Carl v. Ossietzky
UNIVERSITÄT
OLDENBURG

Skriptum zur Vorlesung

„Computerphysik“

1. Version Sommersemester 2000

Dr. V. Hohmann
Fachbereich Physik
Universität Oldenburg

Inhaltsverzeichnis

1 Einleitung

Inhalt der Vorlesung ist nicht die Physik des Computers sondern das Lösen physikalischer Probleme mit Hilfe des Computers. Dem Einsatz des Computers als Werkzeug der Physik kommt heute eine wichtige Bedeutung sowohl in der experimentellen als auch in der theoretischen Physik zu. Insbesondere sind folgende Bereiche zu nennen:

1. Meßwertaufnahme
2. Apparatursteuerung und -kontrolle
3. Meßwertverarbeitung (Funktionenberechnung, Visualisierung)
4. Umsetzung theoretischer Modelle auf den Rechner

Meßwertaufnahme und Apparatursteuerung lassen sich mit Hand und Auge nicht mehr bewältigen und die Hilfe schneller Computer ist bei den meisten Apparaturen notwendig. Die Meßwertverarbeitung erlaubt die Auswertung großer Datenmengen und ersetzt Generationen von ‚menschlichen Computern', die sich zwangsläufig der Berechnung von Funktionswerten aus Tabellen und dem Anfertigen von Diagrammen per Hand widmen mußten. Die Nutzung des Computers für die Modellierung ist sicher die komplexeste Anwendung des Computers in der Physik, die immer dann erforderlich wird wenn analytische (mathematische) Lösungen nicht existieren. Als Beispiele seien das Dreikörperproblem, das nichtlineare (physikalische) Pendel sowie die Lösung der Schrödingergleichung für komplexe Atome genannt. In diesen Fällen werden numerische Verfahren auf dem Computer angewendet, die die (Näherungs-)Lösung durch Berechnungen von Zahlen nach einfachen, wiederkehrenden Regeln (Algorithmen) suchen[1]. Die Validität solcher numerischer Lösungen und der eingesetzten numerischen Verfahren ist immer kritisch zu beleuchten und ein gesundes Mißtrauen gegen die damit erzielten Ergebnisse ist angebracht. Dies läßt sich mit einem Satz zusammenfassen: Es geht um Einsicht, nicht um Zahlen.

Lernziel der Veranstaltung ist es, die Grundlagen dieser Anwendungen, insbesondere der Punkte 3. und 4. anhand einfacher Beispiele zu verstehen und eigene praktische Fähigkeiten zu erlernen, die später auf konkrete Probleme angewandt bzw. erweitert werden können. Auch wenn dies manchmal sehr komplexe Aufgaben sein können: Letztlich handelt sich sind es immer ‚nur' um Berechnungen von Zahlen nach einfachen Regeln.

1.1 Begriffsbestimmung: Computerphysik

Der Begriff „Computerphysik" soll ausgehend von der Mathematik definiert werden:

1. Mathematik (z.B. Analysis)
- beschäftigt sich mit rel. abstrakten Begriffen und arbeitet eher mit Symbolen als mit Zahlen (wobei Zahlen natürlich auch Symbole sind, die aber als abstrakte Menge beschrieben werden, z.B. Menge der natürlichen Zahlen)
- Lösungen sind immer analytisch, d.h. im Prinzip beliebig genau bestimmt; Ausnahmen von dieser Regel werden theoretisch behandelt (z.B. Singularitäten)
- falls analytische Lösungen nicht bekannt sind, werden Existenzbeweise oder Nicht-Existenzbeweise untersucht

2. Numerische Mathematik
- befaßt sich mit der Berechnung von Zahlen nach einfachen, wiederkehrenden Regeln (Algorithmen)
- befaßt sich mit prinzipiellen Fragen der Berechenbarkeit von Zahlen (Konvergenzordnungen)
- Validierung von Algorithmen aufgrund von Fehlerabschätzungen

3. Numerische Mathematik mit dem Computer
- Numerik + (unerkannte) Genauigkeitsverluste (z.B. durch begrenzte Rechengenauigkeit)
4. Computerphysik

[1] Der Computer hilft natürlich auch beim Auffinden analytischer Lösungen, jedoch sind die entsprechenden Programme der Computeralgebra (z.B. Maple oder Mathematica) wissensbasiert, d.h. sie finden keine Lösungen, die den ProgrammentwicklerInnen prinzipiell unbekannt waren.

- Anwendung von 3. auf physikalische Probleme
- zusätzliche Möglichkeit der Validierung von Algorithmen aufgrund physikalischer Randbedingungen

2 Matlab - erste Schritte

Zur Durchführung der Übungen wird Matlab als spezielles Programm für die numerische Mathematik eingesetzt, das einfach zu bedienen ist und einen schnellen Einstieg in die Welt der numerischen Mathematik ermöglicht. Matlab stellt eine sehr gute Ergänzung zu den bekannten Programmen der symbolischen (analytischen) Mathematik (Maple, Mathematica) dar und kommt immer dann zum Einsatz, wenn keine geschlossenen analytischen Lösungen eines Problems angegeben werden können. Die grundlegende Datenstruktur von Matlab sind zweidimensionale Matrizen (daher auch der Name: MATrix LABoratory). Matlab kann mathematische Berechnungen in einer den Programmiersprachen C bzw. FORTRAN ähnlichen Syntax ausführen. Alle grundlegenden mathematischen Funktionen sowie spezielle Funktionen (Bessel etc.) sind enthalten. Die wesentlichen aus der numerischen Mathematik bekannten Algorithmen (Lösung linearer Gleichungssysteme, Eigenwerte/Eigenvektoren, Differentialgleichungen etc.) sind implementiert. Weiterhin beinhaltet das Programm Möglichkeiten zur grafischen Darstellung von Funktionen und Datensätzen. Es kann zweidimensionale xy-Plots, Plots in Polarkoordinaten sowie dreidimensionale Darstellungen erzeugen. Für spezielle Anwendungen (Bildverarbeitung, Signalverarbeitung, etc.) stehen sog. Toolboxes mit weiteren Funktionen zur Verfügung, die nicht im Grundpaket enthalten sind.

Die folgende Tabelle zeigt die Vor- und Nachteile von Matlab gegenüber Programmiersprachen wie C oder Fortran:

Vorteile	**Nachteile**
schnelle Programmentwicklung	unter Umständen langsamere Programmausführung im Vergleich zu optimierten C-Programmen
sehr gute Grafikmöglichkeiten	
viele wichtige Algorithmen sind bereits enthalten	
portable Programmierung	

Im CIP-Raum des Fachbereichs Physik steht Matlab in der Version 5.3 mit der Signalverarbeitungs-Toolbox zur Verfügung. Das Programm läuft unter Windows NT. Da Matlab an Lizenzen gebunden ist, kann es nicht auf andere Rechner transferiert werden! Zur Nutzung ist ein Login für das Windows NT-Netzwerk des Rechenzentrums sowie die Zugehörigkeit zur Nutzergruppe fb8 notwendig (Details siehe http://www.physik.uni-oldenburg.de/docs/cip/start.html)

Im folgenden werden die grundlegende Struktur von Matlab sowie die wichtigsten Befehle erläutert. Daran anschliessend finden sich kleinere Aufgaben, anhand derer der Umgang mit Matlab praktisch erprobt werden kann.

2.1 Bedienung von Matlab

Nach Aufruf von Matlab erscheint ein Kommandozeileneditor, in dem Kommandos eingegeben werden, die nach Drücken der Return-Taste (↵) vom Programm abgearbeitet werden. Bereits vorher abgearbeitete Befehlszeilen können durch Drücken der Pfeiltasten (↑↓) wieder hervorgeholt werden. Gibt man einen Anfangsbuchstaben ein und bedient dann die Pfeiltasten, erscheinen nur die vorherigen Befehle, die mit diesem Buchstaben beginnen. Die Kommandozeile kann so editiert werden, wie es aus Textverarbeitungsprogrammen bekannt ist (Markieren von Zeichen/Wörtern mit der Maus, Ersetzen und Einfügen von Zeichen/Wörtern, etc.).

Matlab sucht die auf der Kommandozeile eingegebenen Befehle in den im Suchpfad angegebenen Verzeichnissen. Mit dem Befehl path kann der Suchpfad ergänzt werden, falls z.B. ein eigenes Verzeichnis mit selbstgeschriebenen Kommandos in den Suchpfad aufgenommen werden soll.

Durch Eingabe des Befehls diary *dateiname* merkt sich Matlab alle eingegebenen Befehle sowie die „Antworten“ von Matlab in der angegebenen Datei. Damit kann die Abfolge der Befehle und Ergebnisse nachträglich nachvollzogen werden.

Matlab merkt sich die Werte aller Variablen, die während des Programmablaufs erzeugt werden im Speicher (dies wird als „Workspace“ bezeichnet). Mit den Befehlen save und load kann der Workspace in eine Datei abgespeichert und zur Fortsetzung der Arbeit später wieder aus der Datei geladen (rekonstruiert) werden. Mit dem Befehl clear können die Variablen im Workspace gelöscht werden. Die Befehle können durch Anhängen

von Variablennamen auf diese beschränkt werden. Zum Beispiel würde der Befehl save test.dat x y nur die Variablen x und y in der Datei test.dat speichern. Die Befehle who und whos listen die aktuell im Workspace vorhanden Variablen auf, wobei whos genauere Informationen liefert.

Die Grafik erscheint in zusätzlichen Fenstern auf dem Bildschirm. Mit dem Befehl figure wird ein neues Grafikfenster erzeugt, so daß mehrere Grafiken erzeugt werden können.

2.2 Hilfefunktionen

Hilfe zu den einzelnen Befehlen erhält man durch Eingabe von help *Befehlsname*, wobei help ohne weitere Argumente eine Liste von möglichen Hilfethemen liefert. Ist der Name eines Befehls nicht bekannt, so empfiehlt sich die Suche unter relevanten Hilfethemen sowie die Benutzung des Befehls lookfor *Stichwort*, wobei das Stichwort etwas mit dem gesuchten Befehl zu tun haben sollte (in Englisch!).

Zugriff auf die gesamte Dokumentation erhält man, wenn der Befehl helpdesk eingegeben wird und ein Internet-Browser (Netscape) gestartet ist. Dann erscheint die Dokumentation im HTML-Format. Interessant für Anfänger ist insbesondere das Handbuch „Getting started“, das eine detaillierte Einführung in Matlab enthält.

Weiterhin können „Examples and Demos“ vom Help-Menü aus aufgerufen werden. Dort finden sich Beispiele nach Themengebieten geordnet. Meist werden auch die benutzten Matlab-Befehle mit angezeigt, so daß die Beispiele als Vorlage für eigene Anwendungen dienen können.

2.3 Variablen

Namen für Variablen können frei vergeben werden. Groß- und Kleinschreibung wird dabei berücksichtigt. Variablen werden durch Zuweisung eingeführt und müssen nicht deklariert werden. Der Typ der Variablen (Skalar, Vektor, Matrix) wird von Matlab anhand der Zuweisung festgestellt. Beispiele:

```
a = pi                  % a wird der Wert Pi=3.1415... zugewiesen
b = [1 2]               % b wird der (Zeilen-)Vektor (1, 2) zugewiesen
c = [1 ; 2]             % c wird der (Spalten-)Vektor (1, 2) zugewiesen
d = [[1 2 3];[4 5 6i]]  % d wird eine 2X3 Matrix zugewiesen
```

(**Hinweise**: Der Text nach dem %-Zeichen wird als Kommentar gewertet.
Die Variable pi ist fest vordefiniert.
Die Variable $i = \sqrt{-1}$ ist zur Eingabe komplexer Größen fest vordefiniert)

Der Wert einer Variablen kann jederzeit angezeigt werden, indem der Name der Variablen ohne Zusätze in der Kommandozeile eingegeben wird. Beispiele:

```
a               % Eingabe der Variablen
a = 3.14        % Antwort von MATLAB (Skalar)

b               % Eingabe der Variablen
b = 1   2       % Antwort von MATLAB (Zeilenvektor)

c               % Eingabe der Variablen
c = 1           % Antwort von MATLAB (Spaltenvektor)
    2

d               % Eingabe der Variablen
d = 1 2 3       % Antwort von MATLAB (2X3 Matrix)
    4 5 6i
```

Es lassen sich auch Elemente oder Bereiche von Matrizen und Vektoren anzeigen bzw. an Variablen zuweisen. Beispiele:

```
d(1,3)              % Anzeigen der ersten Zeile, dritte Spalte
d = 3               % Antwort von MATLAB

d(1,2:3)            % Anzeigen der ersten Zeile, Elemente 2 bis 3
d = 2 3             % Antwort von MATLAB

z = d(2,1:3)        %  Zuweisung von Zeile 2, Spalten 1 bis 3 an z
z = 4 5 6i          % Antwort von MATLAB

d(2,2) = 3          % Zuweisung des Wertes 3 an die Matrix d (Zeile 2, Spalte 2)
d = 1 2 3           % Antwort von MATLAB (2X3 Matrix)
    4 3 6i

d(1,:)          % Adressierung der ersten Zeile
  = 1 2 3       % Antwort von MATLAB

d(1,1:2:3)      % Adressierung der ersten Zeile, jedes 2. Element)
  = 1  3        % Antwort von MATLAB

d(1,[3 1])      % Adressierung der ersten Zeile, 3. und 1. Element)
  = 3  1        % Antwort von MATLAB
```

(**Hinweise:** Jede Operation veranlaßt Matlab seine Arbeit auf dem Bildschirm zu dokumentieren. Dies kann unterbunden werden, indem einem Befehl ein Semikolon (;) nachgestellt wird. Die Indizierung von Elementen von Vektoren/Matrizen beginnt immer bei Index 1.)

2.4 Operatoren

Der Hochkomma-Operator ‘ (**Achtung**: nicht mit einfachen oder doppelten Anführungsstrichen verwechseln!) führt eine komplexe Konjugation durch:

```
y = c';         % Wandle c in einen Zeilenvektor um
y               % Ergebnis zeigen
y = 1 2         % Antwort von Matlab

y = d';         % berechne komplex konjugierte Matrix
y               % Ergebnis zeigen
y = 1  4        % Antwort von MATLAB (3X2 Matrix)
    2  5
    3 -6i
```

Der Operator .‘ führt dagegen eine Transposition durch:

```
y = d.';        % berechne transponierte Matrix
y               % Ergebnis zeigen
y = 1  4        % Antwort von MATLAB (3X2 Matrix)
    2  5
    3  6i
```

Die bekannten Operatoren +,-,*,/ funktionieren unter Matlab wie bei einem Taschenrechner. Die Anwendung dieser Operatoren auf Matrizen und Vektoren kann jedoch zu Verwirrungen führen, da Matlab die Variablen links und rechts der Operatoren analysiert und die Operatoren 'intuitiv' einsetzt, d.h. je nach Dimension der Variablen skalare oder Matrixoperationen einsetzt. Beispiele:

```
y = a * a;      % multipliziere zwei Skalare
y               % Ergebnis zeigen
y = 9.8696      % Antwort von Matlab
```

```
y = b * c;        % multipliziere Zeilen und Spaltenvektor
y                 % Ergebnis ist ein Skalarprodukt
y = 5

y = b .* c'       % multipliziere komponentenweise zwei Zeilenvektoren
y = 1 4           % Ergebnis ist wieder ein Zeilenvektor


y = d .^ d        % komponentenweise Potenzierung
y = d ^ 3         % Potenzierung der Matrix
```

In ähnlicher Weise lassen sich auch Matrizen untereinander und Matrizen mit Vektoren multiplizieren, addieren usw. Falls eine Operation komponentenweise durchgeführt werden soll, die auch als Matrixoperation möglich wäre, muß immer ein Punkt vor den Operator gesetzt werden. Bei allen Operationen nimmt es Matlab sehr genau mit den Dimensionen von Matrizen und Vektoren und gibt bei Unstimmigkeiten die Fehlermeldung "matrix dimensions must agree" aus.

2.5 Schleifen und Bedingungen

Wiederholte Operationen werden innerhalb von Schleifen durchgeführt. Die folgende Schleife berechnet einen Zeilen- und einen Spaltenvektor:

```
for i=1:5               % Beginn der Schleife
        p(i) = i^2;     % Einträge in einen Zeilenvektor
        q(i,1) = i^3;   % Einträge in einen Spaltenvektor
end                     % Ende der Schleife
```

Es können mehrere Schleifen ineinander geschachtelt werden. Das Index-Inkrement kann durch den Doppelpunkt-Operator eingestellt werden:

```
for i=1:2:5             % Beginn der Schleife über ungerade Indizees
        q(i) = i;       % Einträge für ungerade Inizees
        q(i+1) = -i;    % Einträge für gerade Inizees
end                     % Ende der Schleife
```

Wird der Befehl break innerhalb einer Schleife verwendet, so bricht die Ausführung der Schleife ab. Es gibt auch Schleifen, die ausgeführt werden, solange eine bestimmte Bedingung erfüllt ist:

```
x=100;
while( x > 1)           % Beginn der Schleife
        x = x/2;
end                     % Ende der Schleife
```

(**Hinweis**: Schleifen in Matlab sind relativ langsam. Wann immer möglich sollten sie durch Vektoroperationen ersetzt werden.).

Folgende Operatoren stehen zur Erstellung von Relationen zur Verfügung:

Operator	Funktion
==	gleich
~=	ungleich
>	größer
>=	größer gleich
<	kleiner
<=	kleiner gleich
&	logisches „und“
|	logisches „oder“
~	logisches „nicht“

Die bedingte Ausführung von Befehlen erfolgt über sog. If-Konstruktionen:

```
if( x == 0 | x == 1 )          % Beginn der If-Konstruktion
        status = 1;
elseif( x < 0 & x ~= -1 )      % Achtung: elseif muß in einem Wort geschrieben werden
        status = -1;
else
        status = 0;
end                            % Ende der If-Konstruktion
```

2.6 Generierung von Matrizen/Vektoren

Der Doppelpunkt-Operator (:) dient zur Adressierung von Teilen von Matrizen/Vektoren (siehe Abschnitt *Variablen*), aber auch zur Generierung von Matrizen/Vektoren. Beispiele:

```
x = [1:10];                % Erzeuge Zeilenvektor mit den ganzen Zahlen zwischen 1 und 10
x                          % Ergebnis zeigen
x = 1 2 3 4 5 6 7 8 9 10   % Antwort von Matlab

x = [1:3 ; 1 3 4];         % Erzeuge (2X3)-Matrix
x                          % Ergebnis zeigen
x = 1 2 3                  % Antwort von Matlab
    1 3 4
```

Es kann auch eine Schrittweite angegeben werden:

```
x = [0:pi/4:pi];                    % Erzeuge Zeilenvektor mit Zahlen von 0 bis Pi,
                                    % Schrittweite Pi/4.
x                                   % Ergebnis zeigen
x = 0 0.7854 1.5708 2.3562 3.1416   % Antwort von Matlab
```

Zur Generierung von Matrizen/Vektoren stehen weiterhin folgende Funktionen zur Verfügung:

zeros - generiert Matrizen/Vektoren und füllt sie mit Nullen
ones - generiert Matrizen/Vektoren und füllt sie mit Einsen
rand - generiert Matrizen/Vektoren und füllt sie mit gleichverteilten Zufallszahlen im Intervall (0,1)
randn - generiert Matrizen/Vektoren und füllt sie mit normalverteilten Zufallszahlen (Mittelwert 0, Varianz 1);

Als Argumente werden jeweils die gewünschte Anzahl der Zeilen und Spalten übergeben. Beispiel:

```
x = randn(2,3);  % erzeuge (2X3)-Matrix mit Zufallszahlen (normalverteilt)
```

2.7 Funktionen

Matlab enthält alle möglichen mathematischen und numerischen Funktionen. Alle Funktionen werden auf den gesamten Vektor/Matrix angewandt. Beispielsweise erzeugt folgender Befehl einen Vektor, der die Werte der Sinusfunktion zwischen 0 und 2 Pi mit einer Auflösung von Pi/10 enthält:

```
x = sin([0:pi/10:2*pi]);
```

Die folgende Tabelle enthält grundlegende und spezielle mathematische Funktionen:

Funktion	Beschreibung
sqrt(x)	Quadratwurzel
sin(x)	Sinus
asin(x)	Arcussinus
sinh(x)	Sinus Hyperbolicus
asinh(x)	Arcussinus Hyperbolicus
cos(x)	Cosinus

tan(x)	Tangens
exp(x)	Exponentialfunktion
expm(X)	Matrix-Exponentialfuktion
log(x)	natürlicher Logarithmus
logm(x)	Matrix-Logarithmus
log10(x)	Zehnerlogarithmus
rem(i,n)	Modulofunktion
round(x)	Runden
floor(x)	Abrunden
ceil(x)	Aufrunden
inv(X)	Inverse einer Matrix berechnen
fft(x)	schnelle Fouriertransformation
abs(x)	Absolutbetrag
real(x)	Realteil
imag(x)	Imaginärteil
sum	Summe über Vektorelemente
prod	Produkt der Vektorelemente
max	Maximumssuche
min	Minimumssuche
mean	Mittelwert
std	Standardabweichung
cov	(Ko-)Varianz
median	Medianwert
bessel	Besselfunktionen
erf	Gaussche Fehlerfunktion
gamma	Gammafunktion
size	berechne Größe einer Matrix/Vektor
hist	Histogramm
sort	Sortieralgorithmus

2.8 Visualisierung

Zur grafischen Darstellung von Daten stehen die verschiedensten Visualisierungsmöglichkeiten zur Verfügung. Die einfachste Möglichkeit sind xy-Plots, bei denen Datenpunkte (x,y) in einem kartesischen Koordinatensystem dargestellt werden:

```
x = [0:pi/10:2*pi];
y = sin(x);
plot(x,y);
```

Diese Befehle erzeugen eine Peride der sinusfunktion als Grafik. Die folgende Tabelle enthält weitere für die Visualisierung wesentliche Befehle:

Funktion	Beschreibung
plot(x,y)	xy-Plot, lineare Achsen
loglog(x,y)	xy-Plot, doppelt logarithmische Darstellung
semilogx(x,y)	xy-Plot, logarithmische x-Achse
semilogy(x,y)	xy-Plot, logarithmische y-Achse
title('text')	Setzen des Bildtitels
xlabel('text')	Setzen der x-Achsenbeschriftung
ylabel('text')	Setzen der y-Achsenbeschriftung
axis	Setzen der Achsen-Wertebereichs
polar(theta,rho)	Polarplot
contour(z)	2D-Höhenliniendarstellung der Matrix z
image(z)	2D-Farbdarstellung der Matrix z
imagesc(z)	wie image, jedoch mit Skalierung der Daten
pcolor(z)	ähnlich wie image(z)
colormap	Setzen der Farbskala bei Farbdarstellung

plot3(x,y,z)	3D-Darstellung von Vektoren
contour3(z)	3D-Höhenliniendarstellung der Matrix z
mesh(z)	3D-Gitterplot der Matrix z
surf(z)	3D-Gitterplot der Matrix z mit Farbdarstellung

Zur Erprobung der 2D- und 3D-Darstellungen kann mit dem Befehl peaks eine Testmatrix erzeugt werden. Beispielsweise plottet der Befehl imagesc(peaks) eine 2-D-Farbdarstellung der von peaks erzeugten Matrix.

2.9 Eingabe und Ausgabe

Der Befehl input liest eine Eingabe von der Tastatur:

```
x = input('Wert von x eingeben: ')
```

Dieser Befehl druckt den angegebenen Text auf den Bildschirm und wartet auf eine Eingabe von der Tastatur. Die Eingabe wird der Variablen x zugewiesen.

Der Befehl disp druckt Texte und Werte von Variablen auf dem Bildschirm:

```
disp('Der Wert von x ist: ')
disp(x)
```

Diese Befehle drucken den Text sowie den Wert der Variablen x.

Für die formatierte Ausgabe steht auch der Befehl fprintf zur Verfügung wie er aus der Programmiersprache C bekannt ist. Ebenso stehen die Befehle fopen, fread und fwrite und fclose zum Lesen von bzw. Schreiben in Dateien zur Verfügung (Hinweis: nicht mit den Befehlen load und save zu verwechseln, diese speichern den Workspace in einem speziellen Matlab-eigenen Datenformat ab).

2.10 Erstellung eigener Befehle („M-Files“)

In Matlab lassen sich auf einfache Weise eigene Programme und Funktionen selbst schreiben. Hierfür schreibt man ein sog. **Script**. Das ist eine Textdatei, in der Matlab-Befehle zusammengefaßt werden. Der Name der Datei wird von Matlab als Name des Befehls interpretiert, die Extension '.m' der Datei ist festgelegt. Daher heißen diese Dateien auch M-Files. Wie solche M-Files aufgebaut sein müssen, soll im folgenden anhand eines Beispiels erläutert werden. Für weitere Informationen hierzu kann auch die Matlab Hilfe-Funktion verwendet werden. Dazu muß help script oder help function eingeben werden.

(**Hinweis**: Damit Matlab die selbstgeschriebenen M-Files findet, muß das Verzeichnis, in dem sich die Dateien befinden, im Suchpfad sein (Kommando path) oder die Dateien müssen im aktuellen Verzeichnis sein (Kommando cd).)

Ein Script wird dadurch erstellt, daß die gewünschten Befehle zeilenweise in die Textdatei geschrieben werden. Dazu kann ein normaler Texteditor (z.B. notepad) verwendet werden, aber auch der Matlab-eigene Texteditor (Aufruf des Editors durch Auswählen von Open oder New im Menüpunkt File). Das Skript wird durch Eingabe des Dateinamens auf der Kommandozeile gestartet. Matlab arbeitet dann alle Befehle ab und behält die im Skript erzeugten Variablen im Workspace.

(**Hinweis**: Wenn ein M-File geändert wird, liest Matlab es u.U. nicht neu ein, es wird dann die alte Version ausgeführt (Fehler bei Netzwerk-Installation von Matlab, z.B. im CIP-Raum). Das Neu-Einlesen wird erzwungen, wenn man den Befehl clear all eingibt, was aber den Nebeneffekt hat, daß alle Variablen im Workspace gelöscht werden.

Man kann aus einem Skript eine Funktion machen, die sich wie eine Matlab-eigene Funktion verhält. Dazu sind bestimmte Formalia einzuhalten, die im folgenden anhand eines Beispiels erläutert werden sollen:

```
function [mean,stdev] = stat(x)
        %
        % [mean,stdev] = stat(x)
        %
        % Diese Funktion berechnet Mittelwert und Standardabweichung eines Datensatzes
        %
        % Übergabeparameter:
        %    x - Datensatz als Vektor
        %
        % Rückgabewerte:
        %         mean    - Mittelwert
        %         stdev    - Standardabweichung
        %

        n = length(x);
        mean = sum(x) / n;
        stdev = sqrt(sum((x - mean).^2)/n);
```

Die erste Zeile beginnt mit dem Schlüsselwort function und deklariert den Namen der Funktion (hier: stat), die Übergabeparameter in runden Klammern (hier: x) sowie die Rückgabewerte (hier: mean und stdev). Der nachfolgende Kommentarblock wird von Matlab ausgegeben, wenn die Hilfefunktion für dieses Kommando aufgerufen wird (help stat). Danach erfolgt die eigentliche Funktion, die aus beliebigen Matlab-Kommandos aufgebaut sein kann. Wichtig ist, daß dabei die Rückgabewerte aus den Übergabeparametern berechnet werden. Dabei können auch Hilfsvariablen generiert werden (hier: n). Die Funktion verwaltet ihren eigenen Workspace, d.h. der Funktion sind nur die Übergabewerte bekannt und es werden nur die Rückgabewerte an den übergeordneten Workspace zurückgegeben. Alle Hilfsvariablen sind nach Beendigung der Funktion gelöscht. Die Funktion kann nun folgendermaßen von der Kommandozeile aus aufgerufen werden:

```
noise = randn(1,100);              % Zufallszahlen generieren
[mnoise,snoise] = stat(noise);     % Statistik berechnen
disp('Mittelwert und Standardabweichung sind:');
mnoise
snoise
```

2.11 Aufgaben

Anmerkung: Speichere die Lösungen aller Aufgabenstellungen in Skript-Dateien ("M-Files").

I. Datengenerierung

a. generiere einen Zeitvektor von 0ms bis 10 ms bei einer Abtastfrequenz von 10 kHz.
b. generiere über der Zeitbasis einen Sinus mit einer Frequenz von 1 kHz,.
c. plotte Sinus aus b. über der Zeitbasis aus a.
d. wie c., jedoch nur jeden 2. Abtastwert und jeden 4. Abtastwert.
e. speichere die generierten Daten in einem Datenfile (Hinweis: benutze dazu den Befehl "save")
f. ersetze die erste Beispiel-Schleife im Abschnitt *Schleifen und Bedingungen* durch Vektoroperationen.

II. Umgang mit Matrizen

a. definiere die Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

und die Vektoren $b = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$ und $d = \begin{pmatrix} 1 & 2 \end{pmatrix}$ als Zeilenvektoren.

b. berechne Ab^T ; bA ; b * c als Skalarprodukt und (!) komponentenweise.

c. löse das lineare Gleichungssystem $Ax=b^T$ (Hinweis: "help slash") und überprüfe die Lösung durch Einsetzen.

d. berechne $A_2 d^T$, wobei $A_2 = \begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix}$ (A_2 als Untermatrix von A schreiben!).

III. 2D- und 3D-Darstellung von Matrizen

a. generiere eine 2-dimensionale Gauss-Glocke mit Varianz von 8 auf einer (25x25)-Matrix.
b. plotte die Matrix in 2-D-Farbdarstellung mit Höhenlinien.
c. wie b, jedoch 3D-Farbdarstellung.
d. **Optionale Aufgabe**: Wie läßt sich a. ohne FOR-Schleife lösen?

Die Beispiele im Menu ‚Help->Examples and Demos' sind sehr instruktiv und man kann dort viele Tricks und Kniffe der Matlab-Programmierung lernen (die Quelltexte für alle Beispiele sind enthalten!). Die Durchsicht dieser Beispiele wird empfohlen.

3 Zahlendarstellung und numerische Fehler

Fehlerarten:

1. Fehler in den Eingabedaten
2. Rundungsfehler
   - durch endliche Genauigkeit der Zahlendarstellung
   - durch arithmetische/mathematische Operationen
3. Bereichsfehler
4. Abschneidefehler
5. Fehlerfortpflanzung (instabile Lösungen bei iterierten Algorithmen)

3.1 Fehlermaße

Definition: Sei x^* eine Approximation an x ($x,x^* \in$ A, A Zahlenmenge, z.B. reelle Zahlen). Dann sind

$$\Delta x = |x - x^*|$$
$$r(x) = \frac{|x - x^*|}{|x|} = \frac{\Delta x}{|x|}$$

der **absolute Fehler** und der **absolute relative Fehler** von x^*.

Meist ist Δx nicht bekannt, sondern nur eine **Fehlergrenze** ε, die den maximalen absoluten Fehler angibt:

$$x^* - \varepsilon \le x \le x^* + \varepsilon \Leftrightarrow x = x^* \pm \varepsilon$$

Meist ist auch nur die Aproximation x*, nicht aber die „wahre“ Zahl x bekannt. Dann wird auch der relative Fehler approximiert:

$$r(x) \approx \frac{\Delta x}{|x^*|} \le \frac{\varepsilon}{|x^*|}$$

Beispiel: Wird als ein Meßwert 0.0123 ± 0.0005 angegeben, so ist $\Delta x \le \varepsilon = 0.0005$. Der Meßwert hat 2 signifikante Ziffern. Die relative Genauigkeit ist $r(g) \le 0.0005/0.0123 \approx 0.04$.

3.2 Zahlendarstellungen und Rundungsfehler

Digitalrechner (Computer) können nicht unendlich viele verschiedene Zahlen darstellen, da für die Darstellung einer Zahl nur eine endliche Zahl von Informationseinheiten (meist in Bits[2] angegeben) zur Verfügung steht. Die Menge A der darstellbaren Zahlen wird als Menge der **Maschinenzahlen** bezeichnet und ist im allgemeinen eine Teilmenge der reellen Zahlen. A hängt von der genauen Art der Zahlendarstellung im Computer ab und ist implizit mit einer **Rundungoperation** *rd(x)* verknüpft, die angibt, wie Elemente der reellen Zahlen durch Maschinenzahlen approximiert werden. Diese kann allgemein allgemein definiert werden:

$$rd: \Re \to A,$$
$$rd(x) \in A: \|x - rd(x)\| \le \|x - g\| \forall g \in A$$

[2] 1 Bit entspricht einer „Entweder-oder“-Entscheidung (0 oder 1).

Der absolute und relative Approximationsfehler durch die endliche Zahlendarstellung ergeben sich, indem in den o.a. Formeln $x^* = \mathrm{rd}(x)$ gesetzt wird. Das Ergebnis einer arithmetischen Operation ist i.A. nicht Element von A, selbst wenn alle Operanden Element von A sind. Das Ergebnis muß wieder der Rundungsoperation *rd(x)* unterworfen werden, damit es dargestellt werden kann. Dadurch können arithmetische Operationen zusätzliche Rundungsfehler einführen, auch wenn das Ergebnis der Operation selbst beliebig genau berechnet wird:

$$x, y \in A \Rightarrow rd(x+y) \in A \text{ , aber } (x+y) \text{ nicht notwendig } \in A$$

Digitalrechner arbeiten mit verschiedenen Zahlendarstellungen und damit mit unterschiedlichen Maschinenzahlen. Im folgenden werden spezielle gebräuchliche Zahlendarstellungen angegeben, wie sie in Digitalrechnern meist benutzt werden.

3.2.1 Ganze Zahlen (Integer)

Ganze Zahlen (Integer-Werte) werden meist durch das **Zweierkomplement** repräsentiert. Dabei wird jede Maschinenzahl *z* folgendermaßen dargestellt:

$$z = \{\alpha_0, \alpha_1, ..., \alpha_{N-1}\} \ , \ \alpha_n \in \{0,1\}$$

mit

$$z = -\alpha_{N-1} * 2^{N-1} + \sum_{n=0}^{N-2} \alpha_n 2^n \text{ mit } \alpha_n \in \{0,1\}$$

Für jeden Koeffizienten α_n wird eine Informationseinheit von 1 Bit benötigt („0“ oder „1“), so daß zur Darstellung der Zahl N Bits benötigt werden (gängige Größen für N sind 8,16,32 und 64). Es ergibt sich dadurch die Bit-Darstellung von *z*. Beispiel für N=8:

Zahl	**Bit-Darstellung**							
+1	0	0	0	0	0	0	0	1
+2	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0
-1	1	1	1	1	1	1	1	1
+117	0	1	1	1	0	1	0	1
-118	1	0	0	0	1	0	1	0
Wertigkeit	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Zahlenbereich: $-2^{N-1} \le z \le 2^{N-1} - 1$

Fehlergrenze: $\Delta z \le \varepsilon = 0.5$

relativer Fehler: $r(z) \approx \frac{\Delta x}{|z^*|} \le \frac{0.5}{|z^*|}$

Die Integer-Darstellung hat einen konstanten absoluten Fehler und damit einen relativen Fehler, der vom Betrag der darzustellenden Zahl abhängt.

Bemerkung: Der Name Zweierkomplement kommt daher, daß sich der Übergang von einer positiven zu der negativen Zahl gleichen Betrags dadurch ergibt, daß ein Bit-weises Komplement in der Bit-Darstellung durchgeführt wird (jede „1“ wird zu einer „0“ und umgekehrt) und dann eine 1 addiert wird.

Bemerkung: Der Vorteil des Zweierkomplements gegenüber anderen denkbaren Integer-Darstellungen liegt darin, daß die Addition zweier Zahlen einfach durch Bit-weises addieren in der Bit-Darstellung erreicht wird. Eine Fallunterscheidung für positive und negative Zahlen ist nicht nötig. Beispiel (beachte den Übertrag):

+2	0	0	0	0	0	0	1	0
-1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	1

Bemerkung: Aufgrund der Wertigkeit der einzelnen Bits in der Bit-Darstellung entspricht eine Multiplikation mit 2 gerade einem Verschieben aller Bits um eine Position nach links („left-shift“) und entsprechend eine Division durch 2 einem „right-shift“.

3.2.2 Gleitkommazahlen (Floating Point)

Eine Maschinenzahl wird im **Gleitkomma-Format** folgendermaßen dargestellt:

$$g = (-1)^s * m * B^e \ , \ 1/B \le m < 1$$

Dabei ist:

s	Vorzeichenbit („0“ oder „1)
m	Mantisse
e	Exponent
B	Basis (fester Wert)

Bemerkung: Die Nebenbedingung 1/B≤ m< 1 macht die Darstellung eindeutig. Man nennt dies die normierte Darstellung. z.B. kann die Zahl 0.5 mit der Basis B=10 durch $0.5*10^0$ oder $0.05*10^1$ dargestellt werden. Durch die Nebenbedingung wird die erste Darstellung festgelegt.

Auf Digitalrechnern wird meist das 32Bit **IEEE Floating-Point Format** verwendet. Als Basis wird B=2 verwendet und der Exponent *e* wird in 8Bit-Zweierkomplementdarstellung dargestellt. Für die Mantisse m wird eine 23Bit-Darstellung mit folgenden Bit-Wertigkeiten verwendet:

Bit	m_0	m_1	m_2	...	m_{21}	m_{22}
Wertigkeit	2^{-1}	2^{-2}	2^{-3}	...	2^{-22}	2^{-23}

Hinweis: Die Normierungsbedingung bewirkt, daß das Bit m_0 der Mantisse m immer 1 sein muß. Es wird oft daher gar nicht mit abgespeichert.

Beispiel: Die Zahl 3 läßt sich folgendermaßen darstellen:

$$3 = \left(2^{-1} + 2^{-2}\right) * 2^2$$

Die Bit-Darstellung der Zahl 3 im IEEE Floating-Point Format ist demnach:

s	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	m_0	m_1	m_2	...	m_{21}	m_{22}
0	0	0	0	0	0	0	1	0	1	1	0	...	0	0

Zahlenbereich (ca.): $-10^{38} \le g \le 10^{38}$

Sei g=$m*B^e \in A$ Maschinenzahl:

Fehlergrenze: $\Delta g \le \varepsilon = \frac{1}{2} * 2^{-M} * B^e$, M: Anzahl der Bits der Mantisse

relativer Fehler: $r(g) \approx \frac{\Delta g}{|g*|} \le \frac{\varepsilon}{m* B^e} \le \frac{\varepsilon}{1/2* B^e} = 2^{-M}$

Die Gleitkomma-Darstellung hat einen Konstanten relativen Fehler. Die Fehlergrenze wird als **Maschinengenauigkeit** bezeichnet.

Hinweis: Oft wird die Maschinengenaugkeit als die kleinste Zahl $\varepsilon > 0$ definiert, die bei der Addition $(1+\varepsilon)$ noch ein Ergebnis gößer als 1.0 liefert.

Hinweis: Neben dem 32Bit IEEE Floating-Point Format („float“) steht bei den meisten Programmiersprachen noch ein entsprechendes 64Bit IEEE Floating-Point Format zur Verfügung („double“), das entsprechend mehr Bits für Exponent und Mantisse aufweist und damit eine höhere Genauigkeit und einen größeren Zahlenbereich aufweist. Matlab rechnet intern mit dem 64Bit-Format.

3.3 Bereichsfehler

Die verschiedenen Zahlendarstellungen decken einen beschränkten Zahlenbereich ab (s.o.). Durch arithmetische Operationen kann dieser leicht überschritten werden. Wichiges Beispiel ist die Berechnung der Fakultät *n!*, die den Zahlenbereich der 64Bit Gleitkommadarstellung bereits für $n>170$ überschreitet. Der bei Überschreitung des Zahlenbereichs auftretende Bereichsfehler ist um Größenordnungen größer als der Rundungsfehler und muß durch geeignete Wahl der Zahlendarstellung und Überwachung der Größenordnungen der untersuchten Zahlen vermieden werden (**Beispiel:** Berechnung der Fakultät durch Anwendung der Stirlingschen Formel und logarithmischer Darstellung).

Weiterin ist es möglich, daß bestimmte physikalische Konstanten den Bereich über- oder unterschreiten (siehe Übungen).

3.4 Abschneidefehler

Fehler, die dem Algorithmus zur Berechnung des Algorithmus inhärent sind, selbst wenn keine Rundungsfehler auftreten (**Beispiele:** Berechnung der Exponentialfunktion und Approximation des Integrals durch Rechteck-Flächen).

3.5 Fehlerfortpflanzung bei arithmetischen Operationen

Addition:

$$
\begin{aligned}
x &= a+b\ ;\ a,b \geq 0\ ,\ \text{o.B.d.A.} \\
x_{\max} &= a+\Delta a+b+\Delta b \\
x_{\min} &= a-\Delta a+b-\Delta b \\
\Delta x &= \Delta a+\Delta b \\
r(x) &= \frac{\Delta x}{x} = \frac{\Delta a+\Delta b}{|a+b|}
\end{aligned}
$$

Subtraktion:

$$
\begin{aligned}
x &= a-b\ ;\ a,b \geq 0\ ,\ \text{o.B.d.A.} \\
x_{\max} &= a+\Delta a-b+\Delta b \\
x_{\min} &= a-\Delta a-b-\Delta b \\
\Delta x &= \Delta a+\Delta b \\
r(x) &= \frac{\Delta x}{x} = \frac{\Delta a+\Delta b}{|a-b|}
\end{aligned}
$$

Falls 2 in etwa gleich große Zahlen voneinander abegzogen werden, wird der relative Fehler sehr groß! Dies wird als **Ziffernverlust** bezeichnet.

Beispiel: Ziffernverlust bei der Berechnung der quadratischen Formel (→ Übungen).

Beispiel: Betrachte die Differenz der Zahlen p=3.1415926536 und q=3.1415957341, die beide ähnlich groß sind und jeweils auf 11 Stellen signifikant sind. Die Differenz p-q=-0.0000030805 hat nur noch 5 signifikante Stellen, obwohl durch die Operation selbst keine neuen Fehler entstanden sind.

Multiplikation:

$$
\begin{aligned}
x &= a*b \;;\; a,b \geq 0 \;,\; \text{o.B.d.A.} \\
x_{\max} &= (a+\Delta a)*(b+\Delta b) \approx a*b + \Delta a*b + a*\Delta b \;,\; \text{da } \Delta a*\Delta b \text{ viel kleiner als } \Delta a*b \\
x_{\min} &= (a-\Delta a)*(b-\Delta b) \approx a*b - \Delta a*b - a*\Delta b \\
\Delta x &\approx \Delta a*b + a*\Delta b \\
r(x) &= \frac{\Delta x}{x} \approx \frac{\Delta a*b + a*\Delta b}{|a*b|} = \frac{\Delta a}{|a|} + \frac{\Delta b}{|b|}
\end{aligned}
$$

Division:

$$
\begin{aligned}
x &= a/b \;;\; a,b \geq 0 \;,\; \text{o.B.d.A.} \\
x_{\max} &= (a+\Delta a)/(b-\Delta b) = \frac{(a+\Delta a)*(b+\Delta b)}{(b-\Delta b)*(b+\Delta b)} \approx a/b + \frac{\Delta a*b + a*\Delta b}{b^2} \\
x_{\min} &= (a-\Delta a)/(b+\Delta b) = \frac{(a-\Delta a)*(b-\Delta b)}{(b+\Delta b)*(b-\Delta b)} \approx a/b - \frac{\Delta a*b + a*\Delta b}{b^2} \\
\Delta x &\approx \frac{\Delta a*b + a*\Delta b}{b^2} \\
r(x) &= \frac{\Delta x}{x} = \frac{\Delta x}{|a/b|} \approx \frac{\Delta a*b + a*\Delta b}{b^2*|a/b|} = \frac{\Delta a}{|a|} + \frac{\Delta b}{|b|}
\end{aligned}
$$

Bei Addition und Subtraktion addieren sich die absoluten Fehler der Operanden, während sich bei Multiplikation und Division die relativen Fehler addieren.

3.6 Fehlerfortpflanzung bei iterierten Algorithmen

Viele Algorithmen enthalten iterierte Abbildungen, bei denen eine bestimmte Rechenvorschrift solange mit den im vorigen Schritt berechneten Daten wiederholt wird, bis eine bestimmte Abbruchbedingung erfüllt ist. Dabei pflanzen sich die in einem Schritt erzeugten Fehler fort. Je nach Problem kann der Gesamtfehler linear ansteigen, exponentiell ansteigen oder auch exponentiell abfallen (**Fehlerdämpfung**). Das Verhalten eines Algorithmus läßt sich aus den in jeder Iteration verwendeten Operationen und den im letzten Abschnitt abgeleiteten Regeln für die Fehlerfortpflanzung bei den grundlegenden arithmetischen Operationen ableiten.

Definition: Sei *E(n)* der Gesamtfehler nach *n* Iterationsschritten und ε die Maschinengenauigkeit. Falls $|E(n)| \approx n\varepsilon$ ist, spricht man von **linearer Fehlerfortpflanzung**. Falls $|E(n)| \approx K^n\varepsilon$ ist, spricht man von **exponentieller Fehlerfortpflanzung**, wobei für K<1 **Fehlerdämpfung** auftritt.

Beispiel: Iterierte Abbildungen führen bei exponentieller Fehlerfortpflanzung zu einer starken Abhängigkeit von den Anfangswerten und stellen somit einen Weg in das deterministische Chaos dar. Ein Beispiel dafür ist die **logistische Parabel**.

3.7 Aufgaben

Anmerkung: Speichere die Lösungen aller Aufgabenstellungen in Skript-Dateien ("M-Files").

I Rundungsfehler

1. Ermittele die Maschinengenauigkeit von Matlab und schätze daraus ab, mit wieviel Bit die Mantisse dargestellt ist. (**Hinweis**: Beginne bei eps=1 und teile eps solange durch 2, bis (1+eps) nicht mehr größer 1 ist).
2. Berechne die Formel x=(10+h)-10 für $h=B^{-n}$ für n=0...21 und B=10. Berechne den absoluten relativen Fehler zwischen x und h und plotte ihn als Funktion von n. Wiederhole dasselbe für B = 2. Warum ergeben sich hier deutliche Unterschiede?

II Bereichsfehler

1. Wie läßt sicher der Bereichsfehler bei Berechnungen mit physikalischen Konstanten (Atomphysik, Quantenmechanik) vermeiden?
2. Nehme eV (Elektronenvolt) als Einheit für die Energie und die Masse des Elektrons als Einheit der Masse. Konvertiere 1kg, 1m und 1s in diese Einheiten.

III Abschneidefehler

1. Die Taylorreihe der Exponentialfunktion lautet:

$$e^x = \lim_{N\to\infty} S(x,N) \text{ mit } S(x,N) = \sum_{i=0}^{N} \frac{x^i}{i!}$$

Berechne die Teilsummen S(x,N) bis N=60 und plotte den absoluten relativen Fehler bezüglich des wahren Werts (Matlab-Funktion exp())als Funktion von N. Teste das Programm für x=10,2,-2 und -10. Warum steigt der Fehler für negative Zahlen?

IV Fehlerfortpflanzung

1. Die quadratische Gleichung $ax^2 + bx + c = 0$ hat die Lösungen:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ und } x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

a. Unter welcher Bedingung kommt es zu einem Stellenverlust im Zähler?
b. beweise, daß sich die Nullstellen auch nach folgender äquivalenter Formel berechnen lassen:

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}} \text{ und } x_2 = \frac{-2c}{b - \sqrt{b^2 - 4ac}}$$

c. Wie kann mit b. der Stellenverlust vermieden werden?
d. berechne die Nullstellen für *a*=1, *b*=-10000.0001 und *c*=1 sowie *a*=1, *b*=-100000.00001 und *c*=1 nach beiden Formeln. Vergleiche mit der korrekten Lösung und überprüfe die gefundene Lösung durch Einsetzen in die Gleichung (Hinweis: Benutze zum Anzeigen den Befehl sprintf, sonst zeigt Matlab nicht ausreichend Stellen an (z.B. sprintf('%5.10g',x1))).

4 Numerische Differentiation und Integration

Numerische Verfahren zur Differentiation und Integration kommen dann zum Einsatz, wenn keine analytischen Lösungen bekannt sind. Es gibt viele Beispiele für analytisch nicht integrierbare Funktionen, z.B. die Gaußsche Fehlerfunktion

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2}\, dy\,,$$

also das Integral über die Gaußverteilung. Im Gegensatz zur Integration ist die Differentiation auf analytische Weise praktisch immer möglich, so daß die numerische Differentiation selbst meist nicht direkt angewandt wird. Sie bildet aber die Grundlage für die Entwicklung von Algorithmen zur Lösung von Randwertproblemen bei gewöhnlichen und partiellen Differentialgleichungen und soll daher detailliert behandelt werden.

Im folgenden werden die wichtigsten numerischen Verfahren zur Differentiation und Integration erläutert. Die Vorgehensweisen zur Entwicklung der Algorithmen und zur Fehlerabschätzung finden sich in späteren Kapiteln in komplexerer Form wieder und können somit als Grundlage für das Folgende verwendet werden.

4.1 Differentiation

Das Prinzip der numerischen Differentiation besteht darin, die zu differenzierende Funktion an dem gesuchten Punkt durch ein Polynom anzunähern, das sich dann gemäß der bekannten Regeln für Polynome ableiten läßt. Der Wert der Ableitung des Polynoms wird als Schätzwert für die Ableitung der Funktion verwendet. Im allgemeinen ist die Approximation und der Schätzwert für die Ableitung um so genauer, je höherer Ordnung das Polynom ist. Dies ist aber keine Gesetzmäßigkeit und es gibt Ausnahmen, insbesondere, wenn sich die abzuleitende Funktion nicht gut durch Polynome hoher Ordnung annähern läßt.

4.1.1 rechtseitige Formel („naiver“ Ansatz)

Aus der Analysis ist bekannt, das sich die Ableitung nach folgender Formel berechnet:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Der Grenzübergang kann numerisch nicht vollzogen werden, aber durch Wahl eines geeigneten, kleinen *h* ergibt sich eine Näherung der Ableitung nach folgender Formel:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \varepsilon_a \quad (h > 0),$$

wobei ε_a der (unbekannte) Abschneidefehler ist, der sich durch den nicht durchgeführten Grenzübergang (endliches *h*) ergibt. Der Abschneidefehler tritt auch auf, wenn der Quotient beliebig genau berechnet würde. Bei numerischer Berechnung des Quotienten ergeben sich zusätzlich folgende Fehlerquellen:

1. Rundungsfehler bei der Berechnung von x+h.
2. Rundungsfehler bei der Berechnung des Zählers (Achtung: Stellenverlust droht).

Zur Vermeidung des 1. Fehleranteils wähle h immer so, daß *x+h* eine exakt darstellbare Zahl ist[3]. Der 2. Rundungsfehler kann nicht vermieden werden. Wenn die Funktion *f* bis auf die Maschinengenauigkeit ε_m genau berechnet wird, ist der absolute Fehler in der Berechnung der Ableitung:

$$\Delta(f'(x)) = \frac{2\varepsilon_m}{h} + \varepsilon_a \quad (h > 0)$$

Zur Berechnung des Abschneidefehlers entwickeln wird die Funktion in eine **Taylorreihe** 1. Ordnung:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\zeta) \quad , x \le z \le x+h$$

ζ ist nach dem Taylorschen Theorem ein Wert innerhalb des betrachteten Intervalls. Die obige Formel läßt sich daraus sofort durch Auflösung nach $f'(x)$ erhalten:

$$\begin{aligned} f'(x) &= \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(\zeta) \\ &= \frac{f(x+h) - f(x)}{h} + O(h) \end{aligned}$$

Der Abschneidefehler ist somit linear in *h*, man spricht von der **Ordnung** *O*(*h*). Der absolute Fehler kann dann folgendermaßen angegeben werden:

$$\Delta(f'(x)) = \frac{2\varepsilon_m}{h} + \frac{1}{2}Mh \quad , \quad M = \max_{x \le \zeta \le x+h}\left(\left|f''(\zeta)\right|\right)$$

Der erste Summand steigt mit fallendem *h*, während der zweite Summand mit fallendem h sinkt. Somit gibt es ein optimales *h* mit minimalem Fehler, was aber von dem Maximum *M* der zweiten Ableitung im betrachteten Intervall abhängt. Da *M* i.a. unbekannt ist, kann man nur sagen, daß es aufgrund der numerischen Fehler nicht sinnvoll ist, *h* beliebig klein zu wählen. Man sollte von einem relativ großen *h* ausgehen und es solange verkleinern, bis sich die Schätzung der Ableitung bezüglich einer vorgegebenen Genauigkeit nicht mehr ändert.

4.1.2 zentrierte Formel

Die Genauigkeit ließe sich steigern, wenn der Abschneidefehler schneller als linear mit *h* fallen würde, z.B. Ordnung $O(h^2)$ oder höher. Die Idee zur Entwicklung solcher Formeln liegt darin, die Taylorentwicklung zu höheren Ordnungen durchzuführen und auch das Intervall *x-h* einzubeziehen:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\zeta_1) \quad , \quad x \le \zeta_1 \le x+h$$

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\zeta_2) \quad , \; x\text{-}h \le \zeta_2 \le x+h$$

Die beiden Formeln werden voneinander abgezogen und nach $f'(x)$ aufgelöst. Dabei fällt die zweite Ableitung weg (wie auch alle weiteren geraden Ordnungen wegfallen würden):

---

[3] Berechne zunächst folgende temporäre Größen: $\begin{aligned} \tilde{x} &= x + h \\ \tilde{h} &= \tilde{x} - x \end{aligned}$.
$x, \tilde{x}$ und $\tilde{h}$ sind exakt darstellbare Zahlen, da sie jeweils einer Variablen im Workspace explizit zugewiesen wurden. Berechne die Formel dann mit $x, \tilde{x}$ und $\tilde{h}$.

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \frac{1}{6}h^2 \frac{f'''(\zeta_1) + f'''(\zeta_2)}{2}$$
$$= \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

Bei dieser sog. **zentrierten Formel** fällt der Abschneidefehler mit h^2, während der Rundungsfehler bei der Berechnung des Quotienten, wie bereits bei der rechtseitigen Formel gezeigt, proportional $1/h$ ist. Im allgemeinen kommt man dadurch auf einen kleineren Gesamtfehler.
Auf ähnliche Weise läßt sich eine Schätzung für die zweite Ableitung finden. Dazu wird wieder die Taylorentwicklung in den Intervallen $x+h$ und $x-h$ durchgeführt und diesmal durch Addition der beiden Entwicklungen nach $f''(x)$ aufgelöst:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + O(h^4)$$
$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + O(h^4)$$
$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4)$$

$$\Rightarrow f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} + O(h^2)$$

Die zweite Ableitung läßt sich also mit Hilfe der zentrierten Formel ebenso wie die erste Ableitung bis zur Ordnung $O(h^2)$ berechnen, jedoch muß die Funktion f an 3 statt nur an 2 Stellen ausgewertet werden.

4.1.3 Richardson-Extrapolation

Auf ähnliche Weise wie im letzten Abschnitt gezeigt lassen sich natürlich auch Formeln höherer Ordnung finden, jedoch können die Formeln höherer Ordnung auch aus den Formeln niedrigerer Ordnung berechnet werden. Dies als **Extrapolation** bezeichnete Vorgehen soll im folgenden eingeführt werden. Wir betrachten dazu die vollständige Taylorentwicklung der Funktion f um den Punkt x. Eine geeignete Umformung ergibt wieder die zentrierte Formel, jedoch wird der Abschneidefehler in allen Potenzen von h berechnet:

$$f(x+h) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x)}{k!} h^k$$
$$f(x-h) = \sum_{k=0}^{\infty} (-1)^k \frac{f^{(k)}(x)}{k!} h^k \Rightarrow$$
$$f(x+h) - f(x-h) = 2\sum_{k=0}^{\infty} \frac{f^{2(k+1)}(x)}{(2k+1)!} h^{(2k+1)} \Rightarrow$$
$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \sum_{k=1}^{\infty} \alpha_k h^{2k} \qquad , \qquad \alpha_k = \frac{f^{2(k+1)}(x)}{(2k+1)!}$$

Die Reihe stellt den (unbekannten) Abschneidefehler dar. Im folgenden wird der Quotient im linken Teil der letzten Gleichung als $D_0(h)$ definiert[4] und für die Schrittweiten h und $2h$ ausgewertet:

[4] $D_0(h)$ entspricht genau der zentrierten Formel.

$$D_0(h) = f'(x) + \sum_{k=1}^{\infty} \alpha_k h^{2k}$$

$$D_0(2h) = f'(x) + \sum_{k=1}^{\infty} 4^k \alpha_k h^{2k}$$

Die nächsthöhere Potenz von *h* im Abschneidefehler läßt sich nun durch geeignete Verrechnung von $D_0(h)$ und $D_0(2h)$ eliminieren:

$$\begin{aligned} 4D_0(h) - D_0(2h) &= (4-1)f'(x) + 4\sum_{k=2}^{\infty} a_k h^{2k} - \sum_{k=2}^{\infty} 4^k a_k h^{2k} \\ &= 3f'(x) + \sum_{k=2}^{\infty} \tilde{a}_k h^{2k} \end{aligned}$$

Die nächste Iteration zur Approximation von $f'(x)$ ergibt sich also folgendermaßen:

$$\begin{aligned} D_1(h) &\equiv \frac{4D_0(h) - D_0(2h)}{3} \\ &= D_0(h) + \frac{D_0(h) - D_0(2h)}{3} \\ &= f'(x) + \frac{1}{3}\sum_{k=2}^{\infty} \tilde{a}_k h^{2k} \end{aligned}$$

Da die Reihe für den Abschneidefehler erst bei *k*=2 beginnt, hat $D_1(h)$ also einen Abschneidefehler der Ordnung $O(h^4)$ und berechnet sich aus den beiden Größen $D_0(h)$ und $D_0(2h)$, die ihrerseits nur der Ordnung $O(h^2)$ sind. Dies läßt sich zur Elimination höherer Ordnungen allgemein fortsetzen:

$$\begin{aligned} D_k(h) &= \frac{4^k D_{k-1}(h) - D_{k-1}(2h)}{4^k - 1} \\ &= D_{k-1}(h) + \frac{D_{k-1}(h) - D_{k-1}(2h)}{4^k - 1} \end{aligned}$$

Diese Rekursionsformel wird als **Richardson-Extrapolation** bezeichnet und läßt sich immer dann anwenden, wenn der Abschneidefehler nur gerade Potenzen in *h* aufweist. Die numerische Differentiation ergibt sich dann durch Berechnung der folgenden Tabelle:

$$\begin{array}{cccc} D_0(h) & . & . & . \\ D_0(h/2) & D_1(h/2) & . & . \\ D_0(h/4) & D_1(h/4) & D_2(h/4) & . \\ D_0(h/8) & D_1(h/8) & D_2(h/8) & D_3(h/8) \end{array}$$

Die Tabelle läßt sich nur zeilenweise berechnen, wobei sich die erste Spalte direkt nach der zentrierten Formel ergibt, während sich die folgenden Spalten aus der Extrapolationsformel ergeben. Das Ergebnis findet sich dann als der letzte Wert in jeder Zeile. Die Berechnung der Zeilen wird abgebrochen, sobald sich das Zeilenergebnis nicht mehr ändert (bezüglich einer vorgegebenen Genauigkeit).

4.2 Integration

Betrachte das bestimmte Integral

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

Die wesentliche Technik zur numerischen Berechnung bestimmter Integrale besteht darin, die Funktion *f* im betrachteten Intervall [*a*,*b*] durch ein Polynom *P* anzunähern, das dann nach den einfachen Regeln der Polynomintegration integriert wird. Das Integral des Polynoms wird als Schätzwert für das Integral der Funktion verwendet:

$$f(x)\big|_{[a,b]} \approx P(x) \qquad , \qquad P(x) = \sum_{n=0}^{N} \alpha_n x^n$$

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx$$

Je höher die Ordnung *N* des Polynoms ist, desto besser ist i.a. auch die Schätzung des Integralwerts, jedoch gilt dies nur für Funktionen, die sich durch Polynome hoher Ordnung approximieren lassen (Dies ist nicht immer der Fall, z.B. bei Unstetigkeiten). Das Problem reduziert sich somit darauf, eine geeignete Polynomapproximation durchzuführen. Allgemein kann man sagen, daß sich ein Polynom *N*-ter Ordnung aus *N* Punkten eindeutig bestimmen läßt. Es reicht zur Approximation der Funktion *f* durch ein Polynom *N*-ter Ordnung also aus, *f* an *N* Punkten im betrachteten Intervall [*a*,*b*] auszuwerten

4.2.1 Trapezregel

Nehmen wir an, wir unterteilen das Interval [*a*,*b*] in *N* Intervalle. Dazu benötigen wir *N*+1 Punkte:

$$x_0 = a, \quad x_N = b, \quad x_0 < x_1 < \ldots < x_{N-1} < x_N$$

Die Funktion *f* wird an diesen Punkten ausgewertet: $f_i \equiv f(x_i), \quad 0 < i < N$

Die einfachste Methode zur Integration ist nun, diese Punkte stückweise linear zu verbinden. Dadurch ergibt sich in jedem Intervall ein Trapez mit folgender Fläche:

$$T_i = \frac{1}{2}(x_i - x_{i-1})(f_i + f_{i-1}), \quad 1 < i < N$$

Die Gesamtfläche ist dann: $T = \sum_{i=1}^{N} T_i$

Diese Formel wird vereinfacht, wenn die Punkte gleichmäßig im Intervall verteilt sind. Sei $h = (b-a)/N$ die Breite jedes Intervalls, also $x_i = a + ih$. Die Fläche jedes Intervalls ist dann $T_i = 1/2\, h(f_i + f_{i-1})$. In der Formel für die Gesamtfläche kommt dann jedes f_i bis auf die Randpunkte doppelt vor:

$$T(h) = 1/2\, h(f_0 + f_N) + h\sum_{i=1}^{N-1} f_i \qquad \left(\equiv \sum_{i=0}^{N} w_i f_i \right)$$

Das Integral läßt sich nach dieser **Trapezregel** um so besser approximieren, je kleiner *h* ist (der auftretende Abschneidefehler aufgrund des endlichen Intervalls wird später angegeben) (**Bemerkung**: Die Schätzung des Integrals ergibt sich als gewichtete Summe der Abtastpunkte der Funktion).

4.2.2 rekursive Trapezregel

Wir wollen nun die Approximation des Integrals nach der Trapezregel iterativ berechnen, indem das Intervall *h* systematisch verkleinert wird. Sinnvoll ist es, *h* ausgehend vom kompletten Intervall (*h*=(*b*-*a*)) jeweils zu halbieren:

$$h_n = \frac{1}{2^n}(b-a), \qquad n = 0,1,\ldots$$

Dann können die im vorigen Iterationsschritt berechneten f_i wieder verwendet werden. Durch einfache Überlegung ergibt sich:

$$T(h_0) = \frac{1}{2} h_0 \left(f(a) + f(b)\right)$$

$$T(h_{n+1}) = \frac{1}{2} T(h_n) + h_{n+1} \sum_{i=1}^{2^n} f\left(a + (2i-1)h_{n+1}\right)$$

Die Folge $T(h_n)$ konvergiert dann gegen den Wert des Integrals.

4.2.3 Fehleranalyse und Romberg-Integration

Durch ein ähnliches Vorgehen wie in Abschnitt 4.1.3 läßt sich zeigen, daß der Abschneidefehler bei der Anwendung der Trapezregel nur gerade Potenzen von *h* enthält:

$$T(h_n) = \int_a^b f(x) + \sum_{k=1}^{\infty} \alpha_k {h_n}^{2k}$$

Damit ist wiederum die Richardson-Extrapolation zur Erhöhung der Fehlerordnung anwendbar:

$$T_0(h_n) \equiv T(h_n) \qquad \text{(rekursive Trapezregel)}$$

$$T_k(h_n) = \frac{4^k T_{k-1}(h_n) - T_{k-1}(h_{n-1})}{4^k - 1}$$

$$= T_{k-1}(h_n) + \frac{T_{k-1}(h_n) - T_{k-1}(h_{n-1})}{4^k - 1}$$

Die numerische Integration ergibt sich dann durch Berechnung der folgenden Tabelle:

$$\begin{array}{cccc} T_0(h_0) & . & . & . \\ T_0(h_1) & T_1(h_1) & . & . \\ T_0(h_2) & T_1(h_2) & T_2(h_2) & . \\ T_0(h_3) & T_1(h_3) & T_2(h_3) & T_3(h_3) \end{array}$$

Die Tabelle läßt sich nur zeilenweise berechnen, wobei sich die erste Spalte direkt nach der rekursiven Trapezregel ergibt, während sich die folgenden Spalten aus der Extrapolationsformel ergeben. Das Ergebnis findet sich dann als der letzte Wert in jeder Zeile. Die Berechnung der Zeilen wird abgebrochen, sobald sich das Zeilenergebnis nicht mehr ändert (bezüglich einer vorgegebenen Genauigkeit).

Die Berechnung des Integrals nach der rekursiven Trapezregel mit zusätzlicher Richardson-Extrapolation heißt **Romberg-Integration**.

4.2.4 Gaußsche Integration

Die Trapezregel beinhaltet die Auswertung der Funktion *f* an Punkten x_i, die äquidistant im Intervall [*a*,*b*] verteilt sind. Der Schätzwert des Integrals lautet dann:

$$\int_a^b f(x)dx \approx T(h) = 1/2\, h\left(f_0 + f_N\right) + h\sum_{i=1}^{N-1} f_i, \qquad f_i = f\left(x_i\right)$$

Er stellt also eine gewichtete Summe der Funktionswerte f_i dar, wobei die Gewichte w_i in diesem Fall 0.5 für die Randpunkte und 1 sonst sind. Es stellt sich die Frage, ob eine nicht-äquidistante Verteilung der Werte x_i eine höhere Genauigkeit liefert, und wenn ja, welche Gewichte dann bei der Aufsummation zu verwenden sind. Auch hier liefert, wie bei vielen mathematischen Problemen, Gauß die Antwort. Er konnte zeigen, daß eine bestimmte Wahl der Gewichte und der x_i eine Approximation von *f* durch ein Polynom der Ordnung 2*N* liefert, wenn die Funktion nur an *N* Stellen ausgewertet wurde. Dadurch erhöht sich die Ordnung auf das Doppelte im Vergleich zur Trapezregel. Die Werte der w_i und x_i finden sich tabelliert für verschiedene *N* in mathematischen Formelsammlungen. Wie bereits erwähnt heißt höhere Polynom-Ordnung nicht immer höhere Genauigkeit. Dies gilt nur, wenn sich die Funktion f auch durch Polynome hoher Ordnung approximieren läßt.

Eine Erweiterung der Gaußschen Integration erlaubt die Einbeziehung von bestimmten Gewichtsfunktionen im Integranden:

$$\int_a^b f(x)g(x)dx \approx \sum_{i=0}^{N} w_i f_i, \qquad f_i = f\left(x_i\right)$$

Die Werte der w_i und x_i finden sich tabelliert für verschiedene Gewichtsfunktionen *g*(*x*) und verschiedene *N* in mathematischen Formelsammlungen. Folgende Gewichtsfunktionen sind gängig:

Integrationsformel	Gewichtsfunktion
Gauß-Legendre	$g(x) = 1$
Gauß-Chebychev	$g(x) = \left(1 - x^2\right)^{-1/2}$
Gauß-Hermite	$g(x) = e^{-x^2}$
Gauß-Laguerre	$g(x) = x^{\alpha} e^{-x}$
Gauss-Jacobi	$g(x) = (1-x)^{\alpha}(1+x)^{\beta}$

Der Vorteil dieser Regeln ist, daß sie genau sind für Integranden, die sich als Produkt eines Polynoms und einer der oben angegebenen Gewichtsfunktionen schreiben lassen..

4.3 Aufgaben[5]

1. Schreibe eine Matlab-Funktion deriv zur numerischen Berechnung der Ableitung einer beliebigen Funktion *f* an einer vorgegebenen Stelle *x* nach der rechtsseitigen und der zentrierten Formel. Halbiere das Intervall *h* ausgehend von $h=1$ solange, bis sich der Wert bezüglich einer vorgegebenen Genauigkeit (z.B. 10^{-5}) nicht mehr ändert. (**Hinweis**: Schreibe *f* als eigenständige Matlab-Funktion und rufe diese zur Berechnung der Funktionswerte von *f* innerhalb von deriv auf).
2. Berechne mit deriv numerisch die Ableitung folgender Funktionen:

   a. $f(x)=sin(x)$, $x = 5/4*\pi$
   b. $f(x)=e^{-x}/x^2$, $x = 0.5$

   Plotte den absoluten Fehler zwischen der numerisch berechneten und der exakten Ableitung als Funktion des Intervalls *h*. Prüfe nach, ob der absolute Fehler mit *h* (rechtsseitige Formel) bzw. h^2 (zentrierte Formel) fällt (**Hinweis**: verwende eine doppelt logarithmische Darstellung).
3. (*) Verbessere die Funktion deriv aus Aufgabe 1. durch Richardson-Extrapolation und berechne erneut die Ableitungen 2.a und b. Bei welcher Intervallgröße *h* wird dieselbe Genauigkeit wie mit der zentrierten Formel erreicht?

Verwende für die folgenden Aufgaben die zur Verfügung gestellte Matlab-Funktion rombf zur Romberg-Integration:

4. Bestimme π durch Berechnung des Integrals über einen Viertel Einheitskreis auf 20 Stellen genau.

   Erstelle zunächst eine Funktion my_function, welche mit den Parametern x und n aufgerufen wird und den Funktionswert verschiedener mit n spezifizierten mathematischen Funktionen an der Stelle x zurückliefert.
5. (*) Bei der Fresnelschen Beugung spielen die Fresnel Integrale eine wichtige Rolle:

   $$C(w) = \int_0^w \cos\left(1/2\pi \cdot x^2\right)dx \qquad S(w) = \int_0^w \sin\left(1/2\pi \cdot x^2\right)dx$$

   Schreibe eine Funktion zur Berechnung von *C(w)* und *S(w)*. Trage *S* gegen *C* für *w* = 0.0, 0.1, 0.2,..., 5.0 gegeneinander auf. Was stellt der Plot dar?

[5] Aufgaben mit (*) sind optional.

5 gewöhnliche Differentialgleichungen

Differentialgleichungen (DGL) haben in der Physik eine besondere Bedeutung, seit Newton den Apfel fallen ließ und dies (und damit die Gravitation) mit der damals neuen Differentialrechnung beschrieb. Gewöhnliche DGLn enthalten nur Ableitungen nach einer Größe, etwa der Zeit *t*. Sind Ableitungen nach mehreren verschiedenen Größen enthalten, spricht man von partiellen DGLn. Beispiel für gewöhnliche DGLn sind z.B. das Federpendel und die Planetenbewegung. Die Schrödingergleichung und die Wellengleichung sind Beispiele für partielle DGLn. Viele DGLn lassen sich analytisch nicht lösen, so daß wiederum numerische Verfahren erforderlich sind.

5.1 mathematische Formulierung des Anfangswertproblems

Zur Ableitung der allgemeinen Form gewöhnlicher DGLn gehen wir als Beispiel von dem Newtonschen Gesetz $\underline{F} = m\underline{a}$ aus, das eine Beziehung zwischen der Kraft und der Beschleunigung vermittelt[6]. Bei gegebener Kraft lassen sich daraus die Bewegungsgleichungen ableiten:

$$\underline{a} = \frac{d^2\underline{r}}{dt^2} = \frac{\underline{F}}{m}$$

Diese vektorielle Gleichung zweiter Ordnung (2. Ableitung kommt vor) läßt sich in ein System aus 2 Gleichungen erster Ordnung umschreiben, indem wir eine neue Zwischenvariable einfügen. Dies ist immer möglich, jedoch i.a. nicht eindeutig: Es gibt mehrere Möglichkeiten der Variablendefinition. In diesem Fall bietet es sich an, als Zwischenvariable die Geschwindigkeit zu verwenden:

$$\frac{d\underline{r}}{dt} = \underline{v}$$
$$\frac{d\underline{v}}{dt} = \frac{\underline{F}\left(\underline{r}(t), \underline{v}(t), t\right)}{m}$$

$$\underline{r}\left(t = t_0\right) = \underline{r}_0$$
$$\underline{v}\left(t = t_0\right) = \underline{v}_0$$

Wir haben nun 2 Gleichungen erster Ordnung. Durch Integration der beiden Ableitungen ergeben sich 2 Integrationskonstanten, die durch Wahl der Anfangsbedingung festgelegt werden. Die Kraft kann sowohl von Ort und Geschwindigkeit (und damit implizit von der Zeit) und auch explizit von der Zeit abhängen. Nehmen wir an, das Problem ist 2-dimensional in kartesischen Koordinaten *x* und *y*. Dann können wir das Problem durch Einführung weiterer Variablen in eine allgemeine Form überführen:

$$\underline{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} \equiv \begin{pmatrix} r_x \\ r_y \\ v_x \\ v_y \end{pmatrix} \quad \text{und} \quad \underline{H} = \begin{pmatrix} H_1 \\ H_2 \\ H_3 \\ H_4 \end{pmatrix} \equiv \begin{pmatrix} v_x \\ v_y \\ F_x(\underline{z})/m \\ F_y(\underline{z})/m \end{pmatrix}$$

Dabei wurde die explizite Zeitabhängigkeit aller Größen jeweils weggelassen. Die generelle Form einer gewöhnlichen DGL als **Anfangswertproblem** ist mit dieser Definition:

$$\frac{d\underline{z}}{dt} = \underline{H}\left(\underline{z}(t), t\right) \quad \text{mit} \quad \underline{z}\left(t_0\right) = \underline{z}_0$$

[6] Alle vektoriellen Größen sind mit einem Unterstrich versehen.

Dabei ist $\underline{z}$ der gesuchte Lösungsvektor (im einfachsten Fall ist das eine skalare Funktion von t) und $\underline{H}$ eine lineare oder nichtlineare vektorielle Funktion in den angegebenen Argumenten $\underline{z}$ und t. $\underline{z}_0$ ist der Anfangswert, t stellt meist die Zeit dar. Die Funktion $\underline{H}$ läßt sich geometrisch als Vektorfeld interpretieren, daß in jedem Punkt des von $\underline{z}$ und t aufgespannten Raumes eine Richtung für $\underline{z}$ vorgibt.
Ein gegebenes Problem sollte in die angegebene allgemeine Form überführt werden. Dabei können sich je nach Wahl der Zwischenvariablen unterschiedliche Formulierungen ergeben. Das Auffinden sinnvoller Zwischenvariablen ist nicht immer einfach, da sich je nach Wahl unterschiedliche, vorher nicht immer abschätzbare numerische Probleme ergeben können. Im Zweifelsfall müssen mehrere Formulierungen gefunden und berechnet werden.

5.2 einfache Lösungsverfahren

Im folgenden werden einfache Lösungsverfahren aufgezeigt, die sich direkt aus den im letzten Kapitel abgeleiteten Formeln zur numerischen Differentiation ergeben. Die verschiedenen Ansätze sind sehr einfach zu berechnen und unterscheiden sich scheinbar relativ wenig. Experimentell stellt man jedoch fest, daß sie jeweils für unterschiedliche Probleme optimal anwendbar sind. Es ist nicht i.a. im voraus zu sagen, welche Methode wann gut ist, da die numerischen Fehler schlecht abschätzbar sind. Hier hilft nur Intuition und Erfahrung. Hinter allen Methoden steckt die Idee, die Ableitungen durch geeignete Näherung zu diskretisieren und damit die Werte der Variablen in einem Zeitschritt aus den Werten der Variablen im letzten Zeitschritt zu berechnen.

5.2.1 Euler-Methode

Gehen wir wiederum von den einfachen Bewegungsgleichungen aus:

$$\frac{d\underline{r}}{dt} = \underline{v}$$
$$\frac{d\underline{v}}{dt} = \frac{\underline{F}(\underline{r}(t), \underline{v}(t), t)}{m} = \underline{a}(\underline{r}(t), \underline{v}(t), t)$$

Nähern wir die Ableitung nach der rechtsseitigen Formel unter Benutzung eines Zeitschritts τ, so ergibt sich daraus:

$$\frac{\underline{r}(t+\tau) - \underline{r}(t)}{\tau} + O(\tau) = \underline{v}(t)$$
$$\frac{\underline{v}(t+\tau) - \underline{v}(t)}{\tau} + O(\tau) = \underline{a}(\underline{r}(t), \underline{v}(t), t)$$

oder

$$\underline{r}(t+\tau) = \underline{r}(t) + \tau\underline{v}(t) + O(\tau^2)$$
$$\underline{v}(t+\tau) = \underline{v}(t) + \tau\underline{a}(\underline{r}(t), \underline{v}(t), t) + O(\tau^2)$$

Beachte die Erhöhung der Ordnung des Abschneidefehlers durch Multiplikation mit τ. Durch Einführung folgender Notation vereinfachen sich die Gleichungen:

$$f_n \equiv f(n\tau) \quad ; \quad n = 0,1,2,\ldots$$

Damit ergibt sich::

$$\underline{r}_{n+1} = \underline{r}_n + \tau\underline{v}_n + O(\tau^2)$$
$$\underline{v}_{n+1} = \underline{v}_n + \tau\underline{a}_n + O(\tau^2)$$

Für den eindimensionalen Fall läßt sich dieser **Euler-Schritt** geometrisch interpretieren (siehe Abbildung 1). Ausgehend vom aktuellen Zustand r_n wird die Steigung des Ableitungs-Feldes v an der Stelle r_n bestimmt. Mit dieser Steigung wird der neue Wert r_{n+1} durch lineare Fortsetzung auf den nächsten Abtast-Zeitpunkt ermittelt.

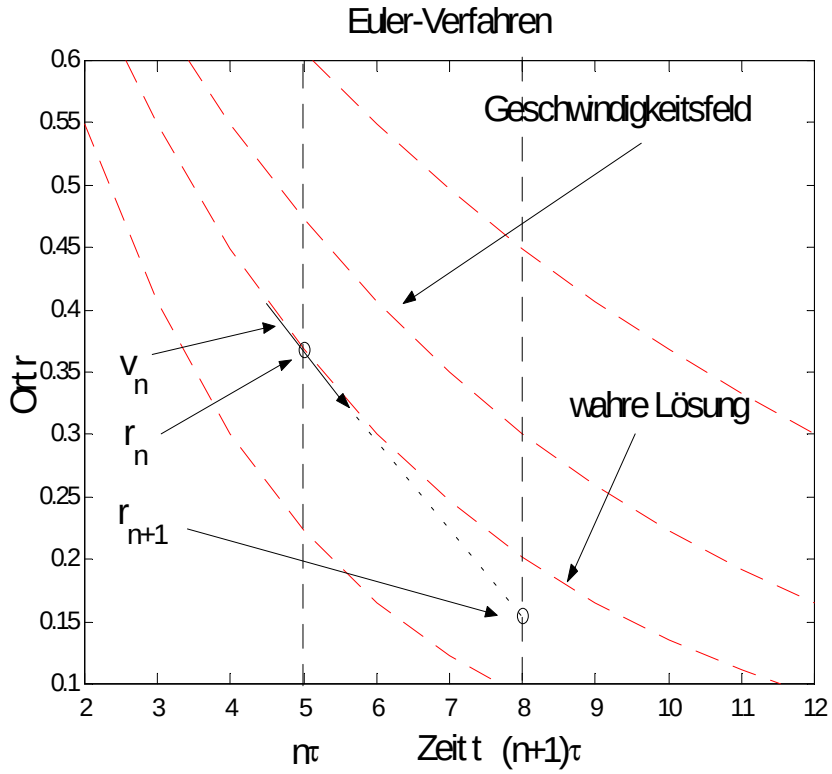


Abbildung 1: Geometrische Interpretation des Euler-Verfahrens (1-dimensional) anhand einer einfachen Lösungsschar (Exponentialfunktionen).

Um die Trajektorie als Folge von diskreten Punkten zu den Zeitpunkten $n\tau$ zu berechnen, ergibt sich damit folgender Algorithmus:

1. Lege die Anfangsbedingungen $\underline{r}_0$ und $\underline{v}_0$ fest.
2. Wähle einen Zeitschritt τ.
3. Berechne die Beschleunigung aus den aktuellen Werten von $\underline{r}$ und $\underline{v}$.
4. Benutze die Euler-Methode zur Berechnung der neuen Werte von $\underline{r}$ und $\underline{v}$.
5. Gehe zu Schritt 3.

5.2.2 Euler-Cromer-Methode

Eine einfache, zunächst nicht hergeleitete Modifikation der Euler-Methode sind die folgenden Gleichungen:

$$\underline{v}_{n+1} = \underline{v}_n + \tau \underline{a}_n + O\left(\tau^2\right)$$
$$\underline{r}_{n+1} = \underline{r}_n + \tau \underline{v}_{n+1} + O\left(\tau^2\right)$$

Die kleine Änderung besteht darin, daß die im betrachteten Zeitschritt neu berechnete Geschwindigkeit zur Berechnung des neuen Ortsvektors verwendet wird. Beachte, daß dadurch die Gleichungen nur in der angegebenen Reihenfolge berechnet werden können. Diese Methode hat denselben lokalen Fehler wie die Euler-Methode funktioniert aber in einigen Fällen viel besser als diese.

5.2.3 Mittelpunkts-Methode

Eine Mischung aus Euler und Euler-Cromer ist die Mittelpunkts-Methode:

$$\underline{v}_{n+1} = \underline{v}_n + \tau \underline{a}_n + O\left(\tau^2\right)$$
$$\underline{r}_{n+1} = \underline{r}_n + \tau \frac{\underline{v}_{n+1} + \underline{v}_n}{2} + O\left(\tau^3\right)$$

Dabei werden die beiden Geschwindigkeitswerte gemittelt, was die Ordnung der Ortsgleichung um 1 erhöht. Durch Einsetzen der ersten in die zweite Gleichung ergibt sich:

$$\underline{r}_{n+1} = \underline{r}_n + \underline{v}_n \tau + \frac{1}{2}\underline{a}_n \tau^2$$

Diese Gleichung ist exakt, falls die Beschleunigung $\underline{a}$ konstant ist. Dies macht die Methode interessant für Probleme, bei denen die Beschleunigung stetig mit langsamer Änderung verläuft. Ist die Beschleunigung zudem nicht von der Geschwindigkeit abhängig, so kann die Ortsgleichung alleine ohne Berechnung der Geschwindigkeit iteriert werden

5.2.4 Verlet-Methode

Die Verlet-Methode hat in der Ortsgleichung die Ordnung 4 und ist somit besonders gut für Probleme geeignet, bei denen es besonders auf diese Variable ankommt. Weiterhin läßt sich die Ortsgleichung, wie bereits bei der Mittelpunkts-Methode auch allein lösen, falls die Kraft bzw. Beschleunigung nur vom Ort abhängt und die Geschwindigkeit nicht von Interesse ist. Zur Ableitung der Verlet-Methode rechnen wir mit der ersten und zweiten Ableitung des Ortes nach der Zeit und nähern diese nach der zentrierten Formel für die erste bzw. zweite Ableitung:

$$\begin{aligned} \frac{d\underline{r}}{dt} &= \underline{v} \\ \frac{d^2\underline{r}}{dt^2} &= \underline{a} \end{aligned} \quad \Rightarrow \quad \begin{aligned} \frac{\underline{r}_{n+1} - \underline{r}_{n-1}}{2\tau} + O\left(\tau^2\right) &= \underline{v}_n \\ \frac{\underline{r}_{n+1} + \underline{r}_{n-1} - 2\underline{r}_n}{\tau^2} + O\left(\tau^2\right) &= \underline{a}_n \end{aligned}$$

Eine Umformung ergibt:

$$\begin{aligned} \underline{v}_n &= \frac{\underline{r}_{n+1} - \underline{r}_{n-1}}{2\tau} + O\left(\tau^2\right) \\ \underline{r}_{n+1} &= 2\underline{r}_n - \underline{r}_{n-1} + \tau^2 \underline{a}_n + O\left(\tau^4\right) \end{aligned}$$

Nachteil der Methode ist, daß sie nicht „selbst-startend“ ist, d.h. die Kenntnis der Anfangswerte $\underline{r}_0$ und $\underline{v}_0$ reicht nicht aus, die Trajektorie zu iterieren. Vielmehr müsssen $\underline{r}_0$ und $\underline{r}_1$ bekannt sein, was i.a. nicht der Fall ist. Um die Iteration zu starten kann $\underline{r}_1$ nach der Euler-Methode berechnet und dann mit der Verlet-Methode weiter iteriert werden.

5.3 globaler und lokaler Abschneidefehler

Bei jedem Zeitschritt ergibt sich je nach Methode ein Abschneidefehler, z.B $O(\tau^2)$ bei der Euler-Methode. Dieser Fehler wird im folgenden als **lokaler Fehler** bezeichnet, da er pro Zeitschritt auftritt. Betrachten wir ein Zeitintervall der Länge $T = N\tau$ zur Berechnung der Trajektorie (also N Zeitschritte), so ergibt sich maximal folgender **globaler Fehler** in der betrachteten Variable am Ende des Zeitintervalls:

$$\begin{aligned} &\text{globaler Fehler} \propto N \times \text{lokaler Fehler} \\ &= NO\left(\tau^n\right) = \left(T/\tau\right)O\left(\tau^n\right) = TO\left(\tau^{n-1}\right) \end{aligned}$$

Z.B. hat die Euler-Methode einen lokalen Abschneidefehler der Ordnung $O(\tau^2)$, aber einen globalen Abschneidefehler von nur $O(\tau)$. Diese Gleichung gibt uns nur eine Abschätzung des globalen Fehlers, da nicht angegebenen werden kann, ob sich die lokalen Fehler akkumulieren oder ausmitteln (d.h. konstruktiv oder destruktiv überlagern). Anschaulich hängt dies davon ab, ob und wenn ja wie viele Wendepunkte das Vektorfeld der Ableitungen im betrachteten Variablenbereich hat sowie, von der Art der Näherung der Ableitung (also von der verwendeten Methode). Diese Unkenntnis über die Fehlerakkumulation bedingt auch die nicht allgemein angebbare Eignung der verschiedenen Methoden für verschiedene Probleme.

5.4 Runge-Kutta-Verfahren 4. Ordnung

Um zu Lösungsformeln höherer Ordnung zu kommen, müssen wir den bisher zur Auffindung einfacher Formeln betriebenen empirischen Weg verlassen. Wir gehen dazu von der generellen Form des Anfangswertproblems aus:

$$\frac{d\underline{z}}{dt} = \underline{H}(\underline{z}(t), t) \quad \text{mit} \quad \underline{z}(t_0) = \underline{z}_0$$

Die einfachen Lösungsmethoden schätzen den mehrdimensionalen Ableitungsvektor $\underline{H}$ aus dem aktuellen Zustand $\underline{z}$ zur Zeit t einmal und berechnen daraus den Zustand zum nächsten Abtastzeitpunkt $t+\tau$. Für den eindimensionalen Fall

$$\frac{dz}{dt} = H(z(t), t)$$

läßt sich dies wie oben gezeigt geometrisch interpretieren. Es ist denkbar, mehrere mögliche Werte von H auf folgende Weise zu berechnen:

$$H_1 = H(z, t)$$
$$H_2 = H\left(z + \tfrac{1}{2}\tau\ H_1, t + \tfrac{1}{2}\tau\right)$$
$$H_3 = H\left(z + \tfrac{1}{2}\tau\ H_2, t + \tfrac{1}{2}\tau\right)$$
$$H_4 = H\left(z + \tau\ H_3, t + \tau\right)$$

Die Werte H_i lassen sich nur in der angegebenen Reihenfolge berechnen. Dabei wird zweimal ein Euler-Schritt mit der halben Schrittweite durchgeführt, wobei beim zweiten Mal die mit dem ersten halben Euler-Schritt geschätzte Ableitung verwendet wird. Zusätzlich wird ein ganzer Euler-Schritt mit der im zweiten halben Schritt geschätzten Steigung durchgeführt. Die Werte H_i lassen sich (wie in Abbildung 1 für den einfachen Euler-Schritt gezeigt) geometrisch interpretieren (siehe Abbildung 2).

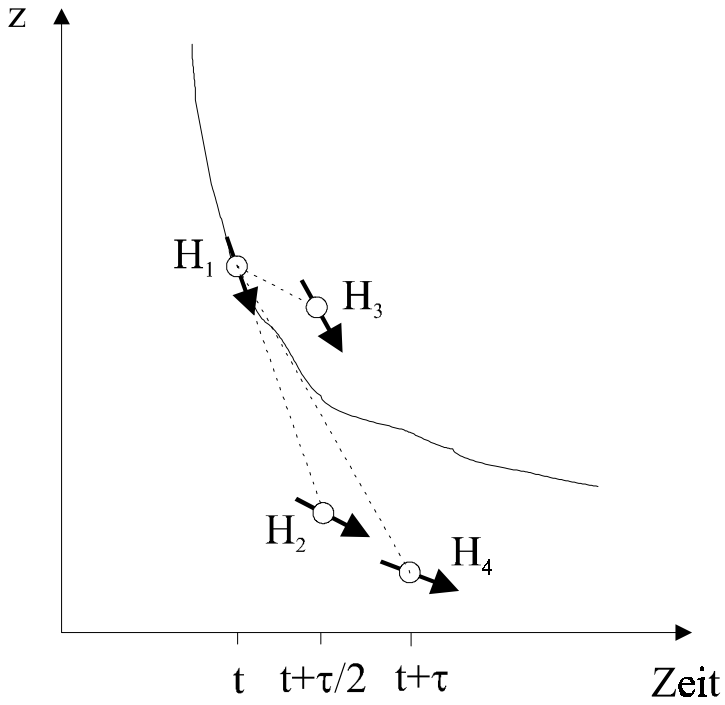


Abbildung 2: Berechnung der Schätzungen der Ableitungen für das Runge-Kutta-Verfahren 4. Ordnung (siehe Text).

Es stellt sich die Frage, wie aus diesen 4 Schätzungen der Ableitung der Zustand zum nächsten Abtastzeitpunkt berechnet wird. Sinnvoll erscheint eine gewichtete Mittelung der einzelnen Steigungen, um dann mit dieser mittleren Steigung einen Euler-Schritt zu machen. Mit Hilfe der Taylorentwicklung (o.B., siehe Textbücher) läßt sich beweisen, daß folgende Gewichte optimal sind, d.h. eine maximale Ordnung ergeben:

$$z(t+\tau) = z(t) + \tau \sum_{i=1}^{4} w_i H_i + O\left(\tau^5\right)$$

$$w_i = \left\{\tfrac{1}{6}, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{6}\right\}$$

Die Summe der Gewichte ist also wie gefordert 1. Die mittleren Steigungen werden doppelt gewichtet. Insgesamt läßt sich dieses Vorgehen vektoriell für die allgemeine Form des Anfangswertproblems folgendermaßen schreiben:

$$\underline{z}(t+\tau) = \underline{z}(t) + \tfrac{1}{6}\tau \left(\underline{H}_1 + 2\underline{H}_2 + 2\underline{H}_3 + \underline{H}_4\right) + O\left(\tau^5\right)$$

$$\begin{aligned}
\underline{H}_1 &= \underline{H}(\underline{z}, t) \\
\underline{H}_2 &= \underline{H}\left(\underline{z} + \tfrac{1}{2}\tau\ \ \underline{H}_1, t + \tfrac{1}{2}\tau\right) \\
\underline{H}_3 &= \underline{H}\left(\underline{z} + \tfrac{1}{2}\tau\ \ \underline{H}_2, t + \tfrac{1}{2}\tau\right) \\
\underline{H}_4 &= \underline{H}\left(\underline{z} + \tau\ \ \underline{H}_3, t + \tau\right)
\end{aligned}$$

Das hier angegebene Runge-Kutta-Verfahren ist 4. Ordnung in jeder Variable (d.h. in jeder Komponente von $\underline{z}$), hat also einen lokalen Abschneidefehler der Ordnung τ^5. Natürlich ließen sich auf die gleiche Weise Formeln noch höherer Ordnung finden, jedoch sind diese komplexer, d.h. erfordern mehr Auswertungen der Funktion $\underline{H}$ pro Zeitschritt. Runga-Kutta 4. Ordnung hat sich als der beste Kompromiß zwischen Genauigkeit und Komplexität herausgestellt. Es stellt daher nach gemeinsamer Aussage aller Textbücher das „Arbeitspferd“ zur Lösung der gew. DGL'n dar, insbesondere wenn es mit einer adaptiven Schrittweitenregelung verknüpft ist, bei der der Zeitschritt τ je nach geschätztem lokalen Abschneidefehler variiert wird (siehe nächster Abschnitt). Die sog. **Prädiktor-Korrektor- Methoden** ermöglichen darüber hinaus eine genauere Kontrolle der lokalen Fehler und der Schrittweiten sowie ermöglichen die Berechnung eines Korrekturterms zur Erhöhung der Genauigkeit. Diese Methoden sollen in diesem Rahmen nicht behandelt werden, siehe dazu die einschlägigen Textbücher.

5.4.1 adaptive Schrittweitenregelung

Die Schrittweite sollte immer so gewählt werden, daß sie ein Bruchteil der System-inhärenten Zeitskalen ist (z.B. 1/50-tel der Schwingungsperiode des Systems)[7]. Dennoch kann das System während der Zeitentwicklung in Zustände mit stark unterschiedlichem Betrag des Ableitungsfeldes geraten. Sind die Ableitungen (d.h. die Änderungen des Systemzustands) groß, so muß ein kleiner Zeitschritt verwendet werden um den lokalen Fehler unter einer vorgegebenen Schranke zu halten. In Bereichen mit langsamer Änderung des Systemzustands reichen dazu längere Schrittweiten aus. Durch die Schrittweitenregelung kann die Anzahl der zu berechnenden Zustände minimiert werden, wobei der zusätzliche Aufwand zur Berechnung der Schrittweiten meist kleiner ist als der eingesparte Berechnungsaufwand.
Die Idee zur adaptiven Berechnung der Schrittweite liegt darin, den lokalen Abschneidefehler für jeden Zeitschritt neu zu schätzen und die Schrittweite so einzustellen, daß er unter einer vorgegebenen Schranke bleibt. Dazu wird der neue Systemzustand zweifach berechnet, einmal durch einen Schritt mit der aktuellen Schrittweite und einmal durch zwei Schritte mit der halben Schrittweite:

$$\underline{z}(t) \rightarrow \quad \rightarrow \quad \rightarrow \underline{z}_1(t+\tau)$$

$$\underline{z}(t) \rightarrow \underline{z}\left(t + \tfrac{1}{2}\tau\right) \rightarrow \underline{z}_2(t+\tau)$$

[7] Problematisch sind dabei Systeme mit stark unterschiedlichen Zeitskalen. Die kürzeste bestimmt immer die Schrittweite, so daß kurze Schrittweiten verwendet werden müssen, auch wenn die langen Zeitskalen überwiegen. Diese sog. **steifen DGL'n** werden oft mit impliziten Schemata gelöst (siehe Textbücher).

Der lokale Abschneidefehler wird als die Differenz der beiden Werte geschätzt:

$$\Delta \underline{z} \approx \left| \underline{z}_2 - \underline{z}_1 \right|$$

Der relative Fehler kann dann folgendermaßen geschätzt werden:

$$r(\underline{z}) = \frac{\Delta \underline{z}}{|\underline{z}|} \approx \frac{\left| \underline{z}_2 - \underline{z}_1 \right|}{\left| \frac{1}{2}\left(\underline{z}_2 + \underline{z}_1 \right) \right|}$$

Da der lokale Abschneidefehler proportional zur fünften Potenz der Schrittweite τ ist (Runge-Kutta 4. Ordnung), lassen sich die aktuellen Werte von Schrittweite und Abschneidefehler mit den zu erwartenden Größen bei Wahl einer neuen Schrittweite τ_{neu} in Beziehung setzen:

$$\left(\frac{\tau}{\tau_{neu}} \right)^5 = \frac{\Delta \underline{z}}{\Delta} = \frac{r(\underline{z})}{r}$$

Diese Gleichung läßt sich wie angegeben auch für die relativen Fehler schreiben, da sich die Normierungsfaktoren kürzen. Bei einer vorgegebenen relativen Fehlergrenze *r* (etwa 10^{-3}) kann somit die neue Schrittweite ausgerechnet werden, die gemäß der vorliegenden Schätzung des relativen Fehlers die vorgegebene Fehlergrenze einhält. Dazu wird die Gleichung nach τ_{neu} aufgelöst:

$$\tau_{neu} = S\tau \left(\frac{r}{r(\underline{z})} \right)^{1/5}$$

Dabei ist zusätzlich ein Sicherheitsfaktor $S \cong 0.9$ eingefügt, damit die Änderung nicht zu stark wird. Zusätzlich sollte sich die neue Schrittweite nicht um mehr als einen vorgegebenen Faktor zu höheren bzw. niedrigeren Werten ändern (etwa Faktor 4 bzw. ¼). Diese Begrenzung ist notwendig, da die Schätzung des Abschneidefehlers ungenau ist und der Quotient in obiger Gleichung singulär werden kann.
Die Schrittweite wird pro Zeitschritt mehrfach gemäß der angegebenen Formel (plus Begrenzung) verkleinert, bis der mit der neuen Schrittweite geschätzte Abschneidefehler kleiner als der vorgegebene ist. Umgekehrt wird die Schrittweite pro Zeitschritt maximal einmal erhöht. Dadurch wird die Gefahr vermindert, daß der Zeitschritt aufgrund ungenauer Schätzung des Abschneidefehlers zu groß wird, was die Genauigkeit extrem reduzieren kann. Wird der Zeitschritt zu klein geschätzt, bedeutet dies zwar einen höheren Rechenaufwand, reduziert jedoch nicht die Genauigkeit.

5.5 Anwendungen verschiedener Lösungsverfahren für Anfangswertprobleme

Die vorgestellten Lösungsverfahren lassen sich auf verschiedene einfache physikalische Systeme anwenden. Im folgenden werden einige davon vorgestellt. Es wird jeweils untersucht (z.T. in Form von Übungsaufgaben), wie sich die verschiedenen Lösungsmethoden im Hinblick auf die physikalischen Probleme verhalten.

5.5.1 schräger Wurf

Der Wurf eines Balles wird in 2 Dimensionen $\begin{pmatrix} x \\ y \end{pmatrix}$ betrachtet. Es wirke die Gravitationskraft in negativer y-Richtung und eine Reibungskraft, die proportional zum Quadrat der Geschwindigkeit ist und gegen die aktuelle Bewegungsrichtung wirkt. Die Bewegungsgleichungen sind dann:

$$\frac{d\underline{r}}{dt} = \underline{v}$$

$$\frac{d\underline{v}}{dt} = \frac{\underline{F}(\underline{r}(t), \underline{v}(t), t)}{m}$$

$$= \frac{\underline{F}_a(\underline{v}) + \underline{F}_g}{m}$$

mit

$$\underline{F}_a(\underline{v}) = -\frac{1}{2} c_w \rho \; A |\underline{v}| \underline{v}$$

$$\underline{F}_g = -mg \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Dabei sind:

$c_w \cong 0.35$ (Luftwiderstandskoeffizient)

$\rho \cong 1.2 \frac{kg}{m^3}$ (Dichte von Luft)

$A\left[m^2\right]$ und $m\left[kg\right]$ als der Querschnittsfläche und Masse des Balles.

$g\left[\frac{kg \cdot m}{s^2}\right]$ Erdbeschleunigung

Ist die Reibung vernachlässigbar, so sind die analytischen Lösungen bekannt. Wird der Ball am Ursprung mit einer Geschwindigkeit v_0 unter einem Winkel Θ zur Horizontalen geworfen, so sind Wurfweite, maximale Höhe und Flugzeit:

$$x_{\max} = \frac{2v_0^2}{g} \sin\Theta \cos\Theta$$

$$y_{\max} = \frac{v_0^2}{2g} \sin^2\Theta$$

$$T = \frac{2v_0}{g} \sin\Theta$$

Dieser Grenzfall kann zur Überprüfung der numerischen Lösung herangezogen werden. Weiterhin kann die Schrittweite physikalisch sinnvoll gewählt werden, indem sie als Bruchteil der zu erwartenden Flugzeit angegeben wird. Der schräge Wurf kann mit der Euler-Methode integriert werden (siehe Aufgaben).

5.5.2 physikalisches Pendel

Wir betrachten ein Pendel, das idealisiert als Massepunkt der Masse *m* modelliert wird, der mit Hilfe eines masselosen Arms der Länge *L* um eine Achse schwingt. Die Bewegungsgleichung dafür lautet (siehe Textbücher):

$$\frac{d\Theta}{dt} = \omega$$

$$\frac{d\omega}{dt} = -\frac{g}{L} \sin\Theta$$

Dabei ist *g* die Erdbeschleunigung. Die Winkelbeschleunigung ist eine nichtlineare Funktion der Auslenkung und es gibt keine analytische Lösung. Für kleine Auslenkungswinkel Θ kann die Gleichung aber linearisiert werden:

$$\sin\Theta \approx \Theta \Rightarrow \frac{d\omega}{dt} = -\frac{g}{L}\Theta$$

Für dieses mathematische Pendel ergibt sich die Lösung:

$$\Theta(t) = \exp\left(i2\pi\frac{t}{T_0}\right) \quad , \quad T_0 = 2\pi\sqrt{\frac{L}{g}}$$

Wird das Verhältnis L/g auf 1 normiert, so ist die Schwingungsperiode T_0 identisch 2π. Die Schrittweite kann dann physikalisch sinnvoll als Bruchteil der Schwingungsperiode eingestellt werden. Mit Hilfe der Energieerhaltung kann überprüft werden, ob die numerische Lösung physikalisch sinnvoll ist. Dazu wird numerisch für jeden Zeitschritt die Gesamtenergie

$$E_{ges} = \frac{1}{2}mL^2\omega^2 - mgL\cos\Theta$$

ausgerechnet und als Funktion der Zeit aufgetragen (siehe Übungen).

Nichtlineare Effekte zeigen sich bei großen Auslenkungswinkeln durch eine Verformung der Schwingungsform relativ zur Sinusfunktion. In diesem Fall hängt auch die Schwingungsperiode vom maximalen Auslenkungswinkel θ_m ab (Ableitung siehe nächster Unterabschnitt):

$$\begin{aligned} T_0 &= 4\sqrt{\frac{L}{g}}\cdot K\left(\sin\left(\frac{1}{2}\theta_m\right)\right) \\ &\approx 2\pi\sqrt{\frac{L}{g}}\left(1+\frac{1}{16}\theta_m^2+\ldots\right) \end{aligned}$$

Für kleine θ_m geht die Formel in die o.a. Formel für den linearen Fall über. Dabei ist K das vollständige elliptische Integral 1. Art:

$$K(x) = \int_0^{\pi/2}\frac{1}{\sqrt{1-x^2\sin^2 z}}dz$$

Wird zusätzlich zu dem nichtlinearen Term der Rückstellkraft eine Reibungskraft proportional zur Winkelgeschwindigkeit sowie eine harmonische Anregung eingeführt, läßt sich an diesem einfachen System der Weg in das **deterministische Chaos** aufzeigen:

$$\begin{aligned} \frac{d\Theta}{dt} &= \omega \\ \frac{d\omega}{dt} &= -\frac{g}{L}\sin\Theta - \alpha\omega + A_0\sin(\omega_0 t) \end{aligned}$$

Reicht die Anregungsamplitude aus, das Pendel zum Überschlag zu bringen, führt es, je nach Einstellung von Dämpfung und Anregung zu einer chaotischen Bewegung. Zur Analyse kann die Trajektorie der Bewegung im **Phasenraum**, d.h. dem durch Θ und ω aufgespannten Raum verfolgt werden. Trägt man nur Punkte der Trajektorie auf, die zu einer bestimmten Phase der Anregung (z.B. Nullphase) angenommen werden, kommt man zum sogenannten **Poincaré-Schnitt**. Ist die Bewegung periodisch ergeben sich nur wenige Punkte, wobei die Anzahl der Punkte die Vielfachheit der Schwingungsperiode relativ zur Anregungsperiode angibt. Der Weg ins Chaos kann dann über die sog. Periodenverdopplung (Bifurkation) nachvollzogen werden: Trägt man die Poincaré-Punkte (nur Θ-Komponente) als Histogramm über der Anregungsamplitude auf, zeigen sich Fenster mit periodischen Bewegungen. Bei Erhöhung der Anregungsamplitude verdoppelt sich die Vielfachheit der Periode. Eine Vielzahl von solchen Periodenverdopplungen führt dann zu chaotischen Bewegungen. Dies ist ein Beispiel

für deterministisches Chaos: Die zugrundeliegende Gleichung ist wohldefiniert und deterministisch. Dennoch ist die Bewegung nicht vorhersagbar, da sie extrem von den Anfangsbedingungen abhängt. Bei numerischer Rechnung akkumulieren zusätzlich die Abschneidefehler.

5.5.2.1 Ableitung der Formel für die Schwingungsperiode

Das ungedämpfte physikalische Pendel schwingt periodisch. Die Maximalauslenkung betrage θ_m. Im Umkehrpunkt, also bei $\theta = \theta_m$, ist die Winkelgeschwindigkeit ω gleich 0. Die Gesamtenergie in diesem Punkt ist dann

$$E_{ges}\Big|_{\Theta=\Theta_m} = -mgL\cos\Theta_m \ .$$

Aufgrund der Konstanz der Gesamtenergie gilt dann für alle Zeitpunkte

$$\frac{1}{2}mL^2\omega^2 - mgL\cos\Theta = -mgL\cos\Theta_m$$

$$\Rightarrow \qquad \omega^2 = \frac{2g}{L}\left(\cos\Theta - \cos\Theta_m\right)$$

$$\Rightarrow \qquad \frac{d\Theta}{dt} = \sqrt{\frac{2g}{L}\left(\cos\Theta - \cos\Theta_m\right)}$$

$$dt = \sqrt{\frac{L}{2g}}\,\frac{d\Theta}{\sqrt{\cos\Theta - \cos\Theta_m}}$$

Das Pendel schwingt im Zeitraum von $t = 0$ bis $t = T_0/4$ (1/4 der gesuchten Schwingungsperiode) von bei $\theta = 0$ bis $\theta = \theta_m$. Beide Seiten der Gleichung können daher integriert werden:

$$\int_0^{T_0/4} dt = \sqrt{\frac{L}{2g}}\int_0^{\Theta_m}\frac{d\Theta}{\sqrt{\cos\Theta - \cos\Theta_m}}$$

$$\Rightarrow \qquad T_0/4 = 2\sqrt{\frac{L}{g}}\int_0^{\Theta_m}\frac{d\Theta}{\sqrt{\sin^2\left(\Theta_m/2\right) - \sin^2\left(\Theta/2\right)}}$$

mit der Ersetzung

$$\cos 2\Theta = 1 - 2\sin^2\Theta$$

Durch entsprechende Umformung mittels des o.a. vollständigen elliptischen Integrals 1. Art ergibt sich die o.a. Formel für die Schwingungsperiode T_0.

5.5.3 Planetenbewegung

Wir betrachten die Bewegung eines Körpers in einem Zentralpotential, z.B. ein Komet im Gravitationsfeld der Sonne. Unter Vernachlässigung aller weiteren Kräfte (etwa Planeten, Sonnenwind) ist die Kraft auf den Körper im Koordinatensystem der Sonne:

$$\underline{F}\left(\underline{r}\right) = -\frac{GmM}{\left|\underline{r}\right|^3}\underline{r}$$

Dabei ist m die Masse des Körpers, M die Masse der Sonnen und G die Gravitationskonstante. Es ist sinnvoll in astronomischen Einheiten zu rechnen. Dann ist $GM=4\pi^2$ AU^3/yr^2, wobei AU=1.496x10^{11}m der mittlere Abstand Sonne-Erde und yr ein Jahr ist. Weiterhin kann die Masse m des Körpers identisch 1 gesetzt werden. Damit kann die Bewegungsgleichung numerisch integriert werden. Wir wissen von den Keplerschen Gesetzen, daß die Lösungen Ellipsen sind. Weiterhin sind Gesamtenergie und Drehimpuls

$$E = \frac{1}{2}m\left|\underline{v}\right|^2 - \frac{GmM}{\left|\underline{r}\right|} \qquad , \qquad \underline{L} = \underline{r}\times\left(m\underline{v}\right)$$

erhalten. Diese physikalischen Randbedingungen erlauben eine Überprüfung der numerischen Lösung. Energie- und Drehimpulserhaltung sind auch bei der numerischen Lösung von Mehrkörperproblemen anwendbar, wenn keine analytische Lösung mehr möglich ist.
Bei der Anwendung der verschiedenen Lösungsverfahren wird klar, daß bei (näherungsweise) Kreisbahnen die Euler-Methode versagt und besonders das Euler-Cromer-Verfahren geeignet ist. Bei exzentrischen Bahnen versagt auch Euler-Cromer. Hier ist das adaptive Runge-Kutta-Verfahren 4. Ordnung die Methode der Wahl, ebenso beim Drei- oder Mehrkörperproblem.

5.5.4 Lorenz-Modell

Früher wurde angenommen, daß Wetter sei nur aufgrund der hohen Anzahl der Variablen unvorhersehbar. Folgerichtig wurde mit der Einführung leistungsfähiger Computer die Hoffnung verknüpft, daß damit eine langfristige Vorhersage möglich sein müßte. Diese Hoffnung hat sich nicht erfüllt. Wir wissen heute, daß das Wetter intrinsisch unvorhersehbar ist, da die zugrundeliegenden Gleichungen nichtlinear ist und deren Lösung extrem sensitiv auf Änderungen der Anfangsbedingungen reagiert („Schmetterlingseffekt“). Dies kann schon anhand von Systemen mit wenigen Variablen gezeigt werden. So wurde von E. Lorenz ein Modell untersucht, daß mit drei Variablen die Konvektion einer Flüssigkeit zwischen zwei parallelen Platten beschreibt, die einen festen Temperaturunterschied aufweisen:

$$\frac{dx}{dt} = \sigma\ (y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz$$

Die Ableitung des Modells findet sich Textbüchern (z.B. H.G. Schuster: „Deterministic Chaos“). Hier sei nur darauf hingewiesen, daß x in etwa die zirkuläre Strömungsgeschwindigkeit, y den Temperaturgradienten zwischen aufsteigenden und absteigenden Flüssigkeitsvolumina und z die Abweichung des Temperaturverlaufs vom Gleichgewichtszustand (nur Wärmeleitung) darstellen. Die Parameter σ und b hängen von den Eigenschaften der Flüssigkeit sowie der Geometrie der Platten ab. Typische Werte sind $\sigma = 10$ und $b = 8/3$. Der Parameter r ist proportional zum äußeren Temperaturgradienten.
Um die Sensitivität des Systems bezüglich der Anfangsbedingungen zu untersuchen, bietet es sich an, die Gleichungen mit 2 leicht unterschiedlichen Anfangsbedingungen zu integrieren (Runge-Kutta mit adaptiver Schrittgröße) und die Zeitverläufe der Variablen im Vergleich zu plotten (siehe Übungen). Weiterhin bietet sich wieder die Darstellung der Trajektorie im Phasenraum an. Sie bewegt sich auf einem aperiodischen Orbit, der im Phasenraum relativ lokalisiert liegt, d.h. auf einem **Attraktor** (siehe Übungen).

5.6 Randwertprobleme

Bisher wurde nur das Anfangswertproblem behandelt, wobei aus einem vollständig bekannten Anfangszustand die Zeitentwicklung iteriert wird. Natürlich muß es immer genügend Nebenbedingungen geben, um die Integrationskonstanten festzulegen. Jedoch müssen diese nicht, wie bei dem Anfangswertproblem, zu einer festen Zeit t_0 festgelegt werden. Dies führt auf die sog. **Randwertprobleme**, bei denen etwa ein Teil des Anfangszustandes bei t_0 und ein Teil des Endzustandes nach einer Zeit T gegeben ist. Zum Beispiel könnte beim schrägen Wurf der Ort zur Zeit t=0 und t=T vorgegeben sein, die Geschwindigkeit jedoch frei sein. Die o.a. Formeln zur Berechnung der Zeitentwicklung lassen sich dann nicht mehr direkt anwenden. Vielmehr müßte man Annahmen über den kompletten Anfangszustand machen, diesen nach den o.a. Formeln iterieren und ihn solange variieren, bis alle Randbedingungen erfüllt sind. Das bedeutet natürlich, daß man die Trajektorie je nach Geschick bei der Variation der Anfangsbedingungen vielfach berechnen muß, bis sich die richtige findet. Bessere Methoden zur Lösung der Randwertprobleme sind **Relaxationsmethoden**, die im Zusammenhang mit partiellen DGL’n behandelt werden.

5.7 Aufgaben

1. Erstelle eine Funktion flugball, welche den schrägen Wurf eines Volleyballes in 2 Dimensionen numerisch kalkuliert. Wende dazu das Euler-Verfahren an. Als Parameter sollten die Anfangsbedingungen des Problems $\vec{x}(0)$ und $\vec{v}(0)$ übergeben werden. Eine graphische Darstellung der Flugbahn sowie Flugzeit und Wurfweite sollten ausgegeben werden.

   Hinweis: Eine sich bewegende Kugel erfährt durch die Luftreibung eine Bremsbeschleunigung von ca.

   $\vec{a}_{\mathrm{Reibung}} = -0.5 c_w \rho \frac{A}{m} \vec{v}|\vec{v}|$ mit

   $c_w \cong 0.35$ (Luftwiderstandskoeffizient)

   $\rho \cong 1.2 \frac{kg}{m^3}$ (Dichte von Luft)

   $A\left[m^2\right]$ und $m\left[kg\right]$ als der Querschnittsfläche und Masse des Balles.

   Die Gravitation sollte natürlich auch nicht vernachlässigt werden ;-)
2. Berechne das Schwingungsverhalten eines physikalischen Fadenpendels mittels der Euler- und der Verlet-Methode. Die Funktion pendel sollte als Parameter die Anfangsbedingungen sowie Schrittweite und Art des Lösungsverfahrens (Euler / Verlet) erwarten. Als Ausgabe sollen der Auslenkungswinkel und die Gesamtenergie in Abhängigkeit von der Zeit dargestellt werden.
   Zur Erinnerung die Bewegungsgleichungen:

   $$\frac{d\omega}{dt} = -\frac{g}{L} \cdot \sin(\Theta)$$

   $$\frac{d\Theta}{dt} = \omega$$

   Die Gesamtenergie des Systems beträgt:

   $$E = \frac{1}{2} m L^2 \omega^2 - m g L \cos(\Theta)$$

   Hinweis:

   Setze zur Vereinfachung $\frac{g}{L} = 1$ und betrachte die normierte Energie $\frac{E}{m \cdot L^2}$.

   Untersuche die Fälle $\left[\Theta_0 = 10^\circ; \tau = 0{,}1\right]$ und $\left[\Theta_0 = 10^\circ; \tau = 0{,}05\right]$ mit der Euler-, sowie $\left[\Theta_0 = 10^\circ; \tau = 0{,}1\right]$ und $\left[\Theta_0 = 170^\circ; \tau = 0{,}1\right]$ mit der Verlet-Methode !

   Was kann über die Eignung der Verfahren für dieses Problem ausgesagt werden ?
3. Betrachte das angetriebene, gedämpfte physikalische Pendel:

   $$\frac{d\Theta}{dt} = \omega$$

   $$\frac{d\omega}{dt} = -\frac{g}{L} \sin\Theta - \alpha\omega + A_0 \sin\left(\omega_0 t\right)$$

   a.) Integriere das System mit der Verlet-Methode. Setze $\omega_0 = \sqrt{\frac{g}{L}} \equiv 1$ und berechne etwa 80 Perioden (t=0 - 80*2π). Schreibe ein Plotprogramm, das die Trajektorie im Phasenraum (ω über Θ) sowie Auslenkung Θ und Gesamtenergie als Funktion der Zeit zeichnet.

   b.) Erzeuge durch geeignete Wahl der Parameter folgende Schwingungsformen:

   (i) Ausschwingen des gedämpften Pendels ohne Anregung (Hinweis: beginne mit $A_0 = 0$ und variiere die Dämpfung α solange, bis das Pendel aus einer Ausgangsposition von Θ =90 Grad mit etwa 5-10 Perioden ausschwingt).

(ii) Grenzzyklus (Hinweis: kleine Dämpfung und kleine Anregung, Schwingung ohne Überschlag).
(iii) Schwebung (Hinweis: keine (!) Dämpfung und kleine Anregung, Schwingung ohne Überschlag).
(iv) chaotische Bewegung (Hinweis: Erhöhe die Anregung, bis es zu einem Überschlag kommt und darüber hinaus).

Erläutere, wie es zu den Schwingungsformen kommt und dokumentiere die gefundenen Parametersätze.

c.) Erzeuge ausgehend davon einen Poincaré-Schnitt, indem nur die Punkte im Phasenraum aufgetragen werden, die zur Nullphase der Anregung gehören:

$$A = A_0 \sin\left(\omega_0 t^*\right) \quad , \quad \omega_0 t^* = n2\pi \quad , \quad n \in N \, .$$

Verwerfe dabei die Punkte, die während des Einschwingvorgangs aufgenommen werden (etwa 20 Perioden, d.h. bis t=40*π). Was läßt sich über die vier Fälle aus b.) aussagen?

d.) Trage die Poincaré-Punkte (Θ-Komponente) als Histogramm über der Anregungsamplitde auf („Bifurkationsdiagramm“). Erzeuge ein Bifurkationsdiagramm mit Anregungsamplituden zwischen etwa 0 und 5 rad/s mit einer Schrittweite von 0.5 rad/s. Verwende den Wert der Dämpfung aus Aufgabe 1.b.i). Identifiziere den interessanten Bereich des Übergangs von periodischer zu chaotischer Bewegung und berechne diesen Bereich mit einer Auflösung von 0.1rad/s neu. Beschreibe das Ergebnis.

4. Betrachte einen Körper im Zentralpotential. Auf ihn wirkt die Kraft:

$$\underline{F}\left(\underline{r}\right) = -\frac{GmM}{\left|\underline{r}^3\right|}\underline{r}$$

a.) Programmiere das adaptive Runge-Kutta-Verfahren 4. Ordnung. Schreibe dazu eine Funktion
[x, t, tau] = rka(x,t,tau,err,derivsRK,param) zur Berechnung eines Zeitschrittes, mit:
x = aktueller wert der abhängigen Variable
t = unabhängige Variable (Normalerweise Zeit
tau = aktuelle Schrittgröße
err = gewünschter lokaler Abschneidefehler (etwa 10^-3)
derivsRK = rechte Seite der DGL (Name der Funktion, die dx/dt zurückgibt
(Aufruf-format: derivsRK(t,x,param)).
param = extra Parameter für derivsRK
x = neuer Wert der abhängigen Variable
t = neuer Wert der abhängigen Variable
tau = empfohlener wert für tau beim nächsten Aufruf von rka

b.) Integriere das System mit Euler-Cromer und rka für eine zirkuläre und eine exzentrische Bahn. Plotte jeweils die Trajektorie im Ortsraum für 10 Perioden sowie den Verlauf der Gesamtenergie. Wie unterscheiden sich die beiden Verfahren?

c.) Trage für die exzentrische Bahn die von rka für jeden Schritt neu ermittelte Schrittgröße als Funktion der zugehörigen radialen Distanz zum Ursprung doppelt logarithmisch auf. Welche Gesetzmäßigkeit läßt sich ableiten? (Hinweis: Keplersche Gesetze)

5. Betrachte das Lorenz-Modell

$$\frac{dx}{dt} = \sigma \left(y - x\right)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz$$

Integriere das System mit rka. Plotte x und z als Funktion von t sowie die Trajektorie im Phasenraum (zx- und yx-Projektion). Verwende die Parameter σ=10, b=8/3 und r=28 sowie die Anfangsbedingungen:

a.) x=1, y=1, z=20
b.) x=1, y=1, z=20.01

Wie unterscheiden sich die zu den verschiedenen Anfangsbedingungen gehörenden Bahnen?

6 Lösung von Gleichungssytemen

Im letzen Abschnitt wurde die Lösung der gew. DGL

$$\frac{d\underline{z}}{dt} = \underline{H}(\underline{z}(t), t) \quad \text{mit} \quad \underline{z}(t_0) = \underline{z}_0$$

behandelt. Ausgehend vom Anfangszustand kann die Zeitentwicklung des Systems als Folge von Zuständen $\underline{z}(n\tau)$ bestimmt werden, etwa mit dem Runge-Kutta-Verfahren. Bisher wurden nur **autonome Systeme** behandelt, bei denen die Kraft und damit $\underline{H}$ nicht explizit von der Zeit abhängt, d.h. $\underline{H}(\underline{z}(t), t) = \underline{H}(\underline{z}(t))$. Für diese Klasse von Systemen existiert oft eine Klasse von Anfangsbedingungen $\underline{z}_0^*$ für die gilt: $\underline{z}(n\tau) \equiv \underline{z}_0^*$. Diese Zustände im *N*-dimensionalen Phasenraum (Zustandsraum) heißen **stationär**. Folgende Äquivalenz ist leicht zu sehen:

$$\underline{z}_0^* \text{ stationärer Zustand} \quad \Leftrightarrow \quad \underline{H}(\underline{z}^*) \equiv \underline{0} \quad ,$$

denn die letztere Bedingung bedeutet $\frac{d\underline{z}}{dt} \equiv \underline{0}$. Die Vektorgleichung $\underline{H}(\underline{z}^*) = \underline{0}$ enthält *N* Gleichungen mit *N* Unbekannten. Zum Auffinden der stationären Zustände müssen also die Wurzeln dieses Gleichungssystems gefunden werden. Dieses Problem zerfällt in 2 Klassen. Falls $\underline{H}$ eine lineare Funktion in $\underline{z}$ ist oder in der Umgebung von erwarteten stationären Zuständen linearisierbar ist, können Methoden der linearen Algebra zur Lösung benutzt werden. Falls $\underline{H}$ nichtlinear ist (z.B. enthält das Fadenpendel die nichtlineare cos-Funktion), so kann das Newtonsche Gradientenverfahren verwendet werden. Beide Klassen von Gleichungen werden im folgenden behandelt. Zusätzlich gibt es spezielle Verfahren, falls $\underline{H}$ ein Polynom in $\underline{z}$ ist, die aber in diesem Rahmen nicht behandelt werden sollen (siehe Textbücher und die Funktion **roots** in Matlab, die Nullstellen von Polynomen findet).

6.1 lineare Gleichungssysteme

Ein lineares Gleichungssystem besteht i.a. aus *M* Gleichungen mit *N* Unbekannten:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N - b_1 &= 0 \\ &\vdots \\ a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N - b_M &= 0 \end{aligned}$$

Dabei sind die x_i die zu bestimmenden Unbekannten und a_{ij} und b_i die (festen) Koeffizienten. Dieses System läßt sich in Matrixform schreiben[8]:

$$\underline{\underline{A}}\,\underline{x} - \underline{b} = \underline{0} \quad \text{oder} \quad \underline{\underline{A}}\,\underline{x} = \underline{b} \ ,$$

wobei folgende Definitionen verwendet werden:

$$\underline{\underline{A}} = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix}, \quad \underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} \quad \text{und} \quad \underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}$$

Dabei ist $\underline{\underline{A}}$ die (*M*x*N*)-Matrix der Koeffizienten, $\underline{x}$ der Lösungsvektor (Vektor der Unbekannten) sowie $\underline{b}$ der Vektor der Koeffizienten 0. Ordnung. Beispiel:

[8] Matrizen werden doppelt unterstrichen notiert.

$$\begin{aligned} 2x_1 + x_2 - 4 &= 0 \\ 4x_1 - 2x_2 - 2 &= 0 \end{aligned} \quad \textit{oder} \quad \begin{pmatrix} 2 & 1 \\ 4 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

Bezüglich der Dimension des Problems (*M*x*N*) werden drei Fälle unterschieden:

1. ***M>N***: Das System ist **überbestimmt**, d.h. es gibt mehr Gleichungen als Unbekannte. In diesem Fall gibt es, wenn die Gleichungen linear unabhängig sind, nur eine **Näherungslösung** (siehe folgendes Kapitel).
2. ***M<N***: Das System ist **unterbestimmt**, d.h. es gibt weniger Gleichungen als Unbekannte. Die Lösungen $\underline{x}$ liegen in einem **Lösungsraum** der Dimension *N* -*M*. Dieser Fall soll hier nicht behandelt werden (siehe Textbücher der linearen Algebra).
3. ***M=N***: Das System hat eine **eindeutige Lösung**, wenn alle Gleichungen linear unabhängig sind (Determinante von $\underline{\underline{A}}$ ist ungleich Null). Falls Gleichungen linear abhängig sind, liegt Fall 2 vor.

Im folgenden sollen alle 3 Fälle behandelt werden. Für den Fall 2 kommen das elementare Gauß-, bzw. Gauß-Jordan-Verfahren zur Anwendung, während für die Fälle 1 und 3 das Verfahren der Singulärwert-Zerlegung vorgestellt wird. Letzteres hat sehr breite Anwendungen in der linearen Algebra..

6.1.1 Elementaroperationen

Zur Lösung des Gleichungssystems werden in den elementaren Verfahren (Gauß-, bzw. Gauß-Jordan-Verfahren) Äquivalenz-Umformungen angewandt, die das Ergebnis (also den Lösungsvektor $\underline{x}$) nicht ändern. Diese Umformungen heißen **Elementaroperationen**. Folgende Elementaroperationen sind möglich:

	Operation	**Entsprechung**
1	Vertauschen von Zeilen in $\underline{\underline{A}}$ und $\underline{b}$	2 Gleichungen in ihrer Reihenfolge tauschen
2	Linearkombinationen von Zeilen in $\underline{\underline{A}}$ und $\underline{b}$ bilden	Linearkombinationen von Gleichungen bilden
3	Spalten in $\underline{\underline{A}}$ vertauschen	Variablen tauschen **(siehe Anmerkung)**

Anmerkung: Die Operation 3 bedeutet, daß die Reihenfolge der Komponenten im Lösungsvektor $\underline{x}$ vertauscht wird. Die Vertauschungen müssen rückgängig gemacht werden, um die ursprüngliche Lösung zu erhalten (falls z.B. die Komponenten Ort und Geschwindigkeit vertauscht werden, wäre dies fatal). Werden mehrere Operationen des Typs 3 angewandt, müssen diese in diese in umgekehrter Reihenfolge rückgängig gemacht werden. Dies erfordert entsprechendes „Bookkeeping“, d.h. Merken der durchgeführten Operationen vom Typ 3.

Beispielgleichung:

$$\begin{aligned} 2x_1 + x_2 &= 4 \\ 4x_1 - 2x_2 &= 2 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} 4x_1 - 2x_2 &= 2 \\ 2x_1 + x_2 &= 4 \end{aligned} \quad \text{1. Operation}$$

$$\Leftrightarrow \quad \begin{aligned} 2x_1 + x_2 &= 4 \\ 6x_1 - x_2 &= 6 \end{aligned} \quad \text{2. Operation (2. Zeile gleich 1. Zeile plus 2. Zeile)}$$

$$\Leftrightarrow \quad \begin{aligned} x_1 + 2x_2 &= 4 \\ -2x_1 + 4x_2 &= 2 \end{aligned} \quad \text{3. Operation } (x_1, x_2 \text{ im Lösungsvektor vertauschen})$$

Alle Gleichungssysteme auf der rechten Seite haben dieselbe Lösung (unter Berücksichtigung der Vertauschung der Lösungskomponenten im Fall 3).

6.1.2 Gauß-Elimination mit Rücksubstitution

Dieses Verfahren ist für den Fall 1 (*M=N*) geeignet. Die Idee dabei ist es, das System durch Anwendung von Elementaroperationen zunächst so umzuformen, daß die Matrix $\underline{\underline{A}}$ eine obere Dreiecksgestalt bekommt:

$$\begin{pmatrix} a'_{11} & \cdots & \cdots & a'_{1N} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a'_{NN} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_N \end{pmatrix}$$

Anmerkung: Die Komponenten von $\underline{A}$ und $\underline{b}$ sind hier als gestrichene Größen geschrieben, da die Zahlenwerte sich durch die Elementaroperationen verändert haben.

Ist das System in der angegebenen Form, lassen sich die Komponenten der Lösung iterativ durch **Rücksubstitution** bestimmen:

$$x_N = b'_N / a'_{NN} \quad \text{, direkt aus der letzten Zeile zu berechnen}$$

$$x_{N-1} = \frac{1}{a'_{N-1N-1}} \left(b'_{N-1} - a'_{N-1,N} x_N \right) \quad \text{, aus der vorletzten Zeile zu berechnen}$$

$$x_i = \frac{1}{a'_{ii}} \left(b'_i - \sum_{j=i+1}^{N} a'_{ij} x_j \right) \;,\; i = N-1, N-2, \ldots, 1 \quad \text{, allg. Form}$$

Aus der ersten Gleichung sowie durch Iteration der dritten Gleichung lassen sich alle Komponenten der Lösung berechnen. Ausgehend von x_N wird x_{N-1} berechnet, usw.

6.1.2.1 Erzeugung der Dreiecksgestalt und Pivoting

Die zur Anwendung der Rücksubstitutionsformel notwendige Dreiecksgestalt läßt sich folgendermaßen erzeugen:

1. Gehe vom Diagonalelement a_{11} aus.
2. Ziehe von jeder Zeile j unterhalb des Diagonalelements das a_{1j}/a_{11}-fache der ersten Zeile ab. Dann entstehen unterhalb des Diagonalelements in der 1. Spalte Nullen.
3. Gehe zur nächsten Spalte und fahre bei Punkt 1 fort, jedoch nun ausgehend von dem der Spalte zugehörigen Diagonalelement.

Problem bei diesem einfachen Verfahren sind die beim Teilen durch das jeweilige Diagonalelement auftretenden numerischen Fehler. Das Element durch das geteilt wird nennt man **Pivot** und das (empirisch begründete) Verfahren zur Minimierung des Fehlers heißt **Pivoting**. Beim Pivoting werden die Zeilen unterhalb und die Spalten rechts vom aktuell betrachteten Diagonalelement (einschließlich der Zeile und Spalte mit dem aktuellen Pivot) so getauscht, daß das vom Betrag her größte Element zum Pivot wird (Elementaroperationen 1 und 3). Beim Teil-Pivoting werden nur Zeilen getauscht, hier spart man sich das notwendige „Bookkeeping“ bei der Vertauschung von Spalten. Empirisch findet man, daß Teil-Pivoting i.a. ausreicht.

Anmerkung: Kommt der Betrag des Pivots (nach Pivoting) in den Bereich der Maschinengenauigkeit, so ist das Verfahren numerisch instabil. Wird das Pivot identisch Null, so kann dies bedeuten, daß das Gleichungssystem tatsächlich singulär ist (Determinante = 0), oder daß der Wert 0 zufällig durch numerische Ungenauigkeiten zustande gekommen ist. Man kann also das Problem anhand der Größe der Pivots nicht eindeutig analysieren. Tritt der Fall einen kleinen Pivots auf, so kann nur festgestellt werden, daß die Lösung instabil **oder** das System möglicherweise sogar singulär ist. Die Lösung sollte in diesem Fall durch Einsetzen überprüft werden.

6.1.2.2 Iterative Verbesserung der Lösung

Das beschriebene Gauss-Verfahren beinhaltet viele Rechenschritte, so daß sich die Rechenfehler aufakkumulieren können. Die numerisch gefundene Lösung wird daher i.a. um einen Betrag größer als die Maschinengenauigkeit von der wahren Lösung abweichen. Im folgenden wird nun ein Verfahren beschrieben, das die Verbesserung der Lösung bis zur Größenordnung der Maschinengenauigkeit erlaubt. Die mit dem Gauß-Verfahren gefundene Näherungslösung für das Gleichungssystem

$$\underline{\underline{A}}\,\underline{x} = \underline{b}$$

sei $\underline{x}'$ und $\underline{x}$ sei die (unbekannte) wahre Lösung. Durch Einsetzen kann die Lösung überprüft werden:

$$\underline{\underline{A}}\,\underline{x}' = \underline{b}'$$

Wir definieren die numerischen Fehler als:

$$\delta\underline{x} = \underline{x}' - \underline{x}$$
$$\delta\underline{b} = \underline{b}' - \underline{b}$$

Dabei ist $\delta\underline{b}$ bekannt (und wie oben bemerkt i.a. größer als die Maschinengenauigkeit), während $\delta\underline{x}$ unbekannt ist, da die wahre Lösung $\underline{x}$ unbekannt ist. $\delta\underline{x}$ kann jedoch ausgehend von der Näherungslösung und unter Ausnutzung der Linearität ausgerechnet werden:

$$\begin{aligned} & \underline{\underline{A}}\,\underline{x}' = \underline{b}' \\ \Leftrightarrow \quad & \underline{\underline{A}}\,(\underline{x} + \delta\underline{x}) = (\underline{b} + \delta\underline{b}) \\ \Leftrightarrow \quad & \underline{\underline{A}}\,\underline{x} + A\,\delta\underline{x} = \underline{b} + \delta\underline{b} \\ \Leftrightarrow \quad & \underline{\underline{A}}\,\delta\underline{x} = \delta\underline{b} \end{aligned}$$

In der dritten Zeile wird die Linearität des Systems augenutzt. In der vierten Zeile wird ausgenutzt, daß $\underline{x}$ die wahre Lösung ist und damit die jeweils ersten Summanden auf beiden Seiten wegfallen. Löst man das in der vierten Zeile angegebene Gleichungssystem (wieder mit Gauß), so erhält man eine Näherungslösung $\delta\underline{x}'$ für den wahren Fehler $\delta\underline{x}$. Damit läßt sich eine verbesserte Schätzung der Lösung angeben:

$$\underline{x}'' = \underline{x}' - \delta\underline{x}'$$

Dieses Verfahren kann iteriert werden bis der Fehler $\delta\underline{b}$ in die Größenordnung der Maschinengenauigkeit kommt. Dazu setzt man die verbesserte Lösung $\underline{x}''$ wieder in das Gleichungssystem ein, berechnet ein neues $\delta\underline{b}$, bestimmt ein neues $\delta\underline{x}$, usw. Nachteil des angegebenen Verfahrens ist es, das man das Gleichungssystem mehrfach lösen muß. Dieser Rechenaufwand ist nur gerechtfertigt, wenn die Lösung besonders genau sein muß.

6.1.3 Gauß-Jordan-Verfahren

Gauß-Elimination mit Rücksubstitution ist das in Bezug auf den Rechenaufwand günstigste Verfahren zur Berechnung der Lösung im Fall 1 ($M=N$). Soll jedoch zusätzlich die zu $\underline{\underline{A}}$ inverse Matrix $\underline{\underline{A}}^{-1}$ berechnet werden, so ist eine Erweiterung des Verfahrens sinnvoll, die als Gauß-Jordan-Verfahren bekannt ist. Dazu wird folgende Matrixgleichung definiert:

$$\underline{\underline{A}}\left[\underline{x} \cup \underline{\underline{Y}}\right] = \left[\underline{b} \cup \underline{\underline{1}}\right]$$

Dabei ist:

$$\left[\underline{b} \cup \underline{\underline{1}}\right] = \begin{pmatrix} b_1 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ b_N & 0 & \cdots & 0 & 1 \end{pmatrix} \text{ und } \left[\underline{x} \cup \underline{\underline{Y}}\right] = \begin{pmatrix} x_1 & y_{11} & \cdots & y_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ x_N & y_{N1} & \cdots & y_{NN} \end{pmatrix}$$

Man kann sich diese Formulierung so vorstellen, daß N+1 Gleichungssysteme gleichzeitig aufgestellt werden, wobei jede Spalte in [$\underline{b}\cup\underline{1}$] und [$\underline{x}\cup\underline{Y}$] ein eigenes, unabhängiges Gleichungssystem darstellt. Diese Systeme können gleichzeitig gelöst werden, indem man die Matrix $\underline{\underline{A}}$ durch Elementaroperationen zur Einheitsmatrix umformt (**Anmerkung**: Jede Elementaroperation wirkt dann auf jede Spalte von [$\underline{b}\cup\underline{1}$]) getrennt, wie oben für den Fall einer Spalte beschrieben. Die rechte Seite der Gleichung stellt dann das gesuchte $\underline{x}$ mit

$$\underline{\underline{A}}\,\underline{x} = \underline{b}$$

und das gesuchte $\underline{\underline{Y}}$ mit

$$\underline{\underline{A}}\,\underline{\underline{Y}} = \underline{\underline{1}}$$

dar. $\underline{\underline{Y}}$ ist also die Inverse von $\underline{\underline{A}}$.

6.1.3.1 Erzeugung der Einheitsmatrix

Die Einheitsmatrix läßt sich durch folgendes Schema erzeugen, das ganz ähnlich dem des Gauß-Verfahrens ist:

1. Gehe spaltenweise durch die Matrix $\underline{\underline{A}}$, beginnend mit Spalte 1. Sei i die aktuelle Spalte.
2. Nehme das Diagonalelement a_{ii} als Pivot und führe Pivoting durch.
3. Teile die Zeile i durch das Pivot. Damit hat das Diagonalelement den Wert 1.
4. Bringe die Werte aller anderen Zeilen in Spalte i auf Null, indem das a_{ji}–fache der i-ten Zeile von der j-ten Zeile abgezogen wird (für alle Zeilen j außer der i-ten Zeile selbst).
5. Gehe zur nächsten Spalte i+1 und fahre bei 2. fort.

Für die numerische Stabilität dieses Verfahrens gilt das oben Gesagte.

6.1.4 Singulärwertzerlegung

Wie oben angedeutet, eignet sich das Gauß- bzw. Gauß-Jordan-Verfahren aufgrund des wiederholten Teilens durch das jeweilige Pivot nicht gut für Gleichungssysteme, die annähernd singulär sind. Weiterhin ist es nicht für über- und unterbestimmte Systeme geeignet. Im folgenden wird nun ein Verfahren vorgestellt, daß numerisch sehr stabil und für alle Klassen von Gleichungssystemen geeignet ist. Weiterhin ermöglicht es eine eindeutige Diagnose der numerischen Stabilität eines Systems. Das Verfahren wird in der Literatur meist als **SVD** (singular value decomposition) bezeichnet. SVD beruht auf einem Theorem der linearen Algebra, dessen Beweis hier nicht angegeben werden soll (siehe dazu Textbücher der linearen Algebra). Es besagt, daß sich jede MxN-Matrix $\underline{\underline{A}}$ mit $M \geq N$ folgendermaßen zerlegen läßt:

$$\underline{\underline{A}} = \underline{\underline{U}} \cdot \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_N \end{pmatrix} \cdot \underline{\underline{V}}^T = \underline{\underline{V}} \cdot diag\left(w_j\right) \cdot \underline{\underline{U}}^T$$

mit :

$\underline{\underline{U}}$: MxN - Matrix

$\underline{\underline{V}}$: NxN - Matrix

w_j : Singulärwerte

Dabei sind $\underline{\underline{U}}$ und $\underline{\underline{V}}$ spaltenweise orthonormal, d.h.:

$$\underline{\underline{U}}^T \cdot \underline{\underline{U}} = \underline{\underline{V}}^T \cdot \underline{\underline{V}} = \underline{\underline{1}}$$

$$\Leftrightarrow \quad \sum_{i=1}^{M} U_{ik} U_{in} = \delta_{kn} \qquad \begin{matrix} 1 \le k \le N \\ 1 \le n \le N \end{matrix}$$

$$\sum_{j=1}^{M} V_{jk} V_{jn} = \delta_{kn} \qquad \begin{matrix} 1 \le k \le N \\ 1 \le n \le N \end{matrix}$$

Es soll hier nicht im Einzelnen behandelt werden, wie die Zerlegung numerisch berechnet wird. Wichtig zu wissen ist, daß zu diesem Zweck stabile und effiziente Algorithmen zur stehen. Im folgenden wird nun betrachtet, wie SVD zur Berechnung der Lösung linearer Gleichungssysteme in den verschiedenen Fällen genutzt werden kann.

Anmerkung: In Matlab steht der Befehl **SVD** zur Verfügung. Achtung: Es gibt mehrere Varianten der Singulärwertzerlegung. Die hier vorgestellte Variante wird von dem SVD-Befehl berechnet, wenn man 0 als zusätzlichen Parameter übergibt (z.B.: [U,S,V]=svd(A,0)).

Anmerkung: Für den Fall *M<N* (unterbestimmtes System) ist SVD auch möglich, jedoch sind dann *N-M* der Singulärwerte identisch Null und die spaltenweise Orthonormalität von $\underline{\underline{U}}$ und $\underline{\underline{V}}$ gilt nicht für diejenigen Spalten, die in der Matrix der Singulärwerte die Nullen enthalten.

6.1.4.1 SVD einer quadratischen Matrix (Fall M=N)

Im Fall *M=N* sind alle beteiligten Matrizen quadratisch. Für $\underline{\underline{U}}$ und $\underline{\underline{V}}$ gilt dann auch die zeilenweise Orthonormalität, d.h. es gilt:

$$\underline{\underline{U}} \cdot \underline{\underline{U}}^T = \underline{\underline{V}} \cdot \underline{\underline{V}}^T = \underline{\underline{1}}.$$

$\underline{\underline{U}}$ und $\underline{\underline{V}}$ sind also orthogonal und

$$\underline{\underline{U}}^T = \underline{\underline{U}}^{-1}$$
$$\underline{\underline{V}}^T = \underline{\underline{V}}^{-1}$$

Die Inverse der Matrix $\underline{\underline{A}}$ ergibt sich dann als

$$\underline{\underline{A}}^{-1} = \underline{\underline{V}} \cdot \begin{pmatrix} 1/w_1 & & \\ & \ddots & \\ & & 1/w_N \end{pmatrix} \cdot U^T = \underline{\underline{V}} \cdot diag\left(1/w_j\right) \cdot U^T$$

und die Lösung eines Gleichungssystems als:

$$\underline{\underline{A}}\underline{x} = \underline{b} \Rightarrow \underline{x} = \underline{\underline{A}}^{-1}\underline{b} = \underline{\underline{V}} \cdot diag\left(1/w_j\right) \cdot \underline{\underline{U}}^T \cdot \underline{b}.$$

Dieses Verfahren geht nur schief, wenn einer oder mehrere der Singulärwerte Null werden oder vom Betrag her in der Größenordnung der Maschinengenauigkeit liegen. Je mehr Singulärwerte betragsmäßig klein werden, desto instabiler wird die Lösung. SVD gibt also zunächst durch die Analyse der Singulärwerte eine klare Diagnose der Situation. Zwei Fälle sind zu unterscheiden:

(i) Die Matrix ist **ill-conditioned**, d.h. $\dfrac{\min\limits_{j \in 1,N}\left(w_j\right)}{\max\limits_{j \in 1,N}\left(w_j\right)} < \varepsilon_m$

(ii) Die Matrix ist **singulär**, d.h. $\exists j : w_j = 0 \quad , j \in 1, N$

6.1.4.1.1 Fall (i): „ill-conditioned“

Im Fall (i) gibt es formal eine eindeutige Lösung, jedoch ist das Gauß- bzw. Gauß-Jordan-Verfahren numerisch instabil (s.o.). SVD liefert dagegen auf folgende Weise eine i.a. robustere und genauere Lösung (ohne Beweis):

$$\underline{x} = \underline{\underline{V}} \cdot diag\left(1/w_j'\right) \cdot \underline{\underline{U}}^T \cdot \underline{b} \quad \text{, mit}$$

$$1/w_j' = \begin{cases} 0 & |w_j| < \varepsilon_m \\ 1/w_j & sonst \end{cases}$$

,d.h. man eliminiert einfach die Singulärwerte. deren Wert nicht genau genug bekannt ist. Die so gefundene Lösung $\underline{x}$ ist dann im Least-Squares-Sinne i.a. besser als die mit anderen Methoden (Gauß-/Gauß-Jordan) gefundene Lösung, d.h.

$$r^2 = \left|\underline{\underline{A}}\underline{x} - \underline{b}\right|^2$$

wird kleiner. Der Beweis dafür soll hier nicht angegeben werden. Eine anschauliche Erklärung dafür ist die, daß man durch das Null-Setzen der $1/w_j$ diejenigen Gleichungen und Kombinationen davon herauswirft, die mehr oder weniger nur Rundungsfehler enthalten.

6.1.4.1.2 Fall (ii): singulär

Der Fall (ii) ist mathematisch etwas komplexer. Die inverse Matrix $\underline{\underline{A}}^{-1}$ existiert in diesem Fall nicht, jedoch kann das Gleichungssystem

$$\underline{\underline{A}}\underline{x} = \underline{b}$$

als Lösung einen Lösungsraum haben. Die Gleichung

$$\underline{x} = \underline{\underline{V}} \cdot diag\left(1/w_j'\right) \cdot \underline{\underline{U}}^T \cdot \underline{b} \quad \text{, mit}$$

$$1/w_j' = \begin{cases} 0 & w_j = 0 \\ 1/w_j & sonst \end{cases}$$

liefert eine „möglichst optimale“ spezielle Lösung (ohne Beweis). „Möglichst optimal“ heißt hier, daß wieder

$$r^2 = \left|\underline{\underline{A}}\underline{x} - \underline{b}\right|^2$$

minimal wird und das $\underline{x}$ die minimale Länge aller Vektoren des Lösungsraums hat. Der Lösungsraum L ergibt sich als Summe des sog. Nullraums und der speziellen Lösung:

$$L = \left\{\underline{x} + \underline{x}' : \underline{x}' \in N\right\}$$

Der **Nullraum** N ist dabei folgendermaßen definiert:

$$N = \left\{\underline{x} : \underline{\underline{A}}\underline{x} = \underline{0}\right\}$$

Der Nullraum enthält also alle Vektoren, die auf Null abgebildet werden.

Anmerkung: Diejenigen Spalten von $\underline{\underline{V}}$, die in der zugehörigen Matrix der Singulärwerte ($diag(w_j)$) Nullen enthalten, spannen den Nullraum auf (ohne Beweis).

Anmerkung: Diejenigen Spalten von $\underline{\underline{U}}$, die in der zugehörigen Matrix der Singulärwerte ($diag(w_j)$) Nullen enthalten, spannen den **Bereich** auf (ohne Beweis). Der Bereich ist die Menge aller Vektoren $\underline{b}$, für die das

Gleichungssystem $\underline{\underline{A}}\underline{x}=\underline{b}$ mindestens eine exakte Lösung hat. Liegt $\underline{b}$ im Bereich von $\underline{\underline{A}}$, so ist die o.a. spezielle Lösung exakt. Liegt b außerhalb des Bereichs, gilt die Least-Squares-Bedingung.

Anmerkung: Der Fall (ii) entspricht mathematisch dem des unterbestimmten Systems ($M<N$), kann also genauso gelöst werden.

6.1.4.2 SVD für überbestimmte Systeme (Fall M>N)

Im dem Fall, daß es mehr Gleichungen als Unbekannte gibt, gibt es i.a. keine eindeutige Lösung des Gleichungssystems. SVD findet hier jedoch wieder die „optimale“ Näherungslösung (ohne Beweis):

$$\underline{x} = \underline{\underline{V}} \cdot diag\left(1/w_j\right) \cdot \underline{\underline{U}}^T \cdot \underline{b}$$

$$\text{mit}: \; r^2 = \left|\underline{\underline{A}}\underline{x} - \underline{b}\right|^2 \quad \text{ist minimal}$$

Anmerkung: Auch hier können wieder Singulärwerte mit einem Betrag kleiner als die Maschinengenauigkeit exisitieren. Ersetze diese in der Matrix der inversen Singulärwerte wieder durch Nullen.

6.2 nichtlineare Gleichungssysteme

Läßt sich eine Gleichungssystem nicht linearisieren, lassen sich die oben beschriebenen eleganten Methoden der linearen Algebra nicht anwenden. Es bleiben nur iterative Lösungsverfahren, von denen im folgenden das bekannte Newton’sche Gradientenverfahren als einfaches Beispiel beschrieben werden soll.

6.2.1 Newton-Verfahren

Das Newton’sche Gradientenverfahren sucht die Nullstelle eines Gleichungssystems, indem es, ausgehend von der aktuellen Schätzung der Nullstelle, diese immer in Richtung des Gradienten verändert. Nehmen wir an, $\underline{z}^*$ sei die Lösung des Gleichungssystems

$$\underline{H}(\underline{z}) = \underline{0} \quad \text{mit} \quad \underline{H} = \left(h_1(\underline{z}) \quad \ldots \quad h_N(\underline{z})\right) \quad \text{und} \quad \underline{z} = \left(z_1 \quad \ldots \quad z_N\right)$$

(**Anmerkung**: $\underline{H}$ und $\underline{z}$ sind hier Zeilenvektoren). Nehmen wir weiter an, $\underline{z}_1$ sei eine Schätzung der Lösung, die um $\delta\underline{z}$ von der wahren Lösung abweicht, d.h.

$$\delta\underline{z} = \underline{z}_1 - \underline{z}^* .$$

Eine Taylorentwicklung erster Ordnung liefert mit dieser Definition und der Annahme, daß $\underline{z}^*$ die Nullstelle sei:

$$\underline{0} \equiv \underline{H}\left(\underline{z}^*\right) = \underline{H}\left(\underline{z}_1 - \delta\underline{z}\right) = \underline{H}\left(\underline{z}_1\right) - \delta\underline{z} \cdot \underline{\underline{D}}\left(\underline{z}_1\right) + O\left(\delta\underline{z}^2\right)$$

mit der Jacobi-Matrix $\underline{\underline{D}}$:

$$\underline{\underline{D}}(\underline{z}) = \left\{D_{ij}\right\} \quad , \quad D_{ij} = \frac{\partial h_j(\underline{z})}{\partial z_i}$$

Unter Vernachlässigung der Terme höherer Ordnung ergibt sich also:

$$\underline{0} = \underline{H}\left(\underline{z}_1\right) - \delta\underline{z} \cdot \underline{\underline{D}}\left(\underline{z}_1\right) + O\left(\delta\underline{z}^2\right)$$

$$\Rightarrow \quad \underline{H}\left(\underline{z}_1\right) \approx \delta\underline{z} \cdot \underline{\underline{D}}\left(\underline{z}_1\right)$$

$$\Rightarrow \quad \delta\underline{z} \approx \underline{H}\left(\underline{z}_1\right) \cdot \underline{\underline{D}}^{-1}\left(\underline{z}_1\right)$$

$$\Rightarrow \quad \underline{z}_2 = \underline{z}_1 - \delta\underline{z} = \underline{z}_1 - \underline{H}\left(\underline{z}_1\right) \cdot \underline{\underline{D}}^{-1}\left(\underline{z}_1\right)$$

$\underline{z}_2$ ist eine neue Schätzung von $\underline{z}^*$, die wieder iterativ in die letzte Gleichung eingesetzt werden kann.

6.3 Aufgaben

1. Gegeben sei die Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 3 & 4 \\ 8 & 9 & 15 \end{pmatrix}$ und der Vektor $b = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$. Löse das lineare Gleichungssystem $Ax = b$ und berechne die inverse Matrix A^{-1}. Verwende dazu das zur Verfügung gestellte Matlab-Skript gaussj.m, das das Gauss-Jordan Eliminationsverfahren durchführt. Vollziehe das Programm nach und vergleiche das Ergebnis mit den eingebauten Matlab-Funktionen inv (Berechnung der Inversen) und \ (Berechnung der Lösung des linearen Gleichungssystems; für Hinweise zur Benutzung bitte „help slash“ eingeben).

2. Gegeben seine die Matrizen

$$A = \begin{pmatrix} 1 & -5 & 3 \\ 2 & -10 & 0 \\ 2 & 8 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 10 & -8 & -9 \\ 6 & 10 & 2 \\ 1 & 7 & 3 \end{pmatrix} \text{ und } \quad C = \begin{pmatrix} 4 & -6 & -2 & 5 \\ -8 & 7 & 5 & 7 \\ -1 & 4 & 0 & -10 \\ -6 & -1 & -4 & 0 \end{pmatrix}$$

   Finde die inversen Matrizen mit Hilfe des Skripts gaussj.m. Warum geht dies in allen Fällen schief? In welchen Fällen müßte es eigentlich eine Lösung geben (berechne dazu die Determinanten) und wie könnte man in diesen Fällen durch Umstellung der Matrizen dennoch mit diesem Skript zu einem Ergebnis kommen (Hinweis: „Pivoting“)?

3. Gegeben sei das Lorenz-Modell

$$\frac{dx}{dt} = \sigma\ (y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz$$

   a) Zeige, daß es die folgenden stationären Zustände besitzt:

   $x^* = y^* = z^* = 0$ und $x^* = y^* = \pm\sqrt{b(r-1)}, z^* = r - 1, wenn \sigma \neq 0$

   b) Suche die stationären Zustände für r=28, σ=10 und b=8/3 mit Hilfe der Newton-Methode. Führe die Berechnung für mehrere verschiedene Anfangsschätzungen der stat. Zustände aus. Wie hängt die Lösung davon ab?

   Hinweis: Verwende die zur Verfügung gestellte Routine newtn, die das Newton-Verfahren implementiert. Schreibe dann eine Routine [H,D]=fnewt(z,p), die aus dem aktuellen Zustand z=(x,y,z) und dem Parametersatz p=(r,σ,b) die Funktion H sowie die Jacobimatrix D berechnet.

7 Modellierung von Meßdaten

Bei der Überprüfung physikalischer Theorien und Modelle spielt die Modellierung von Meßdaten ein wichtige Rolle. Es sollen meist einige Parameter der physikalischen Modelle an Meßdaten angepaßt und dann untersucht werden, inwieweit das angepaßte Modell die Daten beschreibt. Dieser Prozeß wird im folgenden anhand einfacher Beispiele erläutert. Insbesondere sollen dabei die Techniken der Anpassung von Parametern sowie Techniken zur Bewertung der Güte einer Anpassung beschrieben werden.

7.1 Allgemeine mathematische Formulierung des Anpassungsproblems

Nehmen wir an, es liegen N Meßwerte als Wertepaare (x_i,y_i) vor und das anzupassende Modell sagt einen funktionellen Zusammenhang zwischen der abhängigen Größe y und der unabhängigen Größe x voraus:

$$y = f(x, \underline{a}) \quad \text{mit} \quad \underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_M \end{pmatrix},$$

wobei $\underline{a}$ ein Vektor bestehend aus M Modellparametern ist[9]. Natürlich sollte es deutlich mehr Meßwerte als Modellparameter geben, also $N >> M$.

Als Beispiel sei die Aktivität eines radioaktiven Isotops angegeben:

$$y = f(x, \underline{a}) = a_1 \cdot e^{-a_2 x} \quad \text{mit} \quad \underline{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

Dabei entspricht y der Aktivität, x ist die Zeit und die Parameter a_1 und a_2 beschreiben die Skalierung der Aktivität und die Zerfallsrate. Das Modell beschreibt eine exponentielle Abnahme der Aktivität.

Zur Vereinfachung der später verwendeten Ausdrücke schreiben wir die Meßwerte auch in Vektoren:

$$\underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \quad , \quad \underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

Das **allgemeine Vorgehen** zur Anpassung des Modells ist nun folgendes:

1. Definiere eine „Kostenfunktion“ („merit function“) K, die die Abweichung des Modells von den Daten beschreibt. Je größer die Abweichung des Modells von den Daten ist, desto größer ist die Kostenfunktion. Dieser monotone Zusammenhang gilt z.B. für die „Least-Squares“-Kostenfunktion, die als Summe über alle Meßwerte der quadrierten Abweichungen der vom Modell geschätzten Werte der abhängigen Variablen von den tatsächlichen Meßwerten definiert ist, also: $K(\underline{x}, \underline{y}, \underline{a}) = \sum_{i=1}^{N} \left| y_i - f(x_i, \underline{a}) \right|^2$.
2. Minimiere die Kostenfunktion bei gegebenen Meßdaten durch Variation der Parameter $\underline{a}$.

Ergebnis der Anpassung sollte dann sein:

1. Angabe des besten Parametersatzes $\underline{a}_{\text{opt}}$ (bester Fit), der dem minimalen K entspricht.
2. Schätzung des Fehlers der Parameter, d.h. Angabe eines Bereichs in dem die Parameter mit einer gewissen Wahrscheinlichkeit liegen.

[9] Ohne Beschränkung der Allgemeinheit wird das Problem hier für eine unabhängige und eine abhängige Variable formuliert. Falls es mehrere unabhängige Variablen gibt, gehen diese alle als Argument in die Modellfunktion ein. Ebenso könnte das Modell mehrere abhängige Variablen voraussagen, so daß die Modellfunktion vektorwertig wäre.

3. Statistische Aussage über die Güte des Modells („goodness of fit“), d.h. Klärung der Frage, ob das Modell geeignet ist, die Meßdaten ausreichend zu beschreiben.

Die letzten beiden Punkte werden oft vergessen, vielleicht weil sie aufwendig und theoretisch schwer zu fassen sind. Zu einer systematischen Modellanalyse gehören sie jedoch hinzu. Im folgenden werden nun Methoden zur Modellanpassung erläutert. Dabei soll geklärt werden, was jeweils die beste Kostenfunktion ist, wie die Kostenfunktion minimiert wird und wie die Anpassung beurteilt werden soll.

7.2 Aufstellung der Kostenfunktion: Maximum-Likelihood-Methoden

Das Maximum-Likelihood -Konzept (ML-Methode) ermöglicht die Bestimmung der jeweils dem Problem angepaßten Kostenfunktion *K*. Die Idee ist, ausgehend von einem statistischen Modell des Meßprozesses die Wahrscheinlichkeit *W* dafür zu berechnen, daß die gemessenen Daten vom Modell erzeugt werden. Als Kostenfunktion *K* wird dann der negativen Logarithmus dieser Wahrscheinlichkeit verwendet:

$$K(\underline{x}, \underline{y}, \underline{a}) = -\log(W)$$

wobei:

W : Wahrscheinlichkeit, daß die Meßdaten vom Modell erzeugt werden

Als bester Fit $\underline{a}_{\text{opt}}$ wird der Parametersatz angegeben, der *K* minimiert:

$$\underline{a}_{opt} = \underset{\underline{a}}{\operatorname{argmin}}\left(K(\underline{x},\underline{y},\underline{a})\right) \qquad \left(= \underset{\underline{a}}{\operatorname{argmax}}\left(W(\underline{x},\underline{y},\underline{a})\right)\right)$$

Die Minimierung von *K* entspricht aufgrund der streng fallenden Monotonie der –log(*x*)-Funktion gerade der Maximierumg der Wahrscheinlichkeit *W*. Warum heißt dieses Vorgehen nun nicht „Methode der maximalen Wahrscheinlichkeit“, sondern „Maximum-Likelihood-Methode“? Die Antwort ist, daß sich die Angabe der Wahrscheinlichkeit *W* ja auf die Meßdaten bezieht und nicht auf die gesuchten Modellparameter: Wenn das Modell stimmt, gibt es ja nur einen „wahren“ Parametersatz und die Angabe einer Wahrscheinlichkeit des Parametersatzes macht keinen Sinn. Deswegen definiert man den Begriff der **Likelihood** eines Parametersatzes:

> **Die Likelihood eines Parametersatzes $\underline{a}$ entspricht der Wahrscheinlichkeit *W*, daß die gemessenen Daten von dem Modell mit diesem Parametersatz erzeugt werden**

Im folgenden werden einige Beispiele dafür angegeben, wie *W* bzw. *K* aus Annahmen über die Statistik des Meßprozesses berechnet werden können. Insbesondere wird erläutert, welche statistischen Voraussetzungen der Meßprozess erfüllen muß, damit die Least-Squares-Kostenfunktion im Sinne der ML-Konzepts optimal ist. Ausgehend davon werden Erweiterungen der Least-Squares-Methode angegeben, denen leicht veränderte statistische Annahmen über den Meßprozess zugrunde liegen (Chi-Quadrat-Methode und robuste Fit-Methoden).

Anmerkung: Die ML-Methode erfaßt keine systematischen Fehler in den Meßdaten, sondern modelliert nur zufällige Fehler. Systematische Fehler führen zu einer Fehlanpassung der Parameter.

7.2.1 Least-Squares-Methode

Folgende Annahmen liegen der Ableitung der Least-Squares-Kostenfunktion zugrunde:

1. Die Meßfehler aller Meßpunkte sind normalverteilt mit einer festen Varianz σ, d.h. sie schwanken um den „wahren“ Wert, der vom Modell $y = f(x, \underline{a})$ vorhergesagt wird.
2. Alle Meßfehler sind statistisch unabhängig, d.h. der Fehler eines Meßwerts ist unabhängig von dem eines anderen Meßwerts.

In diesem Fall ist die Wahrscheinlichkeit, daß ein Meßwert mit einer bestimmten Abweichung vom vorhergesagten Wert erzeugt wird

$$W_i = \exp\left(-\frac{1}{2}\left(\frac{y_i - f(x_i, \underline{a})}{\sigma}\right)^2\right) \cdot \Delta y \ .$$

Die Gesamtwahrscheinlichkeit, daß alle Meßwerte von dem gegebenen Modell erzeugt werden ist dann aufgrund der statistischen Unabhängigkeit der Meßwerte das Produkt der Einzelwahrscheinlichkeiten

$$W = \prod_i W_i = \prod_i \left[\exp\left(-\frac{1}{2}\left(\frac{y_{i-} f(x_i, \underline{a})}{\sigma}\right)^2\right) \cdot \Delta y\right] .$$

Anmerkung: Der Term Δy ist ein konstanter Term, der aus der von der Gauß-Verteilung gegebenen Wahrscheinlichkeits**dichte** (1.Term der Gleichung) eine Wahrscheinlichkeit macht. Er ist konstant und kann für alle Meßwerte als gleich angenommen werden.

Die Kostenfunktion ergibt sich dann als

$$K(\underline{x}, \underline{y}, \underline{a}) = -\log(W) = \sum_{i=1}^{N} \frac{(y_i - f(x_i, \underline{a}))^2}{2\sigma^2} - N \log(\Delta y)$$

Da die Skalierung der Kostenfunktion beliebig ist (keine Veränderung der Lage des Minimums), können der konstante Faktor $2\sigma^2$ und der konstante 2. Summand $-N \cdot \log(\Delta y)$ weggelassen werden. Damit ergibt sich als **Kostenfunktion des Least-Squares-Verfahrens** und der daraus abgeleitete optimaler Parametersatz wie erwartet:

$$\boxed{\begin{aligned} K(\underline{x}, \underline{y}, \underline{a}) &= \sum_{i=1}^{N} (y_i - f(x_i, \underline{a}))^2 \\ \underline{a}_{opt} &= \underset{\underline{a}}{\operatorname{argmin}}\left(K(\underline{x}, \underline{y}, \underline{a})\right) \end{aligned}}$$

Anmerkung: Aufgrund dieser Ableitung wird klar, welche statistischen Voraussetzungen dem oftmals anschaulich eingeführten Least-Squares-Verfahren zugrunde liegen. Sie sind bei Meßprozessen oft erfüllt, aber nicht immer. Man sollte das Verfahren also nie ungeprüft einsetzen!

7.2.2 Chi-Quadrat-Methode

Eine Generalisierung der Least-Squares-Methode ergibt sich, wenn man unterschiedliche Varianzen für die verschiedenen Meßwerte zuläßt. Diese müssen natürlich vorher bekannt sein, es liegen also Tripel (x_i, y_i, σ_i) für jeden Meßpunkt vor[10]. In diesem Fall kann man in der o.a. Formel für das Least-Squares-Verfahren die Varianz nicht als konstanten Faktor herausziehen. Es ergibt sich somit als zu minimierende Funktion und optimaler Parametersatz

$$\boxed{\begin{aligned} \chi^2(\underline{x}, \underline{y}, \underline{\sigma}, \underline{a}) &= \sum_{i=1}^{N} \left(\frac{y_i - f(x_i, \underline{a})}{\sigma_i}\right)^2 \\ \underline{a}_{opt} &= \underset{\underline{a}}{\operatorname{argmin}}\left(\chi^2(\underline{x}, \underline{y}, \underline{\sigma}, \underline{a})\right) \end{aligned}},$$

wobei $\underline{\sigma}$ ein Vektor ist, der die den Meßwerten entsprechenden Varianzen als Komponenten enthält. Die Kostenfunktion nennt man das **Chi-Quadrat**. Da die Chi-Quadrat-Methode die Least-Squares-Methode als Sonderfall bei konstanter Varianz einschließt, wird im folgenden nur noch das Chi-Quadrat betrachtet.

[10] Zum Beispiel könnte bei der Ablesung eines Meßgeräts der Meßfehler proportional zur abgelesenen Größe y sein.

7.2.3 Robuste Fit-Methoden

Die Annahme, daß die Meßfehler Gauß-verteilt sind, stellt eine recht „harte“ Randbedingung dar. Die Wahrscheinlichkeit, daß z.B. ein Meßwert mehr als 5σ von dem Modellwert abweicht, ist verschwindend gering. Tritt also ein einziger „Ausreißer“ in den Meßwerten auf, so geht das Chi-Quadrat-Verfahren schief. Die Kostenfunktion erhöht sich signifikant, weil die Wahrscheinlichkeit für dieses Ereignis aufgrund der Annahme der Gauß-Verteilung als sehr gering angenommen wird. Das Minimierungsverfahren wird versuchen, die Kosten zu senken, d.h. den Ausreißer mitzumodellieren. Dadurch bekommen dieser ein zu hohes Gewicht in der Kostenfunktion und das Anpassungsverfahren reagiert nicht robust auf Ausreißer. Ist man also nicht sicher, ob die Gauß-Verteilung der Meßfehler zugrunde gelegt werden kann, sollte man überlegen, welche andere Verteilung möglicherweise besser geeignet ist. Hat man diese bestimmt, kann nach dem oben angegebenen Schema die Likelihood und damit die Kostenfunktion bestimmt werden.

Rechnet man z.B. mit Ausreißern der Meßwerte, kann eine Verteilung benutzt werden, die nicht so schnell gegen Null geht wie die Gauß-Verteilung. Dann ist die Wahrscheinlichkeit für große Abweichungen nicht mehr so verschwindend gering. Zu diesem Zweck bietet sich die **Lorentz-Verteilung** an:

$$W_i = \frac{1}{1 + \frac{1}{2}\left(\frac{y_i - f\left(x_i, \underline{a}\right)}{\sigma} \right)^2} \cdot \Delta y$$

Diese ist im Bereich $\pm 2\sigma$ ähnlich der Gauß-Verteilung, fällt jedoch nicht so schnell zu den Seiten hin ab. Als Kostenfunktion ergibt sich für das auf der Lorentz-Verteilung basierende ML-Verfahren

$$\begin{aligned} K\left(\underline{x}, \underline{y}, \underline{a}\right) &= -\log\left(W\right) = -\log\left(\prod_i W_i \right) \\ &\propto \sum_{i=1}^{N} \log\left(1 + \frac{1}{2}\left(\frac{y_i - f\left(x_i, \underline{a}\right)}{\sigma} \right)^2 \right), \end{aligned}$$

wobei wieder der konstante Summand $-N \cdot \log(\Delta y)$ weggelassen wurde. Diese Kostenfunktion ist deutlich weniger anfällig für Ausreißer als das Chi-Quadrat- bzw. Least-Squares-Verfahren (siehe Übungen), jedoch beruht sie auf einer Annahme über die Verteilung der Meßfehler (Lorentz-Verteilung), die nur empirisch begründet ist. Besser ist es in jedem Fall, theoretische Begründungen für die Wahl der Verteilung zu nutzen. Z.B. kann für die Verteilung der Meßungenauigkeit beim radioaktiven Zerfall die Poisson-Verteilung als theoretisch richtige Verteilung angenommen werden (siehe Übungen).

Anmerkung: Wenn möglich sollte man nicht leichtfertig die Annahme einer Gauß-Verteilung verwerfen, denn diese hat den Vorteil, daß man die Aufgaben 2. und 3. der Anpassung (Angabe eines Parameterintervalls, in dem die „wahren“ Parameter mit gewisser Wahrscheinlichkeit liegen und Angabe der Fitgüte, d.h. Eignung des Modells) statistisch „sauberer“ lösen kann (siehe späterer Abschnitt in diesem Kapitel).

7.3 Minimierung von Funktionen

Nachdem die Kostenfunktion festgelegt ist, stellt sich die Frage nach Methoden zur Minimierung derselben. An dieser Stelle sollen 2 Fälle betrachtet werden, mit denen sich ein Großteil der praktisch auftretenden Fälle lösen läßt:

1. Als Kostenfunktion wird das Chi-Quadrat verwendet **und**
   a) das Modell $f\left(x, \underline{a}\right)$ ist linear in den Parametern $\underline{a}$ (nicht unbedingt linear in den Meßwerten!). In diesem Fall kann das Minimierungsproblem direkt ohne langwierigen iterativen Algorithmus gelöst werden (in der Literatur wird dieser Fall i.a. als **„general linear least-squares“**-Methode bezeichnet)
   b) das Modell $f\left(x, \underline{a}\right)$ ist nichtlinear in den Parametern $\underline{a}$. Dann können nur iterative Methoden zur Minimierung der Funktion verwendet werden.

2. Als Kostenfunktion wird nicht das Chi-Quadrat verwendet, d.h. man benutzt andere Verteilungen des Meßfehlers als die Gauß-Verteilung. In diesem Fall bleiben nur iterative Verfahren, unabhängig davon, ob das Modell $f(x,\underline{a})$ linear in den Parametern $\underline{a}$ ist oder nicht.

Im folgenden werden die einzelnen Verfahren erläutert.

7.3.1 „General Least-Squares"-Verfahren

Für dieses Verfahren wird angenommen, daß das Modell linear in den Parametern $\underline{a}$ ist:

$$f(x,\underline{a}) = \sum_{k=1}^{M} a_k f_k(x) \ ,$$

wobei $f_k(x)$ eine **beliebige** Funktion der unabhängigen Variablen *x* ist. Sie kann also auch nichtlinear sein, z.B. eine Potenzfunktion

$$f_k(x) = x^{k-1} \ .$$

Dann ist die Modellfunktion ein Polynom in *x*

$$f(x,\underline{a}) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + \ldots + a_M \cdot x^{M-1} \ .$$

Das Chi-Quadrat ist mit der allgemeinen Formulierung einer linearen Funktion in den Parametern:

$$\chi^2(\underline{x},\underline{y},\underline{\sigma},\underline{a}) = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{k=1}^{M} a_k f_k(x_i)}{\sigma_i} \right)^2 \ .$$

Mit den Definitionen

$$\text{"Designmatrix"}\ \underline{\underline{A}} = \{A_{ij}\} \text{mit} \quad A_{ij} = \frac{f_j(x_i)}{\sigma_i} \quad (N \times M) - \text{Matrix}$$

$$\text{normierter Meßwertevektor} \quad \underline{b} = \begin{pmatrix} y_1/\sigma_1 \\ \vdots \\ y_N/\sigma_N \end{pmatrix}$$

$$\text{Parametervektor} \quad \underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_M \end{pmatrix}$$

kann das Chi-Quadrat folgendermaßen umgeschrieben werden:

$$\chi^2(\underline{x},\underline{y},\underline{\sigma},\underline{a}) = \left| \underline{\underline{A}} \cdot \underline{a} - \underline{b} \right|^2 \ .$$

Das Chi-Quadrat kann also in diesem Fall so umgeformt werden, daß es dem quadratischen Abstand der Vektoren $\underline{\underline{A}}\,\underline{a}$ und $\underline{b}$ entspricht. Der optimale Parametervektor

$$\underline{a}_{opt} = \underset{\underline{a}}{\text{argmin}}\left(\chi^2(\underline{x},\underline{y},\underline{\sigma},\underline{a})\right) = \underset{\underline{a}}{\text{argmin}}\left(\left| \underline{\underline{A}} \cdot \underline{a} - \underline{b} \right|^2\right)$$

entspricht daher genau der Least-Squares-Lösung des Gleichunggsystems $\underline{\underline{A}}\,\underline{a} = \underline{b}$, da diese ja gerade den quadratischen Abstand zwischen $\underline{\underline{A}}\,\underline{a}$ und $\underline{b}$ minimiert. Diese Lösung ist mittels SVD bekannt:

$$\boxed{\underline{a}_{opt} = \underline{\underline{V}} \cdot diag\left(1/w_j\right) \cdot \underline{\underline{U}}^T \cdot \underline{b}},$$

wobei $\underline{\underline{A}} = \underline{\underline{U}} \cdot diag\left(w_j\right) \cdot \underline{\underline{V}}^T$ die SVD der Matrix $\underline{\underline{A}}$ ist.

Mit Hilfe der SVD ist es also möglich, daß Minimierungsproblem in einem Schritt ohne Iteration zu lösen. Dadurch ist das Verfahren besonders effizient und man ist sicher, daß absolute Minimum gefunden zu haben. Weiterhin kann man etwaige numerische Probleme durch „Behandlung" der Singulärwerte lösen (siehe Kapitel über Gleichungssysteme : Nullsetzen des Kehrwertes, falls der Singulärwert zu klein ist).

Anmerkung: In den Lehrbüchern findet man häufig die sog. **Normalgleichungen** als Lösung des „General Least-Squares"-Problems. Diese ergeben sich durch Nullsetzen der Ableitungen des Chi-Quadrats nach den einzelnen Parametern:

$$\begin{aligned}
& 0 \overset{!}{=} \frac{\partial \chi^2}{\partial a_k} && \forall k = 1, \ldots, M \\
\Rightarrow \quad & 0 \overset{!}{=} \sum_{i=1}^{M} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{N} a_j f_j(x_i) \right] f_k(x_i) && \forall k = 1, \ldots, M \\
\Leftrightarrow \quad & \left(\underline{\underline{A}}^T \underline{\underline{A}}\right) \cdot \underline{a} \overset{!}{=} \underline{\underline{A}}^T \cdot \underline{b}
\end{aligned}$$

Die Gleichung in der 3. Zeile beschreibt die in der 2. Zeile angegebenen *M* Normalgleichungen in Matrixform. Dabei wurden die o.a. Definitionen der Designmatrix, des Parametervektors und des normierten Meßwertevektors verwendet. Der optimale Parametervektor ergibt sich hier durch Matrixinversion:

$$\underline{a}_{opt} = \left(\underline{\underline{A}}^T \underline{\underline{A}}\right)^{-1} \cdot \underline{\underline{A}}^T \cdot \underline{b} \ .$$

Durch das Nullsetzen wird ein Extremum der Chi-Quadrat-Funktion gefunden. Es muß überprüft werden, ob es sich tatsächlich um ein Minimum handelt, was einen zusätzlichen Rechenschritt bedeutet. Insgesamt erscheint mir die Lösung über SVD geeigneter als die Lösung über die Normalgleichungen, da sie direkt ein Minimum liefert und numerisch sehr robust ist.

7.3.2 Methoden für nichtlineare Anpassung: Das „Simplex"-Verfahren

Ist das Modell nichtlinear in den Parametern und/oder die Kostenfunktion nicht das Chi-Quadrat (Fälle 1.b. und 2.), bleiben nur iterative Methoden zum Auffinden des Minimums im „Gebirge" der Kostenfunktion. Dazu gibt es mehrere Methoden, die ausgehend von einem Startwert der Parameter nach dem Minimum suchen. Das einfachste Verfahren wäre wieder das Newtonsche Gradientenverfahren, das jedoch den Nachteil hat, das die partiellen Ableitungen der Modellfunktion *f* berechnet werden müssen (was nicht immer möglich ist) und daß das Verfahren je nach Startwerten sicher im nächstliegenden Minimum landet, was möglicher weise ein Neben- und nicht das absolute Minimum ist. Von den Methoden, die weder eine partielle Ableitung benötigen, noch zwangsläufig im nächstliegenden Nebenminimum landen, soll hier die **Simplex-Methode** vorgestellt werden.

Anmerkung: Das Simplex-Verfahren ist in Matlab in der Funktion **FMINS** implementiert.

Anmerkung: Das Simplex-Verfahren und verwandte Verfahren können mit einer gewissen Wahrscheinlichkeit über das nächstliegende Nebenminimum „hinüberspringen". Für keine dieser iterativen Methode kann jedoch garantiert werden, daß sie das absolute Minimum der Kostenfunktion finden!

Das Simplex-Verfahren arbeitet folgendermaßen:

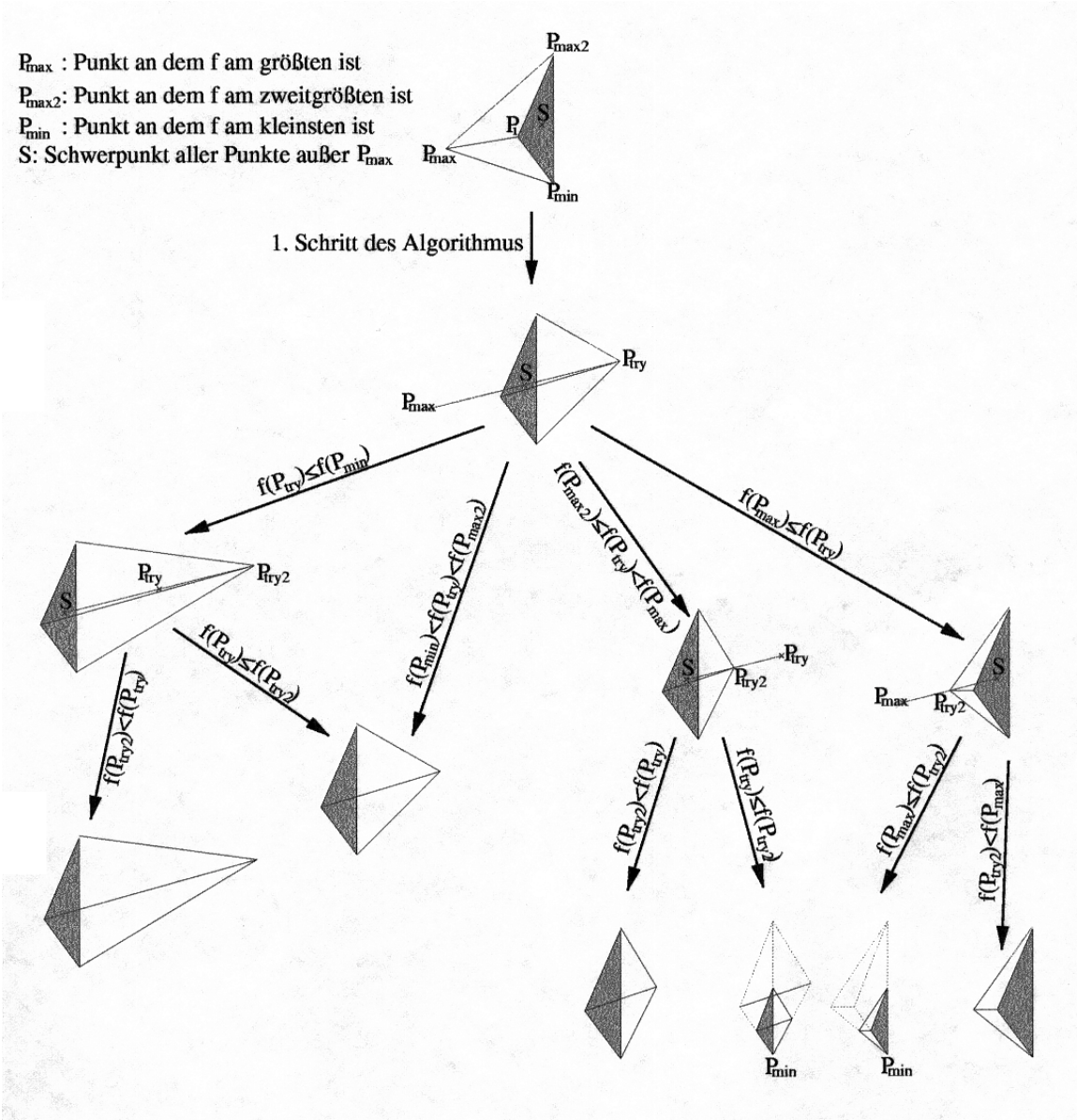
1. Bilde im *N*-dimensionalen Parameterraum einen Simplex. Ein Simplex ist allgemein ein (*N*+1)-Eck im *N*-dimensionalen Raum, als z.B. ein Dreieck in 2 Dimensionen. Verteile die Ecken des Simplex um einen Schätzwert der Parameter (Startwert) im Parameterraum.
2. Berechne die Kostenfunktion *K* auf den Ecken des Simplex.
3. Wende je nach Ergebnis eine der unten beschriebenen **Elementaroperationen** an, die eine Verschiebung des Simplex im Parameterraum bewirken.

4. Fahre bei 2. fort, es sei denn, die Abbruchbedingung ist erfüllt. Die Abbruchbedingung ist i.a. erfüllt, wenn sich die Werte der Kostenfunktion auf den Ecken des Simplex um nicht mehr als eine vorgegebene Differenz unterscheiden oder wenn eine vorgegebene Maximalanzahl von Iterationen überschritten ist.

Folgende Elementaroperationen können angewandt werden:

1. Spiegelung der Ecke über die gegenüberliegende Seite des Simplex, die den maximalen Wert der Kostenfunktion im Vergleich zu allen anderen Ecken aufweist. Damit bewegt sich der Simplex von dem Maximum fort.
2. Wie 1., jedoch mit zusätzlicher Kontraktion des Simplex in der Dimension senkrecht zu der Seite, über die gespiegelt wurde.
3. Wie 2., jedoch Expansion statt Kontraktion.
4. Kontraktion des Simplex in allen Dimensionen.
5. Expansion des Simplex in allen Dimensionen.

Mit jeder Iteration verändert der Simplex seine Lage und Größe im Parameterraum durch die Elementaroperationen und bewegt sich somit „amöbenartig“ in Richtung des Minimums. Die Matlab-Funktion FMINS verwendet empirische Regeln zur Auswahl der Elementaroperationen je nach Verteilung der Werte der Kostenfunktion auf den Ecken des Simplex. Die folgende Abbildung illustriert die Regeln für das Beispiel eines Tetraeders (Simplex in 3 Dimensionen). Ausgehend von einer aktuellen Position des Simplex wird berechnet, wie der Simplex im nächsten Schritt aussieht:



7.4 Fehlerschätzung

Durch Aufstellung und Minimierung der Kostenfunktion wird der optimale Parametersatz $\underline{a}_{opt}$ ermittelt, der gerade dem Minimum der Kostenfunktion entspricht. Es stellt sich nun die Frage nach dem zu erwartenden Fehler der Parameterschätzung, d.h. wie weit weicht der geschätzte Parametersatz $\underline{a}_{opt}$ vom „wahren" Parametersatz $\underline{a}_{wahr}$ ab. Dazu werden i.a. **Konfidenzbereiche** berechnet, die einen Bereich im Parameterraum angeben, in dem der wahre Parametersatz mit einer vorgegebenen Wahrscheinlichkeit liegt. Zur Berechnung der Konfidenzbereiche sind wieder verschiedene Fälle zu unterscheiden (wie bei der Minimierung der Kostenfunktion):

1. Als Kostenfunktion wird das Chi-Quadrat verwendet **und**

   c) das Modell $f(x,\underline{a})$ ist linear in den Parametern $\underline{a}$ („general linear least-squares"). In diesem Fall können die Konfidenzbereiche direkt aus der SVD-Lösung angegeben werden.

   d) das Modell $f(x,\underline{a})$ ist nichtlinear in den Parametern $\underline{a}$. Dann müssen die Konfidenzbereiche durch mehrfache Berechnung des Chi-Quadrats im Raumbereich um den optimalen Parametersatz herum berechnet werden (Iso-Linien der Kostenfunktion).

2. Als Kostenfunktion wird nicht das Chi-Quadrat verwendet, d.h. man benutzt andere Verteilungen des Meßfehlers als die Gauß-Verteilung. In diesem Fall bleiben nur statistische Verfahren zur Bestimmung der Konfidenzbereiche, die eine mehrfache Berechnung des optimalen Parametersatzes aus „synthetischen" Datensätzen erfordern(„Bootstrap"-Methode).

Im folgenden werden die einzelnen Verfahren erläutert.

7.4.1 Chi-Quadrat-Anpassung: Konfidenzbereiche der Parameter

7.4.1.1 Modellfunktion $f(x,\underline{a})$ ist nichtlinear in den Parametern

In diesem Fall wertet man die χ^2-Funktion auf einem Gitter im Bereich um den optimalen Parametersatz aus und bestimmt Iso-χ^2-Kurven, die einer Änderung des χ^2 im Vergleich zum Minimum um verschiedene $\Delta\chi^2$ entsprechen. Folgende Tabelle zeigt den Zusammenhang zwischen dem Wert von $\Delta\chi^2$ und der Wahrscheinlichkeit *p*, mit der der wahre Parametersatz im von der zugehörigen Iso-Kurve berandeten Bereich liegt für verschiedene Anzahlen von Parametern (ohne Beweis):

	M			
p / %	**1**	**2**	**3**	**4**
68.3	1.00	2.30	3.53	4.72
95.4	4.00	6.17	8.02	9.70
99	6.63	9.21	11.3	13.3

Beispiel: bei 2 Parametern (*M*=2) liegt der wahre Parametersatz mit einer Wahrscheinlichkeit von 95.4% innerhalb des Raumbereichs, der von der Iso-Kurve mit $\Delta\chi^2$ =6.17 begrenzt wird, also der Kurve, auf der der Wert des χ^2 um 6.17 höher als im Minimum ist. Diesen Raumbereich bezeichnet man dann als den **Konfidenzbereich auf dem Konfidenzniveau *p* = 95.4%**.

Anmerkung: In der Literatur finden sich oft Angaben über **Konfidenzintervalle**. Diese stellen die Projektion der Konfidenzbereiche auf die Achsen des Raums dar. Für die Konfidenzintervalle gelten immer die in der Tabelle für *M*=1 angegebenen $\Delta\chi^2$ -Werte (ohne Beweis). Beispiel: Ein Konfidenzintervall liegt auf dem 99%-Niveau, wenn es durch Projektion des Konfidenzbereichs mit $\Delta\chi^2$ =6.63 entstanden ist, unabhängig davon, welche Dimension M der Parameterraum hatte.

Anmerkung: Die Konfidenzbereiche sind in diesem Fall immer Ellipsoide (s.u.).

7.4.1.2 „General linear least squares"-Fall

In diesem Fall kann eine analytische Formel für die Berandungen der Konfidenzbereiche angegeben werden, d.h. man muß die Iso-Kurven nicht mehr durch Berechnung des Chi-Quadrats auf einem Gitter extrahieren. Für die Konfidenzniveaus gilt weiterhin die oben angegebene Tabelle, man bekommt die Berandungen aber auf einfachere Weise heraus. Ist eine Variation $\delta\underline{a}$ des Parametervektors ausgehend vom optimalen Vektor $\underline{a}_{opt}$ gegeben, so berechnet sich die Änderung des Chi-Quadrats nach folgender Formel (ohne Beweis):

$$\Delta\chi^2 = \sum_{i=1}^{M} \left(w_i^{\,2} \left(\underline{\underline{V}}_{(i)} \cdot \delta \underline{a} \right)^2 \right)$$

mit

$\underline{\underline{V}}_{(i)}$: i - te Spalte der Matrix $\underline{\underline{V}}$

$\delta \underline{a}$: Variation des Parametervektors ausgehend vom optimalen Parametersatz $\underline{a}_{opt}$

Die Spalten der Matrix $\underline{\underline{V}}$ bilden also die Achsen eines Ellipsoids, das die Berandung der Konfidenzbereiche darstellt (Iso-χ^2-Kurven). Die Achsenabschnitte für die Iso-Kurve bei $\Delta\chi^2 = 1$ stellen gerade die Varianz der Parameter dar (ohne Beweis):

$$\sigma^2(a_k) = \sum_{i=1}^{M} \left(\frac{V_{ki}}{w_i} \right)^2 \quad , \quad k = 1...M$$

Diese Werte stellen für jeweils das Konfidenzintervall des Parameter a_k auf dem 68.3%-Niveau dar.

7.4.2 Die „Bootstrap“-Methode zur Schätzung von Konfidenzbereichen

Im Falle nicht-normaler Verteilungen der Meßfehler (Kostenfunktion ist nicht das Chi-Quadrat) gelten die o.a. Zusammenhänge zwischen Konfidenzbereichen und Konfidenzniveaus nicht. In diesem Fall bleiben nur empirische statistische Methoden zur Ermittlung von Konfidenzbereichen, da analytische Berechnungen in diesem Fall meist nicht möglich sind. Eine mögliche empirische Methode ist das „Bootstrap“-Verfahren, das im folgenden erläutert wird. Dabei werden aus dem vorhandenen Datensatz neue „synthetische“ Datensätze erzeugt, die aber im statistischen Sinne die gleichen Eigenschaften wie der originale Datensatz haben. Einen solchen synthetischen Datensatz kann man z.B. durch zufällige Ziehung von N Meßwertpaaren mit Zurücklegen (!) aus dem Meßdatensatz (N gemessene Paare) erzeugen. Durch das Zurücklegen der gezogenen Paare ist der synthetische Satz nicht mit dem gemessenen Satz identisch und man kann beliebig viele verschiedene synthetische Datensätze erzeugen. Für jeden synthetischen Datensatz wird nun eine Modellanpassung vorgenommen und der optimale Parametersatz $\underline{a}_{opt}$ bestimmt, der natürlich je nach Datensatz verschieden ist. Nun wird die Verteilung der so bestimmten Parametersätze bestimmt und als Konfidenzbereiche diejenigen Bereiche angegeben, in denen ein bestimmter Prozentsatz der Parametersätze liegt.

Anmerkung: Die „Bootstrap“-Methode ist nur anwendbar, falls die Reihenfolge der Meßwerte nicht relevant ist und die Meßfehler aller Meßpunkte die gleiche (nicht-normale) Verteilung haben. Falls dies nicht der Fall ist, werden aufwendigere Monte-Carlo-Methoden benötigt, die aber hier nicht behandelt werden sollen.

7.5 Beurteilung der Modellgüte („goodness of fit“)

Nachdem der optimale Parametersatz und seine Konfidenzbereiche bestimmt wurden, muß noch untersucht werden, inwieweit das Modell die Daten ausreichend beschreibt („goodness of fit“). Dazu wird untersucht, mit welcher Wahrscheinlichkeit Q die verbleibenden Abweichungen der Meßwerte von der Modellvorhersage aufgrund der zufälligen Meßfehler zustande kommen oder ob die Abweichungen systematischer Natur sind. Systematische Abweichungen bedeuten, daß das Modell systematische Effekte in den Meßdaten nicht erfaßt, d.h. seine Vorhersagekraft und Güte beschränkt ist.

Die Fitgüte wird anhand der Wahrscheinlichkeit Q beurteilt, daß die verbleibenden Abweichungen der Meßwerte von der Modellvorhersage aufgrund der zufälligen Meßfehler zustande kommen. Je größer Q, desto größer die Fitgüte.

Falls die Chi-Quadrat-Kostenfunktion verwendet wird, können Ergebnisse der mathematischen Statistik benutzt werden, um Q auszurechnen. Dabei wird benutzt., daß eine K-fache Summe von Quadraten von unabhängigen, N(0,1)-verteilten Zufallsgrößen (normalverteilt mit Mittelwert 0 und Varianz 1) χ^2_K -verteilt (Chi-Quadrat-verteilt mit K Freiheitsgraden) ist. Das Chi-Quadrat entspricht nicht genau dieser Voraussetzung, da die Modellfunktion die Unabhängigkeit der Zufallsgrößen reduziert. Trotzdem wird mit guter Näherung angenommen, daß das Chi-Quadrat bei N Meßwerten und M Modellparametern $\chi^2_{(N-M)}$ –verteilt ist. Daraus läßt sich Q ableiten (ohne Beweis):

$$Q = 1 - \frac{1}{\Gamma((N-M)/2)} \cdot \int_0^{\chi^2/2} y^{(N-M)/2-1} \exp(-y) dy$$
$$= 1 - gammainc(\chi^2/2, (N-M)/2)$$

Dabei stellt der 2. Summand die **inkomplette Gammafunktion** dar. Durch Einsetzen des beobachteten Chi-Quadrat-Wertes im Minimum der Kostenfunktion sowie der Anzahl (*N-M*) der Freiheitsgrade in diese Formel wird *Q* direkt berechnet.

Anmerkung: Die inkomplette Gamma-Funktion steht in Matlab als Befehl **gammainc** zur Verfügung.

Anhand des Wertes von Q können folgende Aussagen über das Modell getroffen werden:

Wert von Q	**Folgerung**
Klein (Q~10^{-18})	Das Modell ist falsch **oder** Die Varianz der zufälligen Meßfehler wurde zu klein geschätzt **oder** Die Annahme normalverteilter Meßfehler stimmt nicht
Q~10^{-3}	Modell OK
Q~1	Zu gut um wahr zu sein. Evtl wurde die Varianz der zufälligen Meßfehler zu groß geschätzt

Es ist also ersichtlich, daß zu einer „sauberen“ Beurteilung des Wertes von *Q* sichergestellt sein muß, daß die Annahmen über die Statistik der Meßfehler korrekt sein müssen.

Der Wert des Chi-Quadrats kann auch mit einer **„Daumenregel“** beurteilt werden. Nimmt man an, daß die zufällige Abweichung der Meßwerte vom vorhergesagten Modellwert im Mittel ungefähr 1σ ist, so ist jeder Summand in der Chi-Quadrat-Funktion ungefähr 1. Das Chi-Quadrat nimmt also bei *N* Summanden (*N* Meßwerten) ungefähr den Wert *N* an. Gleichzeitig wird aber die Anpassung um so leichter, je mehr Parameter vorhanden sind. Dies reduziert den Erwartungswert des Chi-Quadrats in etwa um die Anzahl der Parameter *M*. **Die Daumenregel lautet also, daß der Wert des Chi-Quadrats ungefähr gleich der Anzahl der Freiheitsgrade *N-M* sein sollte, wenn die verbleibenden Abweichungen der Meßwerte von der Modellvorhersage zufällig sind und damit das Modell akzeptabel ist**. Damit ergibt sich folgende Beurteilungstabelle:

Wert von χ^2	**Folgerung**
χ^2>>*N-M*	Das Modell ist falsch **oder** Die Varianz der zufälligen Meßfehler wurde zu klein geschätzt **oder** Die Annahme normalverteilter Meßfehler stimmt nicht
χ^2~*N-M*	Modell OK
χ^2<<*N-M*	Zu gut um wahr zu sein. Evtl wurde die Varianz der zufälligen Meßfehler zu groß geschätzt

Anmerkung: Für die anderen Kostenfunktionen, die nicht auf einer Normalverteilung der Meßfehler beruhen, ist die Bestimmung von *Q* und damit die Beurteilung der Fitgüte nur schwer möglich.

7.6 Methodenübersicht

Die folgende Tabelle zeigt die verschiedenen Fälle bei der Modellanpassung in der Übersicht:

Fall	**Kostenfunktion**	**Minimierung**	**Konfidenzbereiche**	**Fitgüte**
Meßfehler normalverteilt und Modellfunktion linear in den Parametern („general least-squares“)	Chi-Quadrat	über SVD oder Normalgleichungen	Berechnung von Iso-χ^2-Linien über SVD	Chi-Quadrat-Verteilung oder Daumenregel
Meßfehler normalverteilt aber Modellfunktion nichtlinear in den Parametern	Chi-Quadrat	Simplex-Verfahren	Berechnung von Iso-χ^2-Linien durch Abtastung im Parameterraum	Chi-Quadrat-Verteilung oder Daumenregel
Meßfehler nicht normalverteilt	Kostenfunktion nach dem Maximum-Likelihood-Prinzip	Simplex-Verfahren	„Bootstrap“-Methode	Nur möglich, wenn die Wahrscheinlich

	basierend auf angenommener Verteilung der Meßfehler (z.B. Lorentz-Verteilung („robuster“ Fit) oder Poisson-Verteilung)			keitsverteilung der Werte der Kostenfunktion ableitbar ist, evtl. Daumenregel anwendbar

7.7 Aufgaben

1. Das zur Verfügung gestellte Skript linfit.m paßt eine Gerade an Testdaten an. Als Kostenfunktion wird die χ^2-Funktion und zur Funktionenminimierung das Simplex-Verfahren (Matlab-Funktion FMINS) benutzt.

   i. Vollziehe das Skript linfit.m nach und verändere dabei die Testdaten sowie die Startwerte der Parameterschätzung. Welchen Einfluß haben „Ausreißer“?
   ii. Verändere die Kostenfunktion so, daß von Lorentz-verteilten Meßfehlern ausgegangen wird. Wie verändert sich der Einfluß von Ausreißern?

2. Zur Messung der Halbwertzeit eines radioaktiven Isotops wird die Anzahl der Zerfälle alle 10s jeweils für ein Intervall von Δt=1s gezählt. Folgende Werte werden gemessen:

Zeit / s	Aktivität (Anzahl Zerfälle)	Zeit / s	Aktivität (Anzahl Zerfälle)
10	24	60	5
20	17	70	4
30	11	80	2
40	10	90	3
50	6	100	1

   i. Schätze die Halbwertzeit ab. Gehe dabei davon aus, daß die Aktivität exponentiell abnimmt:

   $$A(t) = A_0 e^{-\alpha t}$$

   Gehe weiterhin davon aus, daß die Meßwerte Poisson-verteilt sind:

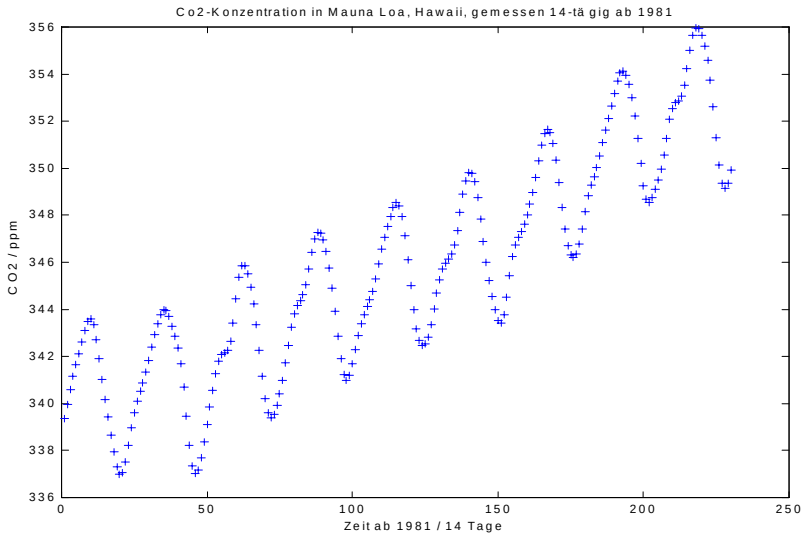
   $$p(k,\lambda) = e^{-\lambda}\frac{\lambda^k}{k!}$$

   Dabei ist k die Anzahl der Zerfälle und λ der Erwartungswert der Anzahl der Zerfälle (Der Erwartungswert ist das Produkt aus der Aktivität und dem Zählintervall: $\lambda = A(t) * \Delta t$)
   (Hinweis: Die zu minimierende Kostenfunktion kann auf Anfrage als Matlab-Skript f_pois.m zur Verfügung gestellt werden)

   ii. Um welchen Prozentsatz ändert sich die geschätzte Halbwertzeit, wenn von Gauss-verteilten Meßwerten ausgegangen wird (Least-Squares-Fit).

3. Die folgende Abbildung zeigt die Meßergebnisse der CO_2-Konzentration auf Hawaii ab 1981 (14-tägig aufgenommen). Die Standardabweichung der Messung beträgt etwa σ = 0.16 ppm.

   Hinweis: Die Daten finden sich in der Datei mauna.dat. Zum Einlesen in Matlab verwende den Befehl „load mauna.dat –ascii“.



a) Bestimme die Anstiegsrate der CO_2-Konzentration in ppm/Jahr. Benutze dazu als Modell, daß die Konzentration linear ansteigt. Wann wird sie gemäß dieses Modells um 15% höher als 1981 liegen. Benutze als Kostenfunktion das Chi-Quadrat (σ = 0.16 ppm für alle Meßwerte) und zur Minimierung das Simplex-Verfahren.

b) Wie a), jedoch unter der Modellannahme, daß die Konzentration quadratisch ansteigt.
Hinweis: Die im Skript linfit.m verwendete lineare Modellfunktion läßt sich einfach auf Polynome n-ter Ordnung erweitern, indem einfach n+1 Startwerte der Parameter angegeben werden.

c) Beurteile die Güte der unter a) und b) benutzten Modelle unter Benutzung der „Daumenregel“ für das Chi-Quadrat. Kann man den Aussagen aus Teilaufgabe a) und b) über den erwarteten Zeitpunkt einer Erhöhung der Konzentration um 15% trauen?

d) Mit welcher einfachen Erweiterung der quadratischen Modellfunktion ließe sich das Modell verbessern? Wiederhole die Anpassung mit dieser erweiterten Modellfunktion und beurteile die Güte dieses Modells unter Benutzung der „Daumenregel“.

e) (*) Wiederhole die Anpassung nach Teilaufgabe d) unter Verwendung des „general least-squares“-Verfahrens (mit SVD).

8 Analog-Digital-Wandlung und Diskrete Fouriertransformation

Bei der Auswertung von Meßsignalen (z.B. Temperatur- und Druckverläufe oder Spannungen und Ströme als Funktion der Zeit) spielen numerische Methoden ein wichtige Rolle. Insbesondere ist hier die **Spektralanalyse**, d.h. Analyse des Frequenzgehalts eines Signals mit Hilfe der **Diskreten Fouriertransformation (DFT)** zu nennen, die im folgenden erläutert werden soll. Zunächst wird dazu der Prozeß der **Analog-Digital-Wandlung (A/D-Wandlung)** beschrieben, der zur Diskretisierung von Meßsignalen zwecks weiterer Verarbeitung im Computer dient. Darauf aufbauend wird die DFT sowie die schnelle Fouriertransformation (FFT) als ein schneller Algorithmus zur Berechnung der DFT eingeführt. Zuletzt wird die **Filterung** von Signalen als Anwendung der DFT beschrieben. Hierbei spielt der sog. **Faltungssatz** eine wichtige Rolle.

8.1 Analog-Digital-Wandlung (A/D-Wandlung)

Physikalische Meßsignale liegen i.A. zeit- und wertekontinuierlich vor, z.B. kann ein Druckverlauf allgemein als reellwertige Funktion der kontinuierlichen Zeit $p = p(t)$ geschrieben werden. Man spricht hierbei von analogen Signalen bzw. kontinuierlichen Funktionen[11]. Da der Computer nur endliche Mengen von Zahlen verarbeiten kann, müssen diese Meßsignale/Funktionen in endliche Zahlenfolgen umgewandelt werden:

$$x(t) \rightarrow \{x(n)\} \quad ; \quad n \in K \subset \mathbb{Z} \ .$$

Dabei besteht die Folge aus den Werten der Funktion zu den Zeitpunkten, die jeweils im Abstand einer vorgegebenen Periode T liegen:

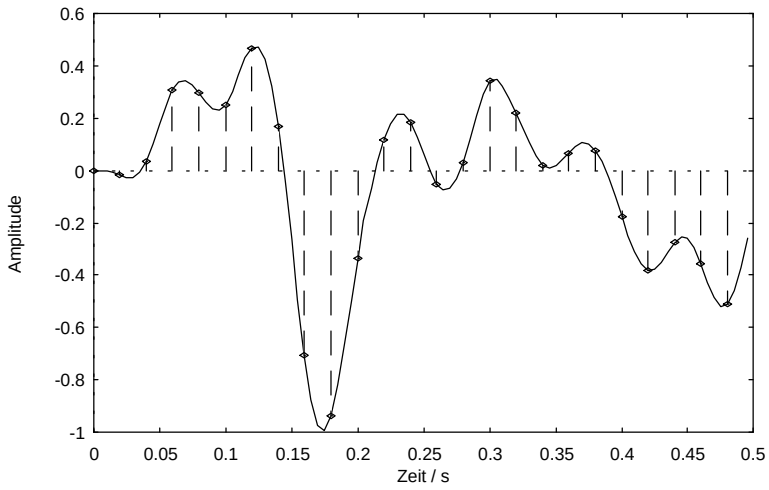
$$x(n) := x(n \cdot T) \quad ; n \in K \subset \mathbb{Z}$$

$$x(n) : Abtastwerte$$

$$T : Abtastperiode$$

$$f_s = 1/T : Abtastfrequenz$$

Dieser Prozeß der Umwandlung in eine Zahlenfolge wird als Abtastung oder Diskretisierung bezeichnet. Seine wichtigste Kenngröße ist die Abtastfrequenz f_s, deren Kehrwert gerade die Abtastperiode T ist. Eine technische Realisierung der Abtastung zur Diskretisierung realer physikalischer Signale heißt Analog-Digital-Wandlung (A/D-Wandlung) oder Digitalisierung.



Zur Analog-Digital-Wandlung: Analoges Signal *x(t)* (durchgezogene Linie), Abtastzeitpunkte (gestrichelte Linien) und Abtastwerte (Rauten). Die Abtastwerte bilden die der Funktion *x(t)* zugeordnete Zahlenfolge *{x(n)} = { x(0), x(1), x(2) , x(3) , . . .}*.

[11] Ist der Zeitverlauf einer realen physikalischen Größe gemeint, so wird im Folgenden der Begriff „Signal“ verwendet. Ist von der abstrakten mathematischen Repräsentation eines Signals die Rede, so wird der Begriff „Funktion“ verwendet.

Anmerkung: Die Abtastung ist mathematisch auch für unendlich lange Signale möglich, im Rechner jedoch kann sie nur auf endlich lange Signale angewandt werden, da die Folge der Abtastwerte sonst unendlich lang würde.

Anmerkung: Aufgrund der endlichen Genauigkeit der Zahlendarstellung im Rechner entsprechen die Abtastwerte nicht genau den Werten der Funktion, sondern weisen einen zufälligen Rundungsfehler auf, der von der Zahlendarstellung abhängt. Die Abweichung zwischen dem im Rechner dargestellten Abtastwert und dem wahren Wert der Funktion zum zugehörigen Zeitpunkt wird als **Quantisierungsfehler** bezeichnet. Wird z.B. eine Sinus-Funktion durch eine Folge von Abtastwerten repräsentiert, so entspricht der Quantisierungsfehler gerade der Genauigkeit, mit dem die Werte der Sinus-Funktion berechnet werden.

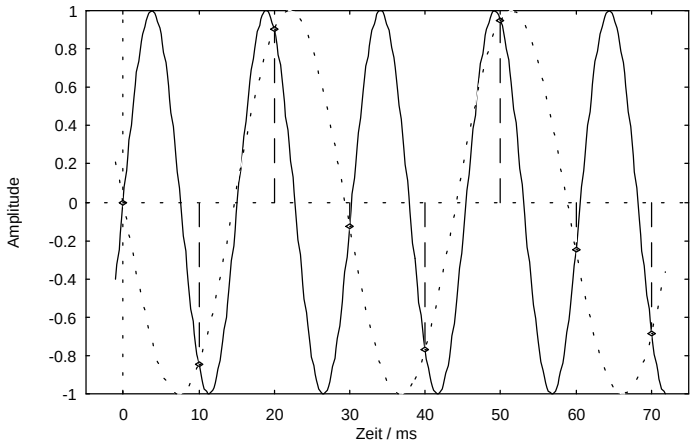
Durch die Abtastung geht der Verlauf der abgetasteten Funktion zwischen den Abtastzeitpunkten verloren. Das sog. **Abtasttheorem** sagt jedoch, daß unter einer bestimmten Bedingung kein Informationsverlust eintritt:

> **Abtasttheorem** (ohne Beweis):
> Eine Funktion $x(t)$ kann aus der Folge ihrer Abtastwerte $\{x(n)\}$ eindeutig rekonstruiert werden, wenn sie nur Frequenzen unterhalb der halben Abtastfrequenz enthält. Mit anderen Worten: Nur dann repräsentiert die Folge der Abtastwerte die Funktion eindeutig.

Anschaulich gesprochen kann sich das Signal nicht beliebig schnell ändern, wenn hohe Frequenzen fehlen. Dadurch wird es aus den Abtastwerten vorhersagbar (interpolierbar).

Anmerkung: Um das Abtasttheorem nicht zu verletzen, müssen bei der A/D-Wandlung die analogen Signale vor der Abtastung tiefpaß-gefiltert werden, d.h. alle Frequenzen oberhalb der halben Abtastfrequenz werden weggefiltert und nur die tiefen Frequenzen durchgelassen.

Anmerkung: Sind in einem Signal Frequenzen enthalten, die höher sind als die halbe Abtastfrequenz, so bilden sich diese in der Folge der Abtastwerte bei tieferen Frequenzen ab („Spiegelfrequenzen“). Dieses Phänomen wird als **Aliasing** bezeichnet (siehe Übungen).



Zum Aliasing-Phänomen: Abtastung einer Sinuswelle hoher Frequenz (durchgezogene Linie) mit zu geringer Abtastrate. Die Abtastwerte (Rauten) repräsentieren auch eine Sinuswelle mit einer geringeren Frequenz (punktierte Linie), so daß eine Mehrdeutigkeit auftritt (Aliasing).

8.2 Diskrete Fouriertransformation (DFT)

Die Fouriertransformation ist allgemein für kontinuierliche Funktionen definiert. Sie beschreibt eine Zerlegung der Funktion in Sinusfunktionen, wobei jede Frequenz vorkommen kann (kontinuierliches Frequenzspektrum). Sie gibt für jede Frequenz an, mit welcher Amplitude und welcher Phase (Verschiebung) die Sinusfunktion dieser Frequenz in der Funktion enthalten ist. Um zu einer Version der Fouriertransformation zu kommen, die mit endlichen Zahlenfolgen rechnet und daher auf dem Computer implementierbar ist, werden im folgenden ausschließlich abgetastete, periodische Funktionen betrachtet. Nehmen wir an, die Abtastperiode sei T und die Periode des Signals $T_0 = N \cdot T$, wobei N eine positive ganze Zahl ist. Dann ist eine solche Funktion eindeutig durch die N Abtastwerte einer Periode

$$\{x(n)\} \quad ; \quad n = 0,\ldots,N-1$$

repräsentiert. $x(N)$ ist aufgrund der Periodizität wieder identisch $x(0)$ usw.. Diese endliche Zahlenfolge kann im Computer verarbeitet werden und repräsentiert dennoch die gesamte, unendlich lange periodische Funktion.

Wie sieht nun das Frequenzspektrum einer abgetasteten, periodischen Funktion aus? Sie kann nur aus Sinusfunktionen zusammengesetzt sein, die ebenfalls die Periode T_0 aufweisen. Falls nicht, würde ja eine Sinusfunktion in zwei aufeinanderfolgenden Perioden der Funktion nicht identisch sein, was der vorausgesetzten Periodizität widerspricht. Alle Sinusfunktionen mit der Periode T_0 / k mit k beliebige natürliche Zahl haben auch die Periode T_0. Das Spektrum enthält somit nur Frequenzen $f_k = k / T_0$, es ist also diskret mit der Abtastperiode $f_0 = 1 / T_0$ (dies entspricht der bekannten Fourierreihe). Eine Voraussetzung für die Darstellbarkeit im Rechner ist damit schon erfüllt: Das Spektrum kann als Zahlenfolge dargestellt werden. Aber ist diese Folge auch endlich? Diese Frage wird im folgenden Absatz behandelt.

Aufgrund der Symmetrie der Fouriertransformation (Hin- und Rücktransformation sind nur durch ein Minuszeichen unterschieden, das einer Zeitumkehr entspricht), gilt das oben benutzte Argument auch umgekehrt: Ist das Frequenzspektrum periodisch mit der Periode $f_s = 1 / T$ (periodisch mit der Abtastfrequenz), so ist die Zeitfunktion diskret mit der Abtastperiode T (was ja gerade der Ausgangspunkt war). Wenn nur Frequenzen $f_k = k / T_0$ vorkommen und die Periode des Spektrums $1 / T = N / T_0$ ist, so gibt es genau N verschiedene Frequenzkomponenten.

Insgesamt ist somit für abgetastete, periodische Funktionen auch das Frequenzspektrum periodisch und abgetastet. Sowohl die Funktion als auch ihr Frequenzspektrum kann im Rechner eindeutig als endliche Zahlenfolge der Länge N dargestellt werden, nämlich durch die Abtastwerte jeweils einer Periode. Die Diskrete Fouriertransformation setzt nun gerade die Abtastwerte der Zeitfunktion und des Spektrums in Beziehung (ohne Beweis):

Diskrete Fouriertransformation der Länge N :

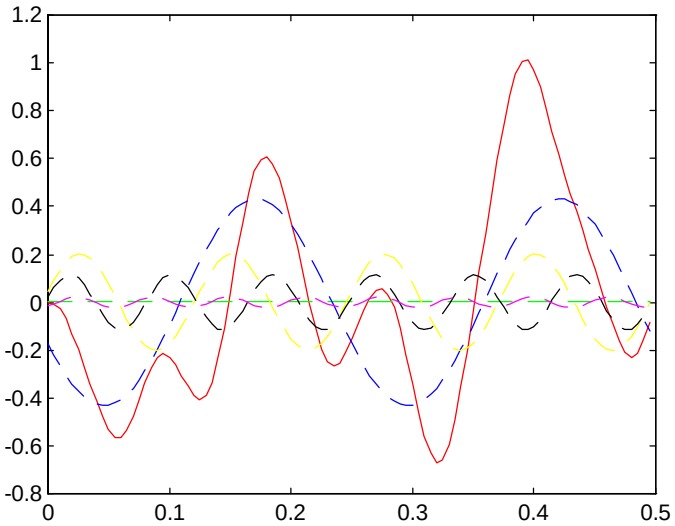
$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot \exp\left(-2\pi i \frac{kn}{N}\right) \quad ; \quad k = 0\ldots N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot \exp\left(2\pi i \frac{kn}{N}\right) \quad ; \quad n = 0\ldots N-1$$

Dabei stellen $X(k)$ bzw. $x(n)$ die Abtastwerte des Spektrums bzw. der Zeitfunktion dar. Die erste Zeile bezeichnet die DFT und die zweite die inverse DFT. Die Beziehung zwischen dem Laufindex k bzw. n und Frequenz bzw. Zeit stellt sich dabei folgendermaßen dar:

$$\begin{aligned} n = 0,1,2,\ldots,N-1 \quad &\triangleq \quad t = 0,T,2T,\ldots,(N-1)T \\ k = 0,1,2,\ldots,N-1 \quad &\triangleq \quad f = 0,\frac{1}{T_0},\frac{2}{T_0},\ldots,\frac{(N-1)}{T_0} \\ &\qquad = 0,1\frac{f_s}{N},2\frac{f_s}{N},\ldots,(N-1)\frac{f_s}{N} \end{aligned}$$

Diese Formeln finden zur Berechnung von Frequenz- und Zeitachse der per DFT berechneten Zahlenfolgen Anwendung. Die Abtastwerte $x(n)$ können sowohl reell- als auch komplexwertig sein, wobei physikalische Signale immer reellwertig sind. Die $X(k)$ sind in beiden Fällen komplexwertig. Der Betrag $|X(k)|$ stellt dabei die Amplitude und das Argument $\arg(X(k))$ die Phase (Verschiebung) dar, mit der eine Sinusfunktion dieser Frequenz in der Funktion enthalten ist.



Zur DFT: Periodisches physikalisches Signal (durchgezogene Linie, dargestellt ist nur eine Periode) und 5 verschiedene darin enthaltene Sinuskomponenten. Zu beachten ist die unterschiedliche Frequenz, Amplitude und relative Verschiebung („Phase“) der Komponenten.

Anmerkung: Es ist wichtig zu bemerken, daß die DFT **ausschliesslich** periodische Funktionen betrachtet. Nimmt man sich also einen beliebigen endlichen Zeitabschnitt einer Funktion/eines Signals, so berechnet die DFT nicht das Spektrum dieses Ausschnitts, sondern das Spektrum der periodisch fortgesetzten Version dieses Ausschnitts (zu den Konsequenzen siehe Übungen).

8.2.1 Reellwertige Zeitfunktionen

Ist die Zeitfunktion $x(n)$ reellwertig, so gilt:

$$X(N-k) = X^*(k) \quad ; \quad k = 0,\ldots,\frac{N}{2},$$

wobei * die komplexe Konjugation bedeutet. Es reichen also $N/2$ komplexe Werte $X(k)$ um N reelle Werte $x(n)$ zu repräsentieren und das Spektrum ist eindeutig im Bereich bis zur halben Abtastfrequenz repräsentiert (vgl. obige Formel zur Umrechnung zwischen Laufindex und Frequenz). Für reelle Signale wird daher das Spektrum nur im Bereich zwischen 0 Hz und der halben Abtastfrequenz $f_s/2$ dargestellt. Der Rest ist redundant.

Anmerkung: Das Abtasttheorem und das oben erwähnte Aliasing sind eine Folge der Periodizität des Spektrums. Ist das Spektrum periodisch mit der Abtastfrequenz f_s, so kommt nur eine Periode davon eine physikalische Bedeutung bei. Beispielsweise sind die Werte des Spektrums bei den Frequenzen $f = 0.3\,f_s$ und $f = (0.3\,f_s + f_s)$ immer identisch. Das abgetastete Signal kann Sinusfunktionen dieser Frequenzen also gar nicht unabhängig voneinander enthalten haben, sie sind somit ununterscheidbar. Für reelle Signale gilt dies aufgrund der hier angegebenen Eigenschaft auch für die Frequenzen $f = 0.3\,f_s$ und $f = (0.3\,f_s + f_s/2)$, so daß das Abtasttheorem resultiert.

8.2.2 Leistungsspektrum

Oft ist man an den komplexen Spektralwerten $X(k)$ gar nicht interessiert, sondern fragt danach, mit welcher Leistung ein Sinus einer bestimmten Frequenz im Signal enthalten ist. Die Leistung ist gerade das Betragsquadrat des Spektralwertes $|X(k)|^2$. Sie kann innerhalb typischer physikalischer Signale über die Frequenz hinweg um mehrere Größenordnungen schwanken, so daß zur grafischen Darstellung oft eine Logarithmierung sinnvoll ist. Als **Leistungspektrum in dB (Dezibel)** definiert man daher folgende Größe:

$$L(k)/dB = 10 \cdot \log_{10}\left(\left|X(k)\right|^2\right) \quad ; \quad k = 0,\ldots,N-1$$

Anmerkung: Wenn sich der Betrag $|X(k)|$ gerade verzehnfacht, ändert sich der Wert im Leistungsspektrum um 20 dB.

8.3 Schnelle Fouriertransformation (FFT)

Die DFT hat einen relativ hohen Rechenaufwand, wenn man die Spektralwerte $X(k)$ direkt nach der Formel der DFT berechnet. Für jeden der N Spektralwerte sind N Multiplikationen und Additionen zu berechnen, so daß der Gesamtaufwand proportional N^2 ist. Bereits Gauß hat jedoch erkannt, daß Teilsummen verschiedener Frequenzen (Index k) identisch sind, wenn N eine Potenz von 2, also $N = 2^1, 2^2, 2^3, 2^4, 2^5, 2^6$.usw., ist. Die FFT nutzt diese Symmetrieeigenschaften aus und berechnet die Teilsummen nur einmal. Dadurch reduziert sich der Rechenaufwand sich von N^2 auf $N \cdot \log_2(N)$, was für große N einen beträchtlichen Unterschied im Aufwand bedeutet. Die FFT stellt also nichts anderes als einen schnellen Algorithmus zur Berechnung der DFT dar. Man kann sagen, daß die DFT nicht die Bedeutung im Bereich der rechnergestützten Signalanalyse erlangt hätte, wenn man nicht diesen schnellen Algorithmus zu ihrer Berechnung gefunden hätte.

Im folgenden wird nun eine Möglichkeit zur Ableitung des FFT-Algorithmus erläutert. Mit der Definition

$$W_N := \exp\left(-\frac{2\pi i}{N}\right)$$

lautet die Formel für die DFT in vereinfachter Schreibweise

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} \quad ; \quad k = 0, \ldots, N-1 \; .$$

Zunächst wird diese Summe in zwei Teilsummen jeweils über die geraden und über die ungeraden Abtastwerte zerlegt:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} \\ &= \sum_{n=0}^{N/2-1} x(2n) \cdot W_N^{k2n} + \sum_{n=0}^{N/2-1} x(2n+1) \cdot W_N^{k(2n+1)} \\ &= \sum_{n=0}^{N/2-1} x(2n) \cdot \left(W_N^2\right)^{kn} + W_N^{-k} \cdot \sum_{n=0}^{N/2-1} x(2n+1) \cdot \left(W_N^2\right)^{kn} \end{aligned}$$

Mit der Identität (siehe Definition)

$$W_N^2 = W_{N/2}$$

ergibt sich:

$$X(k) = \sum_{n=0}^{N/2-1} x(2n) \cdot W_{N/2}^{kn} + W_N^k \cdot \left(\sum_{n=0}^{N/2-1} x(2n+1) \cdot W_{N/2}^{kn} \right)$$

Die beiden verbliebenen Summen stellen gerade die Formel für die DFT der geraden bzw. ungeraden Abtastwerte dar, die im folgenden als $X_g(k)$ und $X_u(k)$ bezeichnet werden:

$$\begin{aligned} X_g(k) &= \sum_{n=0}^{N/2-1} x(2n) \cdot W_{N/2}^{kn} \quad ; \quad k = 0, \ldots, N/2 - 1 \\ X_u(k) &= \sum_{n=0}^{N/2-1} x(2n+1) \cdot W_{N/2}^{kn} \quad ; \quad k = 0, \ldots, N/2 - 1 \end{aligned} \; .$$

Für diese DFT‘s der Länge $N/2$ läuft der Index k zwischen 0 und $N/2 - 1$, für die Spektralwerte $X(k)$ jedoch von 0 bis N-1. Aufgrund der Identität

$$W_{N/2}^{(k+N/2)n} = W_{N/2}^{kn}$$

stellen jedoch die Teilsummen für $k = N/2$ bis N-1 wieder die DFT der geraden bzw. ungeraden Abtastwerte dar. Dadurch ergibt sich insgesamt mit der Modulofunktion mod:

$$X(k) = X_g\left((k)_{\mathrm{mod}N/2}\right) + W_N^k \cdot X_u\left((k)_{\mathrm{mod}N/2}\right) \quad ; \quad k = 0,\ldots,N-1$$

Die gesuchten Spektralwerte $X(k)$ ergeben sich also aus den beiden Spektralwerten $X_g(k)$ und $X_u(k)$ durch eine einzige Multiplikation und Addition. Jedes $X_g(k)$ und $X_u(k)$ wird dabei aufgrund der Modulo-Funktion zweimal verwendet. Hierin besteht die Rechenersparnis.

Insgesamt wurde durch diese Umformungen die DFT der Länge N in 2 DFT's jeweils der Länge $N/2$ sowie eine „Nachbearbeitung" zerlegt. Die beiden DFT's arbeiten auf den Abtastwerten mit geradem bzw. ungeraden Index und die „Nachbearbeitung" berechnet aus den Ergebnissen der Teil-DFT's das Ergebnis der Gesamt-DFT. Die Nachbearbeitung erfordert insgesamt N Operationen (eine Operation pro Spektralwert). Nun kann man die beiden Teil-DFT's in einer nächsten Stufe wieder auf dieselbe Weise zerlegen. Man erhält je 2 DFT's der Länge $N/4$ und je eine Nachbearbeitung mit $N/2$ Operationen. Insgesamt fallen also in dieser nächsten Stufe 4 DFT's der Länge $N/4$ und 2 Nachbearbeitungen mit insgesamt wieder N Operationen an. Diese Zerlegung kann man sooft machen, bis die DFT nur noch auf 2 Abtastwerten arbeitet. Man braucht dann $N/2$ DFT's der Länge 2, die jeweils den Aufwand 2 haben, also auch wieder insgesamt N Operationen.

In jeder Stufe braucht es also N Operationen (für die DFT's der Länge zwei in der untersten bzw. für die „Nachbearbeitungen" in den darauffolgenden Stufen) und insgesamt gibt es $\log_2(N)$ Stufen (z.B. läßt sich N=8 zweimal durch 2 teilen, bis man bei der DFT-Länge 2 angekommen ist und es gibt somit insgesamt 3 Stufen (eine Stufe mit DFT's der Länge 2 und 2 Stufen mit „Nachbearbeitungen")). Der Gesamtaufwand der FFT ist somit proportional $N \cdot \log_2(N)$, wie oben angegeben.

Anmerkung: Man erreicht eine gewisse Reduzierung des Rechenaufwandes für die DFT auch für Längen N, die nicht einer Zweierpotenz entsprechen. Man zerlegt dazu N in ein Produkt aus möglichst vielen Zweierpotenzen, führt für diese Teilmengen der Abtastwerte getrennt eine FFT durch und baut das Ganze nachträglich geschickt zusammen. Dies wird als **mixed-radix FFT** bezeichnet. Sie ist in Matlab implementiert, der Befehl `fft` führt, je nach Länge des übergebenen Vektors, entweder eine reine FFT oder eine mixed-radix FFT durch. Der Rechenaufwand der mixed-radix FFT liegt, je nachdem wie gut sich die Zahl zerlegen läßt, zwischen dem der DFT und dem der FFT. Da man meist nicht direkt weiß, wie gut eine Zahl zerlegbar ist, sollte man möglichst immer Zweierpotenzen verwenden. Um auf eine entsprechende Länge zu kommen, sollten an das Signal Nullen angehängt werden, die das Spektrum nicht verändern.

8.4 Filterung und Faltungssatz

Unter dem Begriff **Filterung** versteht man in der Signaltheorie die Veränderung des Frequenzspektrums eines Signals/einer Funktion. Sind beispielsweise in einem zu analysierenden physikalischen Signal bestimmte Frequenzbereiche besonders interessant, so können sie durch Filterung angehoben werden, während die uninteressanten, etwa mit Rauschen überlagerten Frequenzbereiche abgeschwächt werden können. Grundsätzlich lassen sich 4 Grundtypen von Filtern unterscheiden:

1. **Tiefpaßfilter**: Läßt nur Frequenzen **unterhalb** einer gegebenen Grenzfrequenz durch.
2. **Hochpaßfilter**: Läßt nur Frequenzen **oberhalb** einer gegebenen Grenzfrequenz durch.
3. **Bandpaßfilter**: Läßt nur Frequenzen **innerhalb eines Bereichs** zwischen einer gegebenen unteren und oberen Grenzfrequenz durch.
4. **„Notch-Filter"**: Läßt nur Frequenzen **außerhalb eines Bereichs** zwischen einer gegebenen unteren und oberen Grenzfrequenz durch.

Im Frequenzbereich läßt sich der Prozeß der Filterung auf einfache Weise mathematisch formulieren. Da die Spektralwerte die Amplitude und Phase der Sinusfunktionen zugehörigen Frequenz angeben, kann die Filterung als Multiplikation der Spektralwerte mit einem frequenzabhängigen Faktor beschrieben werden:

$$Y(k) = X(k) \cdot H(k) \quad ; \quad k = 0,\ldots,N-1$$

Dabei sind $X(k)$ und $Y(k)$ die Spektralwerte des ursprünglichen und des gefilterten Signals. Die Folge der (komplexen) Faktoren $H(k)$ beschreibt das Filter eindeutig und kann zur Realisierung des gewünschten Filtertyps frei vorgegebenen werden. Sie heißt **Übertragungsfunktion** des Filters.

Welche Operation im Zeitbereich entspricht der Multiplikation im Frequenzbereich, d.h. wie müssen x(n) und h(n) verknüpft werden, damit y(n) resultiert (Groß- und Kleinbuchstaben jeweils zusammengehörige Paare von Zeitsignal und Spektralwerten)? Dazu wendet man den **Faltungssatz der DFT** an (ohne Beweis).

Seien $\{x(n)\}$ und $\{h(n)\}$ zwei Folgen der Länge N und werde daraus durch zyklische Faltung eine Ausgangsfolge $\{y(n)\}$ berechnet:

$$\begin{aligned} &\textit{zyklische Faltung}: \\ &y(n) = (x \otimes h)(n) \\ &\quad = \sum_{m=0}^{N-1} x(m) \cdot h\left((n-m)_{\bmod N}\right) \quad ; \quad n = 0,\ldots,N-1 \end{aligned}$$

Der Faltungssatz besagt nun, daß die Spektralwerte $Y(k)$ der Ausgangsfolge $\{y(n)\}$ gerade das Produkt der Spektralwerte von $\{x(n)\}$ und $\{h(n)\}$ ist:

$$\begin{aligned} &\textit{Faltungssatz}: \\ &\quad y(n) = (x \otimes h)(n) \quad ; \quad n = 0,\ldots,N-1 \\ &\Leftrightarrow \quad Y(k) = X(k) \cdot H(k) \quad ; \quad k = 0,\ldots,N-1 \\ &mit: Y = DFT(y); X = DFT(x); H = DFT(h) \end{aligned}$$

Eine Faltung im Zeitbereich entspricht also gerade einer Multiplikation im Frequenzbereich. Dies gilt auch umgekehrt: Multiplikation im Zeitbereich entspricht Faltung im Frequenzbereich.

Die Realisierung eines Filters durch Multiplikation im Frequenzbereich mit der Übertragungsfunktion $H(k)$ entspricht also im Zeitbereich der zyklischen Faltung mit der Folge der $h(n)$, die sich durch inverse DFT aus den $H(k)$ berechnet. Die Folge der $h(n)$ wird als **Impulsantwort** bezeichnet.

Eine Filterung kann also auf 2 Arten realisiert werden:

1. Anwendung des Faltungssatzes: Berechne zunächst die DFT der Zeitfunktion, multipliziere diese mit der gewünschten Übertragungsfunktion und bilde dann die inverse DFT, um zur Zeitfunktion des Ausgangssignals zu kommen:

$$y = IDFT\left(DFT(x) \cdot H\right)$$

2. Bilde die Ausgangsfolge direkt durch Faltung im Zeitbereich.

$$y(n) = (x \otimes h)(n) \quad ; \quad n = 0,\ldots,N-1$$

8.4.1 Faltung und zyklische Faltung

Ganz allgemein ist die Faltung für beliebig lange Signale definiert als:

$$\begin{aligned} y(n) &= (x \times h)(n) \\ &= \sum_{m=-\infty}^{\infty} x(m) \cdot h(n-m) \quad ; \quad n \in \mathbb{Z} \end{aligned}$$

Für Indizees ausserhalb des Definitionsbereiches von x oder h wird bei dieser allgemeinen Definition jeweils eine Null als Wert eingesetzt. Deswegen kann die Summe immer von $-\infty$ bis ∞ laufen.

Da die Faltung ein aufwendiger Algorithmus ist, wird sie oft durch Anwendung des Faltungssatzes realisiert, indem man erst in den Frequenzbereich geht, dann multipliziert und dann zurücktransformiert. Haben die Signale die Länge einer Zweierpotenz, kann hierzu die FFT angewandt werden. Dann ist der gesamte Rechenaufwand für die Hin- und Rück-FFT sowie die Multiplikation kleiner als bei direkter Benutzung der Faltungssumme, was etwa ab Signallängen von etwa N=128 gilt. Bei der Realisierung der Faltung über den Faltungssatz muß jedoch folgendes beachtet werden: Sind die Signale x und h periodisch und gleich lang, ist die allgemeine Form der Faltung identisch zur zyklischen Faltung. Die zyklische Faltung liefert genau eine Periode des durch Faltung zweier periodischer Signale entstandenen periodischen Signals. Da die DFT ausschliesslich periodische Signale betrachtet, gilt der Faltungssatz der DFT bezüglich der zyklischen Faltung und nicht bezüglich der allgemeinen Faltungssumme, auch wenn man nur endliche lange Signale betrachtet! Die durch Anwendung des Faltungssatzes realisierte zyklische Faltung und die allgemeine Faltungssumme ergeben daher i.a. unterschiedliche Ergebnisse (siehe Übungen).

Anmerkung: Die allgemeine Form der Faltungssumme wird oft als mathematische Definition eines Filters verwendet. Hier wird auch die Bezeichnung von h als Impulsantwort klar: Benutzt man als Eingangssignal einen Puls ($x(1)$=1, $x(n)$=0 sonst), so ist die Ausgangsfunktion y identisch zu h.

8.4.2 Entfaltung

Als **Entfaltung** bezeichnet man die Umkehrung einer Filterung, z.B. wenn man die Filterwirkung eines bei der Meßwertaufnahme wirkenden Filters (z.B. frequenzabhängige Änderung der Empfindlichkeit eines Meßfühlers) rechnerisch rückgängig machen will. Es ist also eine Impulsantwort h^* gesucht, die die Wirkung eines bekannten Filters (Impulsantwort h, Übertragungsfunktion H) genau aufhebt:

$$\begin{aligned} sei: \quad & y(n) = (x \times h)(n) \\ gesucht: \quad & h^*(n) \\ mit: \quad & x(n) = \left(y \times h^*\right)(n) = \left(x \times h \times h^*\right)(n) \end{aligned}$$

Mit Hilfe des Faltungssatzes läßt sich h^* einfach bestimmen:

$$\begin{aligned} & DFT(y) = DFT(x) \cdot DFT(h) \\ \Rightarrow \quad & DFT(x) = DFT(y) \cdot \frac{1}{DFT(h)} \\ \Rightarrow \quad & x = IDFT\left(DFT(y) \cdot \frac{1}{DFT(h)}\right), \\ & = y \times IDFT\left(\frac{1}{DFT(h)}\right) \\ \Rightarrow \quad & h^* = IDFT\left(\frac{1}{DFT(h)}\right) \end{aligned}$$

Man erkennt an dieser Gleichung, daß die Entfaltung formal sehr einfach ist: Im Frequenzbereich teilt man zur Entfaltung einfach durch die Übertragungsfunktion H, die damit wegfällt. Im Zeitbereich entspricht dies einer Faltung mit der inversen DFT der inversen Übertragungsfunktion $1/H$. Jedoch ist das Teilen durch H numerisch sehr problematisch wenn, wie in der Praxis oft gegeben, der Wert von H für eine oder mehr Frequenzen gegen 0 geht. Spezielle Entfaltungsmethoden, die hier nicht behandelt werden sollen, dienen genau dazu, diese Problematik zu lösen.

8.5 Aufgaben

1. Generiere die verschiedenen unten angegebenen Signale mit der Abtastfrequenz f_s = 10 kHz und der Dauer von 10 ms (entsprechend 100 Abtastwerte). Berechne jeweils das Leistungsspektrum mit Hilfe der FFT (Matlab-Funktion fft) und plotte 4 Perioden davon (Frequenzachse -2 f_s bis +2 f_s). Plotte dazu auch jeweils 4 Perioden der Zeitsignale (Zeitachse –20 ms bis +20 ms). Wie ergeben sich die sichtbaren Unterschiede in den Spektren (qualitative Erklärung anhand der Zeitfunktionen).

   (a) Sinus der Frequenz 1 kHz.
   (b) Sinus der Frequenz 925 Hz
   (c) Sinus der Frequenz 1050 Hz

2. Zum Aliasing-Phänomen: Generiere einen Sinus der Frequenz 1 kHz mit den Abtastfrequenzen f_s = 1.5, 1.7, 1.9 und 2.1 kHz und der Dauer von jeweils 50 Perioden (50 ms). Berechne jeweils das Leistungsspektrum und plotte 4 Perioden davon (Frequenzachse -2 f_s bis +2 f_s). Identifiziere die Aliasing-Komponente. Nach welcher Formel kann deren Frequenz berechnet werden?

3. Zum Faltungssatz: Es seien folgende abgetastete Funktionen definiert:

   $$x1 = \{0, 0, 0, 1, 0, 0, 0, 0, 0, 0\}$$
   $$x2 = \{1, 2, 3, 4, 5, 0, 0, 0, 0, 0\}$$
   $$x3 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

   Berechne die Faltung $x2 * x1$ und $x3 * x1$ durch Einsetzen in die Faltungssumme und durch Anwendung des Faltungsatzes der DFT (Hinweis: Matlab hat die eingebauten Funktionen fft und ifft für die diskrete Fourier-Transformation). Was fällt an dem Ergebnis auf und wie läßt sich dies erklären?

9 Partielle Differentialgleichungen

Während gewöhnliche Differentialgleichungen (DGL'n) Funktionen einer Veränderlicher beschreiben, etwa $x=x(t)$, behandeln partielle DGL'n Funktionen mehrerer Veränderlicher, etwa die Temperatur als Funktion von Ort und Zeit, $T = T(x,t)$. Partielle DGL'n werden allgemein in 3 Typen unterteilt, wobei es jedoch keine allgemeingültigen numerischen Lösungsverfahren dafür gibt, wie etwa das Runge-Kutta-Verfahren bei den gewöhnlichen DGL'n. Im folgenden sollen daher wichtige Beispiele der 3 Typen numerisch behandelt werden.

9.1 Klassifikation partieller Differentialgleichungen

9.1.1 Parabolische Gleichungen

Parabolische DGL'n beschreiben im weitesten Sinne Diffusionsprozesse. Beispiele sind die **Wärmeleitungsgleichung**

$$\frac{\partial}{\partial t} T(x,t) = \kappa \frac{\partial}{\partial x^2} T(x,t) \; .$$

(mit: T die Temperatur, x und t Orts- und Zeitvariable, sowie κ Wärmeleitungskoeffizient) sowie aus der Quantenmechanik die **zeitabhängige Schrödingergleichung**

$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = H \Psi(x,t)$$

mit dem **Hamilton-Operator** H.

9.1.2 Hyperbolische Gleichungen

Hyperbolische DGL'n beschreiben Transportprozesse. Beispiele sind die **Advektionsgleichung**

$$\frac{\partial a(x,t)}{\partial t} = -c \frac{\partial a(x,t)}{\partial x}$$

und die **Wellengleichung**

$$\frac{\partial^2 A(x,t)}{\partial t^2} = c^2 \frac{\partial^2 A(x,t)}{\partial x^2} \; .$$

Dabei ist A die Amplitude der Welle, x und t Orts- und Zeitvariable, sowie c die Ausbreitungsgeschwindigkeit.

9.1.3 Elliptische Gleichungen

Der Prototyp einer elliptische DGL ist die **Poisson-Gleichung** der Elektrostatik, die das Potential einer Ladungsverteilung beschreibt:

$$\frac{\partial^2 \Phi(x,y)}{\partial x^2} + \frac{\partial^2 \Phi(x,y)}{\partial y^2} = -\frac{1}{\varepsilon_0} \rho(x,y)$$

Dabei ist Φ das Potential, x und y Ortsvariablen und ρ die Ladungsdichte-Verteilung. Ist die Ladungsdichte identisch Null, so heißt die Gleichung **Laplace-Gleichung**.

Anmerkung: Alle Gleichungen sind hier der Übersicht halber ein- bzw. zweidimensional geschrieben. Formulierung in mehreren Dimensionen ist möglich (siehe Textbücher).

Anmerkung: Die hier angegebenen Gleichungen sind Prototypen der 3 Klassen. In realen Problemen tauchen oft Mischungen auf, z.B. Wellengleichung mit Dämpfung.

9.2 Anfangswertprobleme

Als Anfangswertprobleme behandelt man Gleichungen, die sowohl eine Orts-, als auch eine Zeitabhängigkeit aufweisen. Dabei muß, wie bei den gewöhnlichen DGL'n. der Zustand zur Zeit $t = 0$ bekannt sein. Für die Wärmeleitungsgleichung ist dies z.B.

$$T_0 = T(x, t = 0)$$

und für die Wellengleichung (2 Integrationskonstanten, da Gleichung 2. Ordnung)

$$A_0 = A(x, t = 0) \text{ und } A_0' = \frac{dA(x, t = 0)}{dx} .$$

$T(x,t)$ bzw $A(x,t)$ für $t>0$ ist dann zu berechnen. Die Festlegung des Anfangszustandes ist jedoch zur Lösung nicht ausreichend. Zusätzlich müssen **Randbedingungen** formuliert werden. Dazu wird die Lösung z.B. auf den Bereich der x-Werte

$$x \in \left[-L/2, L/2\right]$$

mit beliebigem, festen L eingeschränkt. Es gibt nun mehrere Möglichkeiten, die Lösung auf dem Rand einzuschränken (hier angegeben für die Wärmeleitungsgleichung):

1. Feste Werte der Lösung auf dem Rand (Dirichlet-Randbedingung):

$$T\left(x = -L/2, t\right) = T_a \quad , \quad T\left(x = L/2, t\right) = T_b$$

2. Periodische Randbedingungen:

$$T\left(x = -L/2, t\right) = T\left(x = L/2, t\right)$$

oder

$$\left.\frac{dT}{dx}\right|_{x=-L/2} = \left.\frac{dT}{dx}\right|_{x=L/2}$$

Die Lösung wird also im Inneren eines von t=0 und x=± L/2 begrenzten Gebiets gesucht. Warum die Lösung auf dem Rand eingeschränkt werden muß, wird später anhand der Lösungsalgorithmen klar.

9.2.1 Diskretisierung

Für die numerische Bearbeitung werden sowohl Zeit- als auch Ortsvariable durch Abtastung diskretisiert. Die Abtastzeitpunkte sind

$$t_n := n \cdot \tau \quad ; \quad n = 0, 1, 2, 3, 4, \ldots$$

mit der Abtastperiode/Schrittweite τ. Der Bereich x=± L/2 wird in N Intervalle geteilt, d.h. die Abtastperiode/Schrittweite der Ortsvariablen ist

$$h = \frac{L}{N}$$

und die Abtastpunkte (Anzahl: N+1) sind

$$x_i := i \cdot h - L/2 \quad ; \quad i = 0, 1, 2, \ldots, N .$$

Die Lösung wird dann für die Variablenpaare (x_i , t_n) gesucht, d.h. die Lösungs-Punkte

$$T_i^n := T\left(x_i, t_n\right)$$

werden gesucht (hier für die Wärmeleitungsleichung geschrieben). Mit diesen Definitionen der Abtastzeitpunkte und –orte werden im folgenden die verschiedenen Lösungsmethoden beschrieben.

9.2.2 Wärmeleitungsgleichung: FTCS-Schema

Das FTCS-Schema (Forward-Time-Centered-Space) nutzt die rechtsseitige Ableitung zur Diskretisierung der Zeitvariablen (Forward Time) und die zentrierte Ableitung für die Ortsvariable. Für die Wärmeleitungsgleichung bedeutet dies:

$$\frac{\partial T(x,t)}{\partial t} \to \frac{T\left(x_i, t_n + \tau\right) - T\left(x_i, t_n\right)}{\tau} = \frac{T_i^{n+1} - T_i^n}{\tau}$$
$$\frac{\partial^2 T(x,t)}{\partial x^2} \to \frac{T\left(x_i + h, t_n\right) + T\left(x_i - h, t_n\right) - 2 \cdot T\left(x_i, t_n\right)}{h^2} = \frac{T_{i+1}^n + T_{i-1}^n - 2 \cdot T_i^n}{h^2}$$

.

Einsetzen in die Wärmeleitungsgleichung liefert

$$\boxed{\begin{aligned} & \frac{T_i^{n+1} - T_i^n}{\tau} = \kappa \cdot \frac{T_{i+1}^n + T_{i-1}^n - 2 \cdot T_i^n}{h^2} \\ \Rightarrow \quad & T_i^{n+1} = T_i^n + \frac{\kappa\tau}{h^2}\left(T_{i+1}^n + T_{i-1}^n - 2 \cdot T_i^n\right) \end{aligned}}.$$

Aus den Anfangswerten T_i^0 und den Randwerten T_0^n und T_N^n können mit dieser Formel iterativ die Lösungswerte T_i^n berechnet werden.

Anmerkung: Aufgrund der zentrierten Ableitung ist die Vorgabe der Randwerte notwendig!

Die Schrittgrößen τ und *h* müssen gemäß der im physikalischen Problem auftauchenden Längen- und Zeitskalen gewählt werden. Hierbei ist insbesondere das Verhältnis beider Größen wichtig, denn es widerspricht der physikalischen Intuition, daß die Lösung mit einem großen Zeitschritt τ aus eng benachbarten Raumpunkten (*h* relativ klein) berechnet werden kann. Der maximale Zeitschritt beträgt bei gegebenem *h* (ohne Beweis):

$$\tau_{\max} = \frac{h^2}{2\kappa}$$

Für diese Gleichung läßt sich eine anschauliche Begründung geben (siehe Übungen), oder sie läßt sich mit Hilfe der von-Neumann-Stabilitätsanalyse (siehe Abschnitt 9.2.5) ableiten.

9.2.3 Zeitabhängige Schrödingergleichung: Implizites Schema (Crank-Nicholson)

Die **zeitabhängige Schrödingergleich**ung

$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = H \Psi(x,t)$$

mit der Wellenfunktion Ψ hat für ein Teilchen der Masse *m* und das Potential *V* den **Hamilton-Operator**

$$H = -\frac{\hbar^2}{2m} \frac{\partial}{\partial x^2} + V(x) .$$

Das Betragsquadrat der Wellenfunktion beschreibt die Aufenthaltwahrwscheinlichkeitsdichte

$$P(x,t) = \left|\Psi(x,t)\right|^2 .$$

Mit dem FTCS-Schema lautet die Schrödingergleichung diskretisiert:

$$i\hbar\frac{\Psi_j^{n+1}-\Psi_j^n}{\tau}=-\frac{\hbar^2}{2m}\frac{\Psi_{j+1}^n+\Psi_{j-1}^n-2\Psi_j^n}{h^2}+V_j\cdot\Psi_j^n$$
$$\Leftrightarrow\quad \Psi_j^{n+1}=\Psi_j^n-\frac{i\tau}{\hbar}\left(-\frac{\hbar^2}{2m}\frac{\Psi_{j+1}^n+\Psi_{j-1}^n-2\Psi_j^n}{h^2}+V_j\cdot\Psi_j^n\right)$$

Dabei wurde für Zeit- und Ortsindex die oben angegebene Notation verwendet. Schreibt man die Wellenfunktion als zeitabhängigen Spaltenvektor

$$\underline{\Psi}^n:=\begin{pmatrix}\Psi_1^n\\ \vdots\\ \Psi_N^n\end{pmatrix}$$

, so kann die diskretisierte Gleichung als Matrixgleichung geschrieben werden, die das Schema für alle Abtastpunkte des Ortes gleichzeitig löst:

$$\begin{aligned}&\underline{\Psi}^{n+1}=\underline{\Psi}^n-\frac{i\tau}{\hbar}\underline{\underline{H}}\cdot\underline{\Psi}^n\\ \Leftrightarrow\quad &\underline{\Psi}^{n+1}=\left(\underline{\underline{1}}-\frac{i\tau}{\hbar}\underline{\underline{H}}\right)\cdot\underline{\Psi}^n\end{aligned}\qquad (*).$$

Dabei ist $\underline{\underline{H}}$ die Matrix des diskretisierten Hamiltonoperators mit den Komponenten

$$H_{i,j}=-\frac{\hbar^2}{2m}\frac{\delta_{i,j+1}+\delta_{i,j-1}-2\delta_{i,j}}{h^2}+V_i\delta_{i,j}\ .$$

Die Matrixgleichung (*) löst die Schrödingergleichung nach dem FTCS-Schema. Da man solche Matrix-Formulierungen in der Literatur oft findet, sei diese Form hier besonders hervorgehoben.

Ausgehend von der Gleichung (*) wurde die Crank-Nicholson-Methode als wichtigster Typ der sogenannten **impliziten Schemata** entwickelt. Dabei wird auf der rechten Seite der Gleichung nicht nur der alte Wert von $\underline{\Psi}$, sondern der Mittelwert aus altem und neuem Wert eingesetzt:

$$\begin{aligned}&\underline{\Psi}^{n+1}=\underline{\Psi}^{n+1}-\frac{i\tau}{2\hbar}\underline{\underline{H}}\cdot\left(\underline{\Psi}^{n+1}+\underline{\Psi}^n\right)\\ \Leftrightarrow\quad &\left(\underline{\underline{1}}+\frac{i\tau}{2\hbar}\underline{\underline{H}}\right)\underline{\Psi}^{n+1}=\left(\underline{\underline{1}}-\frac{i\tau}{2\hbar}\underline{\underline{H}}\right)\underline{\Psi}^n\end{aligned}\ .$$

Auflösen nach $\underline{\Psi}^{n+1}$ ergibt dann das Crank-Nicholson-Schema:

$$\boxed{\underline{\Psi}^{n+1}=\left(\underline{\underline{1}}+\frac{i\tau}{2\hbar}\underline{\underline{H}}\right)^{-1}\left(\underline{\underline{1}}-\frac{i\tau}{2\hbar}\underline{\underline{H}}\right)\underline{\Psi}^n}.$$

Dieses Schema hat gegenüber dem FTCS-Schema die Vorteile, daß es immer stabil ist und daß die angewandten Matrixoperatoren unitär sind (ohne Beweis).

9.2.3.1 Wellenpaket eines freien Teilchens

Als Beispiel für eine Lösung der Schrödingergleichung soll ein **Gaussches Wellenpaket** betrachtet werden, das in der Quantenmechanik zur Beschreibung eines freien Teilchens Verwendung findet. Als Anfangswert für die Wellenfunktion verwendet man dabei

$$\Psi(x,t=0)=\frac{1}{\sqrt{\sigma_0\sqrt{\pi}}}\exp\left(ik_0x\right)\exp\left(-\left(x-x_0\right)^2\Big/2\sigma_0^2\right)\ .$$

Dabei ist x_0 die aktuelle mittlere Position des Teilchens, σ_0 die aktuelle Breite des Paketes und $p_0=\hbar\cdot k_0$ der aktuelle Impuls des Teilchens. Mit dem Crank-Nicholson-Schema kann aus diesem Anfangszustand die Zeitentwicklung der Wellenfunktion berechnet werden. Aus der Quantenmechanik ist bekannt, daß für das Wellenpaket auch eine analytische Lösung existiert. Sie lautet:

$$\Psi(x,t)=\frac{1}{\sqrt{\sigma_0\sqrt{\pi}}}\frac{\sigma_0}{\alpha}\exp\left(ik_0\left(x-p_0t/2m\right)\right)\exp\left(-\left(x-x_0-p_0t/2m\right)^2\Big/2\alpha^2\right)\ .$$

$$mit: \qquad \alpha^2=\sigma_0^2+i\hbar t/m$$

In der Zeit bleibt also die Form des Wellenpakets erhalten. Der Mittelwert bewegt sich dabei mit der Geschwindigkeit p_0/m und Die Standardabweichung verbreitert sich gemäß

$$\sigma(t)=\sigma_0\sqrt{\left(\frac{|\alpha|}{\sigma_0}\right)^4}=\sigma_0\sqrt{1+\frac{\hbar^2t^2}{m^2\sigma_0^4}}\ .$$

Diese analytische Lösung kann verwendet werden, um das Crank-Nicholson-Schema zu testen (siehe Übungen).

9.2.4 Advektionsgleichung: Lax-Wendroff-Schema

Wir gehen von der Wellengleichung

$$\frac{\partial^2A(x,t)}{\partial t^2}=c^2\cdot\frac{\partial^2A(x,t)}{\partial x^2}\ .$$

aus. Wie bei den gew. DGL'n schreiben wir diese Gleichung 2. Ordnung in 2 Gleichungen 1. Ordnung um. Wir definieren dazu die Zwischenvariablen

$$p=\frac{\partial A}{\partial t}\quad;\quad q=c\cdot\frac{\partial A}{\partial x}\ .$$

Damit können wir die Wellengleichung als Paar von Gleichungen

$$\frac{\partial p}{\partial t}=c\cdot\frac{\partial q}{\partial x}\quad;\quad\frac{\partial q}{\partial t}=c\cdot\frac{\partial p}{\partial x}$$

schreiben. Dabei ist die erste Gleichung durch Einsetzen der Variablen in die Wellengleichung entstanden. Die zweite Gleichung ergibt sich durch Vergleich der gemischten Ableitungen von *p* nach *x* und *q* nach *t*. Diese beiden Gleichungen lassen sich wieder in Vektorschreibweise schreiben:

$$\frac{\partial \underline{a}(x,t)}{\partial t} = c \cdot \underline{\underline{B}} \cdot \frac{\partial \underline{a}(x,t)}{\partial x}$$

$$mit: \quad \underline{a}(x,t) = \begin{pmatrix} p(x,t) \\ q(x,t) \end{pmatrix} \qquad ; \qquad \underline{\underline{B}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Die Wellengleichung ist nicht die einfachste Gleichung dieser Form. Wenn man

$$\underline{\underline{B}} = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$$

setzt, so ergibt sich die sog. **Advektionsgleichung**[12]

$$\boxed{\frac{\partial a(x,t)}{\partial t} = -c \frac{\partial a(x,t)}{\partial x}},$$

wobei *a* die gesuchte Größe ist, die man sich als Amplitude einer Welle vorstellen kann. Im folgenden betrachten wir diese einfachste Form einer hyperbolischen Gleichung. Analytisch läßt sich leicht durch Einsetzen zeigen, daß jede Funktion der Form:

$$a(x,t) = f_0(x - ct)$$

mit einer beliebigen Funktion f_0 Lösung der Advektionsgleichung ist, wobei *c* eine Ausbreitungsgeschwindigkeit darstellt. Beispielsweise ist ein **Cosinus-modulierter Gaußpuls**

$$a(x,t) = \cos\left(\frac{2\pi}{\lambda}(x - c \cdot t)\right) \cdot \exp\left(-\frac{(x - c \cdot t)^2}{2\sigma^2}\right)$$

eine Lösung. Die Funktion stellt ein kurzes Wellenpaket der Wellenlänge λ und der räumlichen Ausdehnung σ dar, das sich mit der Geschwindigkeit *c* in *x*-Richtung bewegt.

Da im Fall der Advektionsgleichung die analytische Lösung bekannt ist, kann sie zum Test von numerischen Lösungsverfahren für hyperbolische Gleichungen verwendet werden. Im folgenden wird dazu das Lax-Wendroff-Schema vorgestellt.

Anmerkung: Die Advektionsgleichung stellt die einfachste Form einer **Erhaltungsgleichung**

$$\frac{\partial \underline{a}}{\partial t} = -grad\left(\underline{F}(\underline{a})\right) \quad \text{oder 1-D:} \quad \frac{\partial a}{\partial t} = -\frac{\partial F(a)}{\partial x}$$

Dabei kann $\underline{a}$ etwa die Massen-, Impuls- oder Energiedichte und das zugehörige $\underline{F}(\underline{a})$ der zugehörige Massen-, Impuls- oder Energiefluß sein. Als Beispiel sei die Kontinuitätsgleichung für die Masse in einer Strömung angegeben (1-Dimensional):

$$\frac{\partial \rho(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left(\rho(x,t) \cdot v(x,t)\right).$$

Dabei ist ρ die Massendichte und *v* die Strömungsgeschwindigkeit.

[12] Advektion: horizontale Heranführung von Luftmassen oder horizontale Verfrachtung von Wassermassen im Meer (lt. Wörterbuch).

9.2.4.1 Lax-Wendroff-Schema

Die Diskretisierung der Advektionsgleichung kann theoretisch wieder nach dem FTCS-Schema erfolgen, jedoch führt dies nicht zu einer stabilen Lösung (siehe Abschnitt 9.2.5 und Übungen). Das Lax-Wendroff-Schema ist höherer Ordnung und stellt ein stabiles Lösungsverfahren für die Advektionsgleichung dar, wenngleich es auch nicht ohne numerische Abweichungen ist (siehe Übungen). Man geht dazu von der Taylor-Entwicklung 2. Ordnung für die Zeitentwicklung aus:

$$a(x,t+\tau) \approx a(x,t) + \tau \cdot \frac{\partial a(x,t)}{\partial t} + \frac{\tau^2}{2} \cdot \frac{\partial^2 a(x,t)}{\partial t^2}$$

Dann wird aus der Advektionsgleichung die zweite Ableitung nach der Zeit bestimmt:

$$\frac{\partial a}{\partial t} = -\frac{\partial F(a)}{\partial x} \qquad mit: \quad F(a) = c \cdot a$$

$$\Rightarrow \quad \frac{\partial^2 a}{\partial t^2} = -\frac{\partial}{\partial t}\frac{\partial F}{\partial x} = -\frac{\partial}{\partial x}\frac{\partial F}{\partial t}$$

Mit

$$\frac{\partial F(a)}{\partial t} = \frac{dF}{da} \cdot \frac{\partial a}{\partial t} = F'(a) \cdot \frac{\partial a}{\partial t} = -F'(a) \cdot \frac{\partial F(a)}{\partial x}$$

ergibt sich durch Einsetzen folgende Form der zweiten Ableitung

$$\frac{\partial^2 a}{\partial t^2} = \frac{\partial}{\partial x} F'(a) \frac{\partial F(a)}{\partial x}$$

und damit für die Taylorentwicklung:

$$a(x,t+\tau) \approx a(x,t) - \tau \cdot \left(\frac{\partial}{\partial x} F\left(a(x,t)\right) \right) + \frac{\tau^2}{2} \cdot \left(\frac{\partial}{\partial x} F'\left(a(x,t)\right) \frac{\partial F\left(a(x,t)\right)}{\partial x} \right)$$

Diese Formel erlaubt wieder die Berechnung eines Zustandes zur Zeit $t+\tau$ aus Zuständen zur Zeit t. Sie wird nun diskretisiert (Nomenklatur der Indizees wie bei der Wärmeleitungsgleichung). Dazu wird für die erste Ableitung der Zeit die zentrierte Formel verwendet:

$$\left. \frac{\partial}{\partial x} F(a) \right|_{x_i, t_n} \quad \rightarrow \quad \frac{F_{i+1}^n - F_{i-1}^n}{2h}$$

Die Berechnung der mehrfachen Ableitung im letzten Term ist recht komplex. Die äußere Ableitung ist eine rechtseitige Formel, ebenso die innerste Ableitung. Die Ableitung F' wird aus dem Mittelwert der in jeweils in den beiden Termen der äußeren rechtsseitigen Ableitung auftretenden Amplitudenwerten berechnet:

$$\frac{\partial}{\partial x}\left(F'(a)\frac{\partial F(a)}{\partial x}\right)\Big|_{x_i,t_n} \rightarrow \frac{\left(F'(a)\frac{\partial F(a)}{\partial x}\right)\Big|_{x_{i+1},t_n} - \left(F'(a)\frac{\partial F(a)}{\partial x}\right)\Big|_{x_i,t_n}}{h}$$

$$\left(F'(a)\frac{\partial F(a)}{\partial x}\right)\Big|_{x_{i+1},t_n} \rightarrow F'\left[\left(a_{i+1}^n + a_i^n\right)/2\right]\cdot\frac{F_{i+1}^n - F_i^n}{h}$$

$$\left(F'(a)\frac{\partial F(a)}{\partial x}\right)\Big|_{x_i,t_n} \rightarrow F'\left[\left(a_i^n + a_{i-1}^n\right)/2\right]\cdot\frac{F_i^n - F_{i-1}^n}{h}$$

Insgesamt ergibt sich damit:

$$a_i^{n+1} = a_i^n - \tau\cdot\frac{F_{i+1}^n - F_{i-1}^n}{2h} + \frac{\tau^2}{2}\frac{1}{h}\cdot\left(F'^n_{i+1/2}\cdot\frac{F_{i+1}^n - F_i^n}{h} - F'^n_{i-1/2}\cdot\frac{F_i^n - F_{i-1}^n}{h}\right)$$

mit

$$F_i^n \equiv F\left(a_i^n\right) \quad und \quad F'^n_{i\pm1/2} \equiv F'\left[\left(a_{i\pm1}^n + a_i^n\right)/2\right].$$

Für die Advektionsgleichung mit

$$F(a) = c\cdot a$$

$$\Rightarrow \quad F_i^n = c\cdot a_i^n \quad und \quad F'^n_{i\pm1/2} = c$$

vereinfacht sich dies Form wesentlich zu

$$\boxed{a_i^{n+1} = a_i^n - \frac{c\tau}{2h}\left(a_{i+1}^n - a_{i-1}^n\right) + \frac{c^2\tau^2}{2h^2}\left(a_{i+1}^n + a_{i-1}^n - 2a_i^n\right)}$$

Hiermit kann, wie bei dem FTCS-Schema, aus drei benachbarten Punkten zu einer Zeit t ein Punkt zur Zeit $t+\tau$ berechnet werden. Im Gegensatz zum FTCS-Schema ist es jedoch stabil, sofern der Zeitschritt maximal τ_{max}, mit

$$\tau_{\max} = \frac{h}{c}$$

ist. Der Beweis ist durch von-Neumann-Stabilitätsanalyse (siehe Abschnitt 9.2.5) möglich (siehe dazu auch die Übungen).

9.2.5 von-Neumann-Stabilitätsanalyse

Die von-Neumann-Stabilitätsanalyse dient der Bestimmung von Stabilitätsregeln für die verschiedenen Lösungsschemata für partielle DGLn. Man geht dazu von einer kleinen Störung aus, die man in die Iterationsgleichungen „injiziert“ und schaut nach, ob diese Störung divergiert (instabiles Schema) oder konvergiert („Dämpfung“ der Störung, stabiles Schema). Wählt man die Störung geschickt, so kann dies analytisch entschieden werden, d.h. ohne tatsächliche numerische Iteration der Gleichung. Speziell verwendet man bei der von-Neumann-Analyse folgende Störung, die eine raumfeste Welle mit zeitabhängiger Amplitude darstellt:

$$a_j^n = \xi^n \cdot e^{i \cdot k \cdot h \cdot j}$$

$$mit:$$

$$\xi : Verstärkungsfaktor$$

$$k = \frac{2\pi}{\lambda} : Wellenzahl\ (beliebig\ wählbar)$$

$$h : Schrittweite\ im\ Ortsbereich$$

$$n : Zeitindex\ (linke\ Seite) / Exponent\ (rechte\ Seite)$$

$$j : Ortsindex$$

$$i : komplexe\ Zahl\ i$$

Die Ortsabhängigkeit (komplexe *e*-Funktion) und die Zeitabhängigkeit (Potenzfunktion) sind hier separiert, d.h. sind als Produkt dargestellt (**Anmerkungen**: Der Index *n* bedeutet auf der linken Seite der Gleichung den Index der Zeit (Diskretisierung) und auf der rechten Seite die Potenz der Amplitude. Um eine Verwechslung mit der komplexen Zahl *i* zu vermeiden, wurde hier *j* als Zeitindex verwendet.). Die Amplitude ξ wird auch als **Verstärkungsfaktor** bezeichnet. Ist der Verstärkungsfaktor für eine Iterationsgleichung größer als 1, ist sie instabil. Stabil ist sie nur für Werte kleiner gleich 1. Die Berechnung von ξ soll nun anhand des FCTS-Schemas für die Advektionsgleichung exemplarisch gezeigt werden.

9.2.5.1 Stabilität des FTCS-Schemas für die Advektionsgleichung

Das FTCS-Schema lautet für die Advektionsgleichung (Einsetzen der rechtsseitigen bzw. zentrierten Formel für Zeit bzw. Ort):

$$a_j^{n+1} = a_j^n - \frac{c\tau}{2h}\left(a_{j+1}^n - a_{j-1}^n\right).$$

Einsetzen der Störung ergibt

$$\begin{aligned} \xi^{n+1} \cdot e^{ikhj} &= \xi^n \cdot e^{ikhj} - \frac{c\tau}{2h}\left(\xi^n \cdot e^{ikh(j+1)} - \xi^n \cdot e^{ikh(j-1)}\right) \\ &= \xi^n \cdot e^{ikhj}\left(1 - \frac{c\tau}{2h}\left(e^{ikh} - e^{-ikh}\right)\right) \end{aligned}.$$

Teilen beider Seiten durch $\xi^n \cdot e^{ikhj}$ ergibt direkt einen Ausdruck für den Verstärkungsfaktor:

$$\begin{aligned} \xi &= 1 - \frac{c\tau}{2h}\left(e^{ikh} - e^{-ikh}\right) \\ &= 1 - i\frac{c\tau}{h}\sin(kh) \end{aligned}$$

Der Betrag des Verstärkungsfaktors ist dann:

$$|\xi| = \sqrt{1 + \left(\frac{c\tau}{h}\right)^2 \sin^2(kh)}$$

Dieser Ausdruck ist unabhängig von der Wahl des Zeitschritts immer größer als 1. Hiermit ist analytisch gezeigt, daß das Schema instabil ist, ohne die Gleichung tatsächlich numerisch iterieren zu müssen. Dies ist durch den geschickten Ansatz der Störung möglich geworden. Zur Anwendung der von-Neumann-Stabilitätsanalyse auf das Lax-Wendroff-Schema siehe die Übungen.

9.3 Randwertprobleme: Poisson- und Laplacegleichung

Zeitunabhängige Gleichungen werden als **Randwertprobleme** beschrieben. Dazu zählen z.B. die Poisson- und Laplacegleichung der Elektrostatik. Hier muß als Anfangsbedingung die Lösung auf dem Rand eines Gebiets vorgegeben werden. Für 2 Dimensionen kann dies ein Rechteck der Kantenlängen L_x und L_y sein[13]. Die x-Variable läuft dann von x=0 bis x= L_x und die y-Variable von 0 bis L_y. Die Randbedingungen lauten

$$\Phi(x=0,y)=\Phi_1 \quad \Phi(x=L_x,y)=\Phi_2$$
$$\Phi(x,y=0)=\Phi_3 \quad \Phi(x,y=L_y)=\Phi_4$$

(Dirichlet-Randbedingung) bzw.

$$\Phi(x=0,y)=\Phi(x=L_x,y)$$
$$\Phi(x,y=0)=\Phi(x,y=L_y)$$

(periodische Randbedingungen). Die Lösung wird dann im Inneren des Rechtecks gesucht. Für die numerische Lösung werden x- und y-Variable wieder durch Abtastung diskretisiert. Dazu werden N bzw. M Intervalle verwendet:

$$\begin{aligned} h_x &= \frac{L_x}{N}; \quad x_i := i\cdot h_x \quad ; \quad i=0,1,2,\ldots,N \\ h_y &= \frac{L_y}{M}; \quad y_j := j\cdot h_y \quad ; \quad j=0,1,2,\ldots,M \end{aligned} .$$

Die Lösung wird dann für die Variablenpaare (x_i , y_j) gesucht, d.h. die Lösungs-Punkte

$$\Phi_{i,j} := \Phi(x_i,x_j)$$

werden gesucht. Im folgenden werden grundlegende Lösungmethoden für die Laplace- und Poissongleichung untersucht.

9.3.1 Laplacegleichung: Relaxationsmethoden (Jacobi-Methode)

Die Laplacegleichung

$$\frac{\partial^2\Phi(x,y)}{\partial x^2}+\frac{\partial^2\Phi(x,y)}{\partial y^2}=0$$

ist in ihrer räumlichen Ableitung bis auf den Vorfaktor (Wärmeleitungskoeffizient) identisch zur Wärmeleitungsgleichung in 2 Dimensionen

$$\frac{\partial T(x,y,t)}{\partial t}=\kappa\cdot\left(\frac{\partial^2 T(x,y,t)}{\partial x^2}+\frac{\partial^2 T(x,y,t)}{\partial y^2}\right)$$

Die Lösung der Wärmeleitungsgleichung nähert sich für große Zeiten (t gegen unendlich) einem stationären Zustand an (stationäre Temperaturverteilung):

$$\lim_{t\to\infty} T(x,y,t)=T_s(x,y)$$
$$\Leftrightarrow \quad \frac{\partial^2 T_s(x,y)}{\partial x^2}+\frac{\partial^2 T_s(x,y)}{\partial y^2}=0$$

[13] Andere spezielle Lösungen existieren für zylindrische und sphärische Geometrien.

Der stationäre Zustand T_s ist also gleichzeitig eine Lösung der Laplacegleichung. Die Idee der **Relaxationsmethoden** allgemein besteht nun darin, in eine stationäre Gleichung künstlich eine Zeitabhängigkeit einzuführen und diese dann bis zu einem stationären Zustand zu iterieren. Letzerer ist dann die Lösung für die stationäre Gleichung. Für die Laplace-Gleichung heißt dies im einfachsten Fall, daß man, wie oben gezeigt, die Gleichung formal wie die Wärmeleitungsgleichung schreibt und dabei das Potential zeitabhängig macht:

$$\frac{\partial \Phi(x,y,t)}{\partial t} = \mu \cdot \left(\frac{\partial^2 \Phi(x,y,t)}{\partial x^2} + \frac{\partial^2 \Phi(x,y,t)}{\partial y^2} \right)$$

Dabei ist μ ein beliebiger Konvergenzfaktor (entsprechend dem Wärmeleitungskoeffizienten). Nach dem FTCS-Schema kann die Gleichung folgendermaßen diskretisiert werden:

$$\Phi_{i,j}^{n+1} = \Phi_{i,j}^{n} + \frac{\mu\tau}{h_x^{\,2}} \left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n} - 2 \cdot \Phi_{i,j}^{n} \right) + \frac{\mu\tau}{h_y^{\,2}} \left(\Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n} - 2 \cdot \Phi_{i,j}^{n} \right) .$$

Die Nomenklatur für Zeit- und Ortsindex ist wie oben. Beachte jedoch, daß hier das Schema auf beide Ortsdimensionen angewandt wurde. Demnach wird der neue Zustand im Punkt (x_i,y_j) aus den Werten in diesem und allen direkt benachbarten Punkten berechnet.

Die Stabilitätsbedingung für dieses Schema lautet ähnlich wie für das FTCS-Schema der 1-dimensionalen Wärmeleitungsgleichung (ohne Beweis):

$$\frac{\mu\tau}{h_x^{\,2}} + \frac{\mu\tau}{h_y^{\,2}} \leq 1/2 \ .$$

Betrachten wir die Gleichung für den Fall, daß die Schrittweiten h_x und h_y gleich sind, d.h.

$$h_x = h_y = h \ ,$$

so lautet die Stabilitätsbedingung

$$\frac{\mu\tau}{h^2} \leq 1/4 \ .$$

Da nur der stationäre Zustand interessant ist, wird der maximal mögliche Zeitschritt gewählt. Einsetzen von

$$\frac{\mu\tau}{h^2} = 1/4$$

in das Relaxationsschema liefert

$$\boxed{\Phi_{i,j}^{n+1} = \frac{1}{4} \left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n} \right)} .$$

Beachte, daß durch die Wahl des max. Zeitschritts der zentrale Punkt (x_i,y_j) gar nicht in die Berechnung des Potentials in diesem Punkt eingeht, sondern nur die benachbarten Punkte. Dieses Schema ist ein spezielles Relaxationsschema zur Lösung der Laplacegleichung und wird als **Jacobi-Methode** bezeichnet. Ausgehend von der Jacobi-Methode kann durch Variation der Iterationsregeln die Konvergenzgeschwindigkeit erhöht werden. Die sich daraus ergebenden Schemata werden als **Überrelaxationsschemata** bezeichnet. Bekannte Verfahren sind **Gauß-Seidel** und **simultane Überrelaxation**. Da sie auf eher empirische Weise aus der Jacobi-Methode abgeleitet sind, sollen sie hier nicht weiter behandelt werden (siehe Textbücher).

9.3.1.1 Berechnung von Anfangswerten

Die Konvergenz der Relaxationsmethoden hängt von der Wahl der Anfangswerte ab, mit denen die gesuchten Werte im Inneren des Rechtecks initialisiert werden. Sind sie geschickt gewählt, so kann das System mit deutlich weniger Schritten konvergieren. Für die Laplacegleichung ist aus der Literatur bekannt, daß sich ihre Lösung innerhalb eines Rechtecks mit der Methode der **Separation der Variablen** analytisch bestimmen läßt. Typischerweise führt dies auf unendliche Reihen als Lösung, die zwar analytisch genau definiert sind, jedoch i.a. schlecht konvergieren. Eine numerische Lösung ist daher trotz Existenz der analytischen Lösung wünschenswert. Ist nun die unendliche Reihe bekannt, so bietet es sich an, das erste Glied der Reihe als Anfangswert für die Relaxationsmethoden zu verwenden. Als Beispiel sei die Lösung der Laplace-Gleichung auf einem Rechteck mit den Randbedingungen

$$\Phi(x=0,y)=\Phi(x=L_x,y)=\Phi(x,y=0)=0$$
$$\Phi(x,y=L_y)=\Phi_0$$

genannt. Der Ansatz der Separation der Variablen

$$\Phi(x,y)=X(x)\cdot Y(y)$$

führt mit diesen Randbedingungen zu der Lösung (siehe Textbücher)

$$\Phi(x,y)=\Phi_0\cdot\sum_{N=1,3,5,\ldots}^{\infty}\frac{4}{\pi n}\sin\left(\frac{n\pi x}{L_x}\right)\sinh\left(\frac{n\pi y/L_x}{n\pi L_y/L_x}\right).$$

Nimmt man das erste Glied der Reihe als Anfangswert der Jacobi-Methode so erhöht sich die Konvergenzgeschwindigkeit deutlich gegenüber suboptimaler Wahl der Anfangswerte (siehe Übungen).

9.3.2 Poissongleichung

Die Poissongleichung

$$\frac{\partial^2\Phi(x,y)}{\partial x^2}+\frac{\partial^2\Phi(x,y)}{\partial y^2}=\frac{1}{\varepsilon_0}\rho(x,y)$$

hat i.a. für eine beliebige vorgegebene Ladungsverteilung ρ keine analytischen Lösungen durch Separation der Variablen. Im folgenden sollen daher 2 numerische Lösungsmethoden vorgestellt werden. Die Ladungsdichte wird dabei wie das Potential diskretisiert:

$$\rho_{i,j} := \rho\left(x_i, x_j\right)$$

9.3.2.1 Jacobi-Methode

Die Jacobi-Methode läßt sich direkt zur Lösung der Poissongleichung erweitern. Der in Abschnitt 9.3.1 verwendetete Formalismus führt auf die Relaxationsgleichung

$$\boxed{\Phi_{i,j}^{n+1}=\frac{1}{4}\left(\Phi_{i+1,j}^{n}+\Phi_{i-1,j}^{n}+\Phi_{i,j+1}^{n}+\Phi_{i,j-1}^{n}+\frac{1}{\varepsilon_0}h^2\rho_{i,j}\right)}.$$

Da i.a. keine analytischen Lösungen existieren, können die Anfangswerte nicht optimal gewählt werden. Es muß daher mit langsamer Konvergenz gerechnet werden.

9.3.2.2 Methode der multiplen Fouriertransformation

Während bisher nur Dirichlet-Randbedingungen betrachtet wurden, ermöglicht die Methode der multiplen Fouriertransformation die Lösung der Poissongleichung auf einem Quadrat mit periodischen Randbedingungen[14]. Die Anzahl der Abtastwerte N und M sowie die Schrittweiten h_x und h_y in x- und y-Richtung müssen gleich sein:

$$N = M$$
$$h_x = h_y = h$$

Ausgehend davon werden zunächst die partiellen Ableitungen nach dem Ort durch die zentrierte Formel diskretisiert:

$$\left.\frac{\partial^2\Phi}{\partial x^2}\right|_{i,j} \rightarrow \frac{\Phi_{i+1,j}+\Phi_{i-1,j}-2\cdot\Phi_{i,j}}{h^2}$$
$$\left.\frac{\partial^2\Phi}{\partial y^2}\right|_{i,j} \rightarrow \frac{\Phi_{i,j+1}+\Phi_{i,j-1}-2\cdot\Phi_{i,j}}{h^2}$$

Einsetzen in die Poissongleichung liefert damit:

$$\frac{1}{h^2}\left(\Phi_{i+1,j}+\Phi_{i-1,j}+\Phi_{i,j+1}+\Phi_{i,j-1}-4\cdot\Phi_{i,j}\right)=\frac{1}{\varepsilon_0}\rho_{i,j} \quad (*)$$

Wir definieren nun die 2-dimensionale diskrete Fouriertransformation (2-D DFT) des Potentials und der Ladungsdichte als

[14] Da die diskrete Fouriertransformation nur periodische Funktionen betrachtet, ist es nicht weiter überraschend, daß diese Methode periodische Randbedingungen verwendet.

$$F_{m,n} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \Phi_{j,k} \cdot W_N^{jm} \cdot W_N^{kn} \quad ; \quad m,n = 0,\ldots,N-1$$

$$R_{m,n} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \rho_{j,k} \cdot W_N^{jm} \cdot W_N^{kn} \quad ; \quad m,n = 0,\ldots,N-1$$

$$mit: W_N = \exp\left(-\frac{2\pi i}{N}\right)$$

(vergleiche dazu die 1-dimensionale DFT). Die Gleichung (*) wird nun auf beiden Seiten Fourier-transformiert. Damit ergibt sich:

$$\frac{1}{h^2}\left(W_N^{-m} + W_N^{m} + W_N^{-n} + W_N^{n} - 4\right) \cdot F_{m,n} = -\frac{1}{\varepsilon_0} R_{m,n}$$

$$\Leftrightarrow \quad \left(2\cos\left(\frac{2\pi m}{N}\right) + 2\cos\left(\frac{2\pi n}{N}\right) - 4\right) \cdot F_{m,n} = -\frac{h^2}{\varepsilon_0} R_{m,n} \qquad (**)$$

Dabei wurde verwendet, daß die DFT linear ist und daß der **Verschiebungssatz** gilt (Beweis siehe Anmerkung unten):

$$Verschiebung: \quad \Phi'_{j,k} := \Phi_{j+l,k}$$

$$\Rightarrow \quad F'_{m,n} = W_N^{-lm} \cdot F_{m,n}$$

Eine Verschiebung bedeutet also eine Multiplikation der Spektralkomponenten mit einem komplexen Faktor.

Insgesamt können nun die Spektralkomponenten des Potentials aus den (bekannten) Spektralkomponenten der vorgegebenen Ladungsverteilung bestimmt werden, indem man die Gleichung (**) nach den $F_{m,n}$ auflöst:

$$F_{m,n} = P_{m,n} \cdot R_{m,n}$$

$$mit: \quad P_{m,n} = -\frac{h^2}{2\varepsilon_0}\left(\frac{1}{\cos\left(\frac{2\pi m}{N}\right) + \cos\left(\frac{2\pi n}{N}\right) - 2}\right)$$

$$und: \quad \Phi = IDFT(F)$$

Das gesuchte Potential ergibt sich somit durch inverse Fouriertransformation der Spektralkomponenten $F_{m,n}$, die sich wiederum durch Filterung der Spektralkomponenten der Ladungsverteilung mit dem Filter P ergeben.

Anmerkung: Die Bestimmung der Spektralkomponenten $F_{m,n}$ aus den $R_{m,n}$ entspricht einer speziellen Filterung mit der Übertragungsfunktion $P_{m,n}$.

Anmerkung: Zum Beweis des Verschiebungssatzes:

$$\Phi'_{j,k} := \Phi_{j+l,k}$$

$$\Rightarrow \quad F'_{m,n} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \Phi'_{j,k} \cdot W_N^{jm} \cdot W_N^{kn}$$

$$\Leftrightarrow \quad F'_{m,n} = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \Phi_{j+l,k} \cdot W_N^{jm} \cdot W_N^{kn}$$

$$\Leftrightarrow \quad F'_{m,n} = \sum_{j'=l}^{N-1+l}\sum_{k=0}^{N-1} \Phi_{j',k} \cdot W_N^{(j'-l)m} \cdot W_N^{kn}$$

$j' = j + l$

$$\Leftrightarrow \quad F'_{m,n} = W_N^{-lm} \cdot \left(\sum_{j'=l}^{N-1+l}\sum_{k=0}^{N-1} \Phi_{j',k} \cdot W_N^{j'm} \cdot W_N^{kn} \right)$$

$$\Leftrightarrow \quad F'_{m,n} = W_N^{-lm} \cdot F_{m,n}$$

Φ *periodisch mit* N

9.4 Aufgaben

1. Das Skript dftcs löst die Wärmeleitungsgleichung nach dem FTCS-Schema für eine delta-förmige Temperaturverteilung über der x-Achse als Anfangswert. Durch Diffusion „zerfällt“ diese Verteilung mit der Zeit. Überprüfe die Stabilität der Lösung für verschiedene Parametersätze. Verwende dazu N=41 und mehrere Werte des Zeitschritts τ im Bereich 10^{-3} und 10^{-5}. Für $\tau = 2.0\text{x}10^{-4}$ erprobe mehrere verschiedene Werte von N.

2. Zeige durch Einsetzen, daß die Gaußfunktion

$$T(x,t) = \frac{1}{\sigma(t)\sqrt{2\pi}} \exp\left[\frac{-(x-x_0)^2}{2\sigma^2(t)}\right]$$

mit

$$\sigma(t) = \sqrt{2\kappa t}$$

Lösung der Wärmeleitungsgleichung ist. Leite aus der Zeitentwicklung der Varianz σ eine Stabilitätsregel ab (**Tip:** Berechne die Zeitdauer für die Erhöhung der Varianz von 0 auf h, wobei h der Gitterabstand der räumlichen Koordinate ist). Stimmt dies mit den Ergebnissen aus Aufgabe 1 überein?

3. Berechne die Zeitentwicklung der Wellenfunktion eines freien Teilchens (Gaussches Wellenpaket) nach der Crank-Nicholson-Methode mit periodischen Randbedingungen. Wähle dabei folgende Parameter: L=100; N=30 **und** 80; m=1; τ=1; $\hbar = 1$; $\sigma_0 = 1$ und mittlere Geschwindigkeit $p_0/m = \hbar k_0/m = 0.5$.

   Gehe folgendermassen vor:
   - Berechne den Anfangszustand $\underline{\Psi}^0$. Stelle dabei sicher, daß die Randwerte periodisch sind.
   - Berechne die Matrix des Hamiltonoperators. Stelle sicher, daß sie den periodischen Randbedingungen genügt (welche Bedingungen sind dabei an die Matrix zu stellen?).
   - Berechne die Gesamtmatrix zur Berechnung von $\underline{\Psi}^{n+1}$ aus $\underline{\Psi}^n$.
   - Iteriere das Schema, bis das Wellenpaket einmal zirkulär durch das System gelaufen ist (berechne dazu die Anzahl der dazu notwendigen Schritte aus L, τ und der mittleren Geschwindigkeit).
   – Plotte $\underline{\Psi}^0$ (Real- und Imaginärteil) sowie $\left|\underline{\Psi}^n\right|^2$ für alle Zeitpunkte (3D-Plot).

   Inwieweit stimmt die numerische Lösung mit der analytischen Lösung überein? Warum entspricht die numerische Lösung für N=30 nicht der Erwartung (Stichwort: Abtasttheorem)?

4. Das Skript aftcs löst die Advektionsgleichung nach dem FTCS-Schema mit periodischen Randbedingungen und einem Cosinus-modulierten Gauß-Puls als Anfangswert.

   a.) Wende das Skript mit einem Zeitschritt τ=0.002 und N=50 Gitterpunkten an. Woran läßt sich erkennen, daß die damit erhaltene numerische Lösung nicht korrekt ist?
   b.) Löse die Gleichung nach dem Lax-Wendroff-Schema. Modifiziere dazu das Skript entsprechend und beurteile die numerische Lösung. Setze dazu Zeitschritte knapp unterhalb des maximalen stabilen Zeitschritts und den maximalen Zeitschritt selbst ein.

5. Wende die von-Neumann-Stabilitätsanalyse auf das Lax-Wendroff-Schema an. Berechne dazu die Formel für den Betrag des Verstärkungsfaktors $|\xi|$ (schriftliche Rechnung). Leite die Stabilitätsbedingung

$$\tau_{\max} = \frac{h}{c}$$

ab, indem das Maximum von $|\xi|$ für folgende Fälle diskutiert wird:

$$\frac{\tau c}{h} : \begin{cases} < 1 \\ = 1 \\ > 1 \end{cases}$$

Hinweis: Betrachte die Maxima der Werte der einzelnen Terme von $|\xi|$ in Abhängigkeit von dem Argument $x = k \cdot h \cdot j$ der Sinus- und Cosinusfunktion. Zur Veranschaulichung plotte $|\xi|$ als Funktion von x für x im Intervall 0 bis 2π für die drei Fälle.

6. Das Skript jacobi löst die Laplace-Gleichung auf einem Quadrat. Das Potential auf dem Rand ist auf drei Seiten Null und auf einer Seite ($y=L$) 1. Als Anfangswerte für die inneren Punkte wird das erste Glied der unendlichen Reihe verwendet, die die analytische Lösung darstellt (Lösung durch Separation der Variablen). Kritisch ist der Rechenaufwand, der im wesentlichen durch die Anzahl der Gitterpunkte und die Wahl der Anfangswerte bestimmt wird.

   a.) Benutze das Skript für verschiedene räumliche Auflösungen (Anzahl Gitterpunkte N=10 bis 30). Wie stark steigt der Rechenaufwand mit der räumlichen Auflösung? Plotte dazu die Anzahl der zur Konvergenz notwendigen Iterationen über N und passe ein Potenzgesetz an die Daten an. Bestimme den Exponenten.
   b.) Wiederhole a.), jedoch mit schlecht gewählten Anfangswerten (z.B. Potential Φ=0 für alle inneren Punkte.

7. Löse die Poisson-Gleichung

$$\frac{\partial^2 \Phi(x,y)}{\partial x^2} + \frac{\partial^2 \Phi(x,y)}{\partial y^2} = -\frac{1}{\varepsilon_0} \rho(x,y)$$

auf einem Quadrat der Kantenlänge 1 für folgende Ladungsverteilungen:

a. Eine Punktladung der Stärke 1 bei [x,y]=[0.5 0.5].
b. Eine Punktladung der Stärke 1 bei [x,y]=[0.4 0.6] (Was fällt im Vergleich zu a. auf?).
c. Zwei Punktladungen bei [x,y] = [0.5 0.45] und [0.5 0.55] mit den Stärken 1 und -1 (Dipol).

Verwende zur Lösung die Methode der multiplen Fourier-Transformation (Skript fftpoi.m). Vollziehe das Skript soweit wie möglich nach.