m + (x_m - x_{i-1}) \frac{\operatorname{sgn}[f(x_{i-1}) - f(x_i)] f(x_m)}{\sqrt{f(x_m)^2 - f(x_{i-1}) f(x_i)}}. \tag{6.5}$$

6.1.4 Nullstellenbestimmung nach Brent

Garantierte Konvergenz bei gutem Konvergenzverhalten bietet das Verfahren nach Brent, das, grob gesprochen, eine Kombination eines modifizierten Sekantenverfahrens und eingeschobener Bisektionsschritte darstellt. Ausgehend von den drei Zahlenpaaren $[x_a, f(x_a)]$, $[x_b, f(x_b)]$ und $[x_c, f(x_c)]$, wobei x_b die bisher beste Annäherung an x_0 sei, errechnet sich der neue Näherungswert als

$$x = x_b + \frac{S[T(R-T)(x_c - x_b) - (1-R)(x_b - x_a)]}{(T-1)(R-1)(S-1)}. \tag{6.6}$$

Die Hilfsgrößen S, R und T sind durch

$$S = f(x_b)/f(x_a), \quad R = f(x_b)/f(x_c), \quad T = f(x_a)/f(x_c) \tag{6.7}$$

bestimmt. Ist man mit dem Iterationsergebnis nicht zufrieden, ersetzt man es durch das Resultat eines Bisektionsschrittes.

6.1.5 Nullstellen von Polynomen

Bekannterweise haben Polynome vom Grad n ebensoviele Nullstellen und können als Produkt

$$P_n(x) = (x - x_1)(x - x_2) \dots (x - x_n), \quad x_n \in \mathbb{C} \tag{6.8}$$

angeschrieben werden. Da komplexe Nullstellen paarweise auftreten ($x_0 = a \pm ib$), ist leicht einsichtig, dass Polynome ungeraden Grades zumindest eine reelle Nullstelle aufweisen. Nachdem diese abgespalten wurde (Polynomdivision, siehe unten), muss die Nullstellensuche auf die komplexe Zahlenebene ausgedehnt werden.

Bei *Mullers Methode* handelt es sich um eine komplexe Verallgemeinerung der Sekantenmethode, wobei aber eine quadratische statt der linearen Interpolation angewandt wird. Ausgehend von drei x-Werten (können äquidistant auf der reellen Achse liegen) kommt folgende Iterationsformel zur Anwendung:

$$x_{i+1} = x_i - (x_i - x_{i-1}) \frac{2C}{B \pm \sqrt{B^2 - 4AC}}, \tag{6.9}$$

wobei die Hilfsgrößen q, A, B und C durch

$$\begin{aligned} q &= \frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}} \\ A &= qP_n(x_i) - q(1+q)P_n(x_{i-1}) + q^2 P_n(x_{i-2}) \\ B &= (2q+1)P_n(x_i) - (1+q)^2 P_n(x_{i-1}) + q^2 P_n(x_{i-2}) \\ C &= (1+q)P_n(x_i) \end{aligned} \tag{6.10}$$

gegeben sind. Das Vorzeichen in Gleichung (6.9) wird so gewählt, dass der Betrag des Nenners maximal wird.

Ein weiteres Verfahren wurde von *Laguerre* formuliert, der mithilfe einer gewagten Abschätzung für den Abstand der tatsächlichen Nullstellen vom derzeit untersuchten Punkt x ein Iterationsschema generiert. Zunächst definiert man wiederum zwei Hilfsgrößen

$$\frac{d \ln |P_n(x)|}{dx} = \sum_{i=1}^{n} \frac{1}{x - x_i} = \frac{P_n'}{P_n} = G(x) \tag{6.11}$$

$$-\frac{d^2 \ln |P_n(x)|}{dx^2} = \sum_{i=1}^{n} \frac{1}{(x - x_i)^2} = \left(\frac{P_n'}{P_n}\right)^2 - \frac{P_n''}{P_n} = H(x)\ . \tag{6.12}$$

Unter der Annahme, dass die Nullstelle x_1 einen Abstand a, alle anderen Nullstellen aber Abstand b von x haben, können die Ausdrücke für G und H wesentlich vereinfacht werden

$$\frac{1}{a} + \frac{n-1}{b} = G \tag{6.13}$$

$$\frac{1}{a^2} + \frac{n-1}{b^2} = H \tag{6.14}$$

Damit ergibt sich für a die Abschätzung

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}} \tag{6.15}$$

welche zum neuen Iterationswert $x_{i+1} = x_i - a$ führt. Auch in Gl. (6.15) wird der Nenner maximiert. Zunächst konvergiert das Laguerre Verfahren immer zu einer der reellen Nullstellen. Hat das Polynom keine reele Nullstelle, so liefert der Algorithmus, meist ohne Probleme, eine der komplexen Nullstellen.

6.2 Auffinden mehrerer/aller Nullstellen von Polynomen

Eine einfache Methode, die für reelle Nullstellen funktioniert, besteht darin, dass man das zu untersuchende Intervall in n Unterintervalle teilt und diese auf einen Vorzeichenwechsel der Funktion innerhalb deren Grenzen untersucht. Danach werden durch eine

der oben beschriebenen Methoden in den gefundenen Unterintervallen die Nullstellen von $f(x)$ bestimmt.

Aus Polynomen können Nullstellen auch abdividiert werden. Dadurch wird ein Polynom geringeren Grades erzeugt und die Prozedur wiederholt. Im Falle eines komplexen Paares von Nullstellen dividiert man durch

$$[x - (a + ib)][x - (a - ib)] = x^2 - 2ax + (a^2 + b^2). \tag{6.16}$$

Dabei ist zu beachten, daß die gefundenen Nullstellen nur als gute Näherung an die tatsächlichen Nullstellen des Polynoms betrachtet werden sollten, da sich numerische Fehler mit jeder Division aufaddieren. Alle gefundenen Nullstellen sollten daher nach Beendigung der Routine „nachkorrigiert“ werden.

- Einschub: einfache algebraische Manipulation von Polynomen:

  Für ein Polynom vom Grade $n-1$, das durch $P_{n-1} = \sum_{i=1}^{n} c_i x^{i-1}$ (die Koeffizienten sind im Array `c(1:n)` gespeichert, `c(1)` ist demnach der konstante Term des Polynoms) gegeben ist, führt man eine Multiplikation mit dem Faktor $(x-a)$ durch einen Programmteil durch, der

  1. c_n zu c_{n+1} macht und
  2. eine Zeile enthält, die den neuen Koeffizienten c_j durch $c_{j-1} - c_j * a$ berechnet (von $j = n$ beginnend).

  Nehmen Sie ein Blatt Papier und überprüfen Sie's!

  Will man das Polynom P_{n-1} durch $(x-a)$ dividieren, sind folgende Rechenschritte erforderlich:

  1. c_n in einem Zwischenspeicher $ZS1$ ablegen und $c_n = 0$ setzen (der Grad des Polynoms wird um eins erniedrigt)
  2. von $j = n-1$ beginnend c_j in $ZS2$ ablegen und auf $c_j = ZS1$ setzen. Danach wird $ZS1$ durch $ZS1 = ZS2 + ZS1 * a$ neu berechnet.

  Auch wenn das recht kompliziert klingt, so genügen doch auch hier Papier und Bleistift, um sich an einem einfachen Beispiel von der Korrektheit des Algorithmus zu überzeugen.

Der Vollständigkeit halber sei noch die Möglichkeit erwähnt, die Nullstellen eines Polynoms durch die Berechnung der Eigenwerte einer Matrix **A** zu bestimmen. Sie ist definiert durch die Gleichung

$$P_n(x) = \det[\mathbf{A} - x\mathbb{1}] \tag{6.17}$$

Für Polynome hohen Grades und auf für lineare Algebra optimierten Computern kann sich dies als die schnellste und stabilste Methode herausstellen.

6.3 χ^2-Anpassung

Bevor wir zur Beschreibung der Anpassungsmethoden kommen, sollten ein paar Worte darüber gesagt werden, worum es bei einer Anpassung eigentlich geht. Es ist nicht das Ziel einer Anpassung, durch eine Reihe von Datenpunkten eine möglichst glatte Kurve zu legen, um die graphische Darstellung zu optimieren, sondern es handelt sich bei der χ^2-Anpassung um eine Methode zu entscheiden, wie gut das physikalische Bild, das man sich von einem Prozesss gemacht hat, die gemessene Realität erklären kann. *Eine Anpassung alleine kann Ihnen niemals eine Erklärung für die gemessenen Daten liefern.* Zwei Beispiele:

- Sie untersuchen den radioaktiven Zerfall einer Ihnen unbekannten Substanz. Da Sie wissen, dass Zerfallsprozesse durch eine abklingende Exponentialkurve beschrieben werden können, passen Sie eine Funktion der Form $c \cdot e^{-\Gamma t}$ an Ihre Datenpunkte an, ermitteln daraus z.B. die Halbwertszeit des Materials und bestimmen mithilfe einer Datensammlung das untersuchte Element.

- Sie haben eine Experiment durchgeführt, für dessen Ergebnis Ihnen zwei mögliche Erklärungen einfallen. Eine Anpassung kann Ihnen die Wahl zwischen den beiden Modellen erleichtern, kann Ihnen aber auch sagen, dass keines der beiden Modelle in der Lage ist, die gemessenen Daten zufriedenstellend zu erklären.

Hat man also einen Satz von N Datenpunkten (x_i, y_i) gemessen und hat man die Vermutung, dass die Daten durch eine Funktion mit M Parametern

$$y(x) = y(x; a_1 \dots a_M) \tag{6.18}$$

beschrieben werden können, so sollte die Anpassungprozedur folgende Informationen liefern:

1. die Parameter a_i,

2. eine Abschätzung über den Fehler jedes einzelnen Parameters und

3. ein statistisches Maß für die Qualität der angepassten Funktion.

Geben Sie sich niemals mit dem ersten Punkt alleine zufrieden, oftmals täuscht die graphische Darstellung das Auge über die Unzulänglichkeit des zugrundegelegten Modells hinweg (eine geschickt gewählte Darstellung der Daten hilft oft zu verschleiern, dass man keine Ahnung hat, was man gemessen oder gerechnet hat!).

Der intellektuelle Sprung, den man wagen muss, liegt in der Umkehrung der Frage. Sie lautet nun: Wie wahrscheinlich ist es, einen bestimmten Satz von Parametern $(a_1 \dots a_M)$ im Laufe meiner Anpassung zu finden? Die Aufgabe besteht also darin, jenen Punkt im Parameterraum zu finden, an dem die Wahrscheinlichkeitsverteilung ihr Maximum hat. Dies alles lässt sich in Formeln ausdrücken, und Bücher lassen sich darüber schreiben (eine brauchbare und leicht verständliche Einführung in das Thema bietet z.B. [15]). Hier interessiert uns nur das Ergebnis, das in der Aufgabe resultiert, die Funktion

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; a_1 \dots a_M)}{\sigma_i} \right)^2 \tag{6.19}$$

zu minimieren. Die Parameter σ_i in Gleichung (6.19) bezeichnen die individuelle Standardabweichungen zu jedem Messpunkt (x_i, y_i).

Kennt man σ_i nicht und muss/kann man von einer für alle Datenpunkte identischen Standardabweichung ausgehen, setzt man zunächst alle σ_i auf eins, minimiert Gl. (6.19) und errechnet aus dem Ergebnis σ mittels

$$\sigma^2 = \frac{\sum_{i=1}^{N} [y_i - y(x_i; a_1 \dots a_M)]^2}{N - M}. \tag{6.20}$$

Numerisch gesehen geht es bei einer χ^2-Anpassung darum, die Nullstellen der M Ableitungen von Gl. (6.19) nach den Parametern a_k zu finden, also das Gleichungssystem

$$0 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i)}{\sigma_i} \right) \left(\frac{\partial y(x_i; \dots a_k \dots)}{\partial a_k} \right) \qquad k = 1, \dots, M \tag{6.21}$$

zu lösen.

Wir wollen uns hier auf Modelle beschränken, die linear in den Parametern a_i sind, also Summen von Produkten der a_i mit beliebigen Basisfunktionen X_i. Handelt es sich bei den X_i z.b. um Potenzen von x, optimieren wir die Koeffizienten des Polynoms, bei $X_i = \sin(\alpha_i x)$ oder $X_i = \cos(\alpha_i x)$ die Parameter der harmonische Serie, sodass die Daten bestmöglich wiedergegeben werden. Damit sieht unsere ursprüngliche Gl. (6.19) nun folgendermaßen aus:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right)^2. \tag{6.22}$$

Definiert man das Problem durch Definition einer N-Zeilen $\times M$-Spalten Matrix $\mathbf{A}$ und eines Vektors $\vec{b}$ mit N Komponenten durch

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i}, \qquad b_i = \frac{y_i}{\sigma_i} \tag{6.23}$$

um, wird Gl. (6.22) zu

$$\chi^2 = |\mathbf{A} \cdot \vec{a} - \vec{b}|^2 \,. \tag{6.24}$$

Setzt man nun die Basisfunktionen X_i in Gl. (6.21) ein, so führt dies auf das Gleichungssystem

$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right) X_k(x_i) \qquad k = 1, \ldots, M \,, \tag{6.25}$$

das äquivalent zu

$$\left(\mathbf{A}^T \cdot \mathbf{A}\right)_{M\times M} \cdot \vec{a}_M = \mathbf{A}^T_{M\times N} \cdot \vec{b}_N \tag{6.26}$$

ist. Die Lösung eines Gleichungssystems der Form $\mathbf{A}\vec{a} = \vec{c}$ gehört zu den Standardproblemen der linearen Algebra und kann getrost den optimierten Bibliotheksfunktionen überlassen werden. Die Auswahl der geeigneten Routine hängt im wesentlichen davon ab, welche Zahlenwerte die Matrixelemente A_{ij} haben. Dabei ist für die χ^2-Anpassung die Methode der *singular value decomposition* am wenigsten empfindlich auf Rundungsfehler.

Was die Abschätzung der Fehler der angepassten Parameter betrifft, die wir zu Beginn als Bedingung für eine sinnvolle Anpassung gefordert hatten, so fallen diese als Nebenprodukt der Berechnung an. Die Varianzen der a_i sind durch

$$\sigma^2(a_i) = C_{ii} \qquad \text{mit} \qquad C = \left(\mathbf{A}^T\mathbf{A}\right)^{-1} \tag{6.27}$$

gegeben. Hat man das einmal akzeptiert, glaubt man auch, dass es sich bei den Nebendiagonalelementen C_{ij} um die Kovarianzen zwischen a_i und a_j handelt.

Der dritte Punkt unserer Liste war die Abschätzung der Güte des Anpassung. Sie wird als χ^2-Wahrscheinlichkeitsfunktion $Q(\chi^2|\nu)$ bezeichnet. Vereinfacht gesagt beschreibt sie die Wahrscheinlichkeit, dass man trotz eines richtigen Modells für die Anpassung einen besseren Wert für χ^2 finden kann und sollte daher möglichst klein sein.

Berechnet wird sie mithilfe der unvollständigen Γ-Funktion

$$Q(a, x) = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} dt \,, \tag{6.28}$$

wobei für $a = (N - M)/2$ und für $x = \chi^2/2$ eingesetzt wird.

6.4 Aufgaben

- **Aufgabe 6.1: Bisektionsverfahren**
  Erstellen Sie ein Programm, welches die Nullstelle einer beliebigen Funktion in einem gegebenen Intervall $[a, b]$ mittels Bisektionsverfahren bestimmt. Testen Sie das Programm für die Funktion $f(x) = \sin(x) + x/2$ im Intervall $[-10, 10]$.

- **Aufgabe 6.2:Newton-Raphson-Verfahren**
  Erstellen Sie ein Unterprogramm `NEWTON(X,Func,Acc,IFAIL)`, welches die Nullstelle einer beliebigen Funktion mit Hilfe des Newton-Raphson-Verfahrens ausgehend von einer ersten Schätzung der Nullstelle bestimmt. Dabei ist `X` beim Aufruf der Subroutine zuerst der Schätzwert und beim Ausgang der beste Wert für die Nullstelle. `Func` ist die betrachtete Funktion. `Acc` ist die gewünschte Genauigkeit mit der die Nullstelle gefunden werden soll. `IFAIL` ist ein Fehlerparameter (`IFAIL=0`: Nullstelle gefunden, `IFAIL=1`: in 1000 Schritten konnte keine ausreichende Genauigkeit gefunden werden, `IFAIL=2`: Iteration konvergiert zu keinem Funktionswert). Testen Sie das Programm für die Funktion $f(x) = \sin(x) + x/2$ im Intervall $[-10, 10]$.

- **Aufgabe 6.3: Nullstellen eines Polynoms**
  Schreiben Sie ein Programm, welches alle reellen Nullstellen eines Polynoms mit reellen Koeffizienten $c(i)$ bestimmt. Führen Sie eine grobe Nullstellensuche zuerst mittels Bisektionsverfahren durch. Verfeinern Sie den Wert der Nullstelle mit dem in Beispiel 6.2 erstellten Unterprogramm auf der Basis des Newton-Raphson Verfahrens. Nach Bestimmung einer Nullstelle a reduzieren Sie den Grad des Polynoms durch Division durch $(x - a)$.

- **Aufgabe 6.4: Lineare χ^2-Anpassung**
  Brown'sche Bewegung: Ein makroskopisches (= im Mikroskop sichtbares) Teilchen vollführt eine Zufallsbewegung aufgrund von Stößen mit vielen mikroskopisch kleinen (= unsichtbaren) Teilchen. Diese Bewegung wird in der Literatur oft als "random walk" bezeichnet. Das Programm `randwalk.f` liefert Ihnen den Abstand des makroskopischen Teilchens vom Ausgangspunkt als Mittelwert von N random walks mit Schrittweite 1 als Funktion der Schrittanzahl n. Das Ergebnis sollte proportional zu $\sqrt{n}$ sein. Schreiben Sie

  - ein Programm, daß die Funktion $f(x) = a * \sqrt{x}$ an das Ergebnis von `randwalk.f` anpaßt;
  - ein Programm, in dem Sie ausnützen, daß sich die Funktion $f(x)$ linearisieren läßt.

- **Aufgabe 6.5: Nichtlineare χ^2-Anpassung**
  Führen Sie eine nichtlineare χ^2 Anpassung durch. Das Programm `nonlinear.f` erzeugt Datenpunkte, an die die Funktion $f(x) = a \cdot e^{-b \cdot x} + c \cdot e^{-d \cdot x}$ angepaßt werden soll. Verwenden Sie die CERNLIB-Routine `dfunft` und stellen Sie das Ergebnis zusammen mit den Datenpunkten mithilfe von `gnuplot` dar.

Kapitel 7

Das Pendel

7.1 Schwingungsbewegung

7.1.1 Die Bewegungsgleichung und ihre Lösung

Wir betrachten zunächst eine an einer Feder aufgehängte Masse m wie sie in Abb. 7.1 dargestellt ist. Im Gravitationsfeld der Erde sei die Ruhelage der Feder bei $z = \zeta_0$. Bei Abweichung von der Ruhelage erfüllt das System die Bewegungsgleichung

$$m\frac{d^2z}{dt^2} = -C(z - \zeta_0)\,, \tag{7.1}$$

wobei C die Federkonstante ist. Der lineare Zusammenhang zwischen Kraft und Auslenkung ist bei Federn in einem weiten Bereich erfüllt. Die Wirkung der Gravitationskraft ist dabei in der Ruheauslenkung ζ_0 bereits berücksichtigt. Die Lösung von (7.1) ist ein Standardbeispiel bei der Behandlung von Differentialgleichungen zweiter Ordnung. Man erhält die allgemeine Lösung

$$z(t) = A\sin(\omega_0 t + \alpha)\,, \tag{7.2}$$

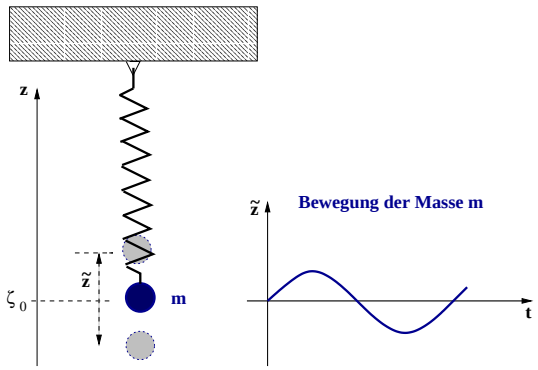

Abbildung 7.1: Anordnung einer Masse m an einer Feder mit der Federkonstante C

welche eine ungedämpfte Schwingung mit der Eigenfrequenz

$$\omega_0 = \sqrt{\frac{C}{m}} \tag{7.3}$$

darstellt. Die Größen A und α ergeben sich aus den Anfangsbedingungen, die stets integraler Bestandteil einer Problemstellung im Zusammenhang mit einer Differentialgleichung sein müssen.

7.1.2 Lagrangeformalismus und Zustand im Phasenraum

Die Bewegungsgleichung (7.1) entspricht im Lagrange-Formalismus der Euler-Lagrange-Gleichung

$$\frac{\partial L}{\partial z} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) = 0\,, \tag{7.4}$$

wobei $\dot{z} = \frac{dz}{dt}$ und $L(z, \dot{z}, t)$ die Lagrangefunktion des Systems ist. Aus Gleichung (7.1) folgt

$$\frac{\partial^2 L}{\partial \dot{z}\partial \dot{z}} = m \quad \Longrightarrow \quad L(z, \dot{z}, t) = \tfrac{1}{2} m\dot{z}^2 + R(z,t)\dot{z} + S(z,t) \tag{7.5}$$

Die Integrationskonstanten R und S können dabei nur von z und t abhängen. Aus der Bewegungsgleichung lassen sich $R(z,t)$ und $S(z,t)$ nicht vollständig bestimmen, da die Lagrangefunktion nur bis auf eine Eichtransformation bestimmt werden kann, d.h.

$$\tilde{L}(z, \dot{z}, t) = L(z, \dot{z}, t) + \frac{d}{dt}\Omega(z,t) \tag{7.6}$$

erfüllt dieselbe Euler-Lagrange-Gleichungen wie $L(z, \dot{z}, t)$, wobei $\Omega(z,t)$ eine beliebige Eichfunktion ist. Insbesondere ergibt sich aus (7.5) und (7.6)

$$\tilde{L}(z, \dot{z}, t) = \tfrac{1}{2} m\dot{z}^2 + \left[R(z,t) + \frac{\partial \Omega}{\partial z}\right]\dot{z} + \left[S(z,t) + \frac{\partial \Omega}{\partial t}\right]\,. \tag{7.7}$$

Ausnützen der Eichfreiheit erlaubt ohne Einschränkung der Allgemeinheit die Annahme

$$\tilde{R}(z,t) = R(z,t) + \frac{\partial \Omega}{\partial z} = 0. \tag{7.8}$$

Stellt man nun die Euler-Lagrange-Gleichung mit $\tilde{L}$ auf und vergleicht die Terme mit jenen der Bewegungsgleichung (7.1), so ergibt sich

$$\tilde{S}(z,t) = S(z,t) + \frac{\partial \Omega}{\partial t} = \frac{1}{2} C\tilde{z}^2 + D(t)\,, \tag{7.9}$$

wobei

$$\tilde{z} = z - \zeta_0\,. \tag{7.10}$$

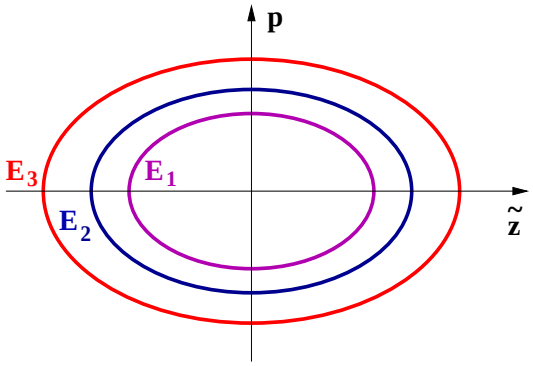

Abbildung 7.2: Phasendiagramm ungedämpfter Schwingungsbewegungen (mathematisches Pendel) mit unterschiedlichen Gesamtenergien $E_3 > E_2 > E_1$.

Wie man aus der Gleichung (7.9) erkennt, hat man in $\tilde{S}(z,t)$ noch eine Eichfreiheit, sodass ohne Einschränkung der Allgemeinheit $D(t) = 0$ gesetzt werden kann. Damit ergibt sich die Lagrange-Funktion der in Abb. 7.1 dargestellten Anordnung mit

$$L(z,\dot{z},t) = \tfrac{1}{2}m\dot{z}^2 - \tfrac{1}{2}C\tilde{z}^2\,, \tag{7.11}$$

wobei wir hier die Unterscheidung zwischen $\tilde{L}$ und L unterdrückt haben. Mit dem verallgemeinerten Impuls des Systems

$$p = \frac{\partial L}{\partial \dot{z}} = m\dot{z} \tag{7.12}$$

erhält man über die Legendre-Transformation die Hamiltonfunktion des linearen harmonischen Oszillators

$$H(z,p,t) = z\cdot p - L(z,\dot{z},t) = \tfrac{1}{2}m\dot{z}^2 + \tfrac{1}{2}C\tilde{z}^2\,. \tag{7.13}$$

Da die Hamiltonfunktion keine explizite Zeitabhängigkeit aufweist, ist die Gesamtenergie des Systems eine Erhaltungsgröße.

Für eine eingehende physikalische Diskussion ist es sinnvoll die Bewegung des Systems im Phasenraum zu betrachten. Die allgemeine Lösung (7.2) liefert im Phasenraumdiagramm eine Ellipse (siehe Abb. 7.3), welche durch die Gesamtenergie charakterisiert werden kann. Das Auftreten einer geschlossenen Kurve (im gegenständlichen Fall einer Ellipse) im Phasenraum zeigt eine periodische Bewegung des Systems an.

7.1.3 Die gedämpfte Schwingungsbewegung

In realistischen Systemen ist stets eine Dämpfung vorhanden, sodass die Energie keine Erhaltungsgrösse ist. Ein typischer Dämpfungsterm ist der Luftwiderstand, welcher in einer laminaren Strömung proportional zur Geschwindigkeit der Masse ist. Die Bewegungsgleichung (7.1) muss nun um diesen Reibungsterm ergänzt werden,

$$m\frac{d^2z}{dt^2} = -C(z-\zeta_0) - \gamma\dot{z}\,, \tag{7.14}$$

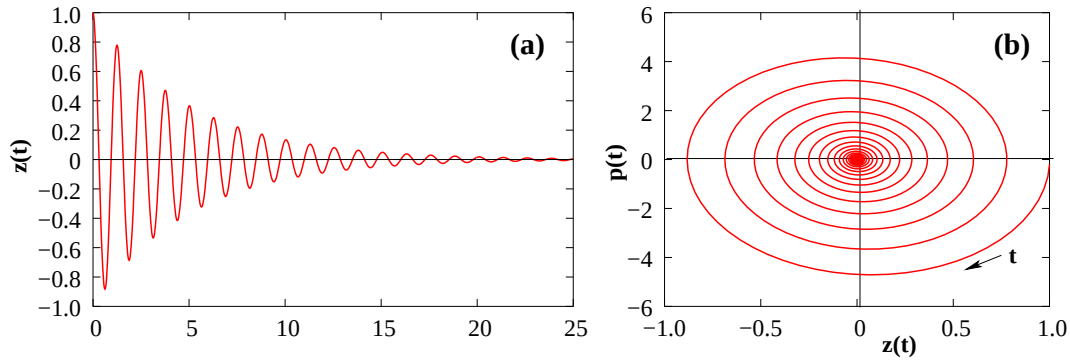

Abbildung 7.3: Die gedämpfte Schwingung. (a) Zeitliche Bewegung; (b) Phasendiagramm der gedämpfter Schwingung.

wobei γ ein konstanter Dämpfungskoeffizient ist. Für diesen einfachen Reibungsterm lässt sich wieder eine allgemeine Lösung der Bewegungsgleichung angeben,

$$z(t) = A \exp(-\gamma t) \sin(\omega t + \alpha)\,, \tag{7.15}$$

welche eine gedämpfte Schwingung mit der Frequenz

$$\omega = \sqrt{\frac{C}{m} - \left(\frac{\gamma}{2m}\right)^2} \tag{7.16}$$

darstellt. Die Größen A und α ergeben sich wieder aus den Anfangsbedingungen. Die Abweichung der Frequenz ω von der Eigenfrequenz ω_0 des ungedämpften Systems steigt mit steigender Dämpfung. Außerdem bewirkt die Dämpfung eine exponentiell fallende Amplitude der Lösung, sodass die Masse m schließlich zum Stillstand kommt.

Der Dämpfungsterm kann nicht durch eine Lagrange-Funktion beschrieben werden, sondern geht in die Euler-Lagrange-Gleichung als nichtkonservativer Term ein,

$$\frac{\partial L}{\partial z} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) = \gamma \dot{z}\,. \tag{7.17}$$

Dabei ist die Lagrange-Funktion jene der ungedämpften Bewegung (7.11) und der verallgemeinerte Impuls durch (7.12) gegeben. Die Dämpfung bewirkt eine ständige Verringerung der Schwingungsenergie, sodass sich eine zum Koordinatenursprung laufende Kurve im Phasenraum ergibt. Diese ist in Abb. 7.3 gemeinsam mit der zeitlichen Entwicklung der Auslenkung dargestellt.

7.1.4 Frequenzspektrum

Neben der zeitlichen Entwicklung interessiert bei allgemeinen Schwingungsbewegungen vor allem das Frequenzspektrum $Z(\omega)$. Dieses ergibt sich über Fouriertransformation aus der zeitlichen Entwicklung

$$Z(\omega) = \int_{-\infty}^{+\infty} dt z(t) e^{i\omega t} \tag{7.18}$$

Ist die Funktion $z(t)$ reel-wertig, so hat die Fouriertransformation die Symmetrie $Z(-\omega) = Z(\omega)^*$.

Setzt man nun für $z(t)$ die Lösung (7.2) des ungedämpften Pendels ein, so ergibt sich die Fouriertransformierte

$$Z(\omega) = \int_{-\infty}^{+\infty} dt e^{i\omega t} A \sin(\omega_0 t + \alpha) = i\pi A \left[e^{-i\alpha} \delta(\omega - \omega_0) - e^{i\alpha} \delta(\omega + \omega_0) \right] . \quad (7.19)$$

Wir haben dabei angenommen, dass die Schwingung zu allen Zeiten im Gang ist. Nimmt man an, dass die Auslenkung $z(t) = 0$ für $t < 0$, so erhält man auf Grund der Identität,

$$\int_0^{\infty} dt e^{i\omega t} = \frac{1}{2} \int_{-\infty}^{+\infty} dt e^{i\omega t} , \quad (7.20)$$

für die Fouriertransformierte

$$Z(\omega) = \int_0^{+\infty} dt e^{i\omega t} A \sin(\omega_0 t + \alpha) = i\frac{\pi}{2} A \left[e^{-i\alpha} \delta(\omega - \omega_0) - e^{i\alpha} \delta(\omega + \omega_0) \right] . \quad (7.21)$$

In beiden Fällen zeigt das Spektrum die typische Form einer periodischen Bewegung, nämlich das Auftreten von Delta-Funktionen bei $\omega = \pm\omega_0$.

Hat das Pendel eine Dämpfung, so hat man die Fouriertransformierte der Lösung (7.15) zu bilden,

$$Z(\omega) = \frac{1}{2} A \left[\frac{e^{i\alpha}}{(\omega + \omega_0) + i\gamma} - \frac{e^{-i\alpha}}{(\omega - \omega_0) + i\gamma} \right] . \quad (7.22)$$

Je nach Stärke der Dämpfung γ kommt es zu einer Verschiebung (Gleichung (7.16) und Verbreiterung der Resonanz (Abb. 7.4).

Für die in diesem Kapitel behandelten Beispiele haben wir nur die Realteile der Fouriertransformierten betrachtet,

$$Z(\omega) = \int_0^{\infty} dt z(t) \cos(\omega t) . \quad (7.23)$$

Diese sind leicht darzustellen und zeigen bei einer periodischen Bewegung (ungedämpfte Federschwingung, periodische erzwungene Federschwingung) gemäss (7.21) das Auftreten einer Linie in der Spektralverteilung. Bei einer gedämpften Schwingung kommt es zu einer Verbreiterung der Linie (siehe Abb. 7.4).

7.2 Numerische Lösung gewöhnlicher Differentialgleichungen

Viele physikalische Probleme erfordern in ihrer mathematischen Formulierung die Lösung einer gewöhnliche Differentialgleichung. Es sind dabei sowohl Anfangs- bzw.

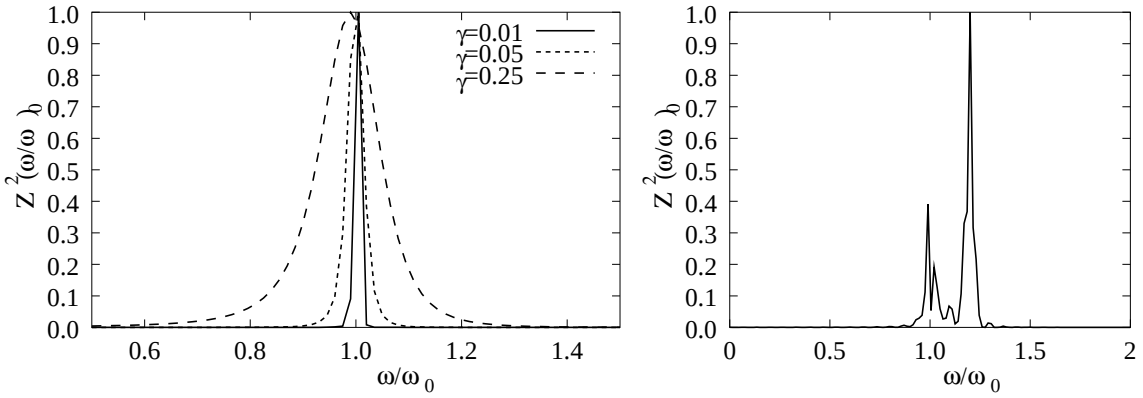

Abbildung 7.4: Die Spektralanalyse einer (a) freien ungedämpften [$\gamma = 0.01$ Nsm^{-1}] und gedämpften Federbewegung [$\gamma = 0.05, 0.025$ Nsm^{-1}] (b) einer getriebenen Feder. Die Verteilungen sind normiert auf den Maximalwert. Die Frequenz ist in Einheiten der Eigenfrequenz gegeben.

Randwertprobleme als auch Eigenwertprobleme zu lösen. Bei der Berechnung der Pendelbewegung, wie wir sie nachstehend behandeln werden, liegt ein typisches Anfangswertproblem vor.

Für die numerische Lösung der gewöhnlichen Differentialgleichungen stehen heute eine Vielzahl von allgemeinen und speziellen Verfahren zur Verfügung, die wichtigsten Klassen von Algortihmen sind:

a. Einschrittverfahren
b. Mehrschrittverfahren
c. Extrapolationsverfahren

Im folgenden werden nur die einfachsten Einschrittverfahren besprochen. Für Algorithmen zu (b.) und (c.) verweisen wir auf die einschlägige Literatur, z.B. Stoer [16].

7.2.1 Systeme gewöhnlicher Differentialgleichungen erster Ordnung

Ein System von gewöhnlichen Differentialgleichungen erster Ordnung hat die Form

$$\begin{array}{rcl} y_1' &=& f_1(x, y_1, \ldots, y_n) \\ y_2' &=& f_2(x, y_1, \ldots, y_n) \\ \vdots & & \vdots \\ y_n' &=& f_n(x, y_1, \ldots, y_n) \end{array} \quad , \tag{7.24}$$

wobei die Differentialgleichungen durch n Funktionen gegeben sind, $f_i, i = 1, \ldots, n$. Das Differentialgleichungssystem lässt sich durch eine Vektornotation kompakt anschreiben

$$\underline{y'} = \underline{f}(x, \underline{y}) \, . \tag{7.25}$$

Die Vektoren sind dabei wie folgt definiert

$$\underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \text{und} \quad \underline{f}(x, \underline{y}) = \begin{pmatrix} f_1(x, y_1, \ldots, y_n) \\ \vdots \\ f_n(x, y_1, \ldots, y_n) \end{pmatrix} . \tag{7.26}$$

Es ist wichtig festzuhalten, dass man jedes Anfangswertproblem einer Einzeldifferentialgleichung n-ter Ordrnung,

$$y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)}) , \tag{7.27}$$

auf ein Anfangswertproblem eines Systems von n Differentialgleichungen erster Ordnung zurückführen kann. Definiert man nun

$$z_1 = y , \quad z_2 = y' , \quad \ldots , \quad z_n = y^{(n-1)} \tag{7.28}$$

so ergibt sich das System von Differentialgleichungen erster Ordnung

$$\begin{array}{rcl} z_1' & = & z_2 \\ z_2' & = & z_3 \\ \vdots & & \vdots \\ z_n' & = & f_n(x, z_1, \ldots, z_n) \end{array} \quad . \tag{7.29}$$

In jedem System von gewöhnlichen Differentialgleichungen erster Ordnung lässt sich durch die zusätzliche Definition von $y_0 = x$, die unabhängig Veränderliche x auf der rechten Seite eliminieren, d.h. man erhält ein System der Form

$$\begin{array}{rcl} y_0' & = & 1 \\ y_1' & = & f_1(y_0, y_1, \ldots, y_n) \\ y_2' & = & f_2(y_0, y_1, \ldots, y_n) \\ \vdots & & \vdots \\ y_n' & = & f_n(y_0, y_1, \ldots, y_n) \end{array} \quad . \tag{7.30}$$

Dies wird als autonomes System von Differentialgleichungen bezeichnet und vereinfacht die numerische Implementierung ohne Genauigkeitsverlust.

Explizite Einschrittverfahren zur Lösung von gewöhnlichen Differentialgleichungen sind durch die Rekursionsvorschrift

$$\underline{y}(x_{i+1}) = \underline{y}(x_i) + h_{i+1}\underline{\Phi}(x_i, \underline{y}(x_i); h_{i+1}) \quad \text{mit} \quad \underline{y}(x_0) = \underline{y}(x_a) \tag{7.31}$$

gegeben, wobei $x_i, i = 0, \ldots, N$ adequat gewählte Stützstellen im zu untersuchenden Intervall $[x_a, x_e]$, $h_{i+1} = x_{i+1} - x_i$ und $\underline{y}(x_i)$ der Wert der Lösung $\underline{y}(x)$ an der Stelle $x = x_i$ sind. Die Funktion $\Phi(x, \underline{y}(x), h)$ ist charakteristisch für das verwendete Einschrittverfahren. Beginnend vom Anfangs- bzw. Randwert $\underline{y}(x_a)$ lässt sich die Lösung $\underline{y}(x)$ sukzessive berechnen. In den folgenden Unterkapiteln werden die einfachsten Einschrittverfahren kurz beschrieben.

7.2.2 Einschrittverfahren erster und zweiter Ordnung

Das einfachste Einschrittverfahren ist ohne Zweifel das Euler- oder Polygonzugverfahren, welches wir hier am Beispiel einer Einzeldifferentialgleichung,

$$y' = f(x, y)\,, \tag{7.32}$$

betrachten wollen. Ist y hinreichend oft differenzierbar, so kann man eine Reihenentwicklung der Lösung $y(x)$ durchführen

$$y(x+h) = \sum_{k=0}^{m} \frac{h^k}{k!}\, y^{(k)}(x) + \frac{h^{m+1}}{(m+1)!} y^{(m+1)}(x+\theta h) \quad \text{mit} \quad 0 \le \theta \le 1\,. \tag{7.33}$$

Man benötigt in dieser Entwicklung die Kenntnis der Ableitungen $y^{(k)}(x)$. Die niedrigsten Ableitungen lassen sich einfach aus der Funktion $f(x, y)$ bestimmen:

$$\begin{aligned} y' &= f(x,y)\,, && (7.34)\\ y'' &= \frac{d^2y}{dx^2} = \frac{df}{dx} + \frac{df}{dy}\frac{dy}{dx} = f_x + f_y f\,, && (7.35)\\ y^{(3)} &= f_{xx} + 2f f_{xy} + f_{yy} f^2 f_x f_y + f f_y^2\,, && (7.36)\\ \cdots &\quad \cdots \quad \cdots && (7.37) \end{aligned}$$

Je nach der Wahl von m lassen sich unter Vernachlässigung des Resttermes verschiedene Verfahren ableiten.

Das *Euler- oder Poligonzugverfahren* ergibt sich bei der Wahl $m = 1$. Vernachlässigung des Resttermes für $m = 1$ führt auf die Rekursionsbeziehung,

$$y(x+h) = y(x) + hf(x, y(x)) \quad \text{d.h.} \quad \Phi(x, y; h) = f(x, y)\,. \tag{7.38}$$

Beim Euler- oder Polygonzugverfahren wird also der Differentialquotient durch den Differenzenquotienten ersetzt, d.h.

$$y' \;\approx\; \frac{y(x+h) - y(x)}{h} \tag{7.39}$$

Will man die Entwicklung (7.33) bis $m = 2$ berücksichtigen, so setzt man die Funktion Φ in folgender Form an,

$$\Phi(x, y; h) = a_1 f(x, y) + a_2 f(x + p_1 h, y + p_2 h f(x, y)) \tag{7.40}$$

Eine Entwicklung von (7.40) bis zur ersten Ordnung in h liefert

$$\Phi(x, y; h) = (a_1 + a_2) f(x, y) + a_2 h\,[p_1 f_x(x, y) + p_2 f_y(x, y) f(x, y)] + o(h^2)\,. \tag{7.41}$$

Wählt man $a_1 + a_2 = 1$ und $p_1 a_2 = p_2 a_2 = \frac{1}{2}$ so sind die Terme der Reihenentwicklung (7.33) bis $m = 2$ durch die Funktion $f(x, y)$ ausgedrückt. Man erhält damit das *Verfahren von Heun*, welches auch als *verbessertes Polygonzugverfahren* bezeichnet wird,

$$\begin{aligned} a_1 = a_2 &= \tfrac{1}{2}\,, \qquad p_1 = p_2 = 1 \\ \Phi(x, y; h) &= \frac{1}{2}\,[f(x, y) + f(x + h, y + hf(x, y))]\,, && (7.42) \end{aligned}$$

und das *modifizierte Euler-Verfahren*

$$\begin{aligned} a_1 = 0\ ; \qquad & a_2 = 1\ ; \quad p_1 = p_2 = \tfrac{1}{2} \\ \Phi(x,y;h) \quad = \quad & f(x+\tfrac{1}{2}h, y+\tfrac{1}{2}hf(x,y))\,. \end{aligned} \tag{7.43}$$

Betrachtet man die Ausdrücke für die Funktion Φ, so sieht man, dass in den letzten beiden Verfahren die Funktion $f(x,y)$ zweimal aufgerufen werden muss, während im Euler-Verfahren nur ein Aufruf von $f(x,y)$ pro Schritt erforderlich ist.

7.2.3 Konsistenz und Konvergenz von Einschrittverfahren

Bei der Diskussion der Güte eines Einschrittverfahrens muss man zwischen Konsistenz und Konvergenz des Verfahrens unterscheiden. Die Konsistenz eines Verfahrens bezieht sich auf die Eigenschaften der Funktion Φ, während Konvergenz die Lösung der Differentialgleichung betrifft.

Man nennt das Einschrittverfahren *konsistent*, wenn die Funktion Φ der Bedingung genügt

$$\lim_{h\to 0} \Phi(x,y;h) = f(x,y) \quad x\epsilon[a,b],\ , y\epsilon R\,. \tag{7.44}$$

Man spricht von einem Verfahren der Ordnung p, falls

$$\tau(x,y;h) = \Delta(x,y;h) - \Phi(x,y;h) = \mathcal{O}(h^p) \tag{7.45}$$

gilt, wobei der Differenzenquotient folgend definiert ist

$$\Delta(x,y;h) = \begin{cases} \frac{z(y+h)-z(x)}{h} & h \neq 0 \\ f(x,y) & h = 0 \end{cases}\ , \tag{7.46}$$

und $z(x)$ die exakte Lösung der Differentialgleichung ist.

Durch Einsetzen in die entsprechenden Ausdrücke sieht man, dass das Euler-Verfahren konsistent in 1. Ordnung ist, während das Verfahren von Heun und das modifizierte Eulerverfahren konsistent in 2. Ordnung sind.

Neben der Konsistenz eines Verfahrens gibt es noch den Begriff der Konvergenz. Ein Einschrittverfahren heißt an der Stelle $x\epsilon[x_a, x_e]$ *konvergent*, wenn

$$\lim_{h\to 0}(y_i^h) - z(x)) = 0 \quad \text{für} \quad x_i = x\epsilon[x_a, x_e]\,, \tag{7.47}$$

wobei y_i^h die numerische Lösung an der Stelle x_i und $z(x = x_i)$ die exakte Lösung der Differentialgleichung ist. Das Einschrittverfahren heißt konvergent zur Ordnung $p > 0$, wenn

$$y_i^h - z(x) = \mathcal{O}(h^p) \quad \text{für} \quad h \to 0, j = 0, 1, \ldots, N \tag{7.48}$$

gilt.

Bei Einschrittverfahren läßt sich leicht ein Zusammenhang zwischen Konsistenz- und Konvergenzordnung herstellen. Es gilt folgender Satz:

Für eine Einzeldifferentialgleichung $y' = f(x,y)$ sei ein Einschrittverfahren $\Phi(x,y;h)$ mit folgenden Eigenschaften gegeben:

- $\Phi(x, y; h)$ ist bezüglich aller Veränderlicher $(x, y; h)$ stetig für $x_a \leq x \leq x_e$, $-\infty < y < +\infty$, $0 \leq h \leq h_0$, wobei h_0 hinreichend klein ist,
- Das Verfahren sei konsistent von der Ordnung $p > 0$,
- Die Funktion $\Phi(x, y; h)$ erfülle bezüglich y eine Lipschitzbedingung,

Für diese so gegebene Einschrittverfahren gilt, dass es konvergent von der Ordnung p.

7.2.4 Rundungsfehler

Die Abschätzung der Verläßlichkeit numerischer Lösungen ist für die Beurteilung der Güte eines Resultates unerläßlich. Bei numerischen Rechnungen lassen sich verschiedene Klassen von Fehlern unterscheiden:

- *Unsicherheit in den Eingabedaten:* Diese können im allgemeinen in der numerischen Rechnung nicht beeinflußt werden. Man muss jedoch die Stabilität bezüglich solcher Unsicherheiten diskutieren.
- *Abbrechfehler:* Darunter versteht man Fehler durch Vernachlässigung von Resttermen bzw. nicht vollständig durchgeführte Iterationen. Solche Fehler lassen sich über den verwendeten Algorithmus steuern.
- *Rundungsfehler:* Diese werde durch die Abbildung der numerischen Werte auf die Menge der Maschinenzahlen verursacht. Durch Auslöschungseffekte kann es zu einer unerwünschten Verstärkung dieser Fehler kommen.

Im Unterkapitel 7.2.3 haben wir die Konvergenz der Verfahren betrachtet. Im nächsten Schritt wollen wir die Gesamtheit aller Fehler, und zwar Rundungs- und Verfahrensfehler gemeinsam betrachten. Sei $z(x)$ die exakte Lösung und η_i die numerische Lösung an der Stelle x_i, dann gelten die Beziehungen

$$\text{Numerische Lösung:} \qquad \tilde{\eta}_{i+1} = \tilde{\eta}_i + h\Phi(x_i, \tilde{\eta}_i; h) + \varepsilon_{i+1}\,, \tag{7.49}$$

$$\text{Exakte Lösung:} \qquad z(x_{i+1}) = z(x_i) + h\Phi(x_i, z_i; h) + \wr(h^{p+1})\,, \tag{7.50}$$

wobei ε_{i+1} der gesamte Rundungsfehler für die Integration von x_i nach x_{i+1} ist. Subtraktion der beiden Gleichungen liefert,

$$r_{i+1} = \tilde{\eta}_{i+1} - z(x_{i+1}) = \tilde{\eta}_i - z(x_i) + h\left[\Phi(x_i, \tilde{\eta}_i; h) - \Phi(x_i, z(x_i); h)\right] \varepsilon_{i+1}\wr(h^{p+1})\,. \tag{7.51}$$

Die Größe des Gesamtfehlers lässt sich unter der Annahme einer Lipschitz Bedingung für Φ abschätzen

$$|r_{i+1}| \leq |r_i| + h\ L|r_i||\varepsilon_{i+1}| + Mh^{p+1}\,. \tag{7.52}$$

Lösung dieser Differenzengleichung liefert die Abschätzung

$$|r_i| \leq \left[Mh^p + \frac{|\varepsilon_i|}{h}\right] \frac{e^{iM(x_e - x_a)} - 1}{L}\,. \tag{7.53}$$

Diese Abschätzung ist für praktische Anwendungen zu grob und hat daher nur qualitativen Wert. Sie zeigt aber das wesentliche Verhalten der Abhängigkeit des Gesamtfehlers von der Schrittweite. Es gibt einen optimalen Wert für die Schrittweite h. Wird h kleiner, so steigt aufgrund der Rundungsfehler der Gesamtfehler an (Abb. 7.5).

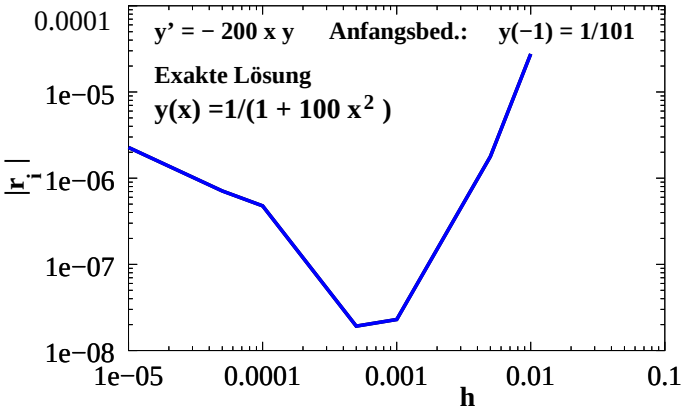

Abbildung 7.5: Beispiel für den Gesamtfehler bei der numerischen Lösung einer Differentialgleichung.

7.2.5 Runge-Kuttaverfahren

Die Standardmethode

Eine der beliebtesten Methoden zur Integration von gewöhnlichen Differentialgleichungen ist das *Verfahren von Runge-Kutta*. Dieses Verfahren wurde 1985 erstmals formuliert und basiert auf einen allgmeinen Ansatz für die Funktion $\Phi(x, y; h)$. In der Standardform ist das Runge-Kutta-Verfahren für eine Funktion $y' = f(x, y)$ durch folgende Funktion $\Phi(x, y; h)$ gegeben:

Die Standardform des Runge-Kutta-Verfahrens

$$\Phi(x, y; h) = \tfrac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right] + O(h^5)$$

$$k_1 = f(x, y) \qquad k_2 = f(x + \tfrac{h}{2}, y + \tfrac{h}{2}k_1)$$
$$k_3 = f(x + \tfrac{h}{2}, y + \tfrac{h}{2}k_2) \qquad k_4 = f(x + h, y + h\ k_3)$$

Entwickelt man $\Phi(x, y; h)$ in eine Taylorreihe nach h, so findet man, dass die oben angegebene Form des Runge-Kutta-Verfahrens konisistent in vierter Ordnung ist. Der Beweis der Konsistenzordnung ist trivial, erfordert aber eine längere Rechnung.

Das Runge-Kutta-Verfahren ist einfach, effizient und bestens geeignet, wenn an die Genauigkeit der Lösungen nicht allzu große Anforderungen gestellt werden. Für

hohe Anforderungen an die Genauigkeit, sollte man auf eine *Predictor-Korrektor-Methode* übergehen [16]. Das Runge-Kutta-Verfahren lässt sich auch auf Systeme von Differentialgleichungen erster Ordnung in analoger Weise anwenden. Die Funktionen $\Phi(x, y; h), k_1, k_2, k_3$ und k_4 sowie die Variable y erhalten dann Vektorcharakter (siehe 7.2.1). Die Programmierung wird besonders einfach, wenn man auf ein autonomes Gleichungssystem übergeht, d.h. man definiert die unabhängige Variable x als die nullte Komponente des Lösungsvektors y (siehe Gleichung (7.30).

Runge-Kutta-Methode mit angepasster Schrittweite

Ein guter Algorithmus für die Integration gewöhnlicher Differentialgleichungen sollte die Güte der Lösung laufend kontrollieren und entsprechende Adaptierung der Parameter selbst vornehmen, um eine bestimmte vorgegebene Genauigkeit zu garantieren. In Einschrittverfahren lässt sich die Genauigkeit über die Schrittweite kontrollieren. In Bereichen starker Änderung der Lösung $y(x)$ wird man zur Erreichung einer bestimmten Genauigkeit viele kleine Schritte benötigen, während man in Bereichen eines weitgehend glatten Funktionsverlaufes von $y(x)$ mit wenigen großen Schritten das Auslangen findet. Sieht man eine entsprechende Anpassung der Schrittweite im Algorithmus vor, so kann man die Effizienz des Algorithmus wesentlich steigern (je nach Funktion $y(x)$ sind Effizienzsteigerungen von 100 und mehr möglich).

Die Festlegung einer der jeweiligen Lösung angepassten Schrittweite erfordert die Bereitstellung einer Information über die Güte der Lösung bei jedem Integrationsschritt. Bei der Standard Runge-Kutta-Methode (konsistent in vierter Ordnung) erreicht man dies am einfachsten durch Verdopplung der Schritte, d.h. man rechnet jeden Schritt zweimal, einmal in einem Schritt und einmal in zwei Schritten,

$$y(x+2h) = y_1(x) + (2h)^5\phi + O(h^6) + \cdots , \tag{7.54}$$

$$y(x+2h) = y_2(x) + 2h^5\phi + O(h^6) + \cdots . \tag{7.55}$$

Die exakte Lösung wurde mit $y(x)$, bezeichnet, die numerischen Lösungen, y_1, y_2, wurden mit einer Schrittweite $2h$ bzw. h gerechnet. Insgesamt erfordert jeder Schritt 11 Aufrufe der Funktion $f(x, y)$. Dies muss man vergleichen mit 8 Aufrufen, wenn man mit der Schrittweite h integriert. Die Rechenzeit erhöht sich zwar um den Faktor $1,375$, aber der Algorithmus mit Schrittverdopplung liefert einen Parameter, $\Delta = y_2 - y_1$, der die Güte der Lösung harakterisiert. Die Forderung, dass Δ kleiner als ein vorgegebener Genauigkeitsparameter `acc` ist, erlaubt eine automatisierte Schrittweitenanpassung des Runge-Kutta-Verfahrens. Erfüllt die gewählte Schrittweite das Kriterium $\Delta <$`acc` nicht, so wird die Schrittweite verkleinert und die Integration noch einmal ausgeführt. Ist das Kriterium $\Delta <$`acc` erfüllt, so kann y_2 als Wert der Lösung verwendet werden und man führt den nächsten Integrationsschritt aus. Alternativ lässt sich auch ein korrigierter Wert der Lösung angeben,

$$\Delta = y_2 - y_1 = 30h^5\phi + O(h^6) , \tag{7.56}$$

$$y^{(corr)} = y_2(x) + \frac{1}{15}\Delta + O(h^6) . \tag{7.57}$$

Die neue Abschätzung ist nun fünfter Ordnung und wird allgemein als *lokale Approximation* bezeichnet. Man kann nun mit diesem Wert die Lösung fortsetzen.

Eine alternative Konstruktion eines Differentialgleichungsintegrators mit automatisierter Schrittweitenanpassung beruht auf den von Fehlberg abgeleiteten *Eingebetteten Runge-Kutta-Formeln*. Fehlberg entdeckte eine Methode fünfter Ordnung, welche sechs Aufrufe von $f(x, y)$ erfordert[1],

$$k_i = hf(x_n + a_i h, y_n + \sum_{j=1}^{i-1} b_{ij} k_j) \quad j = 1, \ldots, 6\,, \tag{7.58}$$

$$y_{n+1} = y_n + \sum_{i=1}^{6} c_i k_i + O(h^6)\,. \tag{7.59}$$

Die Koeffizienten a_i, b_{ij}, c_i sind in Tab. 7.1 gegeben. Eine andere Kombination der gleichen sechs Aufrufe liefert ein Verfahren vierter Ordnung,

$$\tilde{y}_{n+1} = y_n + \sum_{j=1}^{6} \tilde{c}_j k_j + O(h^5)\,, \tag{7.60}$$

Durch Differenzbildung erhält man wieder ein Kriterium für die Güte und damit für die Schrittweitenanpassung,

$$\Delta = y_{n+1} - \tilde{y}_{n+1} = \sum_{j=1}^{6} (c_j - \tilde{c}_j)\ k_j\,. \tag{7.61}$$

Abweichend von den Koeffizienten von Fehlberg, haben sich die von Cash und Karp [17] angegebenen Werte (siehe Tab. 7.1) im allgemeinen bewährt.

7.3 Das getriebene physikalische Pendel

Abschließend zu diesem Kapitel wollen wir kurz das getriebene physikalische Pendel betrachten (Abb. 7.6). Wir nehmen dabei an, dass eine Masse m an einer Stange mit fester Länge L beweglich aufgehängt ist und eine Schwingung im Gravitationsfeld in einer Ebene ausführt. Unter Berücksichtigung des fixen Abstandes L zwischen Masse und Aufhängungspunkt, reduziert sich die Bewegungsgleichung zu

$$mL\frac{d^2\alpha}{dt^2} + \gamma L\frac{d\alpha}{dt} + mg\sin(\alpha) = F\sin(\Omega t + \beta)\,, \tag{7.62}$$

wobei g die Erdbeschleunigung, F und Ω die Stärke bzw. die Frequenz der treibenden Kraft und β deren Phasenverschiebung sind. Es ist dabei festzuhalten, dass die Treiberkraft auf die Masse in Richtung der Drehbewegung angreift.

[1] Allgemein lässt sich sagen, dass ein Runge-Kutta-Verfahren M. Ordnung zumindest M Aufrufe von $f(x, y)$, aber niemals mehr als $M + 2$ Aufrufe erfordert.

i	a_i	b_{ij}					c_i	$\tilde{c}_i$
1	0						$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$

Tabelle 7.1: Parameter von Cash und Karp [17] für die eingebettete Runge-Kutta-Methode fünfter Ordnung.

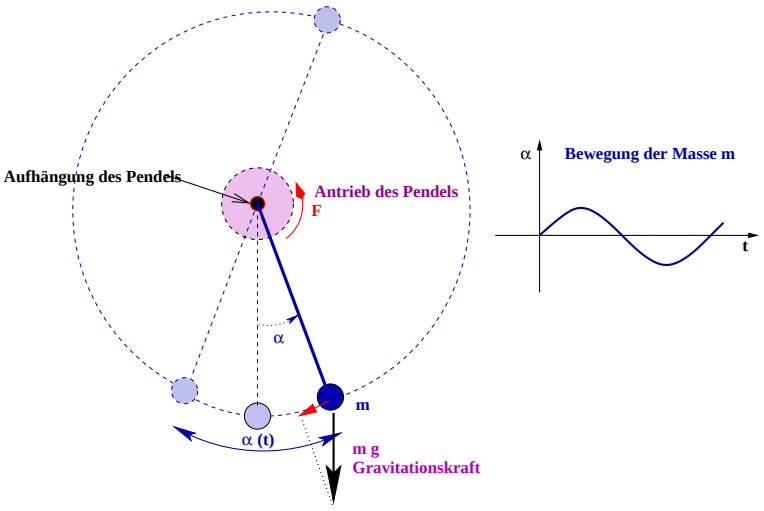

Abbildung 7.6: Anordnung eines getriebenen physikalischen Pendels.

Nach der mathematischen Formulierung des physikalischen Problems, ist der erste Schritt bei der numerischen Umsetzung die Suche nach einer geeigneten Skalierung des Systems, sodass die Variablen in natürlichenËinheiten des betrachteten Systems angegeben werden. Beim physikalischen Pendel bietet sich die Eigenfrequenz des Pendels für die Definition der Zeiteinheit an, d.h. $\tau = t/t_0$ mit

$$t_0 = \frac{1}{\omega_0} = \sqrt{\frac{L}{g}}\,. \tag{7.63}$$

Damit ergibt sich die *nichtlineare* Differentialgleichung zweiter Ordnung für den Auslenkwinkel $\alpha(t)$ in Radian

$$\frac{d^2}{d\tau^2} + \Gamma\frac{d\alpha}{d\tau} + \sin(\alpha) = f\sin(\frac{\Omega}{\omega_0}\tau + \beta)\,, \tag{7.64}$$

wobei die dimensionslosen Koeffizienten Γ und f durch

$$\Gamma = \frac{\gamma}{m}\sqrt{\frac{L}{g}} \quad \text{und} \quad f = \frac{F}{mL} \tag{7.65}$$

gegeben sind.

Wir betrachten zunächst das freie physikalische Pendel, d.h. $F = 0$. Bei kleiner Auslenkung ergibt sich zunächst das gleiche Bild wie bei der einfachen Schwingungsbewegung. Erhöht man die Auslenkung der Schwingung auf über 90 Grad, so erkennt man im zeitlichen Verlauf deutliche Abweichungen von einer Sinuskurve (Abb. 7.7), die sich auch im Phasenraum bemerkbar macht. Im Frequenzspektrum erkennt man nun bei größeren Auslenkungen auch das Auftreten von Oberwellen, die der Eigenfrequenz überlagert sind. Das Auftreten von Oberwellen ist eine direkte Konsequenz der Nichtlinearität der Gleichung. Die Dämpfung führt wie im Fall der Schwingungsgleichung wieder zu einer Verbreiterung der Eigenresonanz.

Führt man nun eine periodische Treiberkraft ein, so kann es bei ausreichend starker Treiberkraft zu einem Überschlag des Pendels kommen, d.h. der Auslenkwinkel übersteigt die Grenzen des Intervall $[-\pi, \pi]$. Bei ausreichend starker Treiberkraft erfolgt dann der übergang zu einer chaotischen Bewegung, welche durch ein exponentiell abfallendeS Spektrum signalisisiert wird. (siehe Abb. 7.8).

7.4 Aufgaben

- **Aufgabe 7.1: Runge-Kutta Algorithmus**
  Schreiben Sie ein FORTRAN Unterprogramm `SUBROUTINE RKBAS(Y,YY,N,H,DGRHS,W)`, in welcher ein Lösungsschritt von `Y(0)`=x_0 nach `YY(0)`=$x + H$ für ein System von N Differentialgleichungen 1. Ordnung mittels des Standard Runge-Kutta-Algorthmus gemacht wird. `Y` ist ein Feld, welches die Funktion am Eingang enthält, `YY` ist ein Feld, welches die Funktionswerte nach dem Schritt enthält. `W` ist ein Arbeitsfeld und `SUBROUTINE DGRHS(Y,N,F)` ist ein Unterprogramm, welches die rechte Seite des Gleichungssystem $y' = f(y)$ an der Stelle `Y` berechnet und im Feld `F` speichert. Testen Sie das Unterprogramm durch Lösen der Differentialgleichung der Federschwingung.

- **Aufgabe 7.2: Physikalisches Pendel**
  Implementieren Sie in das vorhandene Programm für die getriebene Schwingung einen Programmteil, in welchem die Lösung des getriebenen physikalischen Pendels berechnet wird. Sie müssen dabei auch eine entsprechende Routine für die Berechnung der rechten Seite des entsprechenden Gleichungssystems erstellen.

- **Aufgabe 7.3: Phasendiagramm und Frequenzspektrum**
  Untersuchen Sie die Bewegung eines getriebenen Pendels und betrachten Sie den Übergang zu einer chaotischen Bewegung bei Steigerung der Treiberkraft.

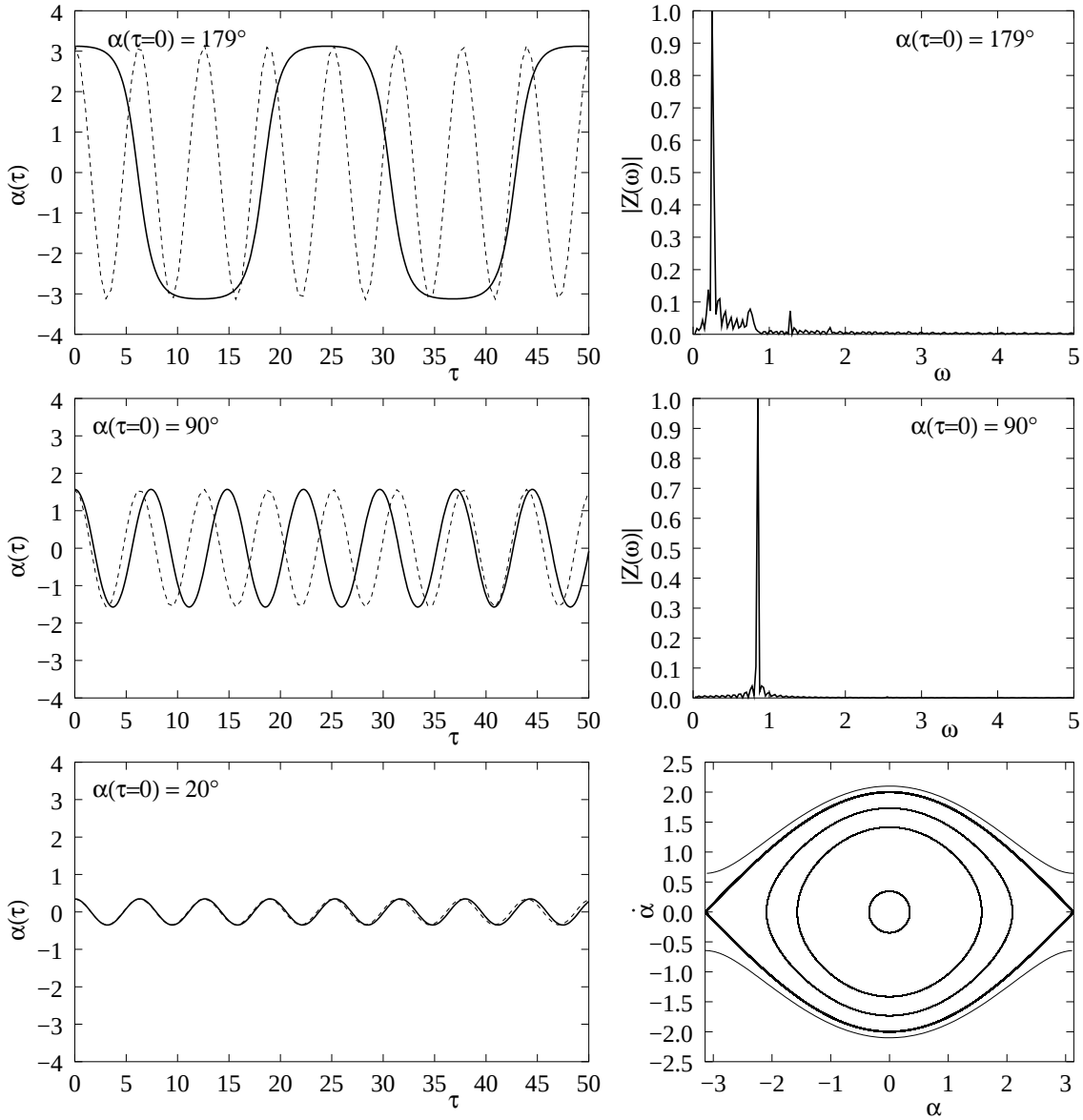

Abbildung 7.7: Die Schwingungen eines freien physikalischen Pendels. In der linken Spalte sind die Lösungen für $\alpha(\tau)$ für verschiedene Startwerte $\alpha(\tau = 0)$ gezeigt. In der rechten Spalte sind die Spektralanalysen der Lösungen mit den Startwerten $\alpha(\tau = 0) = 179$ und 90 Grad angegeben. Ein Phasendiagramm des physikalischen Pendels ist in der rechten Spalte unten gezeigt.

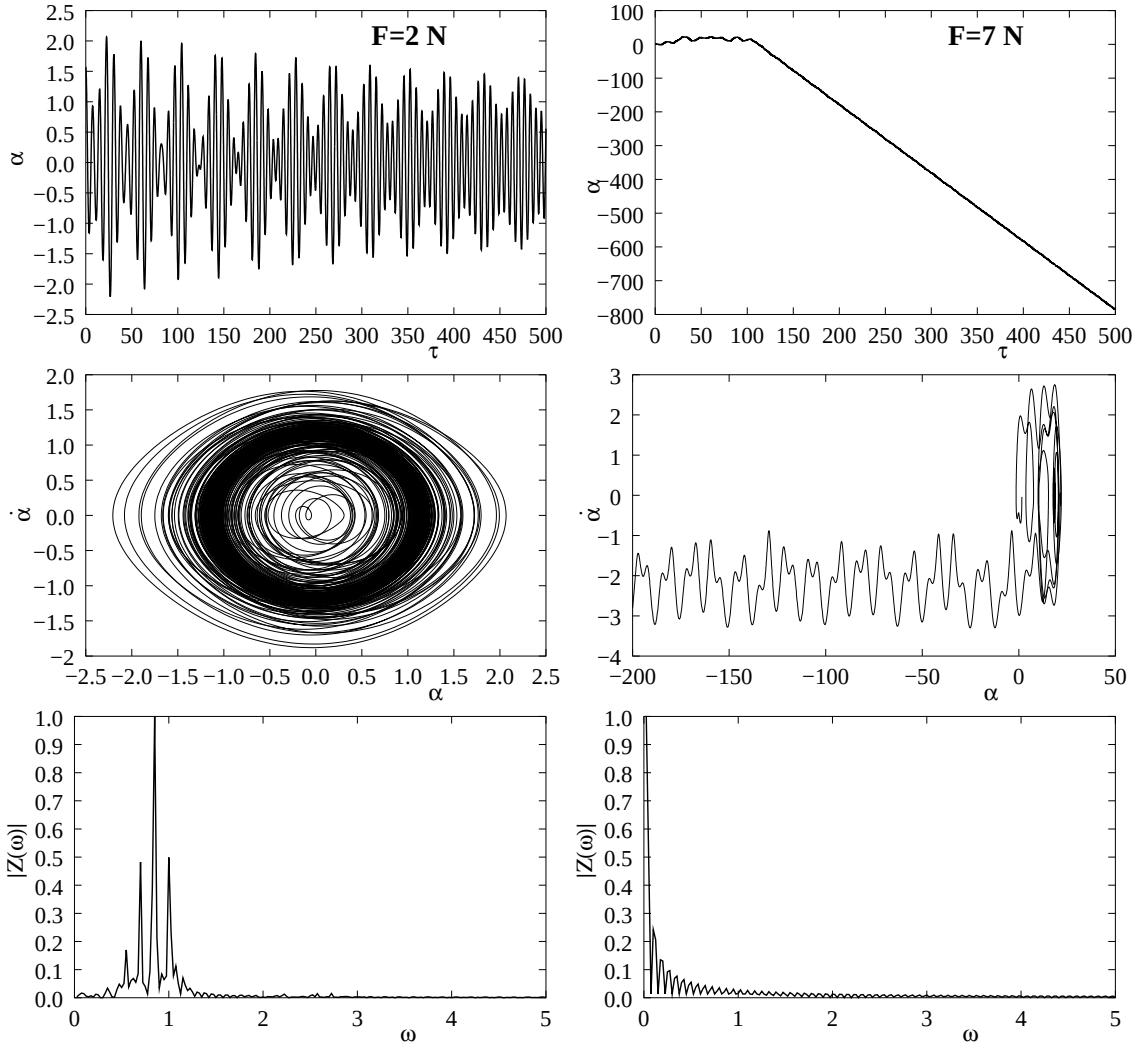

Abbildung 7.8: Lösung, Phasendiagramm und Spektralanalyse für ein mit 2 bzw. 7 N getriebenes ungedämpftes physikalisches Pendel. Die Treiberfrequenz ist mit $\Omega = 3.2$ s^{-1} angenommen. Man erkennt die beiden Extremfälle: (a) eine erzwungene Schwingung mit Schwebungserscheinungen und (b) den Übergang zu chaotischer Bewegung bei großer Treiberkraft.

Kapitel 8

Wienfilter

8.1 Ein Geschwindigkeitsfilter für Ionen

Ein Wienfilter[1] ist ein Geschwindigkeitsfilter für Ionen. Nur Ionen einer bestimmten Geschwindigkeit können den Wienfilter passieren.

Ein Wienfilter besteht aus einem homogenen elektrischen Feld $\vec{E}$ und einem homogenen magnetischen Feld $\vec{B}$, die senkrecht aufeinander und auch senkrecht auf die Bahn der Ionen stehen, siehe Abb. 8.1.

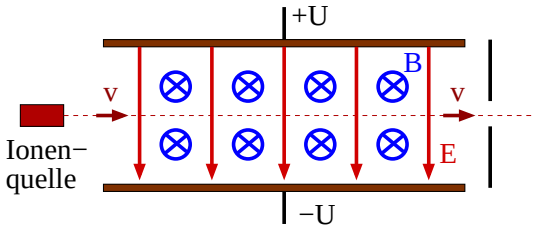

Abbildung 8.1: Schema eines Wienfilters. Die aus der Ionenquelle von links kommenden Ionen treffen auf homogene elektrische und magnetische Felder, die senkrecht zur Ionenbahn und senkrecht aufeinander stehen.

Über die beiden Feldstärken kann man regulieren, welche Geschwindigkeit die Ionen haben, die den Filter passieren können. Für positiv geladene Ionen wirkt in der Abbildung die Coulombkraft nach unten und die Lorentzkraft nach oben. Wenn Kräftegleichgewicht zwischen der elektrischen und der magnetischen Kraft herrscht, fliegen die Ionen unabgelenkt durch den Filter. Für zu langsamen Teilchen überwiegt die elektrische Feldkraft, für zu schnellen Teilchen überwiegt die Lorentzkraft. In beiden Fällen können daher die Ionen die Lochblende am Ende des Filters nicht passieren. Aus

$$\vec{F} = Q(\vec{E} + \vec{v} \times \vec{B}) = 0 \tag{8.1}$$

[1] Wilhelm Wien, 1864 - 1928. Für seine Arbeiten zur Strahlung von glühenden Körpern, Wiensches Verschiebungsgesetz und Wiensches Strahlungsgesetz, erhielt er 1911 den Nobelpreis. Er identifizierte die Kathodenstrahlen und bestimmte die spezifische Ladung und die Geschwindigkeit von Kanalstrahlen.

folgt für aufeinander normale Vektoren $\vec{v}$, $\vec{E}$ und $\vec{B}$,

$$v = \frac{E}{B} \tag{8.2}$$

Die selektierte Geschwindigkeit ist entsprechend dieser Beziehung von der Ladung und Masse der Ionen unabhängig.

Dispersion:

Für das 6 Zoll-Wienfilter Colutron 600-B ist die Dispersion für eine Massendifferenz bei Krypton etwa 2.5 mm nach 1.13 m Driftdistanz.

8.2 Lösung der Laplacegleichung unter Randbedingungen

Das Programm Laplace verwendet ein praktisches Überrelaxationsverfahren zur Lösung der Laplacegleichung unter Randbedingungen. Die Laplacegleichung gehört zu den elliptischen partiellen Differentialgleichungen zweiter Ordnung. Die Ableitungen werden durch endliche Differenzen zwischen den Funtkionswerten auf einem Quadrat ausgedrückt, das den betrachteten Punkt umgibt. Der Laplaceoperator angewendet auf die Funkton $\varphi(x, y)$ lautet dann in diskretisierter Form, siehe die Berechnung der zweiten Ableitung für äquidistante Datenpunkte in Gl. 5.48,

$$\Delta\varphi(x, y) = (\partial_x^2 + \partial_y^2)\varphi(x, y) \longrightarrow \varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1} - 4\varphi_{i,j} \tag{8.3}$$

Für eine allgemeine partielle Differentialgleichung handelt es sich deshalb um die Lösung eines linearen Gleichungssystems der Form

$$Ax = b, \tag{8.4}$$

mit einer vorgegebenen Matrix A und einem vorgegebenen Vektor b, und einem gesuchten Vektor x.

Wenn die partiellen Ableitungen nur mit Hilfe von nächsten Nachbarpunkten diskretisiert sind, wie im Fall des Laplaceoperators in Gl. (8.3), so führt dies auf Gleichungen der Form

$$a_{ij}x_{i+1,j} + b_{ij}x_{i-1,j} + c_{ij}x_{i,j+1} + d_{ij}x_{i,j-1} + e_{ij}x_{i,j} = f_{ij}. \tag{8.5}$$

Ein iteratives Verfahren zur Lösung dieser Gleichung erhält man, wenn man Gl. (8.5) formal nach $x_{i,j}$ auflöst,

$$x_{i,j} = -(a_{ij}x_{i+1,j} + b_{ij}x_{i-1,j} + c_{ij}x_{i,j+1} + d_{ij}x_{i,j-1} - f_{ij})/e_{ij}. \tag{8.6}$$

Dies erlaubt es, einen verbesserten Funktionswert aus den Nachbarwerten auszurechnen.

Im Falle der Laplacegleichung $\Delta\varphi = 0$ lautet die Beziehung 8.6

$$\varphi_{i,j}^{neu} = \frac{1}{4}(\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1}). \tag{8.7}$$

Ein Programm für dieses iterative Verfahren ist das Unterprogramm "relax" im Programm "laplace.c".

In jedem Schritt des Verfahrens kann die Annäherung an die stabile Lösung durch Berechnung der Residuen

$$r_{i,j} = \varphi_{i,j}^{neu} - \varphi_{i,j} \tag{8.8}$$

verfolgt werden. Die Norm des Vektors der Residuen oder das maximale Residuum können als Abbruchkriterium verwendet werden.

Eine Verbesserung der Konvergenz kann man mit dem Simultanen Überrelaxations-Verfahren (SOR) errreichen. In diesem Verfahren werden die Residuen zu einer Berechnung der neuen Funktionswerte verwendet

$$\varphi_{i,j}^{neu} = \varphi_{i,j} + \omega r_{i,j}, \tag{8.9}$$

wobei der Parameter ω die Güte der Konvergenz beeinflussen soll.

Der Fehler wächst meist zuerst um einen Faktor von ungefähr 10 an, um erst dann abzufallen.

Ein praktischer Punkt besteht in der Art, wie das Gitter durchlaufen wird. Dies kann einerseits sequentiell erfolgen, wobei sich anfängliche Fehler jedoch leicht fortpflanzen können. Besser erscheint es, das Gitter in gerade und ungerade Punkte zu unterteilen. Die Funktionswerte an geraden Gitterpunkten sind dann nur von ungeraden Gitterpunkten abhängig und umgekehrt.

Dieses Verfahren soll nun durch Variation eines Relaxationsparameters beschleunigt werden.

Eine einfache Modifikation besteht in der Chebyshev-Beschleunigung, für die der Relaxationsparameter in jedem Halbschritt nach folgender Vorschrift geändert wird

$$\omega_0 = 1. \tag{8.10}$$

$$\omega_1 = \frac{1}{1 - \rho_{Jacobi}^2/2} \tag{8.11}$$

$$\omega_{n+1} = \frac{1}{1 - \rho_{Jacobi}^2 \omega_n/4} \tag{8.12}$$

$$\omega_\infty \rightarrow \omega_{optimal} \tag{8.13}$$

8.3 Berechnung der Bahn eines geladenen Teilchens

Wir wollen die Bahn eines Ions in einem Wienfilter numerisch untersuchen. Dazu sollen die Newtonsche Bewegungsgleichung eines geladenen Teilchens im Feld eines Wienfilters mithilfe der Runge-Kutta Methode integriert werden. Der Ort und die Geschwindigkeit des Teilchens werden bis auf die Genauigkeit $(\delta t)^5$ berechnet. Die Geschwindigkeit wird so angepasst, dass die Energie für jeden Zeitschritt erhalten ist. Eine kleine Zusatzkraft wird hinzugefügt, um den kanonischen Impuls in jedem Schritt zu erhalten.

Die Bewegung des Ions wird durch die Differentialgleichung

$$\frac{d^2\vec{r}(t)}{dt^2} = \frac{\vec{F}(\vec{r}(t), \vec{v}(t), t)}{m} = \vec{a}(\vec{r}(t), \vec{v}(t), t) \tag{8.14}$$

beschrieben. Die numerische Lösung dieser Differentialgleichung erfordert folgende Schritte:

- Definition der Anfangsgeschwindigkeit

$$\vec{v}(t_0) = \left.\frac{d\vec{r}(t)}{dt}\right|_{t_0} \tag{8.15}$$

- Integration von t_0 bis $t_0 + \delta t$

$$\begin{aligned} \vec{v}(t_0 + \delta t) &= \vec{v}(t_0) + \int_{t_0}^{t_0+\delta t} dt\ \vec{a}(\vec{r}(t), \vec{v}(t), t), \\ \vec{r}_0(t_0 + \delta t) &= \vec{r}(t_0) + \int_{t_0}^{t_0+\delta t} dt\ \vec{v}(\vec{r}(t), \vec{v}(t), t). \end{aligned} \tag{8.16}$$

Diese Integration führen wir nun mit einem Standardcode für die Lösung von Differentialgleichungen durch, z.B. der Runge-Kutta-Methode.

8.4 Aufgaben

- **Aufgabe 8.1: Lösung der Laplace Gleichung mit Hilfe des Programmes laplace.c**

  Schreiben Sie für jede 100. Iteration, den Fehler aus.

  - Führen Sie gerade und ungerade Punkte ein.
    Iterieren Sie abwechselnd über gerade und ungerade Punkte.
    Anleitung: Es reicht hier, ein "if" zu verändern.
  - Messen Sie den Fehler mit der Norm des Residuenvektors (Summe der Absolutwerte) und verwenden Sie diesen als modifiziertes Abbruchkriterium
  - Veränderen Sie den Überrelaxation-Faktor und untersuchen Sie die Schnelligkeit der Konvergenz.
  - Führen Sie die Chebyshev-Beschleunigung ein und untersuchen Sie die Abhängigkeit von ρ^2_{Jacobi}.

- **Aufgabe 8.2: Berechnung der Bahn eines geladenen Teilchens in einem Wienfilter**

- Für ein Wien–Filter der Länge L mit den Feldstärken $|E|$ und $|B|$, an dessen Ende eine Kollimatorblende des Durchmessers $2d$ angebracht ist, ist die relative Breite der Geschwindigkeitsverteilung der transmittierten Teilchen bestimmt durch die Gleichung:

$$|\frac{\Delta v}{v_0}| \leq \frac{m}{q}\frac{d}{L^2}\frac{|E|}{|B|^2}$$

  wobei m die Masse der Teilchen und q deren Ladung ist, und $v_0 = |E|/|B|$ die Sollgeschwindigkeit angibt.

- Stellen Sie die Bewegungsgleichung auf und bringen Sie diese in eine für die Runge-Kutta Methode geeignete Form.

- Berechnen Sie die Bahn eines 10 keV Ar^+ Ions in einem Wienfilter der Länge 10 cm mit einer Kollimatorblende deren Durchmesser 50μm beträgt . Das Program `wien.c` sowie die Hilfsroutinen in `rk4.h & rk4.c` können zu diesem Zweck verwendet werden. `wien.c` ruft die externe Funktion `derivs(float t, float * f, float * dfdt)` auf, welche die Ableitungen der Funktionen `f` zu einem Zeitpunkt `t` berechnet. Die Orts- und Geschwindigkeitsvektoren zum Zeitpunkt `t` sind bei Aufruf in dem Feld `f[1..6]`enthalten. Die Variablen `dfdt` beschreiben also die Zeitabhängigkeit der x- y- und z- Komponente der Ortsvektors (`dfdt[1]-dfdt[3]`) bzw. des Geschwindigkeitsvektors (`dfdt[4]-dfdt[6]`). Dieses Unterprogramm muss von Ihnen geschrieben werden. Der Einfachheit halber kann zuerst ein ideales Wienfilter verwendet werden in dem nur innerhalb der Kondensatorplatten ein ideales homogenes Feld herrscht. Überlegen Sie zuerst, wie die Felder dimensioniert werden müssen, sodaß nur jene Teilchen transmittiert werden deren Geschwindigkeit maximal 1% von der Sollgeschwindigkeit abweicht. Das Program `wien.c` schreibt die Teilchenbahn in die Datei `traj.dat`. Überprüfen Sie diese mit Gnuplot.
  Hinweis: Es empfiehlt sich zur Überprüfung des Programms zuerst einen einfacheren Fall zu rechnen in dem kein E-Feld herrscht, sondern nur ein homogenes B-Feld. Wenn die Anfangsgeschwindigkeit nicht parallel zum B-Feld gewählt wird, beschreibt das Teilchen eine Zyklotronbahn die einfach zu überprüfen ist.

- Verwenden Sie das Unterprogramm `randgauss(float mean, float width)` zur Erzeugung eines nicht monoenergetischen Ionenstrahls. Erstellen sie eine Häufigkeitsverteilung der anfänglichen Geschwindigkeitsverteilung und vergleichen Sie diese mit jener nach der Blende des Wienfilters. Vergleichen Sie das Ergebnis mit der o.a. Gleichung für die Breitte der transmittierten Geschwindigkeutsverteilung.

Kapitel 9

Das Wasserstoffatom

9.1 Die radiale Schrödingergleichung und ihre Lösungen

Für explizit zeitunabhängige Hamiltonoperatoren ist es zweckmäßig die Schrödingergleichung

$$H\Psi = i\hbar\frac{\partial}{\partial t}\psi \tag{9.1}$$

durch den Ansatz stationärer Lösungen $\Psi = \psi \exp(-iEt/\hbar)$ in die zeitunabhängige Schrödingergleichung überzuführen. Für ein Einteilchenproblem erhält man dann

$$\left\{-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r})\right\}\psi(\mathbf{r}) = E\psi(\mathbf{r})\,, \tag{9.2}$$

wobei $V(\mathbf{r})$ die Wechselwirkung ist, die ein Teilchen der Masse m erfährt.

Im folgenden beschränken wir uns auf sphärisch symmetrische Potentiale, d.h. $V(\mathbf{r}) = V(r)$, sodass eine Entwicklung von $\psi(\mathbf{r})$ nach Kugelflächenfunktionen $Y_{\ell m}(\theta,\varphi)$ zweckmäßig ist,

$$\psi(\mathbf{r}) = \sum_{\ell=0}^{\infty}\frac{u_\ell(r)}{r}\sum_{m=-\ell}^{+\ell} a_{\ell m}Y_{\ell m}(\theta,\varphi)\,. \tag{9.3}$$

Die Größen $a_{\ell m}$ sind geeignet gewählte Vorfaktoren, welche vom betrachteten physikalischen Problem anhängen. Einsetzen von (9.3) in (9.2) und Ausnützen der Orthogonalität der Kugelflächenfunktionen

$$\int_0^\pi d\theta\sin\theta\int_0^{2\pi} d\varphi Y^*_{\ell m}(\theta,\varphi)Y_{\ell' m'}(\theta,\varphi) = \delta_{\ell\ell'}\delta_{mm'} \tag{9.4}$$

führt zur sogenannten radialen Schrödingergleichung,

$$\left\{-\frac{\hbar^2}{2m}\left(\frac{d^2}{dr^2} - \frac{\ell(\ell+1)}{r^2}\right) + V(r)\right\}u_\ell(r) = Eu_\ell(r)\,. \tag{9.5}$$

In praktischen Rechnungen wird diese Gleichung meist in geeigneter Form skaliert, d.h. es werden die Einheiten an das betrachtete Problem angepasst.

Bedingt durch den Zentrifugalwall $\ell(\ell+1)/r^2$ tritt deshalb für $\ell \neq 0$ eine Singularität bei $r = 0$ auf. Wie bereits in den Methoden der theoretischen Physik gezeigt wurde, sprechen wir bei $r = 0$ von einer außerwesentlichen singulären Stelle. Die Lösung kann an dieser Stelle als verallgemeinerte Potenzreihe angesetzt werden, d.h.

$$u_\ell(r) = \sum_{i=0}^{\infty} c_i r^{\alpha+i} \,. \tag{9.6}$$

Einsetzen des Potenzreihenansatzes (9.6) in die radiale Schrödingergleichung (9.5) und Vergleich der Koeffizienten der niedrigsten Potenz $r^{\alpha-2}$ liefert die zwei möglichen Werte für α. Bezüglich des Verhaltens bei $r = 0$ unterscheidet man daher zwei Typen von Lösungen:

$$\lim_{r\to 0} \left\{ \varphi_\ell(r) r^{-(\ell+1)} \right\} = a \qquad \text{reguläre Lösung}\,, \tag{9.7}$$

$$\lim_{r\to 0} \left\{ \varphi_\ell(r) r^{\ell} \right\} = b \qquad \text{irreguläre Lösung}\,. \tag{9.8}$$

wobei a und b Konstanten sind. Abgesehen von der Normierung ist die reguläre Lösung durch die Randbedingung (9.7) eindeutig bestimmt. Im Gegensatz dazu bilden die irregulären Lösungen (9.8) eine eindimensionale Mannigfaltigkeit. Unter den irregulären Lösungen sind besonders die *Jostlösungen* hervorzuheben, weil sie in analytischen Behandlungen eine wichtige Rolle spielen. Die Jostlösungen sind durch die Randbedingungen

$$\lim_{r\to\infty} \left\{ e^{\mp ikr} \; f_\ell^{\pm}(k,r) \right\} = 1 \tag{9.9}$$

gegeben. Eine lineare Differentialgleichung 2. Ordnung hat zwei linear unabhängige Lösungen. Die beiden Jostlösungen $f_\ell^+(k,r)$ und $f_\ell^-(k,r)$ stellen ein derartiges Duppel dar. Zwei Funktionen $\varphi_\ell(k,r)$ und $\eta_\ell(k,r)$ sind linear unabhängig, wenn ihre *Wronski-Determinante*

$$W\{\varphi_\ell, \eta_\ell\} = \varphi_\ell \eta_\ell' - \varphi_\ell' \eta_\ell \tag{9.10}$$

nicht verschwindet. Die Wronski-Determinante der beiden Jostlösungen ergibt $W\{f_\ell^+, f_\ell^-\} = 2ik$, die beiden Jostlösungen sind deshalb linear unabhängige Funktionen.

9.2 Das Numerov Verfahren

Die radiale Schrödingergleichung (9.5) ist eine gewöhnliche Differentialgleichung zweiter Ordnung. Sie kann auf die Form

$$\left[\frac{d^2}{dr^2} + w(r) \right] y(r) = S(r) \tag{9.11}$$

mit verschwindendem Quellterm, $S(r) = 0$, gebracht werden. Für zwei Teilchen, die über ein Potential $V(r) = V(\mid \vec{r}_2 - \vec{r}_1 \mid)$ wechselwirken, hat $w(r)$ die Form

$$w(r) = \frac{2\mu}{\hbar^2}\,(E - V(r)) - \frac{\ell(\ell+1)}{r^2}, \tag{9.12}$$

wobei μ die reduzierte Masse des Systems, ℓ die Bahndrehimpulsquantenzahl und E die Schwerpunktsenergie sind.

Auch die radiale Poissongleichung für eine sphärisch symmetrische Ladungsverteilung ist von der Form von Gleichung (9.11),

$$\left[\frac{d^2}{dr^2} - \frac{\ell(\ell+1)}{r^2}\right]\varphi(r) = -r\frac{\rho(r)}{\epsilon_0}\,, \tag{9.13}$$

wobei $\varphi(r)/r$ das elektrostatische Potential und $\rho(r)$ die elektrische Ladungsverteilung sind.

Die numerische Lösung der Differentialgleichung (9.11) kann im Prinzip durch ein Einschrittverfahren, z.B. das Runge-Kutta-Verfahren, erfolgen. Diese Vorgangsweise ist zweckmäßig, falls neben der Funktion $y(r)$ auch die Ableitung dy/dr in der weiteren Rechnung benötigt wird. In den meisten Fällen geht allerdings die erste Ableitung nicht explizit ein.

Eine sehr einfache Methode zur numerischen Lösung von Differentialgleichungen von Typ (9.11) geht auf Numerov [18] zurück. Man bezeichnet den Algorithmus auch als *Cowling Methode* oder *Fox-Godwin-Verfahren* [19]. Wir definieren nun ein äquidistantes Gitter $\{r_n = r_0 + nh\}, n = 0, 1, 2, \ldots$ auf dem Definitionsbereich und entwickeln die Lösung $y(r)$ an der Stelle $y_n = y(r_n)$ in eine Taylor-Reihe. Man erhält somit für

$$y_{n\pm1} = y_n \pm h\; y'_n + \frac{h^2}{2}\; y''_n \pm \frac{h^3}{6}\; y'''_n + \frac{h^4}{24}\; y_n^{(\mathrm{iv})} \pm \frac{h^5}{120}\; y_n^{(\mathrm{v})} + O(h^6) \tag{9.14}$$

Aus Gleichung (9.14) ergibt sich der Ausdruck

$$\frac{y_{n+1} - 2\; y_n + y_{n-1}}{h^2} = y''_n + \frac{h^2}{12}\; y_n^{(\mathrm{iv})} + O(h^4)\,. \tag{9.15}$$

Die zweite Ableitung y''_n kann also als Differenzenausdruck plus einem Korrekturterm dargestellt werden, welcher die vierte Ableitung der Funktion enthält. Der charakteristische Schritt des Numerov-Verfahrens besteht nun in der Darstellung der vierten Ableitung durch die Differentialgleichung (9.11),

$$y^{(\mathrm{iv})} = \frac{d^2}{dr^2}y'' = \frac{d^2}{dr^2}\left[-w(r)y(r) + S(r)\right]\,. \tag{9.16}$$

Damit lässt sich der Korrekturterm wieder durch die rechte Seite der Differentialgleichung ausdrücken, sodass wir nun folgenden Differenzenausdruck für die Lösung der

Differentialgleichung erhalten,

$$\begin{aligned} y_n'' &= \frac{y_{n+1}-2y_n+y_{n-1}}{h^2} - \frac{h^2}{12}y_n^{(iv)} + O(h^4) = \frac{y_{n+1}-2y_n+y_{n-1}}{h^2} \\ &+ \frac{h^2}{12}\left[\frac{w_{n+1}y_{n+1}-2w_ny_n+w_{n-1}y_{n-1}}{h^2} - \frac{S_{n+1}-2S_n+S_{n-1}}{h^2} + O(h^2)\right] \\ &= -w_ny_n + S_n + O(h^4)\,. \end{aligned} \tag{9.17}$$

Ordnet man nun die Terme, so ergibt sich die Grundgleichung für die Lösung der Differentialgleichung (9.11) im Numerov-Verfahren

$$\begin{aligned} &y_{n+1}\left[1+\frac{h^2}{12}w_{n+1}\right] - y_n\left[2-\frac{10h^2}{12}w_n\right] + y_{n-1}\left[1+\frac{h^2}{12}w_{n-1}\right] \\ &\quad = \frac{h^2}{12}\left[S_{n+1}+10S_n+S_{n-1}\right] + O(h^6)\,. \end{aligned} \tag{9.18}$$

Das Numerov-Verfahren ist konsistent und konvergent in vierter Ordnung und eignet sich bestens für die Lösung von Anfangswertproblemen. Für die praktische Rechnung ist es zweckmäßig die Größe $Q(r)$ zu definieren,

$$Q_n = \left[1+\frac{h^2}{12}w_n\right] y_n\,. \tag{9.19}$$

Die Rekursionsformel (9.18) reduziert sich dann auf die einfache Form

$$Q_{n+1} + 10Q_n + Q_{n-1} - 12y_n = \frac{h^2}{12}\left[S_{n+1}+10S_n+S_{n-1}\right] + O(h^6)\,, \tag{9.20}$$

welche eine Rekursionsvorschrift für die Hilfsgröße $Q(r)$ darstellt.

9.3 Die numerische Lösung der radialen Schrödingergleichung

Im folgenden beschränken wir uns auf die Betrachtung der radialen Schrödingergleichung, d.h. die Inhomogenität $S(r) = 0$ und $w(r)$ ist durch Gleichung (9.12) gegeben. Die Wellenfunktion physikalischer Probleme ist meist die reguläre Lösung. Gemäß Gl. (9.7) ist die reguläre Lösung durch die Randbedingung am Ursprung charakterisiert,

$$\lim_{r\to 0} y(r) = r^{\ell+1}\,. \tag{9.21}$$

Dies bedeutet, dass die reguläre Lösung $y(r)$ bei $r = 0$ unabhängig von der Bahndrehimpulsquantenzahl ℓ verschwindet, d.h. $y_0 = y(r = 0) = 0$. An der ersten Stützstelle wird vorerst ein beliebiger Wert $y_1 = y(r = h) = a$ für die Lösung angesetzt. Dies stellt keine Einschränkung dar, da aufgrund der Linearität der Schrödingergleichung,

die Lösung nachträglich auf den richtigen Wert normiert werden kann. Ausgehend von den Anfangswerten $y_0 = 0$ und $y_1 = a$ und den damit bekannten Werten der Hilfsfunktion, Q_0 und Q_1, wird dann die Lösung $y_n, n = 2, 3, \ldots$ rekursiv über die Gleichung (9.20) konstruiert:

$$Q_2 = 12y_1 - 10Q_1 - Q_0 \quad \text{und} \quad y_2 = Q_2 \left[1 + \frac{h^2}{12} w_2\right]^{-1} \tag{9.22}$$

$$Q_3 = 12y_2 - 10Q_2 - Q_1 \quad \text{und} \quad y_3 = Q_3 \left[1 + \frac{h^2}{12} w_3\right]^{-1} \tag{9.23}$$

$$\ldots\ldots \tag{9.24}$$

$$Q_{n+1} = 12y_n - 10Q_n - Q_{n-1} \quad \text{und} \quad y_{n+1} = Q_{n+1} \left[1 + \frac{h^2}{12} w_{n+1}\right]^{-1} \tag{9.25}$$

$$\ldots\ldots \tag{9.26}$$

Die Durchführung dieser Rekursion ist unproblematisch. Einzig bei der Bestimmung von Q_0 muss die Singularität des Zentrifugalwalls berücksichtigt werden. Eine genauere Betrachtung liefert:

$$\begin{aligned} Q_0 &= \lim_{r\to 0} \left[1 - \frac{h^2}{12} \frac{\ell(\ell+1)}{r^2}\right] y(r) \\ &= -\frac{h^2}{12} \frac{\ell(\ell+1)}{r^2} a_0 r^{\ell+1} \\ &= -\frac{1}{6} y_1 \delta_{\ell,1}\,, \end{aligned} \tag{9.27}$$

wobei wir die Reihenentwicklung für $y_1 = a_0 h^2$ eingesetzt haben. Der Wert der Hilfsfunktion bei $r = 0$, weicht also nur bei $\ell = 1$ von Null ab.

Ein Beispiel sind die mit diesem Verfahren bestimmten regulären Lösungen der radialen Schrödingergleichung mit $V = 0$ bei verschiedenen Bahndrehimpulsquantenzahlen ℓ. Für die numerische Rekursion wurde eine Schrittweite von $h = \Delta\rho = 0.1$ verwendet. Die erhaltenen Lösungen entsprechen den sphärischen Besselfunktionen und weisen bei dieser Gitterannahme eine mittlere Abweichung von weniger als 0.1% auf.

Im Rahmen theoretischer Rechnungen ist auch die Berechnung von irregulären Lösungen erforderlich. Diese sind im allgemeinen durch asymptotische Randbedingungen, d.h. bei $r \to \infty$, definiert. Betrachten wir als Beispiel die Berechnung der Jostlösungen $f_\ell^\pm(\rho)$. Diese sind durch die asymptotische Randbedingung

$$\lim_{\rho\to\infty} \frac{f_\ell^\pm(\rho)}{H_\ell^\pm(\rho)} = 1 \tag{9.28}$$

festgelegt, wobei $H_\ell^\pm(\rho)$ die einlaufenden $(-)$ und auslaufenden $(+)$ Hankelfunktionen sind. Für die numerische Rechnung der Jostlösungen geht man daher von Stützpunkten r_N und r_{N+1} aus, die außerhalb des Wechselwirkungsbereiches liegen, d.h. $V_N = V_{N+1} = 0$. An diesen Stützstellen sind daher die Lösungen durch

$$y_N = f_\ell^\pm(kr_N) = H_\ell^\pm(k, r_N) \quad \text{und} \quad y_{N+1} = f_\ell^\pm(kr_{N+1}) = H_\ell^\pm(kr_{N+1}) \tag{9.29}$$

gegeben. Die Werte der Lösung y_n für $n = N-1, N-2, \ldots, 1$ werden nach Gleichung (9.20) durch Einwärtsrekursion

$$Q_{n-1} = 12y_n - 10Q_n - Q_{n+1} \quad \text{und} \quad y_{n-1} = Q_{n-1}\left[1 + \frac{h^2}{12}w_{n-1}\right]^{-1} \tag{9.30}$$

rekursiv bestimmt. Der Funktionswert y_0 kann abgesehen von $\ell = 0$ nicht bestimmt werden. Dies entspricht dem singulären Verhalten irregulärer Lösungen für $r \to 0$.

9.4 Berechnung von Bindungszuständen

Ein wichtiges Problem in der Atom- und Kernphysik ist die Bestimmung von Bindungszuständen von Einteilchenproblemen. Dabei ist es unwesentlich, ob ein tatsächliches Einteilchenproblem wie im H-Atom oder ein effektives Einteilchenproblem (z.B. Hartree-Fock-Modell) vorliegt. Die Einteilchenzustände stellen den Ausgangspunkt für Berechnungen der Atom- und Kernstruktur jedes Vielteilchensystems dar. Die mathematische Problemstellung beteht für Bindungszustände in der Bestimmung der Eigenzustände und Eigenwerte im diskreten Spektrum des Einteilchen-Hamiltonoperators H. Für ein sphärisch symmetrisches Potential $V(r)$ lautet die radiale Schrödingergleichung

$$\left\{-\frac{\hbar^2}{2\mu}\left[\frac{d^2}{dr^2} - \frac{\ell(\ell+1)}{r^2}\right] + V(r)\right\} u_{n\ell}(r) = \epsilon_{n\ell} u_{n\ell}(r). \tag{9.31}$$

Die Eigenfunktionen sind unter der Nebenbedingung der Normierbarkeit

$$\int_0^\infty dr u_{n\ell}^2(r) = 1 \tag{9.32}$$

zu bestimmen. Die Forderung der quadratischen Normierbarkeit der Lösung $u_{n\ell}(r)$ liefert eine Einschränkung auf diskrete Energieeigenwerte $\epsilon_{n\ell}$ (siehe z.B. die analytische Ableitung der Bindungsenergien im Wasserstoffatom). Gilt für das Potential $\lim_{r\to\infty} V(r) = 0$, dann haben alle Bindungszustände negative Energieeigenwerte, $\epsilon_{n\ell} < 0$.

Aufgrund ihrer Normierbarkeitsforderung sind die Eigenfunktionen für Bindungszustände reguläre Lösungen der radialen Schrödingergleichung zu einer vorerst unbekannten Bindungsenergie $\epsilon_{n\ell}$. Für $r \to 0$ hat $u_{n\ell}(r)$ die Eigenschaft

$$\lim_{r\to 0} u_{n\ell}(r) r^{-(\ell+1)} = a_{n\ell}\,, \tag{9.33}$$

wobei $a_{n\ell}$ eine Konstante ist, die dazu dient, die Normierungsbedingung (9.32) zu erfüllen. Für asymptotische r-Werte muss die Bindungswellenfunktion dem Grenzwert

$$\lim_{r\to\infty} \frac{u_{n\ell}(r)}{H_{n\ell}^+(i\kappa_{n\ell} r)} = \lim_{r\to\infty} \frac{u_{n\ell}(r)}{e^{-\kappa_{n\ell} r}} = A_{n\ell} \tag{9.34}$$

genügen, wobei

$$\kappa_{n\ell} = \sqrt{-\frac{2\mu\epsilon_{n\ell}}{\hbar^2}}\,. \tag{9.35}$$

Durch Integration der radialen Schrödingergleichung von großen r-Werten beginned, lässt sich Q_0 bestimmen. Ein Energieeigenwert ist dann gefunden, wenn $Q_0 = -\frac{1}{6}\delta_{1\ell}u_{n\ell}(h)$ erfüllt. Der Energieeigenwert lässt sich also also Nullstellensuche der Funktion $Q_0(\epsilon_{n\ell})$ formulieren. Man bezeichnet solche Verfahren allgemein auch als Schießverfahren.

9.5 Aufgaben

- **Aufgabe 9.1: Fox-Goodwin-Algorithmus**

  a) Konstruieren Sie einen Modul auf der Basis des Numerov-Verfahrens zur Lösung der radialen Schrödingergleichung mit vorgegebenen Potential $V(r)$, Schwerpunktsenergie E, reduzierter Masse m und Bahndrehimpulsquantenzahl ℓ.

  b) Berechnen Sie durch numerische Lösung der Differentialgleichung die sphärischen Besselfunktionen $j_\ell(\rho)$ und die sphärischen Neumannfunktionen $n_\ell(\rho)$.

  c) Untersuchen Sie die Genauigkeit der Lösung als Funktion der verwendeten Schrittweite.

- **Aufgabe 9.2: Berechnung von Energieeigenwerten**

  a) Konstruieren Sie ein Modul zur Bestimmung der Energieeigenwerte der radialen Schrödingergleichung mittels Einfachschießverfahren. Verwenden Sie die Integration von außen und den Abgleich auf $Q_0 = -\frac{1}{6}y_1\delta_{\ell 1}$ als Kriterium.

  b) Bestimmen Sie die Energieeigenwerte und die normierten Bindungswellenfunktionen des Wasserstoffatoms bis zur Hauptquantenzahl $N = 3$. Vergleichen Sie die erhaltenen Energieeigenwerte und Bindungswellenfunktionen mit den exakten Lösungen. Überprüfen Sie die Orthogonalität der Bindungswellenfunktionen.

Literaturverzeichnis

[1] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. *Numerical Recipes in Fortran 77* (Cambridge University Press, 1992).

[2] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. *Numerical Recipes in C* (Cambridge University Press, 1992).

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in Fortran 90.* (Cambridge University Press, 1996)

[4] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical Recipes in C++* (Cambridge University Press, 2002).

Kapitel 2

[5] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions* (National Bureau of Standards, Washington, 1972), 10. Auflage. `[Kapitel 2]`

[6] Gillman und Fiebig, Computers in Physics, Jän/Feb (1988) 62. `[Kapitel 2]`

Kapitel 3

[7] Donald Knuth, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms* (Addison-Wesley, 1981). `[Kapitel 3]`

[8] Robert Sedgewick, *Algorithmen* (Addison-Wesley, 1991). `[Kapitel 3]`

[9] Christoph Überhuber, *Computernumerik I und II* (Springer, 1995). `[Kapitel 3]`

[10] Downloads und Hinweise unter `http://www.cs.sunysb.edu/ algorith/files/random-numb`
`[Kapitel 3]`

Kapitel 4

[11] J. Stoer, *Einführung in die Numerische Mathematik I* (Springer, Berlin-Heidelberg, 1972). `[Kapitel 4]`

Kapitel 5

[12] C.F. Gerald und P.O. Wheatley, *Applied Numerical Analysis* (Addison-Wesley, Reading, 1989). [Kapitel 5]

[13] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions* (National Bureau of Standards, Washington, 1972), 10. Auflage. [Kapitel 5]

[14] W. H. Press, B. P. Flannery, S.A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge University Press, 1988). [Kapitel 5]

Kapitel 6

[15] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes in Fortran77* (second edition, Cambridge University Press, 1995). [Kapitel 6]

Kapitel 7

[16] Stoer, Burlisch, *Numerische Mathematik II* (Springer, Heidelberg, 1980). [Kapitel 7]

[17] J.R. Cash, A.H. Karp, ACM Trans. Math. Software **16**, 201 (1990).

phantomabcdef

Kapitel 9

[18] B. Numerov, Publs.observatoire central astrophys.Russ., **2** (1933) 188. [Kapitel 9]

[19] L. Fox, E. T. Goodwin, Proc. Camb. Phil. Soc. **45** (1949) 373. [Kapitel 9]