Harald Nahrstedt

Algorithmen für Ingenieure

Technische Realisierung mit Excel und VBA

2. Auflage

STUDIUM

Springer Vieweg

Algorithmen für Ingenieure

Leserstimmen

„Viele brauchbare, nützliche und praktische Beispiele. Die Realisierung mit der Standardsoftware MS Office Excel ist hervorragend."

Prof. Dr. Ulrich Schwellenberg, FH Düsseldorf

„Das Buch knüpft an den für Ingenieure bereits bekannten Problemen an, löst sie mit klaren, in Struktogrammen dargestellten Algorithmen und mit Mitteln von Excel und Visual Basic."

Dr. Inge Adamski, TU Dresden

„Mir gefallen die ausführlichen Beispiele mit komplettem Quellcode."

Dr. S. Behrens, BTH Cottbus

„Das Buch zeigt, dass sich viele anspruchsvolle Fragestellungen im Ingenieurwesen mit bekannten Mitteln, wie Visual Basic und Excel, lösen lassen."

Prof. Dr. Karlheinz Tooten, FH Bochum

„Viele praktische Beispiele zum Üben direkt verbunden mit algorithmischen Lösungen. Die Unterschiedlichkeit der Aufgaben und zugehörigen Algorithmen ist auch ein gutes Nachschlagewerk für erfahrene Anwender."

Privatdozent Dr.-Ing. Udo Küppers, Uni Bremen

„Sehr viele Beispiele, gut erläutert und strukturiert. Die Beispiele lassen sich gut nachvollziehen und programmieren."

Professor Dr. Mutz, FH Hannover

Harald Nahrstedt

Algorithmen für Ingenieure

Technische Realisierung mit Excel und VBA

2., überarbeitete Auflage

Mit 165 Abbildungen und 65 Tabellen

STUDIUM

Springer Vieweg

Harald Nahrstedt
Möhnesee bei Soest, Deutschland

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säure freiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Ver brennung Schadstoffe freisetzen.

ISBN 978-3-8348-1692-4 ISBN 978-3-8348-1980-2 (eBook)
DOI 10.1007/978-3-8348-1980-2

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Einbandentwurf: KünkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist eine Marke von Springer DE.
Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Warum dieses Buch

In Laufe meiner langjährigen beruflichen Erfahrung musste ich viele ingenieurmäßige Aufgaben lösen. Dies gelang mir immer wieder, nach dem ich den für die Lösung richtigen Algorithmus gefunden hatte. Die Entwicklung eines Modells, die Suche nach dem Algorithmus und dessen formale Anwendung waren für mich immer die kreativsten Phasen eines Lösungsprozesses. Die Umsetzung des Algorithmus in die jeweilige Programmiersprache war dann „nur noch" handwerkliches Können. So ergaben sich mit den Jahren die verschiedensten Lösungen im technischen Bereich mit den unterschiedlichsten Algorithmen. Damit wuchs aber auch mein Interesse an verschiedene Arten von Algorithmen und ihren Möglichkeiten.

Mit Freude stelle ich fest, dass gerade in den letzten Jahren durch die sinnvolle Verknüpfung von Ingenieurwissenschaften und Informatik neue Maßstäbe gesetzt werden. Studienrichtungen wie Maschinenbauinformatik oder Ingenieurinformatik machen Hoffnung auf eine schnelle Weiterentwicklung. Ähnlich wie einst die Ingenieurdisziplin aus der Physik hervorgegangen ist, scheint auch die Informatik hier einen ähnlich klärenden Prozess zu erfahren. Hin zu einem Softwareingenieurwesen und weg vom Künstlertum und Tüftlerdasein.

Ziel dieses Buches ist es, sowohl dem Ingenieurstudenten als auch dem praktizierenden Ingenieur Algorithmen und deren Anwendungsmöglichkeiten zu zeigen. Dabei kann und soll der Umfang dieses Buches nur einfache Anwendungen zeigen und so das Prinzip erklären und das Interesse an einer Vertiefung des Stoffes wecken. So beschränken sich die mathematischen Herleitungen auf einfache Formen, ohne Untersuchung von Stetigkeit oder gültigen Bereichen.

Auch die Biologie hat schon immer die Ingenieurwissenschaften beeinflusst (Bionik) und erfährt im Moment über die Informatik mit Prozessen aus der Natur neue Impulse. Dabei habe ich das Thema *Neuronale Netze* zunächst ausgespart. Vielleicht erlaubt mir eine spätere Ausgabe auch diesem Thema einige Seiten zu widmen.

Ingenieure arbeiten oft in Teams. Dennoch werden durch Aufteilung bestehender Aufgaben die Probleme meist durch Einzelpersonen gelöst oder zumindest stammt die Kernidee einer Lösung von dieser. Es ist die Schlüsselqualifikation eines Ingenieurs, diese Problemlösungen zu liefern. In diesem Sinne ist ein Ingenieur auch forschend tätig, soweit dies ihm zeitlich möglich ist. Denn zeiteffektiv zu

arbeiten, steht bei ihm an vorderster Stelle und so benötigt er auch ein gutes Zeitmanagement. In diesem Sinne sollen die dargestellten Algorithmen auch zur schnelleren Lösungsfindung dienen.

Zum Aufbau

Im ersten Kapitel gebe ich eine kurze Übersicht zur geschichtlichen Entwicklung von Algorithmen. Ebenso werden die Eigenschaften und Klassen von Algorithmen erläutert. Eine Betrachtung verwandter Begriffe schließt sich an.

Die restlichen Kapitel haben eine grundlegende Einführung zum Thema und die Anwendung des Algorithmus bis zur Realisierung an einem praktischen Beispiel. Dazu verwende ich die Entwicklungsumgebung von Microsoft Office Excel 2003.

Danksagung

Ich danke all denen im Hause Vieweg+Teubner die, stets im Hintergrund wirkend, zum Gelingen dieses Buches beigetragen haben.

An den Leser

Dieses Buch soll auch zum Dialog zwischen Autor und Leser auffordern. Daher finden Sie sowohl auf der Homepage des Verlages www.viewegteubner.de, als auch auf meiner Homepage www.harald-nahrstedt.de ein Forum für ergänzende Programme, Anregungen und Kommentare so wie Lösungen zu den Übungsaufgaben.

Möhnesee, im September 2011 Harald Nahrstedt

Inhalt

1 Algorithmen 1

1.1 Geschichtliches 1
1.2 Formale Definition 3
1.3 Aspekte der Algorithmen 3
1.4 Algorithmenklassen 4
1.5 Das Konzept einer Problemlösung 6
1.6 Heuristik 7

2 Lösungen von Gleichungen 11

2.1 Lösungen von Quadratischen Gleichungen 11
2.2 Lösungen von Kubischen Gleichungen 14
2.3 Lösungen von Gleichungen höheren Grades 19

3 Lösungen linearer Gleichungssysteme 29

3.1 Lösungen linearer Gleichungssysteme 29
3.2 Lineare Optimierung mit der Simplex-Methode 34

4 Funktionen 45

4.1 Interpolation von Funktionen durch Polynome 45
4.1.1 Interpolation nach Newton 45
4.1.2 Interpolation mittels kubischer Splines 50
4.2 Approximation von Funktionen durch Polynome 55
4.3 Numerische Integration 62

5 Differentialgleichungen 75

5.1 Gewöhnliche Differentialgleichungen 75
5.2 Partielle Differentialgleichungen 94

6 Vektoren und Matrizen 103

6.1 Matrizendefinitionen 103
6.2 Lösungen von Gleichungssystemen 122
6.3 Differenzenverfahren für gewöhnliche Differentialgleichungen 128
6.4 Eigenwertprobleme 132

7 Pseudozufallszahlen ... 139

7.1 Integration nach der Monte-Carlo-Methode ... 139
7.2 Probabilistische Simulation ... 144

8 Algorithmen auf Datenstrukturen ... 163

8.1 Permutationen ... 163
8.2 Regression und Korrelation ... 169
8.3 Arrays und Datenfelder ... 176
8.4 Arbeiten auf Listenstrukturen ... 180
8.5 Arbeiten auf Baumstrukturen und Graphen ... 189

9 Verhaltens-Algorithmen ... 199

9.1 Das Prinzip Teile und Herrsche ... 199
9.2 Die Greedy-Methode ... 201
9.3 Rückverfolgung oder Backtracking ... 205
9.4 Rückwärtsrechnen oder rekursive Prozeduren ... 212

10 Bioalgorithmen ... 215

10.1 Der Ameisenalgorithmus ... 215
10.2 Evolutionsstrategien ... 224
10.3 Genetische Algorithmen ... 230

11 Künstliche Intelligenz ... 237

11.1 Fuzzy Logic ... 237
11.2 Expertensysteme ... 246

Literaturverzeichnis ... 247
Index ... 249

1 Algorithmen

Unter einem Algorithmus versteht man eine genau definierte Handlungsvorschrift zur Lösung eines Problems. Er stellt eine der wichtigsten mathematischen Begriffe dar, vergleichbar etwa mit dem Begriff der Funktion. Schaut man jedoch genauer hin, dann stellt man fest, dass der Algorithmus nicht nur für die Mathematik und Informatik von zentraler Bedeutung ist, sondern auch eng mit zentralen Themen der Geistesgeschichte verbunden ist. Seine Anwendung reicht in die Gebiete der Philosophie, der Medizin, der Biologie, der Kulturgeschichte, der Politik, u. v. a. Es gibt kaum eine Wissenschaft, in der der Algorithmus nicht bekannt ist und genutzt wird.

1.1 Geschichtliches

Lösungsvorschriften sind sehr viel älter, als es ihre Anwendung in den heutigen Computern vermuten lässt.

Einer der ältesten festgehaltenen Algorithmen der Menschheit wurde um 1700 v. Chr. in Keilschrift auf Tontafeln verfasst und beschreibt das Wurzelziehen nach einer babylonisch sumerischen Methode.

Den ersten schriftlich festgehaltenen Algorithmus verfasste Euklid in seinem Buch „Die Elemente“ im 3. Jh. v. Chr. Nebenbei bemerkt, diente dieses Buch etwa 2000 Jahre als Lehrbuch – eine phantastische Leistung. In diesem Buch beschreibt Euklid unter anderem auch den bis heute benutzten Euklid Algorithmus.

Lineare Gleichungssysteme mit zwei Unbekannten wurden schon um 1700 v. Chr. von den Babyloniern gelöst. Um 180 v. Chr. verfasste der chinesische Mathematiker Shang Cang ein allgemeines Lösungsverfahren für lineare Gleichungssysteme mit fünf Ungekannten, in dem unschwer der Gauß Algorithmus zu erkennen ist. Bei Leonardo Fibonacci von Pisa (1170-1220) findet man diese Lösungsmethode wieder. Man vermutet, dass sie ihren Weg von den Babyloniern über die Griechen (Diophant) oder Araber in den Westen genommen hat.

Das Wort Algorithmus wird auf den Autor des Buches Hisab al-dschabr wa-l-muqabala mit Namen Muhammad ibn Musa al Chwarizmi zurückgeführt. Dieser lebte um 830 n. Chr. am Hofe des Kalifen Mamum in Baghdad und er beschrieb in seinem Buch die Regeln der Arithmetik, wie sie bis dahin von den Indern entwickelt waren. Es ist sein Verdienst, dass sich damit die Algebra auch im Westen schnell verbreitete. Dazu trug auch im Mittelalter eine lateinische Übersetzung mit der Bezeichnung „algoritmi de numero Indorum“ bei. Stand in dem ursprüngli-

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_1,

chen Buch noch das Wort Algorism für Regeln der Arithmetik mit arabischen Zahlen, so wurde es später zu dem pseudogriechischen Wort „algorithmos“ übersetzt.

Es war dann der spanische Philosoph und Theologe Raimundus Lullus, der im Jahre 1656 seine berühmte „Ars Magna“ veröffentlichte. Darin beschreibt er einen allgemeinen Algorithmus als ein System grundlegender Begriffe, deren Kombination alle möglichen Wahrheiten erzeugen sollte. Dazu stellte er sich die mechanische Ausführung seines Algorithmus als eine Anordnung von sechs konzentrischen Kreisen vor, die systematisch gegeneinander verschoben, alle möglichen Kombinationen ergeben sollten.

Erst im Jahre 1842 wurde der erste für einen Computer gedachte Algorithmus von Ada Lovelace skizziert. Er war gedacht für die von Charles Babbage konstruierte Analytical Engine. Da er diese aber nicht vollendete, wurde auch der Algorithmus nie implementiert. Ada Lovelace gilt dennoch als die erste Programmiererin.

Leibnitz war es dann, der bereits in seiner Jugend eine Methode aufzustellen versuchte, die jedes Problem auf algorithmische Weise lösen sollte. Seine Dissertation „De arte combinatoria“ aus dem Jahre 1880 hatte zum Ziel, neue Begriffe aus wenigen grundlegenden Begriffen abzuleiten. Die Basis sollte ein Alphabet menschlichen Denkens sein, dargestellt durch geeignete Symbole. Ihre Kombination sollte zu allen bekannten Begriffen führen und auch neue erzeugen. Die Durchführbarkeit seines Systems sah Leibnitz durch Abbildungen aller Begriffe auf Zahlen und die der grundlegenden Begriffe auf Primzahlen. Rechnungen auf diesem System sollten dann alle Arten von Problemen lösen, bis hin zu unterschiedlichen menschlichen Meinungen. So sollten sich damit außerdem Prozesse des menschlichen Denkens abbilden und auf Maschinen nachvollziehen lassen.

Kurt Gödel zeigte dann im Jahre 1931 mittels Beweis, dass eine bestimmte Klasse mathematischer Probleme durch keinen Algorithmus aus einer exakt definierten Klasse von Algorithmen gelöst werden kann. In dieser Zeit wurde eine ganze Reihe von Ansätzen zur Definition eines Algorithmus entwickelt. Insbesondere den Begriff der Berechenbarkeit untersuchte Alan Turing mit seiner Turing-Maschine. Ebenso entstanden die Markov-Algorithmen.

Mit dem Aufkommen digitaler Rechenanlagen durch die Raumfahrt, bekamen die Algorithmen eine neue Bedeutung. Ging es früher nur um den Lösungsweg, so bekommen Begriffe wie Komplexität und Rechenzeit eine immer größere Bedeutung. Während die Rechenzeit ihre Grenzen in den Erkenntnissen der atomaren Physik findet, entstehen aus der Komplexität Begriffe wie „künstliche Intelligenz“ und führen zu scheinbaren Widersprüchen.

1.2 Formale Definition

Obwohl wir doch alle begreifen, was unter einem Algorithmus zu verstehen ist, ist uns die Wissenschaft eine exakte Definition des Begriffs bis heute schuldig geblieben. Im Laufe der Zeit wurden jedoch folgende Eigenschaften für einen Algorithmus abgeleitet:

- Alle verwendeten Größen müssen bekannt sein
- Die Umarbeitung geschieht in Arbeitstakten
- Die Beschreibung des Algorithmus ist vollständig
- Die Beschreibung des Algorithmus ist endlich
- Alle angegebenen Operationen sind zulässig
- Angabe einer Sprache für die Regeln

Die in diesem Buch dargestellten Algorithmen können nur als Beispiel für eine Kategorie gleichartiger Methoden dargestellt werden, da sich über jeden Bereich leicht ein eigenes Buch schreiben ließe. Der Leser soll diese Darstellungen als Anreiz empfinden, die Thematik weiter zu vertiefen.

1.3 Aspekte der Algorithmen

Algorithmen verbinden in ihrer Anwendung oft sehr unterschiedliche Wissensgebiete miteinander. So wie der Algorithmus von Euklid Geometrie und Algebra verbindet, gibt es Algorithmen im naturwissenschaftlichen Bereich als Simulation auf Wahrscheinlichkeiten. Algorithmen in den Sprachwissenschaften zur Beschreibung sprachlicher Regeln. Algorithmen in den Sozialwissenschaften zur Darstellung bestimmter Verhaltensmuster. Algorithmen in der Wirtschaft zur Darstellung von Wirkzusammenhängen. Und viele andere Gebiete mehr. Ja sogar in der Kunst werden Algorithmen zu Arbeitstechniken genannt. Ebenso lassen sich Algorithmen eines Fachgebiets auch auf andere übertragen. Als Beispiel sei der Ameisenalgorithmus aus der Bionik genannt.

Dieses Buch befasst sich mit der Anwendung von Algorithmen im ingenieurwissenschaftlichen Bereich und bedarf daher einer besonderen Sichtweise. Anders als der Naturwissenschaftler, kann sich der Ingenieur nicht ausgiebig mit allen Randbedingungen eines Problems befassen. Die Zeit, die ihm für ein Projekt zur Verfügung steht, ist ebenso endlich wie der finanzielle Rahmen. Er muss mit begrenzten Informationen Entscheidungen treffen und dabei Risiken abschätzen. Auf digitalen Rechenanlagen kann er sich mit Hilfe fachrelevanter Algorithmen diese Informationen durch Berechnungen und Simulationen in kürzester Zeit effizient besorgen.

Dies setzt allerdings Kenntnisse in einer gängigen Programmiersprache und entsprechendes Werkzeug voraus. Während die schnelllebige Hardware und Systemsoftware von geringer Bedeutung ist, sind die allgemeine Darstellung der Algo-

rithmen und ihre Umsetzung in eine leicht verständliche Programmiersprache Gegenstand unserer Betrachtung.

1.4 Algorithmenklassen

Die Einteilung der Algorithmen geschieht je nach Sichtweise in unterschiedlichen Klassen. Ich will daher nachfolgend einige Begriffe erläutern.

Die ältesten uns bekannten Algorithmen im klassischen Sinne sind die numerischen Algorithmen wie der Euklid Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen oder das Sieb des Eratosthenes zur Ermittlung der Primzahlen einer vorher festgesetzten Zahlenmenge. Aber auch Algorithmen zur Lösung von Gleichungen und Gleichungssystemen sind hier zu finden. In der Neuzeit kamen mit der Differential- und Integralrechnung auch Algorithmen zur Näherung, Integration, Differentiation, Interpolation und Approximation auf. Ebenso verschiedene Matrizen-Verfahren. Die Geschichte der numerischen Algorithmen ist Teil der Geschichte der Mathematik.

Unter deterministischen Algorithmen versteht man diejenigen, die bei gleicher Eingabe immer das gleiche Ergebnis liefern. Enthält ein Algorithmus Elemente eines Zufallsereignisses, so spricht man von nicht-deterministischen oder randomisierten Algorithmen. Das Monte-Carlo-Verfahren gehört zu diesen Algorithmen.

Unter iterativen Algorithmen versteht man diejenigen, die mit Rechenschritten, ausgehend von bekannten Größen, Zwischenergebnisse erzielen, die wiederum als Basis für eine erneute Ausführung der Rechenschritte dienen. Man unterscheidet diese Verfahren noch nach Algorithmen mit einer vorher bekannten Anzahl von Iterationsschritten, z. B. bilde n-Fakultät

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n \,,$$

mit n Iterationsschritten oder nach einer vorher unbekannten Anzahl von Iterationsschritten, z. B. bestimme die Euler Zahl e hinreichend genau

$$\text{bilde } x = \left(1 + \frac{1}{n}\right)^n \text{, solange } \left|x_n - x_{n-1}\right| < \varepsilon \,,$$

wobei ε eine sehr kleine Zahl darstellt.

Rekursive Algorithmen sind eine besondere Art von iterativen Algorithmen, bei denen die Rechenschritte aus dem eigentlichen Algorithmus bestehen, so dass dieser wiederholt angewendet wird, z. B. rekursive Bestimmung von n-Fakultät

$$n! = n \cdot (n-1)!,$$

ergibt sich aus der Bestimmung von (n-1)-Fakultät und diese wiederum aus (n-2)-Fakultät

$$(n-1)! = (n-1) \cdot (n-2)!,$$

zurück schreitend bis man das definierte 0! = 1 erreicht. Bei rekursiven Algorithmen ist besonders auf die Endlichkeit der Prozedur zu achten.

Nach der Art der Problemstellung unterscheidet man zwischen Entscheidungs- und Optimierungsalgorithmen. Entscheidungsalgorithmen stellen in ihrer komplexesten Form ein Expertensystem dar. Optimierungsalgorithmen gibt es mit den unterschiedlichsten Methoden. Eine Einteilung lässt sich vornehmen zwischen den Methoden die eine optimale Lösung und denen die zwar eine Lösung bieten, die aber nicht unbedingt die optimale sein muss. Ebenso zwischen denen, die nur eine Lösung bieten und denen, die alle Lösungen der Methode anzeigen.

Die Anwendungen von Optimierungsmethoden auf militärische, industrielle, wirtschaftliche, staatliche und soziale Prozesse sind unter dem Begriff Operation Research bekannt geworden. Erstmals im zweiten Weltkrieg eingesetzt, wurden diese Methoden Anfang der fünfziger Jahre auch für die Industrie und staatliche Stellen interessant. Mit dem Aufkommen digitaler Rechenanlagen werden sie bei Simulationen, Kostenoptimierungen, Lagerhaltungstheorien, Warteschlangenproblemen, Netzwerkanalyse, Transportproblemen, Auftragsplanungen und vieles mehr. Das größte Problem besteht heute in der Bestimmung des besten Verfahrens.

Auf der Suche nach immer neuen Algorithmen wurde auch wieder die Natur als Vorbild entdeckt. Es wurde und werden Mechanismen und Entscheidungsprozesse in der Natur gefunden, die sich auch in anderen Gebieten verwenden lassen. Für diese, als kognitive Algorithmen bezeichneten Methoden, spielen die Möglichkeitslogik (Fuzzy-Logic) und künstliche neuronale Netze (KNN) eine wichtige Rolle.

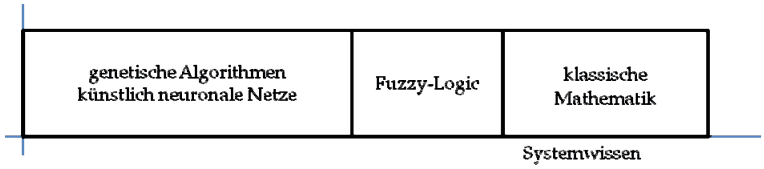


Abbildung 1-1: Systemwissen versus Algorithmenklassen

Ebenso die Gruppe der genetischen Algorithmen, die Fortpflanzungsmechanismen simulieren. Hierbei geht es mitunter um für das menschliche Verständnis sehr einfache Fähigkeiten, deren Realisierung im Computer allerdings großen Aufwand erfordert. Es werden auch Kombinationen dieser Methoden zur Lösung eingesetzt, die als hybride Algorithmen bezeichnet werden. Bezeichnend ist, dass mit abnehmendem Systemwissen die Anwendungsmöglichkeiten kognitiver Algorithmen steigen.

Der Begriff Künstliche Intelligenz (KI) stellt eher ein Forschungsgebiet, als eine einzelne Methode, geschweige denn einen Algorithmus dar. Mit den Expertensystemen ist ein erster sinnvoller Ansatz gefunden, dem sicher auch andere folgen werden. Was Algorithmen letztlich leisten können, lässt sich heute noch nicht beantworten.

Um es mit einem saloppen Ausspruch zu sagen, den ich erst kürzlich gelesen habe: „Die Zukunft hat gerade erst begonnen!“. Und dies gilt besonders für Algorithmen. Mit der immer weiter fortschreitenden Entwicklung der Computer sind neuronale Netze und parallele Algorithmen Wirklichkeit geworden.

Aber auch an den Postulaten der Algorithmen wird gerüttelt. Ein Algorithmus setzt voraus, dass alle Eingangsdaten bekannt sind. Es gibt jedoch Situationen, in denen Algorithmen erste Daten liefern müssen, bevor alle Daten vorliegen. Die Entscheidungen treffen für etwas, was erst noch passiert. Die zukünftige Neu- und Weiterentwicklung von Algorithmen wird sicher spannend bleiben.

1.5 Das Konzept einer Problemlösung

Computer sind ein unverzichtbares Hilfsmittel bei der Bearbeitung von Problemlösungen geworden. Wurden sie bisher zur Ermittlung und Speicherung von Daten eingesetzt, so werden sie immer mehr auch aktiv zur Problemlösung herangezogen. Mit Hilfe neuerer Algorithmen und immer schnellerer Systeme lassen sich selbst umfangreiche Probleme in kürzester Zeit lösen. Dabei hat die Simulation komplexer naturwissenschaftlich-technischer Zusammenhänge einen hohen Stellenwert.

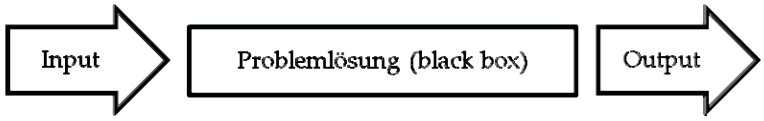


Abbildung 1-2: black box

Die Problemlösung beginnt in der Regel bei der Beschreibung des Problems als Black Box. Über die Definition des Outputs, lässt sich der Input oft direkt herleiten. Und damit auch deren Umwandlung über mathematisch-physikalische Zusammenhänge.

Mitunter sind jedoch auch Vereinfachungen und Abstraktionen notwendig. Immer mit der Maßgabe, ein adäquates Modell des Problems zu erhalten, an dem sich die Lösung des Problems nachvollziehen lässt. Die Abstraktion des Modells muss so lange erfolgen, bis ein geeigneter Algorithmus eingesetzt werden kann. Liegt der Algorithmus fest, erfolgt die Lösungsfindung unabhängig vom Computer, in Form eines Flussdiagramms, eines Struktogramms oder einer Pseudosprache. Erst danach sollte die Auswahl des Computers, die Wahl des Betriebssystems und der Programmiersprache erfolgen. Es folgt die Programmierung mit der Wahl der Prozeduren und Datenstrukturen. Die Entwicklung schließt mit umfangreichen Tests und Ergebnisanalysen ab. Entspricht die gefundene Lösung nicht den Anforderungen, müssen einzelne Schritte oder der gesamte Zyklus wiederholt werden. Diesen Zusammenhang zeigt anschaulich die Abbildung 1-3.

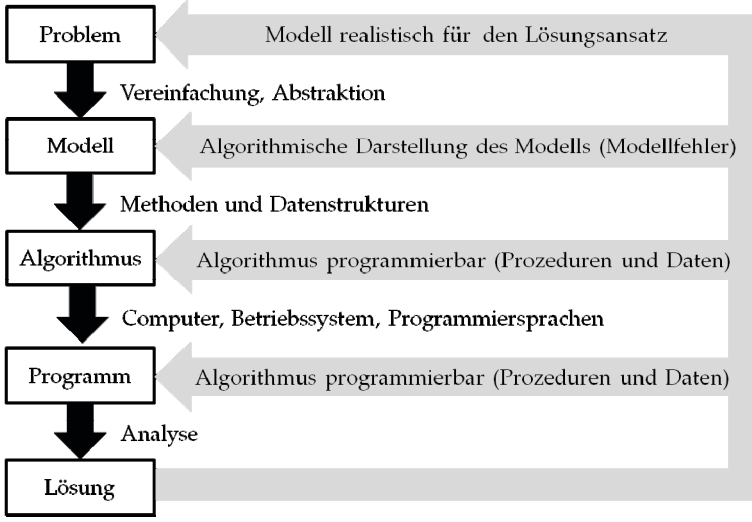


Abbildung 1-3: Schema der Lösungsfindung

1.6 Heuristik

Man kann kein Buch über Algorithmen schreiben, ohne wenigstens die Heuristik – die Wissenschaft des Problemlösens zu erwähnen.

Die Geschichte der Algorithmen ist immer auch ein Teil der Heuristikgeschichte. So haben Euklid, Mohammed ben Musa al Khovaresni, Lullus, Descartes, Leibnitz, u. a. immer erst im heuristischen Sinne geforscht und ihre Ergebnisse waren nicht nur Algorithmen.

Die ersten Ansätze entwickelte im 4. Jahrhundert der griechische Mathematiker Pappos von Alexandria mit folgender Methode:

1. Betrachte dein Problem als gelöst.
2. Suche den Lösungsweg durch Rückwärtsgehen (Analyse).
3. Den Beweis liefert das Vorwärtsgehen (Synthese).

Heute kennen wir verschiedene Methoden für ein heuristisches Vorgehen. Da ist an erster Stelle das Mindmapping zu nennen. Diese von Tony Buzan entwickelte Methode erstellt eine *Gedächtnislandschaft*.

Die dient zur schnellen Orientierung und hilft neue Möglichkeiten zu entdecken.

Eine weitere Methode ist das Brainstorming. Diese, von Alex F. Osborn entwickelte Methode, ist wohl die am häufigsten angewendete Kreativitätsmethode. Der Grundgedanke bei dieser Methode besteht darin, frei und ohne Zensur durch Assoziation eine große Anzahl von Ideen zu produzieren.

Zur Lösung im heuristischen Sinne ist weder ein Algorithmus noch ein Computer erforderlich. Manchmal muss man zur Lösungsfindung noch „vor“ dem Algorithmus einsteigen. An dieser Stelle möchte ich ganz besonders den Namen Fritz

Zwicky (1898-1974) nennen, der mit seiner Konstruktion und Auswertung des Morphologischen Kastens auch mir immer wieder neue Denkansätze geliefert hat. Ich will nachfolgend seine fünf Schritte zur Lösungsfindung kommentarlos wiedergeben.

1. Erstellung einer genauen Umschreibung oder Definition sowie der zweckmäßigen Verallgemeinerungen des vorgegebenen Problems.
2. Bestimmung aller Parameter, die die Lösung beeinflussen.
3. Erstellung des Morphologischen Kastens, in dem alle möglichen Lösungen des Problems ohne Beurteilung eingetragen werden.
4. Analyse aller im Morphologischen Kasten enthaltenen Lösungen bezüglich ihrer Realisierbarkeit oder anderer Werte.
5. Wahl der nach der Analyse optimalen Lösung sowie deren Realisierung und Konstruktion.

Zwickys Morphologischer Kasten erlaubt die Aufgabe von Vorfixierungen und Denkblockaden. Je nach Form auch als Matrix oder Tableau bezeichnet, ist die Zwicky Box die bekannteste Methode der Morphologie. Die Parameter des Problems werden in der ersten Spalte einer Tabelle eingetragen. Sie müssen voneinander unabhängig sein. Für die einzelnen Parameter werden ihre Ausprägungen (mögliche Lösungen) zeilenweise aufgelistet. Die Ausprägungen sollen sich dabei nicht am Parameter, sonder am Gesamtproblem orientieren. Durch Kombination der Ausprägungen werden dann Lösungsmöglichkeiten für das Gesamtproblem gebildet.

Tabelle 1-1: Zwicky Box

Parameter	**Ausprägungen**			
Rahmenform	gebogene Form	Standard	Doppelrahmen	Liegeform
Material	Alu	Carbon	Stahl	Kunststoff
Felgen	Alufelgen und Speichen	Stahlfelgen und Speichen	Kunststoff-Felgen und Speichen	Felgen ohne Speichen
Reifen	Vollgummi	Schlauchlose Reifen	Mantel und Schlauch	
Bremse	Scheibenbremse	Mittelzugbremse	Trommelbremse	Reifenbremse
Lenker	Rennlenker	Gebogener Lenker	Gerade Form	Bequeme Form
Schaltung	Nabenschaltung	Kettenschaltung	Ohne Schaltung	Kombination Ketten-/ Nabenschaltung

Die Lösungsmöglichkeiten können dann noch einmal mit Hilfe des Erfüllungsgrades zum Gesamtproblem beurteilt werden.

Tabelle 1-2: Lösungsmöglichkeiten

Parameter	Ausprägung	Erfüllungsgrad
Rahmenform	Gebogene Form	60%
Material	Alu	90%
Felgen	Alufelgen und Speichen	85%
Reifen	Mantel und Schlauch	50%
Bremse	Reifenbremse	45%
Lenker	Gerade Form	68%
Schaltung	Nabenschaltung	92%

Ein für Ingenieure typisches Denkbeispiel heißt Bierdeckelaufgabe [1]. Dabei geht es darum, neun Punkte auf einem Bierdeckel mit möglichst wenigen geraden Linien zu verbinden. Die übliche Lösung finden Sie in Abb. 1-4.

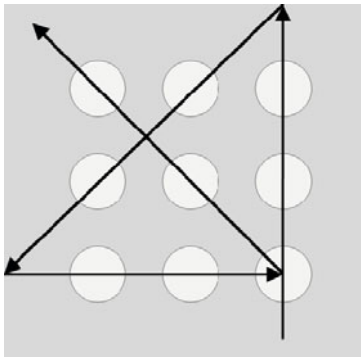


Abbildung 1-4: Übliche Lösung des Bierdeckelproblems

Gibt man die Restriktion „auf dem Bierdeckel zeichnen“ auf und ebenso die Restriktion, dass die Linien durch die Mitte der Punkte gehen müssen, dann erhält man eine Lösung mit drei Geraden (Abb. 1-5).

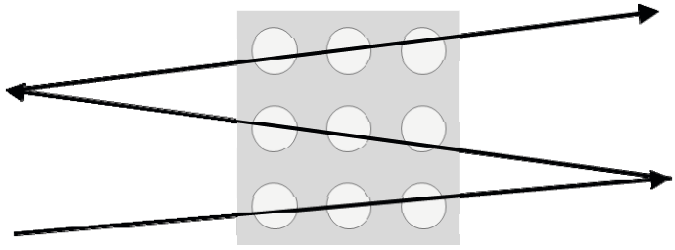


Abbildung 1-5: Lösung nach der Aufgabe von Restriktionen

Gibt man als weitere Restriktion die Unantastbarkeit des Bierdeckels auf, und zerschneidet man diesen, so gelingt sogar eine Lösung mit einer Linie.

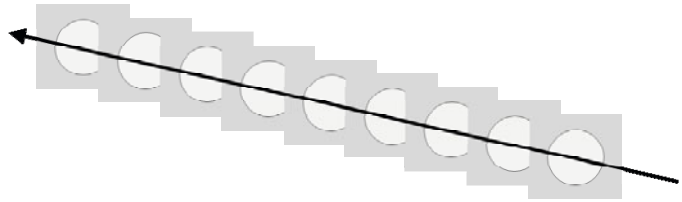


Abbildung 1-6: Eine dreidimensionale Lösung

Tabelle 1-3: Teil eines Morphologischen Kastens

Linienführung	**auf dem Bierdeckel**	**über den Bierdeckel hinaus**	**dreidimensional**
Bierdeckel	keine Änderung	falten	zerschneiden
Punkte	normal	stark vergrößert	./.
usw.			

Es gibt sogar eine Fuzzy-Lösung neben vielen weiteren Lösungen. Die Befreiung von Fixierungen gelingt fast methodisch durch den morphologischen Kasten.

Aber auch Groner & Groner [2] nennen heuristische Problemlösungsmethoden.

- Metakognitive Planung der einzusetzenden Heuristiken (Welche Heuristik?).
- Aufgabenanalyse (Input, Output).
- Abstraktion durch Reduktion und/oder Amplifikation (weglassen oder ergänzen).
- Wahl der Repräsentation (Problemdarstellung).
- Analogien (gibt es ein ähnlich gelöstes Problem).
- Teillösungen (top-down-design).
- Hypothesen prüfen (angenommen dass …, stimmt dann auch …).
- Trennung von Einflussgrößen (ist y abhängig von x oder x und y abhängig von z).
- Aufgabe von Fixierungen, Inkubation (Arbeit beiseite legen und später wieder aufgreifen).
- Nutzung des eigenen Unwissens (wie würde ich die Aufgabe lösen, bevor ich recherchiert habe, was andere getan haben).

Diese Punkte sind weder vollständig noch in der Reihenfolge anzuwenden. Die Heuristik ist ein weites und immer spannendes Betätigungsfeld.

Heute weiß man auch, dass Problemlösungen in einer Gruppe oft besser gefunden werden können, als durch den einzelnen Experten. Man spricht hier von kollektiver Intelligenz. Für den Erfolg sind jedoch einige Voraussetzungen nötig. Es sind in den letzten Jahren dazu Methoden und Techniken entwickelt worden. Die bekannteste Methode ist das Brainstorming.

2 Lösungen von Gleichungen

Gleichungen gehören nach den Zahlen zu den ersten mathematischen Errungenschaften der Menschheit. Bevor sich eine algebraische Schreibweise für Gleichungen gebildet hatte, wurden diese in Worte gefasst. Noch heute werden Dreisatzaufgaben gerne mit Worten beschrieben.

2.1 Lösungen von Quadratischen Gleichungen

Quadratische Gleichungen haben die allgemeine Form

$$x^2 + ax + b = 0 \tag{2.1.1}$$

Sind die Koeffizienten dieser Gleichung reell, treten drei Lösbarkeitsfälle auf. Die Lösungen sind entweder reell und verschieden, reell und fallen zusammen oder konjugiert komplex ohne reelle Lösung. Die Lösungsformel für die reellen Fälle lautet allgemein

$$x_{1,2} = -\frac{a}{2} \pm \sqrt{\left(\frac{a}{2}\right)^2 - b} \tag{2.1.2}$$

Der Radikand der Wurzel wird als Diskriminante (lat. discriminare, trennen, scheiden) bezeichnet und ist ausschlaggebend für die Art der Lösungen.

$$D = \left(\frac{a}{2}\right)^2 - b \tag{2.1.3}$$

Ist D>0 gibt es zwei reelle Lösungen. Ist D=0, nur eine reelle Lösung. Und ist D<0 werden die Lösungen imaginär.

Beispiel 2-1: Härtebestimmung nach Brinell

Bei der Härtebestimmung eines Werkstoffs mittels der Kugeldruckprobe nach Brinell wird die Eindringtiefe h einer kleinen Stahlkugel von bekanntem Radius r in einen zu prüfenden Werkstoff aus dem Radius x des Eindringkreises bestimmt.

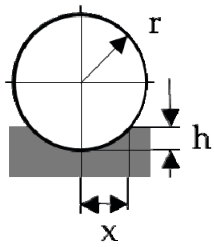


Abbildung 2-1: Kugeldruckprobe nach Brinell

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_2,

Nach dem Satz des Pythagoras erhält man

$$r^2 = (r - h)^2 + x^2 \quad . \tag{2.1.4}$$

Dies führt durch Umstellung auf die Normalform der Quadratischen Gleichung

$$h^2 + 2rh + h^2 = 0 \quad . \tag{2.1.5}$$

Die Lösungen lauten

$$h_{1,2} = r \pm \sqrt{r^2 - x^2} \quad . \tag{2.1.6}$$

Dabei ist nur eine Lösung sinnvoll, die eine Höhe h ermittelt, die kleiner ist als r

$$h = r - \sqrt{r^2 - x^2} \quad . \tag{2.1.7}$$

Die einfache Darstellung des Berechnungsablaufs als Struktogramm finden Sie in Tabelle 2-1.

Tabelle 2-1: Struktogramm zur Bestimmung der Eindringtiefe

Eingaben Kugelradius r, gemessener Abdruckradius x
$h = r - \sqrt{r^2 - x^2}$
Ausgabe der Eindringtiefe h

Zur Programmierung benutzen wir ein neues Tabellenblatt in unserer neu angelegten Mappe. Der Mappe geben wir den Namen *Algorithmen* und diesem Tabellenblatt den Namen *Brinell*.

Das zugehörige Codefenster bekommt den Namen *tblBrinell*. Schreibformen und Notationen entnehmen Sie bitte dem Kapitel 1 – Einführung in VBA meines Buches [3].

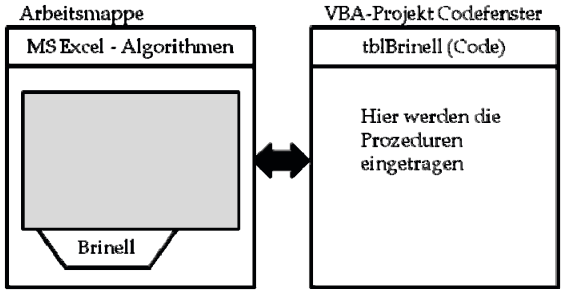


Abbildung 2-2: Eine Tabelle in einer Arbeitsmappe besitzt unter VBA ein Codefenster

In dieses Codefenster geben wir auch die in der Codeliste 2-1 dargestellten Prozeduren ein.

Code 2-1: Bestimmung der Eindringtiefe

```
Option Explicit

'Prozedur zur Erstellung eines Formblatts
Sub Brinell_Formblatt()

'Tabelle löschen
   Worksheets("Brinell").Activate
   Worksheets("Brinell").Cells.Clear

'Tabelle beschriften
   Range("A1").Value = "Kugelradius"
   Range("A2").Value = "Abdruckradius"
   Range("A3").Value = "Eindringtiefe"
   Range("C1").Value = "mm"
   Range("C2").Value = "mm"
   Range("C3").Value = "mm"
   Range("A1").ColumnWidth = 20
   Range("B1").ColumnWidth = 10
   Range("C1").ColumnWidth = 5
   Columns("B").Select
   Selection.NumberFormat = "0.000"
   Range("B1").Select
End Sub

'Prozedur zur Berechnung der Eindringtiefe
Sub Brinell_Auswertung()
   Dim r, x, h As Double

'Eingabewerte lesen
   r = Cells(1, 2)
   x = Cells(2, 2)
'Berechnung
   h = r - Sqr(r * r - x * x)
'Ausgabe
   Cells(3, 2) = h
   Range("B1").Select
End Sub
```

Zum Abschluss sollen die Prozeduren noch über eine Symbolleiste, die wir auch *Algorithmen* nennen, aufgerufen werden. Die Symbolleiste erhält den Menüpunkt *Eindringtiefe* mit den Unterpunkten *Neues Formblatt* und *Auswertung*. Die Menüunterpunkte erhalten bei der Definition noch keine Prozedur-Zuweisung. Beim ersten Aufruf fragt das System nach dieser Zuordnung und wir können aus einer Übersicht wählen.

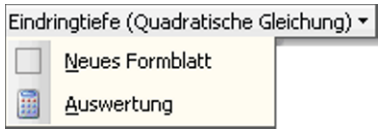


Abbildung 2-3: Menü Algorithmen Menüpunkt Eindringtiefe

In der Tabellenansicht wird eine Berechnung zunächst durch den Aufruf des Formblatts gestartet. Nach einer Eingabe von Daten für r und x erhalten wir das Ergebnis (Abbildung 2-4).

	A	B	C
1	Kugelradius	10,000	mm
2	Abdruckradius	5,000	mm
3	Eindringtiefe	1,340	mm

Abbildung 2-4: Auswertung mit Beispieldaten

Übung 2-1

Schreiben Sie ein Programm, so dass für einen vorgegebenen Kugelradius die Eindringtiefen für einen Abdruckradius-Bereich von x_1 bis x_2 in einer Tabelle erstellt werden.

2.2 Lösungen von Kubischen Gleichungen

Die Lösungsformel für kubische Gleichungen geht auf den Mathematiker del Ferro zurück. Es handelt sich dabei eher um ein Verfahren in mehreren Schritten. Die Normalform der kubischen Gleichung lautet

$$x^3 + ax^2 + bx + c = 0 \tag{2.2.1}$$

In einem ersten Schritt wird die Gleichung mit einem Faktor multipliziert. Danach eine kubische Ergänzung ermittelt und in einem weiteren Schritt eine Substitution durchgeführt. So erhält man eine reduzierte kubische Gleichung, deren Lösung sich durch zwei Kubikwurzeln darstellen lässt. Dieses so erhaltene Gleichungssystem ist nach dem Wurzelsatz von Vieta lösbar.

Wir ersparen uns das Herleiten der Formel und betrachten den Lösungsalgorithmus als Struktogramm in Tabelle 2-2.

Tabelle 2-2: Bestimmung der Lösungen von kubischen Gleichungen

Kubische Gleichung $x^3 + ax^2 + bx + c = 0$
$p = b - \frac{a^2}{3}$
$q = \frac{2 \cdot a^3}{27} - \frac{a \cdot b}{3} + c$
$(y = x + \frac{a}{3})$

Diskriminate $D = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3$
D<0 Drei verschiedene reelle Lösungen
$r = \sqrt{\frac{-p^3}{27}}$
$\cos(\varphi) = \frac{-\frac{q}{2}}{r}$
$y_1 = 2 \cdot \sqrt[3]{r} \cdot \cos\left(\frac{\varphi}{3}\right)$
$y_2 = 2 \cdot \sqrt[3]{r} \cdot \cos\left(\frac{\varphi}{3} + \frac{2 \cdot \pi}{3}\right)$
$y_3 = 2 \cdot \sqrt[3]{r} \cdot \cos\left(\frac{\varphi}{3} + \frac{4 \cdot \pi}{3}\right)$
D=0 Drei reelle Lösungen, von denen zwei identisch sind
$y_1 = 2 \cdot \sqrt[3]{-\frac{q}{2}}$
$y_2 = y_3 = -\sqrt[3]{-\frac{q}{2}}$
D>0 Eine reelle Lösung und zwei konjugiert komplexe Lösungen
$u = \sqrt[3]{-\frac{p}{2} + \sqrt{D}}$
$v = \sqrt[3]{-\frac{p}{2} - \sqrt{D}}$
$y_1 = u + v$
$y_{2,3} = \frac{-(u+v) \pm i \cdot \sqrt{3} \cdot (u-v)}{2}$

Die Umsetzung dieses Algorithmus als Programm finden Sie in der nachfolgenden Codeliste 2-2. Legen Sie dazu ein Tabellenblatt mit dem Namen *tblKubischeGleichung* an.

Code 2-2: Lösungen kubischer Gleichungen

```
Option Explicit

Sub Formular()
   Worksheets("Kubische Gleichung").Cells.Clear
```

```
   Cells(3, 1) = "Gleichung:"
   Cells(3, 2) = "x" & ChrW(179) & " + "
   Range("C3").Interior.ColorIndex = 15
   Cells(3, 4) = "x" & ChrW(178) & " + "
   Range("E3").Interior.ColorIndex = 15
   Cells(3, 6) = "x" & " + "
   Range("G3").Interior.ColorIndex = 15
   Cells(3, 8) = "="
   Cells(3, 9) = "0"
   Cells(3, 3).Activate

   Cells(5, 1) = "D="
   Cells(6, 1) = "x1="
   Cells(7, 1) = "x2="
   Cells(8, 1) = "x3="
   Range("B5:C5").MergeCells = True
   Range("B6:C6").MergeCells = True
   Range("B7:C7").MergeCells = True
   Range("B8:C8").MergeCells = True
End Sub

Private Sub Worksheet_Change(ByVal Target As Range)
   If Target.Row = 3 Then
      Select Case Target.Column
      Case 3
         Cells(3, 5).Activate
      Case 5
         Cells(3, 7).Activate
      Case 7
         Call Auswertung
         Cells(3, 3).Activate
      End Select
   End If
End Sub

Sub Auswertung()
   Dim a, b, c, D, p, q, r, s, u, v, w, z As Double
   Dim y1, y2, y3, y2i, y3i, x1, x2, x3 As Double

   a = Cells(3, 3)
   b = Cells(3, 5)
   c = Cells(3, 7)

   p = b - (a * a) / 3
   q = (2 * a * a * a) / 27 - (a * b) / 3 + c
   D = (q / 2) ^ 2 + (p / 3) ^ 3
   Cells(5, 2) = D
   If D < 0 Then
      r = Sqr(-(p / 3) ^ 3)
      z = -(q / 2) / r
      w = Atn(-z / Sqr(-z * z + 1)) + 2 * Atn(1)
      y1 = 2 * r ^ (1 / 3) * Cos(w / 3)
      y2 = 2 * r ^ (1 / 3) * Cos(w / 3 + 8 * Atn(1) / 3)
      y3 = 2 * r ^ (1 / 3) * Cos(w / 3 + 16 * Atn(1) / 3)
      x1 = y1 - a / 3
      x2 = y2 - a / 3
      x3 = y3 - a / 3
```

```
      Cells(6, 2) = x1
      Cells(7, 2) = x2
      Cells(8, 2) = x3
      Cells(7, 4) = ""
      Cells(8, 4) = ""
      Cells(7, 5) = ""
      Cells(8, 5) = ""
   ElseIf D = 0 Then
      y1 = 2 * (-q / 2) ^ (1 / 3)
      y2 = (-q / 2) ^ (1 / 3)
      y3 = y2
      x1 = y1 - a / 3
      x2 = y2 - a / 3
      x3 = y3 - a / 3
      Cells(6, 2) = x1
      Cells(7, 2) = x2
      Cells(8, 2) = x3
      Cells(7, 4) = ""
      Cells(8, 4) = ""
      Cells(7, 5) = ""
      Cells(8, 5) = ""
   Else
      s = Sqr(D)
      z = -q / 2 + s
      If z >= 0 Then
         u = z ^ (1 / 3)
      Else
         u = -(-z) ^ (1 / 3)
      End If
      z = -q / 2 - s
      If z >= 0 Then
         v = z ^ (1 / 3)
      Else
         v = -(-z) ^ (1 / 3)
      End If
      y1 = u + v
      y2 = -(u + v) / 2
      y2i = Sqr(3) * (u - v) / 2
      y3 = -(u + v) / 2
      y3i = -Sqr(3) * (u - v) / 2
      x1 = y1 - a / 3
      x2 = y2 - a / 3
      x3 = y3 - a / 3
      Cells(6, 2) = x1
      Cells(7, 2) = x2
      Cells(8, 2) = x3
      Cells(7, 4) = "+ i"
      Cells(8, 4) = "+ i"
      Cells(7, 5) = y2i
      Cells(8, 5) = y3i
   End If
End Sub
```

Aufgerufen werden die Prozeduren Formblatt und Auswertung über entsprechende Menüpunkte in der Symbolleiste. Die Eingabe der Formel ist so gestaltet, dass mit jeder Eingabe zur weiteren Eingabe eines Koeffizienten gesprungen wird.

Mit der Eingabe des letzten Koeffizienten erfolgen dann der Aufruf der Bewertung und die Ausgabe der Lösungen. Die Eingabe kann dann erneut beginnen.

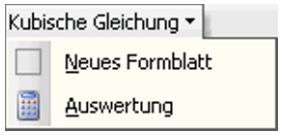


Abbildung 2-5: Menü Kubische Gleichung

Beispiel 2-2: Trichtervolumen

Mit Hilfe des erstellten Programms soll ein Berechnungsproblem gelöst werden. Ein Blechtrichter nach Abbildung 2-6 hat die Querschnittsform eines gleichseitigen Dreiecks und ein Volumen von 1814 cm³. Gesucht ist der obere Durchmesser des Kegels, denn hier soll ein zylindrisches Rohr angeschweißt werden.

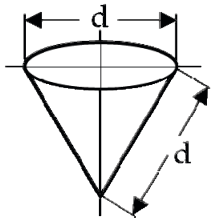


Abbildung 2-6: Geschweißter Blechtrichter

Das Volumen bestimmt sich aus der Formel

$$V = \frac{\pi}{3} \cdot \left(\frac{d}{2} \right)^2 \cdot h \ . \tag{2.2.2}$$

Aus der Gleichschenkligkeit folgt für die Höhe

$$h = \frac{d}{2} \sqrt{3} \ . \tag{2.2.3}$$

Damit ergibt sich die kubische Gleichung

$$d^3 - \frac{24 \cdot V}{\pi \cdot \sqrt{3}} = 0 = d^3 - 8000 \ . \tag{2.2.4}$$

Das Programm liefert die Lösung. Der Durchmesser ist 20 cm groß.

	A	B	C	D	E	F	G	H	I
1									
2									
3	Gleichung:	x³ +	0	x² +	0	x +	-8000	=	0
4									
5	D=		16000000						
6	x1=		20						
7	x2=		-10	+ i	17,3205081				
8	x3=		-10	+ i	-17,3205081				

Abbildung 2-7: Berechnungsformular

Übung 2-2: Kugelbehälter

Welchen Durchmesser hätte ein Kugelbehälter bei gleichem Volumen?

2.3 Lösungen von Gleichungen höheren Grades

Jede algebraische Gleichung kann in der allgemeinen Form

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0 \tag{2.3.1}$$

dargestellt werden. Man spricht von einer Gleichung n-ten Grades. Für n > 4 gibt es keine Lösungsformel. Das ist mathematisch bewiesen. Betrachtet man diese Gleichung als Funktion

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \tag{2.3.2}$$

so ergeben sich einige Lösungsverfahren. Durch die Entwicklung der Computertechnologie haben die so genannten Iterationsverfahren eine große Bedeutung erfahren. Lassen sich die Lösungsmethoden doch in Form von Iterationsschleifen programmieren, die solange durchlaufen werden, bis die gewünschte Genauigkeit erreicht ist.

Betrachten wir die Methode Regula Falsi (Abbildung 2-8). Bezeichnen wir die zu suchende Nullstelle (Wurzel) einer stetigen Funktion f(x) mit x_0. Bei der Methode ersetzt man die Kurve f(x) im Intervall (x_1, x_2) durch eine Sekante (lineare Interpolation). Der Schnittpunkt mit der x-Achse x_3 ist die angenäherte Lösung für x_0.

Eine wiederholte Anwendung dieser Methode mit den Näherungswerten liefert eine Lösung mit hinreichender Genauigkeit. Voraussetzung ist die Stetigkeit der Funktion. Eine ähnliche Methode ist das Newton Verfahren. Diese Verfahren funktionieren nicht nur bei Funktionen, die als Polynome gegeben sind, sondern generell für stetige Funktionen.

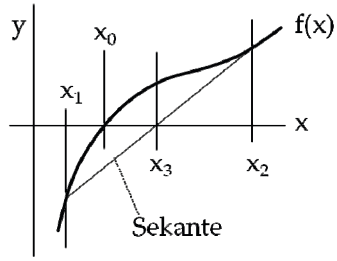


Abbildung 2-8: Methode Regula Falsi

Beispiel 2-3: Minimaler Materialverbrauch

Für eine zylindrische Konservendose (Abbildung 2-9), mit einem vorgegebenen Inhalt V, soll zur Herstellung möglichst wenig Blech verbraucht werden. Das Volumen bestimmt sich aus der Gleichung

$$V = \pi \cdot r^2 \cdot h \tag{2.3.3}$$

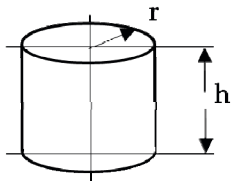


Abbildung 2-9: Zylindrischer Behälter

Die Oberfläche, die eigentliche Zielgröße, bestimmt sich aus

$$O = 2 \cdot \pi \cdot r^2 + \frac{2 \cdot V}{r}, \tag{2.3.4}$$

wenn man für h die umgestellte Volumengleichung einsetzt.

Nun suchen wir nicht nach einer Nullstelle dieser Funktion, sondern nach einem Extremwert, nämlich dem Minimum. Es gilt für stetige Funktionen, das an der Stelle eines Extremwertes der Funktion f(x) ihre erste Ableitung y'=f'(x) das Vorzeichen wechselt. Ein Extremwert der Ableitung ist wiederum an ihrer Ableitung, der zweiten Ableitung zu erkennen.

Ist diese an dieser Stelle positiv, dann liegt ein Minimum vor, andernfalls ein Maximum. Die Ableitungen der Oberfläche ergeben

$$O' = 4 \cdot \pi \cdot r - \frac{2 \cdot V}{r^2} \tag{2.3.5}$$

und

$$O'' = 4 \cdot \pi + \frac{4 \cdot V}{r^3}. \tag{2.3.6}$$

Stellen wir zunächst den Algorithmus für die Methode Regula Falsi auf. Tabelle 2-3 zeigt die Methode in der allgemeinen Form eines Struktogramms. Ich benutze dazu eine Tabellenform. So lässt sich ein Struktogramm schnell erstellen.

Tabelle 2-3: Struktogramm zur Methode Regula Falsi

Eingabe Bestimmung zweier Startwerte x_1 und x_2, so dass $f(x_1)>0$ und $f(x_2)<0$ Abschaltwert der Iterationsschleife ε		
	$x = x_1 - y_1 \frac{x_2 - x_1}{y_2 - y_1}$	
	Ist f(x)>0	
	Ja	Nein
	$x_1=x$	$x_2=x$
	So lange wie $\lvert f(x)\rvert > \varepsilon$	
Ausgabe x		

Nun ersetzen wir die allgemeinen Bedingungen und Formeln durch die speziellen unseres Beispiels und erhalten so einen Algorithmus, wie in Tabelle 2-4 dargestellt.

Tabelle 2-4: Struktogramm zur Bestimmung des Durchmessers für einen minimalen Materialverbrauch

Eingabe Bestimmung zweier Startwerte d_1 und d_2, so dass $O'(d_1)>0$ und $O'(d_2)<0$. Abschaltwert der Iterationsschleife ε		
	$d = d_1 - O'(d_1) \frac{d_2 - d_1}{O'(d_2) - O'(d_1)}$	
	Ist O'(d)>0	
	Ja	Nein
	$d_1=d$	$d_2=d$
	So lange wie $\lvert f(x)\rvert > \varepsilon$	
Bestimme O''(d)		
Ausgabe d und Angabe ob Minimum oder Maximum		

Wir erstellen ein weiteres Tabellenblatt in unserer Mappe *Algorithmen*. Dem Tabellenblatt geben wir den Namen *Minimum* und dem dazugehörigen Codefenster den Namen *tblMinimum*.

In diesem Codefenster erstellen wir die Programmanweisungen nach Codeliste 2-2. Übernehmen Sie auch die Kommentare, damit Sie bei einer späteren Betrachtung des Programmcodes auch schneller den Inhalt verstehen. Sie sollten auch bei

allen nachfolgenden Programmen nicht mit Kommentaren sparen. Hier gilt die Devise: „Lieber zu viel als zu wenig."

Code 2-3: Bestimmung der minimalen Oberfläche

```
Option Explicit

'Prozedur zur Erstellung eines Formblatts

Private Sub Minimum_Formblatt()

'Tabelle löschen
   Worksheets("Minimum").Activate
   Worksheets("Minimum").Cells.Clear


'Tabelle beschriften
   Range("A1").Value = "Volumen V ="
   Range("A2").Value = "Startwert d1 ="
   Range("A3").Value = "Startwert d2 ="
   Range("A4").Value = "Abschaltgrenze"
   Range("C1").Value = "cm" + ChrW(179)
   Range("C2").Value = "mm"
   Range("C3").Value = "mm"
   Range("E1").Value = "Oberfläche [cm^2]"
   Range("F1").Value = "O'(d)"
   Range("G1").Value = "O''(d)]"
   Range("B6").Value = "d [cm]"

   Range("A1").ColumnWidth = 15
   Range("B1").ColumnWidth = 10
   Range("C1").ColumnWidth = 5
   Range("D1").ColumnWidth = 1
   Range("E1").ColumnWidth = 15
   Range("F1").ColumnWidth = 15
   Range("G1").ColumnWidth = 15
   Columns("B").Select
   Selection.NumberFormat = "0.000"
   Range("B1").Select
End Sub
Private Sub Minimum_Testdaten()
   Cells(1, 2) = 50
   Cells(2, 2) = 10
   Cells(3, 2) = 100
   Cells(4, 2) = 0.01
End Sub

Private Function Ob(V, d)  'Oberfläche in cm^3
   Dim r As Double
   Dim pi As Double
   pi = 4 * Atn(1)         'Konstante pi
   r = d / 2               'Radius in cm
   Ob = 2 * pi * r * r + 2 * V / r
End Function

Private Function Ob1(V, d) '1. Ableitung der Oberfläche
   Dim r As Double
```

```
   Dim pi As Double
   pi = 4 * Atn(1)          'Konstante pi
   r = d / 2                'Radius in cm
   Ob1 = 4 * pi * r - 2 * V / (r * r)
End Function

Private Function Ob2(V, d) '2. Ableitung der Oberfläche
   Dim r As Double
   Dim pi As Double
   pi = 4 * Atn(1)          'Konstante pi
   r = d / 2                'Radius in cm
   Ob2 = 4 * pi + 4 * V / (r * r * r)
End Function

Private Sub Minimum_Auswertung()
   Dim d, d1, d2, V, e As Double
   Dim i As Integer

'Eingabewerte lesen
   V = Cells(1, 2)          'Volumen in cm^3
   d1 = Cells(2, 2) / 10    'Startwert 1 in cm
   d2 = Cells(3, 2) / 10    'Startwert 2 in cm
   e = Cells(4, 2)          'Abschaltkriterium

'Startüberprüfung
   If Ob1(V, d1) > 0 Then
      If Ob1(V, d2) <= 0 Then
         'Startwerte korrekt
      Else
         MsgBox "Startwerte falsch!", _
            vbInformation & vbOKOnly
         Exit Sub
      End If
   Else
      If Ob1(V, d2) > 0 Then
         d = d1: d1 = d2: d2 = d
      Else
         MsgBox "Startwerte falsch!", _
            vbInformation & vbOKOnly
         Exit Sub
      End If
   End If

'Berechnung
   Cells(2, 5) = Ob(V, d1)
   Cells(2, 6) = Ob1(V, d1)
   Cells(2, 7) = Ob2(V, d1)
   Cells(3, 5) = Ob(V, d2)
   Cells(3, 6) = Ob1(V, d2)
   Cells(3, 7) = Ob2(V, d2)

   i = 6
   Do
      d = d1 - Ob1(V, d1) * (d2 - d1) _
          / (Ob1(V, d2) - Ob1(V, d1))
      If Ob1(V, d) > 0 Then
         d1 = d
```

```
      Else
         d2 = d
      End If
      i = i + 1
      Cells(i, 2) = d
      Cells(i, 5) = Ob(V, d)
      Cells(i, 6) = Ob1(V, d)
      Cells(i, 7) = Ob2(V, d)
'Abbruchkriterium
   Loop While Abs(Ob1(V, d)) > e

End Sub
```

Auch diese Prozeduren werden über die Symbolleiste Algorithmen angebunden. Die Symbolleiste erhält den Menüpunkt *Minimale Oberfläche* mit den Unterpunkten *Formblatt, Testdaten* und *Auswertung*. Die Prozeduren müssen bei der Definition nicht zugewiesen werden, denn beim ersten Aufruf des Symbols wird nach der zugehörigen Prozedur gefragt und im Dialog werden alle vorhandenen Prozeduren angezeigt. Durch Anklicken erfolgt die Zuordnung.

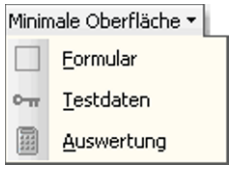


Abbildung 2-10: Menü Minimale Oberfläche

Mit Hilfe der eingebauten Testdaten ergibt sich die Auswertung nach Abbildung 2-11. In dem gezeigten Tabellenblatt sind die Zeilen 11 bis 59 ausgeblendet, so dass man nur die Werte zum Beginn und zum Ende der Iteration sieht. Bei einem zylindrischen Behälter vom 50 cm^3 Inhalt ergibt sich für einen Durchmesser von 3,993 cm die kleinste Oberfläche zu 75,13 cm^2. Welchen Wert hat die Zylinderhöhe?

	A	B	C	D	E	F	G
1	Volumen V =	50,000	cm³		Oberfläche [cm^2]	O'(d)	O''(d)]
2	Startwert d1 =	10,000	mm		177,0796327		
3	Startwert d2 =	100,000	mm		201,5707963		
4	Abschaltgrenze	0,010					
5							
6		d [cm]					
7		8,830			145,1230549	50,35019769	14,89038823
8		7,942			124,2653093	43,56095191	15,76010511
9		7,251			110,1629796	37,94831455	16,76390524
10		6,701			100,3823833	33,19668794	17,88350659
60		3,994			75,13251203	0,019506824	37,67957562
61		3,994			75,13251069	0,016712208	37,68237383
62		3,994			75,1325097	0,014317894	37,68477138
63		3,994			75,13250898	0,012266562	37,68682561
64		3,994			75,13250845	0,01050909	37,68858566
65		3,993			75,13250806	0,009003392	37,69009362

Abbildung 2-11: Auswertung der Testdaten

Übung 2-3: Volumentabelle

Schreiben Sie ein Programm das für mehrere Volumen z. B. vom 100 bis 1000 cm^3 mit einer Schrittweite von 100 die jeweiligen optimalen Durchmesser und Oberflächen bestimmt und stellen Sie die Verhältnisse in einem Diagramm dar.

Die nachfolgende Betrachtung führt zur Suche nach einem Maximum.

Beispiel 2-4: Maximales Volumen

Ein Transportbehälter soll so aus einem quadratischen Blech mit der Kantenlänge geformt werden, dass sein Volumen ein Maximum darstellt. Zur Herstellung werden die vier kleinen Quadrate an den Ecken ausgestanzt und die seitlichen Laschen gefalzt.

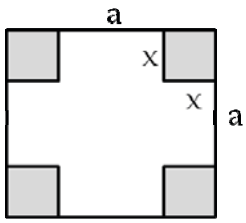


Abbildung 2-12: Blechzuschnitt

Das Volumen des Behälters bestimmt sich aus der Gleichung

$$V = x(a-2x)^2 \ . \tag{2.3.7}$$

Die Ableitungen ergeben hier

$$V' = 12x^2 - 8ax + a^2 \tag{2.3.8}$$

und

$$V'' = 24x - 8a \ . \tag{2.3.9}$$

Auch hier wird wiederum nach einer Nullstelle der Ableitung gesucht. Diesmal wollen wir uns das Verfahren nach Newton ansehen (Abbildung 2-13). Im Gegensatz zur Methode Regula Falsi wird statt der Sekante eine Tangente zur Ermittlung einer weiteren Näherung benutzt.

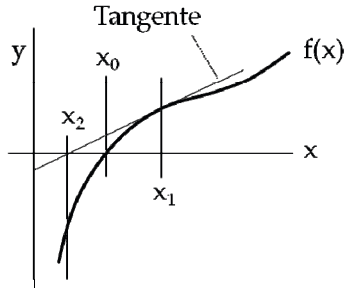


Abbildung 2-13: Methode nach Newton

Auch diese Methode liefert nach endlichen Iterationsschritten eine brauchbare Lösung.

Tabelle 2-5: Struktogramm zur Methode nach Newton

Eingabe Bestimmung eines Startwertes x_1 Abschaltwert der Iterationsschleife ε
$x = x_1 - \frac{f(x_1)}{f'(x_1)}$
$x_1 = x$
So lange wie $\lvert f(x) \rvert > \varepsilon$
Ausgabe x

Damit sich tatsächlich Näherungswerte ergeben, muss im betrachten Intervall gelten

$$\left| \frac{f(x) \cdot f'(x)}{[f'(x)]^2} \right| < 1 \ . \qquad (2.3.10)$$

Interessant ist noch, dass diese Methode mit einem Startwert auskommt. Für beide Methoden ist eine Voraussetzung, dass es sich um zweimal stetig differenzierbare Funktionen handelt. Nur dann ist ein Abbruch der Iterationen gewährleistet.

Für das Maximum-Problem ergibt sich damit der in Tabelle 2-6 dargestellte Algorithmus. Wir erstellen ein weiteres Tabellenblatt in unserer Mappe Algorithmen. Der Tabelle geben wir den Namen *Maximum* und dem dazugehörigen Codefenster den Namen *tblMaximum*.

Tabelle 2-6: Struktogramm zur Bestimmung des maximalen Volumens

Eingabe Eingabe der Kantenlänge a in cm Bestimmung des Startwertes x_1 Abschaltwert der Iterationsschleife ε
$x = x_1 - \frac{V(x_1)}{V'(x_1)}$
Bestimme V(x), V′(x) und V′′(x)
So lange wie $\lvert V'(x) \rvert > \varepsilon$

In *tblMaximum* tragen wir den Programmcode aus der Codeliste 2-4 ein.

Code 2-4: Bestimmung des maximalen Volumens

```
Option Explicit

'Prozedur zur Erstellung eines Formblatts
Private Sub Maximum_Formblatt()

'Tabelle löschen
   Worksheets("Maximum").Activate
   Worksheets("Maximum").Cells.Clear
'Tabelle beschriften
   Range("A1").Value = "Kantenlänge a ="
   Range("A2").Value = "Startwert x ="
   Range("A3").Value = "Abschaltgrenze"
   Range("C1").Value = "cm"
   Range("C2").Value = "cm"
   Range("E1").Value = "Volumen [cm^3]"
   Range("F1").Value = "V'(x)"
   Range("G1").Value = "V''(x)]"
   Range("B6").Value = "x [cm]"
   Range("A1").ColumnWidth = 15
   Range("B1").ColumnWidth = 10
   Range("C1").ColumnWidth = 5
   Range("D1").ColumnWidth = 1
   Range("E1").ColumnWidth = 15
   Range("F1").ColumnWidth = 15
   Range("G1").ColumnWidth = 15
   Columns("B").Select
   Selection.NumberFormat = "0.000"
   Range("B1").Select
End Sub

Private Sub Maximum_Testdaten()
   Cells(1, 2) = 50
   Cells(2, 2) = 2
   Cells(3, 2) = 0.01
End Sub

Private Function Vol(a, x)    'Volumen in cm^3
   Vol = x * (a - 2 * x) ^ 2
End Function

Private Function Vo1(a, x)    '1. Ableitung des Volumens
   Vo1 = 12 * x * x - 8 * a * x + a * a
End Function

Private Function Vo2(a, x)    '2. Ableitung des Volumens
   Vo2 = 24 * x - 8 * a
End Function

Private Sub Maximum_Auswertung()
   Dim a, x, e As Double
   Dim i As Integer

'Eingabewerte lesen
   a = Cells(1, 2)         'Kantenlänge in cm
   x = Cells(2, 2)         'Startwert in cm
```

```
    e = Cells(3, 2)          'Abschaltkriterium

'Berechnung
    Cells(2, 5) = Vol(a, x)
    Cells(2, 6) = Vo1(a, x)
    Cells(2, 7) = Vo2(a, x)
    i = 6
    Do
       x = x - Vo1(a, x) / Vo2(a, x)
       i = i + 1
       Cells(i, 2) = x
       Cells(i, 5) = Vol(a, x)
       Cells(i, 6) = Vo1(a, x)
       Cells(i, 7) = Vo2(a, x)

'Abbruchkriterium
    Loop While Abs(Vo1(a, x)) > e
End Sub
```

Die Symbolleiste Algorithmen erhält einen neuen Menüpunkt *Maximales Volumen*. Dessen Unterpunkte sind wiederum *Formblatt*, *Testdaten* und *Auswertung*. Beim ersten Aufruf werden die Menüpunkte mit den Prozeduren verknüpft.

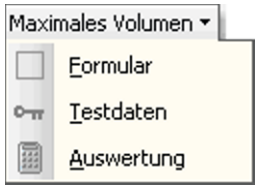


Abbildung 2-14: Menü Maximales Volumen

Mit Hilfe der eingebauten Testdaten ergibt sich die nachfolgende Auswertung. Bei einer Kantenlänge vom 50 cm ergibt sich ein Einschnitt von x=8,333 cm für ein maximales Volumen von 9259,26 cm^3.

	A	B	C	D	E	F	G
1	Kantenlänge a =	50,000	cm		Volumen [cm^3]	V'(x)	V''(x)]
2	Startwert x =	2,000	cm		4232	1748	-352
3	Abschaltgrenze	0,010					
4							
5							
6		x [cm]					
7		6,966			9062,046846	295,9230372	-232,8181818
8		8,237			9258,326837	19,38675004	-202,3130347
9		8,333			9259,259229	0,110190344	-200,0132224
10		8,333			9259,259259	3,64209E-06	-200,0000004

Abbildung 2-15: Auswertung der Testdaten

Übung 2-4: Kantenlänge

Schreiben Sie ein Programm das für mehrere Kantenlängen z. B. von 50 bis 250 cm, mit einer Schrittweite von 10 cm, das jeweilige Maß x zum optimalen Volumen bestimmt und stellen Sie die Verhältnisse in einem Diagramm dar. Gibt es eine feste Beziehung zwischen a und $x_{optimal}$?

3 Lösungen linearer Gleichungssysteme

Systeme linearer Gleichungen, kurz lineare Gleichungssysteme genannt, bestehen aus mehreren Gleichungen mit mehreren Unbekannten. In der Technik führen viele Probleme auf lineare Gleichungssysteme.

3.1 Lösungen linearer Gleichungssysteme

Eigentlich hatten wir es in unserem vorherigen Beispiel von Kapitel 2 mit einem System von zwei Gleichungen mit zwei Unbekannten zu tun. Der allgemeine Fall liegt vor, wenn m Gleichungen mit n Unbekannten gegeben sind.

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2 \\ &\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = c_m \end{aligned} \qquad (3.1.1)$$

Diese Form heißt lineares Gleichungssystem. Die reellen Zahlen a_{ik} (i=1,…,m; k=1,…,n) sind die Koeffizienten des Systems. Die reellen Zahlen c_i werden als Absolutglieder bezeichnet. Das Gleichungssystem wird als homogen bezeichnet, wenn die Absolutglieder verschwinden.

Eine Methode zur Bestimmung der Lösung ist das Gauß Verfahren, auch als Gauß Eliminationsverfahren bezeichnet. Man entfernt durch Multiplikation von Gleichungen mit einer Zahl und Addition zu einer anderen aus (n-1) von n Gleichungen eine Unbekannte. Entfernt aus (n-2) der neuen (n-1) Gleichungen eine zweite Unbekannt. Das Verfahren wird solange wiederholt, bis nur eine Gleichung mit einer Unbekannten vorliegt. Aus ihr wird die Unbekannte bestimmt und durch rückwirkendes Einsetzen alle anderen.

Tabelle 3-1: Algorithmus der Gauß-Elimination

Eingabe der Koeffizienten des Gleichungssystems
I = 1 (1) n-1
j = i+1 (1) n
$c = a_{ji} / a_{ii}$
k = i + 1 (1) n + 1
$a_{jk} = a_{ik} - c\ a_{ik}$
i= n (-1) 1
$\sum a = a_{i,n+1}$
k = i + 1 (1) n
$\sum a = \sum a - a_{ik}\ x_k$
$Xi = \sum a / a_{ii}$
Lösung ausgeben

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_3,

Beispiel 3-1: Temperaturverteilung in einem Kanal

Die Temperaturverteilung innerhalb eines Kanals mit rechteckigem Querschnitt wird durch die Laplace-Gleichung beschrieben. An der Rohrwand werden unterschiedliche Temperaturen gemessen. Es sollen die Temperaturen an den angegebenen Punkten bestimmt werden unter der Annahme, dass ein innen liegender Punkt den Mittelwert aller benachbarten Punkte hat.

Jeder Punkt geht zu einem Viertel in die Gleichungen ein und da wir nicht umständlich mit Brüchen arbeiten wollen, multiplizieren wir die Gleichungen mit 4 und so ergeben sich die nachfolgenden Gleichungen.

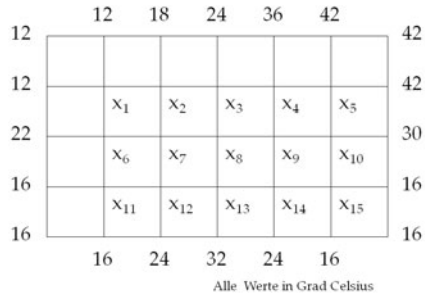


Abbildung 3-1: Gitternetz

$$
\begin{aligned}
x_1 &: 4x_1 - x_2 - x_6 = 24\\
x_2 &: -x_1 + 4x_2 - x_3 - x_7 = 18\\
x_3 &: -x_2 + 4x_3 - x_4 - x_8 = 24\\
x_4 &: -x_3 + 4x_4 - x_5 - x_9 = 36\\
x_5 &: -x_4 + 4x_5 - x_{10} = 84\\
x_6 &: -x_1 + 4x_6 - x_7 - x_{11} = 22\\
x_7 &: -x_2 - x_6 + 4x_7 - x_8 - x_{12} = 0\\
x_8 &: -x_3 - x_7 + 4x_8 - x_9 - x_{13} = 0\\
x_9 &: -x_4 - x_8 + 4x_9 - x_{10} - x_{14} = 0\\
x_{10} &: -x_5 - x_9 + 4x_{10} - x_{15} = 30\\
x_{11} &: -x_6 + 4x_{11} - x_{12} = 34\\
x_{12} &: -x_7 - x_{11} + 4x_{12} - x_{13} = 24\\
x_{13} &: -x_8 - x_{12} + 4x_{13} - x_{14} = 32\\
x_{14} &: -x_9 - x_{13} + 4x_{14} - x_{15} = 24\\
x_{15} &: -x_{10} - x_{14} + 4x_{15} = 32
\end{aligned}
\qquad (3.1.2)
$$

Da wir den Algorithmus als Struktogramm bereits vorliegen haben, erstellen wir ein Tabellenblatt *Temperaturverteilung* und geben unter *tblTemperaturverteilung* den Programmcode aus Codeliste 3-1 ein.

Code 3-1: Bestimmung der Temperaturverteilung nach der Gauß-Elimination

```
Option Explicit

'Gauss-Elimination
Private Sub Werte(n, A, x)
   Dim Shp As Shape
   Dim i, j As Integer

'Tabelle löschen
   Worksheets("Temperaturverteilung").Activate
   Worksheets("Temperaturverteilung").Cells.Clear
   For Each Shp In Shapes
      Shp.Delete
   Next

'Werte
   Range("A1:P15").ColumnWidth = 5
   Range("A20:G24").Select
   Selection.NumberFormat = "0.00"

   For i = 1 To n
      For j = 1 To n + 1
         Cells(i, j) = 0
         If i = j Then Cells(i, j) = 4
      Next j
   Next i
   Cells(1, 2) = -1: Cells(1, 6) = -1
   Cells(2, 1) = -1: Cells(2, 3) = -1
   Cells(2, 7) = -1
   Cells(3, 2) = -1: Cells(3, 4) = -1
   Cells(3, 8) = -1
   Cells(4, 3) = -1: Cells(4, 5) = -1
   Cells(4, 9) = -1
   Cells(5, 4) = -1: Cells(5, 10) = -1
   Cells(6, 1) = -1: Cells(6, 7) = -1
   Cells(6, 11) = -1
   Cells(7, 2) = -1: Cells(7, 6) = -1
   Cells(7, 8) = -1: Cells(7, 12) = -1
   Cells(8, 3) = -1: Cells(8, 7) = -1
   Cells(8, 9) = -1: Cells(8, 13) = -1
   Cells(9, 4) = -1: Cells(9, 8) = -1
   Cells(9, 10) = -1: Cells(9, 14) = -1
   Cells(10, 5) = -1: Cells(10, 9) = -1
   Cells(10, 15) = -1
   Cells(11, 6) = -1: Cells(11, 12) = -1
   Cells(12, 7) = -1: Cells(12, 11) = -1
   Cells(12, 13) = -1
   Cells(13, 8) = -1: Cells(13, 12) = -1
   Cells(13, 14) = -1
   Cells(14, 9) = -1: Cells(14, 13) = -1
   Cells(14, 15) = -1
   Cells(15, 10) = -1: Cells(15, 14) = -1
   Cells(1, 16) = 24
   Cells(2, 16) = 18
   Cells(3, 16) = 24
   Cells(4, 16) = 36
```

```
   Cells(5, 16) = 84
   Cells(6, 16) = 22
   Cells(7, 16) = 0
   Cells(8, 16) = 0
   Cells(9, 16) = 0
   Cells(10, 16) = 30
   Cells(11, 16) = 34
   Cells(12, 16) = 24
   Cells(13, 16) = 32
   Cells(14, 16) = 24
   Cells(15, 16) = 32
End Sub

Private Sub Werte_Lesen(n, A, x)
   Dim i, j As Integer

'Bestimmung belegter Zeilen
'und Definition der notwendigen Datenfelder
   Cells(Rows.Count, 1).End(xlUp).Select
   n = ActiveCell.Row
   ReDim A(n, n + 1), x(n) As Double
   For i = 1 To n
      For j = 1 To n + 1
         A(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Private Sub Subtrahiere_Gleichung(A, n, i, j)
   Dim k As Integer
   Dim c As Double

   c = A(j, i) / A(i, i)
   For k = i + 1 To n + 1
      A(j, k) = A(j, k) - c * A(i, k)
   Next k
End Sub
Private Function Summe(A, x, n, i)
   Dim s As Double
   Dim k As Integer

   s = A(i, n + 1)
   For k = i + 1 To n
      s = s - A(i, k) * x(k)
   Next k
   Summe = s
End Function

Sub Auswertung()
   ReDim A(1, 1), x(1) As Double
   Dim n, i, j, k As Integer

   n = 15
   ReDim A(n, n + 1), x(n) As Double
   Call Werte(n, A, x)
   Call Werte_Lesen(n, A, x)
   For i = 1 To n - 1
```

```
      For j = i + 1 To n
         Call Subtrahiere_Gleichung(A, n, i, j)
      Next j
   Next i

   For i = n To 1 Step -1
      x(i) = Summe(A, x, n, i) / A(i, i)
      j = Int((i - 1) / 5) + 1
      k = i - (j - 1) * 5
      Cells(j + 20, k + 1) = x(i)
      Cells(i, n + 3) = x(i)
   Next i

   Cells(20, 1) = 12
   Cells(20, 2) = 12
   Cells(20, 3) = 18
   Cells(20, 4) = 24
   Cells(20, 5) = 36
   Cells(20, 6) = 42
   Cells(20, 7) = 42
   Cells(21, 7) = 42
   Cells(22, 7) = 30
   Cells(23, 7) = 16
   Cells(24, 7) = 16
   Cells(24, 6) = 16
   Cells(24, 5) = 24
   Cells(24, 4) = 32
   Cells(24, 3) = 24
   Cells(24, 2) = 16
   Cells(24, 1) = 17
   Cells(23, 1) = 18
   Cells(22, 1) = 22
   Cells(21, 1) = 12

   Range("A20:G24").Select
   Charts.Add
   ActiveChart.ChartType = xlSurface
   ActiveChart.SetSourceData _
      Source:=Sheets("Temperaturverteilung").Range( _
      "A20:G24"), PlotBy:=xlRows
   ActiveChart.Location Where:=xlLocationAsObject, Name:= _
      "Temperaturverteilung"
   With ActiveChart
      .HasTitle = True
      .ChartTitle.Characters.Text = "Temperaturverteilung"
      .Axes(xlCategory).HasTitle = False
      .Axes(xlSeries).HasTitle = False
      .Axes(xlValue).HasTitle = False
   End With
   ActiveChart.Corners.Select
   With ActiveChart
      .Elevation = 25
      .Rotation = 211
   End With
End Sub
```

Die Symbolleiste *Algorithmen* erhält einen neuen Menüpunkt *Temperaturverteilung* mit einem einzigen Unterpunkt *Auswertung*. Beim ersten Aufruf wird dieser Menüpunkt mit der Prozedur Auswertung verknüpft.

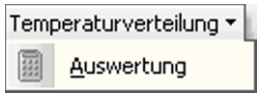


Abbildung 3-2: Menü Temperaturverteilung

Nach Eingabe der Koeffizienten und Aufruf der Auswertung stehen die Ergebnisse in der Tabelle in Spalte R von x_1 bis x_{15} aufsteigend.

Zusätzlich zur Ausgabe der berechneten Werte habe ich im Bereich (A20:G24) die Temperaturwerte nach Lage eingestellt. Damit ist es dann möglich, mittels eines Makros, daraus eine grafische Darstellung zu erstellen. Die Anweisungen aus dem Makro habe ich dann in die Prozedur Auswertung eingefügt.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	4	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0	24		16,0313341
2	-1	4	-1	0	0	0	-1	0	0	0	0	0	0	0	0	18		20,3130166
3	0	-1	4	-1	0	0	0	-1	0	0	0	0	0	0	0	24		25,1700181
4	0	0	-1	4	-1	0	0	0	-1	0	0	0	0	0	0	36		31,1461428
5	0	0	0	-1	4	0	0	0	0	-1	0	0	0	0	0	84		35,9601126
6	-1	0	0	0	0	4	-1	0	0	0	-1	0	0	0	0	22		19,8123199
7	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0	0	0	0		22,050714
8	0	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0	0	0		25,2209131
9	0	0	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0	0		27,4544407
10	0	0	0	0	-1	0	0	0	-1	4	0	0	0	0	-1	30		28,6943075
11	0	0	0	0	0	-1	0	0	0	0	4	-1	0	0	0	34		19,1672316
12	0	0	0	0	0	0	-1	0	0	0	-1	4	-1	0	0	24		22,8566063
13	0	0	0	0	0	0	0	-1	0	0	0	-1	4	-1	0	32		26,2084797
14	0	0	0	0	0	0	0	0	-1	0	0	0	-1	4	-1	24		24,7563993
15	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	4	32		21,3626767
16																		
17																		
18																		
19																		
20	12,00	12,00	18,00	24,00	36,00	42,00	42,00											
21	12,00	16,03	20,31	25,17	31,15	35,96	42,00											
22	22,00	19,81	22,05	25,22	27,45	28,69	30,00											
23	18,00	19,17	22,86	26,21	24,76	21,36	16,00											
24	17,00	16,00	24,00	32,00	24,00	16,00	16,00											
25																		
26																		
27																		
28																		
29																		
30																		
31																		
32																		
33																		
34																		

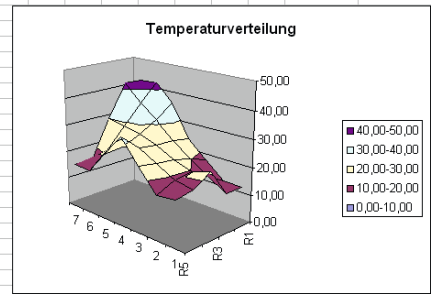


Abbildung 3-3: Temperaturfeld der Testdaten

Übung 3-1: Randtemperaturen

Schreiben Sie als Ergänzung eine Prozedur, mit deren Hilfe Sie die Randtemperaturen verändern können.

3.2 Lineare Optimierung mit der Simplex-Methode

Bleiben wir weiter bei den Gleichungssystemen. Bei vielen wirtschaftlichen Problemen gilt es für eine Ziel- oder Kostenfunktion mit n Variablen

$$f = c_0 + c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \qquad (3.2.1)$$

einen Extremwert (minimale Kosten, maximale Ausnutzung, etc.) zu finden, für die zusätzlich gegebene Nebenbedingungen (Gleichungen oder Ungleichungen) erfüllt sind.

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = a_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le a_2 \\ &usw. \end{aligned} \tag{3.2.2}$$

mit $x_1>=0,\ldots, x_n>=0$.

Zur Lösung gibt es einige Verfahren. Wir befassen uns an dieser Stelle mit der Simplexmethode von Dantzig. Die Methode wollen wir an Hand eines Beispiels studieren.

Beispiel 3-2: Produktionsoptimierung

Eine Produktion erstellt mit zwei Maschinengruppen M_1, M_2 zwei Produkte P_1, P_2 mit folgenden Zeiten und Gewinnen:

Tabelle 3-2: Produktionsübersicht

	M_1	M_2	Gewinn
P_1 (A)	6	2	3 Euro
P_2 (B)	4	4	5 Euro
	160 Std./Woche	120 Std./Woche	

Die Firma arbeitet mit 40 Stunden/Woche und verfügt über 4 Maschinen der Gruppe M_1 und 3 Maschinen der Gruppe M_2. Unter der Annahme, dass immer die ganze Produktion verkauft werden kann, ist der maximale Gewinn gesucht.

Kommen wir zur Aufstellung des Gleichungssystems. A sei die wöchentliche Produktion des Produkts P_1 und B die wöchentliche Produktion des Produktes P_2. Mit der Gruppe M_1 stehen 160 Stunden und mit der Gruppe M_2 120 Stunden pro Woche zur Verfügung. Damit gilt:

Gesucht: Maximum von

$$3A + 5B \Rightarrow \text{Maximum} \tag{3.2.3}$$

Restriktionen:

$$\begin{aligned} &6A + 4B \le 160 \\ &2A + 4B \le 120 \\ &A, B \ge 0 \end{aligned} \tag{3.2.4}$$

Grafisch ist dieses Problem direkt lösbar, wenn man die Restriktionen als Funktionen im gültigen Bereich zeichnet.

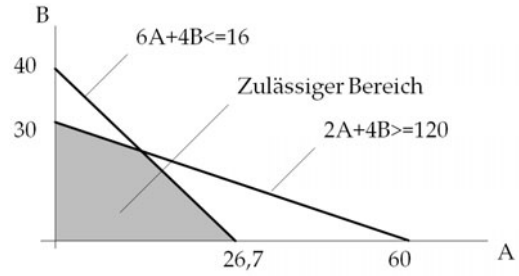


Abbildung 3-4: Grafische Lösung

Doch wir interessieren uns für die Simplex-Methode. Dazu erstellt man für das Problem eine Tabelle der Form Tabelle 3-3.

Als Pivotspalte wird diejenige gewählt, in der in der untersten Zeile der größte negative Wert steht. In diesem Fall -5. Als Pivotzeile wird diejenige gewählt, bei der sich der Quotient aus Wert der letzten Spalte geteilt durch Wert der Pivotspalte den kleinsten Wert ergibt. In diesem Fall ist 160/4=40 > 120/4=30. Somit ist das Pivotelement die 4 in der zweiten Zeile.

Tabelle 3-3: Auswertungstabelle

6	4	160
2	4	120
-3	-5	0

Die Tabelle wird nun folgendermaßen umgeformt. Die Elemente in der Pivotzeile werden durch das Pivotelement dividiert. Die Elemente der Pivotspalte werden durch das Pivotelement dividiert und bekommen das umgekehrte Vorzeichen. Das Pivorelement selbst wird durch seinen Reziprokwert ersetzt. Alle übrigen Elemente werden nach der Rechteckregel umgeformt.

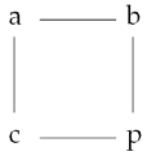


Abbildung 3-5: Rechteckregel

Sei a ein beliebiges Element aus der alten Tabelle, b und c die im Rechteck zum Pivotelement p angeordneten Elemente der alten Tabelle (siehe Abbildung 3-5), dann ergibt sich für den neuen Wert

$$a = a - b \cdot \frac{c}{p} \cdot \tag{3.2.5}$$

Die neue Tabelle sieht also bereits nach der ersten Umformung wie in Tabelle 3-4 dargestellt aus.

Tabelle 3-4: Auswertungstabelle nach der ersten Umformung

4,00	-1,00	40
0,50	0,25	30
-0,50	1,25	150

Die Tabelle wird solange umgeformt, bis in der unteren Zeile nur noch positive Elemente stehen.

Tabelle 3-5: Auswertungstabelle nach der letzten Umformung

0,25	-0,25	10
-0,125	0,75	25
0,125	1,125	155

Zusätzlich werden die Elemente der untersten Zeile und die Elemente der letzten Spalte mit fortlaufenden Zahlen markiert. Bei jeder Umwandlung werden dann die Marken von Pivotzeile und Pivotspalte vertauscht. Am Ende stehen in den Zeilen 1, 2,…, n in der hinteren Spalte die Werte für x_1, x_2,…, x_n.

Das Verfahren liefert das gleiche Ergebnis, wie wir es von der grafischen Lösung kennen, nämlich dass das Maximum bei

$$3 \cdot 10 + 5 \cdot 25 = 155 \text{ Euro}$$

liegt.

Wir wollen nun den allgemeinen Algorithmus unter Verwendung eines Tabellenblattes und von Arrays erstellen.

Tabelle 3-6: Algorithmus der Simplex-Methode

Eingabe der Ungleichungs-Koeffizienten Zeilenwerte m und Spaltenwerte n		
Daten lesen	i=1, 1, m+1	
		j = 1, 1, n+1
		a(i,j) = Cells(i+2,j)
z=0		
	i=1, 1, n / Markierung setzen	
	z=z+1	
	Cells(m+5,i)=z	
	i=1, 1, m / Markierung setzen	
	z=z+1	
	Cells(i+2,n+3)=z	
	ps=0 Startwert der Pivotspalte	
	Bestimme Pivot-	i=1, 1, n
		Ist ps=0 oder a(m+1,i)<min

spalte	Ja		Nein
	Min=a(m+1,i) und ps=i		./.
pz=0 Startwert der Pivotzeile			
Bestimme Pivotzeile	i=1, 1, m		
	Ist a(i,ps)>0		
	Ja		Nein
	$d = \frac{a(i, n+1)}{a(i, ps)}$		./.
	Ist pz=0 oder d<Min		
	Ja	Nein	
	Min=d und pz=i		
	Umformung i=1, 1, m+1		
	Rechteckregel j=1 bis n+1Schritt 1		
	Ist i≠pz und j≠ps		
	Ja		Nein
	$b(i, j) = a(i, j) - a(i, ps) \cdot \frac{a(pz, j)}{a(pz, ps)}$		./.
	i=1, 1, n+1		
	Ist i≠ps		
	Ja		Nein
	$b(pz, i) = \frac{a(pz, i)}{a(pz, ps)}$		./.
	i=1, 1, m+1		
	Ist i≠pz		
	Ja		Nein
	$b(i, ps) = -\frac{a(i, ps)}{a(pz, ps)}$		./.
	$b(pz, ps) = \frac{1}{a(pz, ps)}$		
	Neue Tabelle eintragen i=1, 1, m+1		
	j=1, 1, n+1		
	a(i,j)=b(i,j)		
	Cells(i+2,j)=b(i,j)		
Marken vertauschen Z=Cells(pz+2,n+3) Cells(pz+2,n+3)=Cells(m+5,ps)			

	Cells(m+5,ps)=z		
	k=0		
	Schleifen-bedingung	i=1, 1, n+1	
		Ist a(m+1,i)<0	
		Ja	Nein
		k=1	./.
	solange k=1		

Dieser Algorithmus wird in einem neuen Tabellenblatt *Simplex* unter *tblSimplex* programmiert. Den Programmcode finden Sie in der nachfolgenden Codeliste 3-2.

Code 3-2: Simplexmethode

```
Option Explicit

Private Sub Testdaten()
   Dim i, j As Integer

'Tabelle löschen
   Worksheets("Simplex").Activate
   Worksheets("Simplex").Cells.Clear

'Werte
   Range("A1:C3").ColumnWidth = 10
   Range("A1:B1").Select
   Selection.NumberFormat = "#0"
   Range("A3:C5").Select
   Selection.NumberFormat = "0.000"

   Cells(1, 1) = 2
   Cells(1, 2) = 2

   Cells(3, 1) = 6
   Cells(3, 2) = 4
   Cells(3, 3) = 160
   Cells(4, 1) = 2
   Cells(4, 2) = 4
   Cells(4, 3) = 120
   Cells(5, 1) = -3
   Cells(5, 2) = -5
   Cells(5, 3) = 0
End Sub

Private Sub Auswertung()
   Dim m, n, i, j, k, pz, ps, min, z As Integer
   Dim d As Double
   ReDim A(1, 1), B(1, 1) As Double

   m = Cells(1, 1)
   n = Cells(1, 2)
   ReDim A(m + 1, n + 1), B(m + 1, n + 1)


'Daten lesen
   For i = 1 To m + 1
```

```
      For j = 1 To n + 1
         A(i, j) = Cells(i + 2, j)
      Next j
   Next i

'Marken setzen
   z = 0
   For i = 1 To n
      z = z + 1
      Cells(m + 5, i) = z
   Next i
   For i = 1 To m
      z = z + 1
      Cells(i + 2, n + 3) = z
   Next i

'Auswertungsschleife
   Do

'Bestimme PivotSpalte mit größtem negativen Wert
'in der untersten Zeile
      ps = 0
      For i = 1 To n
         If ps = 0 Or A(m + 1, i) < min Then
            ps = i
            min = A(m + 1, i)
         End If
      Next i

'Bestimme PivotZeile durch Division der Werte
'der letzten Spalte durch Elemente in der PivotSpalte,
'vorausgesetzt dieser Wert ist > 0
      pz = 0
      For i = 1 To m
         If A(i, ps) > 0 Then
            d = A(i, n + 1) / A(i, ps)
            If pz = 0 Or d < min Then
               min = d
               pz = i
            End If
         End If
      Next i

'Umformung
      For i = 1 To m + 1
         For j = 1 To n + 1
            If Not i = pz And _
               Not j = ps Then
               B(i, j) = A(i, j) - A(i, ps) * _
                         A(pz, j) / A(pz, ps)
            End If
         Next j
      Next i
      For i = 1 To n + 1
         If Not i = ps Then
            B(pz, i) = A(pz, i) / A(pz, ps)
         End If
```

```
      Next i
      For i = 1 To m + 1
         If Not i = pz Then
            B(i, ps) = -A(i, ps) / A(pz, ps)
         End If
      Next i
      B(pz, ps) = 1 / A(pz, ps)

'Neue Tabelle eintragen
      For i = 1 To m + 1
         For j = 1 To n + 1
            A(i, j) = B(i, j)
            Cells(i + 2, j) = B(i, j)
         Next j
      Next i

'Marken vertauschen
      z = Cells(pz + 2, n + 3)
      Cells(pz + 2, n + 3) = Cells(m + 5, ps)
      Cells(m + 5, ps) = z

'Schleifenbedingung
      k = 0
      For i = 1 To n + 1
         If A(m + 1, i) < 0 Then k = 1
      Next i
   Loop While k = 1

End Sub
```

Die Symbolleiste *Algorithmen* erhält wieder einen neuen Menüpunkt *Simplex* mit den Unterpunkten *Textdaten* und *Auswertung*. Beim ersten Aufruf werden diese Menüpunkte mit den Prozeduren Testdaten und Auswertung verknüpft.

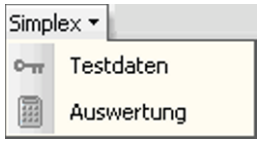


Abbildung 3-6: Menü Simplex

Die Testdaten entsprechen dem betrachteten Beispiel und nach deren Aufruf und der anschließenden Auswertung ergeben sich die bereits bekannten Werte.

	A	B	C	D	E
1	2	2			
2					
3	0,250	-0,250	10,000		1
4	-0,125	0,375	25,000		2
5	0,125	1,125	155,000		
6					
7	3	4			

Abbildung 3-7: Ergebnis der Testdaten

Übung 3-3: Produktionsplanung

Ein Unternehmen stellt zwei Produkte her, die auf drei Maschinen hergestellt werden können. Die entsprechenden Daten zeigt die nachfolgende Tabelle.

Tabelle 3-7: Produktionsdaten

Produkt 1	Produkt 2	Einheit		max. Kapazität [Min]
4	2	Min/St	Maschine 1	220
2	4	Min/St	Maschine 2	240
2	2	Min/St	Maschine 3	140
154	110	DM/St	Deckungsbeitrag pro Einheit	

Die Zielsetzung besteht darin, ein Produktionsprogramm zu bestimmen, welches den Gesamtdeckungsbeitrag maximiert.

Ist
x_1 = Anzahl der produzierten Menge von Produkt 1
x_2 = Anzahl der produzierten Menge von Produkt 2
dann ergibt sich aus der maximalen Kapazität der Maschine

$$\begin{aligned} 4x_1 + 2x_2 &<= 220 \\ 2x_1 + 4x_2 &<= 240 \\ 2x_1 + 2x_2 &<= 140 \end{aligned} \quad (3.2.6)$$

mit der Randbedingung $x_1, x_2 >= 0$, die Zielfunktion

$$154x_1 + 110x_2 \rightarrow Max. \quad (3.2.7)$$

In Matrizenform lässt sich das Problem darstellen mit

$$Max \quad c^T x \quad (3.2.8)$$

und den Nebenbedingungen

$$Ax <= b, x >= 0. \quad (3.2.9)$$

Mit den Werten c^T=(154,110); x=x1, x2; $A = \begin{pmatrix} 4 & 2 \\ 2 & 4 \\ 2 & 2 \end{pmatrix}$; b^T=(220, 240, 140).

Für die Simplexmatrix gilt die Forderung Ax = b, statt wie unter Gleichung 3.2.9 Ax <= b. Die Umwandlung erzielt, wenn man für jede Nebenbedingung eine sogenannte Schlupfvariable einführt , das Gleichungssystem

$$\begin{aligned} 4x_1 + 2x_2 &= 220 \\ 2x_1 + 4x_2 &= 240 \\ 2x_1 + 2x_2 &= 140 \end{aligned} \quad (3.2.10)$$

Ermitteln Sie auch für dieses Problem, zur Produktionsplanung mit mehreren Engpässen, die Lösung.

Als weiteres Beispiel wählen wir die Suche nach einem Minimalwert.

Beispiel 3-3: Zuschnittoptimierung

In einer Produktion werden aus 5 x 3 Meter großen Spanplatten nachfolgende Bauteile geschnitten.

Tabelle 3-8: Bauteile-Übersicht

Typ	Länge x Breite	Stück
A	3 x 1,5	120
B	4 x 1,5	80
C	3 x 1	220
D	2 x 2	60

Tabelle 3-9: Zuschnittformen

Zuschnittform	A	B	C	D
1	3	0	0	0
2	1	1	0	0
3	1	0	3	0
4	1	0	1	1
5	2	0	0	1
6	0	2	0	0
7	0	2	1	0
8	0	1	2	0
9	0	0	5	0
10	0	0	2	2
11	0	0	3	1

Für den Zuschnitt gibt es mehrere Möglichkeiten. Die sinnvollsten werden nachfolgend dargestellt. Schnittbreiten bleiben unberücksichtigt.

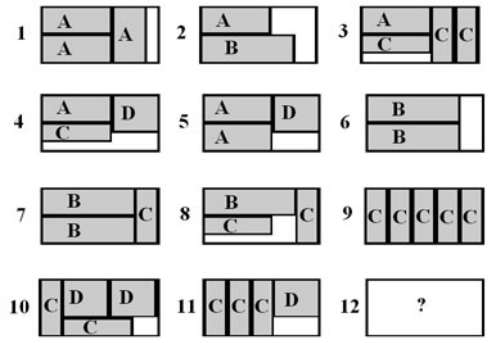


Abbildung 3-8: Zuschnitte

Daraus ergeben sich nach den Tabellen 3-7 und 3-8 die Ungleichungen

$$\begin{aligned}
&3\cdot x_1+1\cdot x_2+1\cdot x_3+1\cdot x_4+2\cdot x_5+0\cdot x_6+0\cdot x_7+0\cdot x_8+0\cdot x_9+0\cdot x_{10}+0\cdot x_{11}\le 120\\
&0\cdot x_1+1\cdot x_2+0\cdot x_3+0\cdot x_4+0\cdot x_5+2\cdot x_6+2\cdot x_7+1\cdot x_8+0\cdot x_9+0\cdot x_{10}+0\cdot x_{11}\le 80\\
&0\cdot x_1+0\cdot x_2+3\cdot x_3+1\cdot x_4+0\cdot x_5+0\cdot x_6+1\cdot x_7+2\cdot x_8+5\cdot x_9+2\cdot x_{10}+3\cdot x_{11}\le 220\\
&0\cdot x_1+0\cdot x_2+0\cdot x_3+1\cdot x_4+1\cdot x_5+0\cdot x_6+0\cdot x_7+0\cdot x_8+0\cdot x_9+2\cdot x_{10}+1\cdot x_{11}\le 60\\
&x_1\ge 0, x_2\ge 0, x_3\ge 0, x_4\ge 0, x_5\ge 0, x_6\ge 0, x_7\ge 0, x_8\ge 0, x_9\ge 0, x_{10}\ge 0, x_{11}\ge 0
\end{aligned} \tag{3.2.11}$$

Mit Hilfe des Simplex-Programms ergibt sich die optimale Lösung. Interessant ist, dass die Zuschnittform 5 trotz des geringen Verschnitts nicht zur Anwendung kommt.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	4	11												
2														
3	0,600	-1,200	-1,000	1,800	1,800	-0,600	-1,400	-1,000	-0,200	0,600	0,800	16,000		4
4	0,000	2,000	0,000	0,000	0,000	1,000	2,000	1,000	0,000	0,000	0,000	80,000		2
5	0,400	-0,800	1,000	1,200	0,200	-0,400	-0,600	0,000	0,200	-0,600	-0,800	24,000		3
6	-0,600	1,200	1,000	-1,800	-0,800	0,600	1,400	1,000	0,200	0,400	1,200	44,000		11
7	0,400	0,200	0,000	0,200	0,200	0,600	0,400	0,000	0,200	0,400	0,200	164,000		
8														
9	12	6	9	1	5	13	7	8	14	15	10			

Abbildung 3-9: Auswertung der Testdaten

Für diese Lösung werden 164 Rohplatten benötigt. In diesem Fall zeigen sich auch die Schwächen des Systems. Eine nicht angegebene Kombination wird auch nicht ausgewertet. So gibt es in diesem Fall noch eine bessere Lösung. Das Finden der fehlenden Zuschnittsform überlasse ich dem Leser.

Tabelle 3-10: Zuschnittlösung

Zuschnittform	Stück	A	B	C	D
4	16	16		16	16
2	80	80	80		
3	24	24		72	
11	44			132	44
Summe		120	80	220	60

Übung 3-4: Verschnitt-Optimierungssystem

Erweitern Sie die Simplexmethode zu einem Verschnitt-Optimierungssystem, bei dem nur die Rohspanplatten-Maße und die erforderlichen Bauteilmaße angegeben werden. Das Programm soll dann die Zuschnittlösung ausweisen und ebenso die Verschnittflächen. Geben Sie außerdem den Verschnitt als prozentualen Anteil zur Gesamtfläche aus.

4 Funktionen

Immer dann, wenn eine Größe von einer anderen Größe abhängt, sprechen wir von einer Funktion oder Abbildung. Im mathematischen Sinne handelt es sich um eine Vorschrift, die jedem Element der Menge A in eindeutiger Weise ein Element der Menge B zuordnet. In allgemeiner Schreibweise

$$f : A \to B$$

Da die Technik voll von solchen Abhängigkeiten ist, können unzählige Funktionen genannt werden. Ebenso gibt es viele Algorithmen, die den Umgang mit Funktionen aus unterschiedlichen Gründen beschreiben.

4.1 Interpolation von Funktionen durch Polynome

Nicht immer liegt eine Gleichung oder ein Gleichungssystem vor. Oftmals werden in der Praxis Werte gemessen und daraus Funktionen abgeleitet. Eine Methode ist die Funktion durch ein Polynom zu ersetzen.

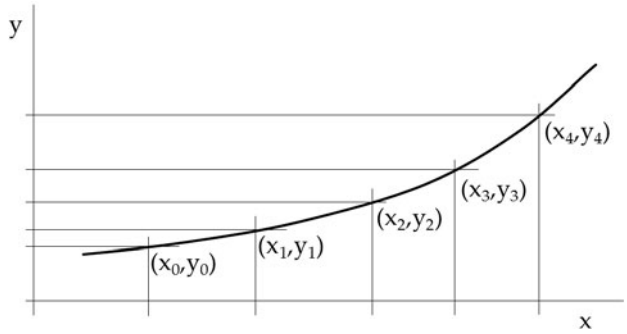


Abbildung 4-1: Stützstellen eines Interpolationspolynoms

4.1.1 Interpolation nach Newton

Dieses Verfahren geht von dem Ansatz

$$P(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \ldots + b_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \tag{4.1.1}$$

für ein Näherungspolynom aus. Die Koeffizienten b_0, b_1, …, b_n werden so bestimmt, dass das Näherungspolynom durch die Punkte (x_0,y_0), (x_1,y_1), …, (x_n,y_n) verläuft. Setzt man in den Ansatz von Newton für x die Werte x_0, x_1, …, x_n ein, so erhält man das Gleichungssystem

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_4,

$$
\begin{aligned}
y_0 &= b_0 \\
y_1 &= b_0 + b_1(x_1 - x_0) \\
y2 &= b_0 + b_1(x_1 - x_0) + b_2(x_2 - x_0)(x_2 - x_1) \\
&\ldots \\
y_n &= b_0 + b_1(x_1 - x_0) + b_2(x_2 - x_0)(x_2 - x_1) + \ldots \\
&+ b_n(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1})
\end{aligned}
\tag{4.1.2}
$$

Dieses System lässt sich schrittweise für b_0, b_1, …, b_n auslösen mit

$$
\begin{aligned}
b_0 &= y_0 \\
b_1 &= \frac{y_1 - y_0}{x_1 - x_0} = [x_1 x_0] \\
b_2 &= [x_2 x_1 x_0], usw.
\end{aligned}
\tag{4.1.3}
$$

Die Koeffizienten bestimmen sich aus den dividierten Differenzen, die allgemein definiert sind zu

$$
[x_n x_{n-1} \ldots x_1 x_0] = \frac{[x_n x_{n-1} \ldots x_1] - [x_{n-1} x_{n-2} \ldots x_0]}{x_n - x_0} \tag{4.1.4}
$$

Setzt man diese Koeffizienten in den Ansatz ein, erhält man das Interpolationspolynom nach Newton

$$
\begin{aligned}
y = y_0 &+ [x_1 x_0] \cdot (x - x_0) + [x_2 x_1 x_0] \cdot (x - x_0) \cdot (x - x_1) + \ldots \\
&+ [x_n x_{n-1} \ldots x_1 x_0] \cdot (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1})
\end{aligned}
\tag{4.1.5}
$$

Tabelle 4-1: Bestimmung eines Interpolationspolynoms nach Newton

Eingabe der x_i und y_i für i=1,…,n		
Daten lesen	i=1, 1, m+1	
		j = 1, 1, n+1
		a(i,j) = Cells(i+2,j)
z=0		
Markierung setzen	i=1, 1, n	
	z=z+1	
	Cells(m+5,i)=z	
Markierung setzen	i=1, 1, m	
	z=z+1	
	Cells(i+2,n+3)=z	

Beispiel 4.1: Stahlseilverlauf

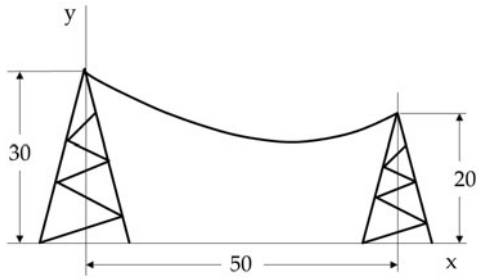


Abbildung 4-2: Stahlseilverlauf

Ein Stahlseil zwischen zwei Masten, siehe Abbildung 4-2, hat den in der Tabelle 4-2 dargestellten Verlauf.

Tabelle 4-2: Stützstellen des Seilverlaufs

x	0	10	20	30	35	40	50
y	30	18	11,5	10	10,5	12,5	20

Ein Programm soll die Koeffizienten eines Interpolationspolynoms nach Newton bestimmen und den gesamten Verlauf des Seils mit einer Schrittweite von 1 Meter. In den vorgegebenen Stützstellen müssen natürlich die vorgegebenen Funktionswerte erreicht werden, ansonsten liegt ein Fehler vor.

Code 4-1: Bestimmung der Koeffizienten eines Interpolationspolynoms nach Newton

```
Option Explicit

Sub Newton_Leer()
   Call Verlauf_Entfernen
   ThisWorkbook.Worksheets("Newton").Cells.Clear
End Sub

Sub Newton_Testdaten()
   Call Newton_Leer
   Cells(1, 1) = 0: Cells(1, 2) = 30
   Cells(2, 1) = 10: Cells(2, 2) = 18
   Cells(3, 1) = 20: Cells(3, 2) = 11.5
   Cells(4, 1) = 30: Cells(4, 2) = 10
   Cells(5, 1) = 35: Cells(5, 2) = 10.5
   Cells(6, 1) = 40: Cells(6, 2) = 12.5
   Cells(7, 1) = 50: Cells(7, 2) = 20
End Sub

Private Sub Werte_Lesen(n, A)
   Dim i, j As Integer
'
'Bestimmung belegter Zeilen
'und Definition der notwendigen Datenfelder
   Cells(Rows.Count, 1).End(xlUp).Select
   n = ActiveCell.Row
```

```
   ReDim A(n, n) As Double
   For i = 1 To n
      For j = 1 To 2
         A(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Private Sub Steigungen(n, A)
   Dim i, j As Integer

   For j = 3 To n + 1
      For i = 1 To n - j + 2
         Cells(i, j) = (Cells(i + 1, j - 1) - _
                       Cells(i, j - 1)) / _
                       (Cells(i + j - 2, 1) - Cells(i, 1))
      Next i
   Next j
   Exit Sub
End Sub

Private Sub Verlauf(n, A)
   Dim v, b, x, y, z As Double
   Dim i, j, k, p As Integer

   p = 0
   v = Cells(1, 1)
   b = Cells(n, 1)
   For x = v To b Step 1
      y = Cells(1, 2)
      For j = 3 To n + 1
         z = Cells(1, j)
         For k = 1 To j - 2
            z = z * (x - Cells(k, 1))
         Next k
         y = y + z
      Next j
      p = p + 1
      Cells(n + 2 + p, 1) = x
      Cells(n + 2 + p, 2) = y
   Next x
End Sub

Sub Auswertung()
   ReDim A(1, 1) As Double
   Dim n As Integer

   Call Werte_Lesen(n, A)
   Call Steigungen(n, A)
   Call Verlauf(n, A)
End Sub

Sub Verlauf_Zeigen()
    Range("A10:B60").Select
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    ActiveChart.SetSourceData Source:=Sheets("Newton").Range("A10:B60"), PlotBy _
```

```
        :=xlColumns
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Newton"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Seilverlauf"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrima-ry).AxisTitle.Characters.Text = "Weite [m]"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Höhe [m]"
    End With
    ActiveChart.Legend.Select
    Selection.Delete
End Sub

Sub Verlauf_Entfernen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Newton").Shapes
      Shp.Delete
   Next
End Sub
```

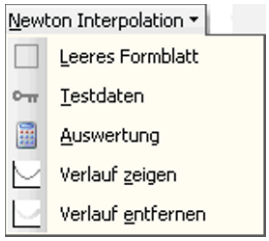


Abbildung 4-3: Menü zur Interpolation nach Newton

Übung 4.1: Interpolation nach Lagrange

Erstellen Sie eine Prozedur zur Bestimmung der Interpolationskoeffizienten nach dem Verfahren von Lagrange. Dieses geht von dem Ansatz

$$P(x) = L_0(x)y_0 + L_1(x)y_1 + L_2(x)y_2 + \ldots + L_n(x)y_n \qquad (4.1.6)$$

aus, in dem die Koeffizienten $L_i(x)$ der Stützwerte y_i wiederum Polynome n-ten Grades in x sind.

$$L_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \qquad (4.1.7)$$

	A	B	C	D	E	F	G	H
1	0	30	-1,2	0,0275	-8,3333E-05	-7,1429E-06	1,0119E-06	-5,3571E-08
2	10	18	-0,65	0,025	-0,00033333	3,3333E-05	-1,6667E-06	
3	20	11,5	-0,15	0,01666667	0,00066667	-3,3333E-05		
4	30	10	0,1	0,03	-0,00033333			
5	35	10,5	0,4	0,02333333				
6	40	12,5	0,75					
7	50	20						
8								
9								
10	0	30						
11	1	29,0965508						
12	2	28,004608						
13	3	26,785108						
14	4	25,4894034						
15	5	24,1601563						
16	6	22,832192						
17	7	21,5333155						
18	8	20,285088						
19	9	19,1035658						
20	10	18						
21	11	16,9814984						
22	12	16,051648						
23	13	15,2110993						
24	14	14,458112						
25	15	13,7890625						
26	16	13,198912						
27	17	12,6816368						
28	18	12,2306194						
29	19	11,839002						
30	20	11,5						
31	21	11,2071783						
32	22	10,954688						
33	23	10,7374655						
34	24	10,551392						
35	25	10,3934152						
36	26	10,261632						
37	27	10,155333						
38	28	10,075008						
39	29	10,0223133						
40	30	10						
41	31	10,0118045						
42	32	10,0622994						
43	33	10,1567068						
44	34	10,300672						
45	35	10,5						
46	36	10,760352						
47	37	11,0869043						
48	38	11,483968						
49	39	11,9545709						
50	40	12,5						
51	41	13,1193058						
52	42	13,808768						
53	43	14,561323						
54	44	15,365952						
55	45	16,2070313						
56	46	17,0636434						
57	47	17,9088505						
58	48	18,708928						
59	49	19,4225608						
60	50	20						

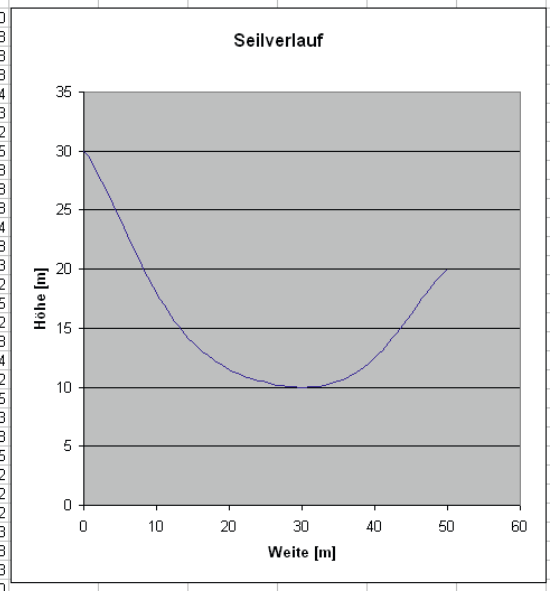


Abbildung 4-4: Auswertung der Testdaten

4.1.2 Interpolation mittels kubischer Splines

Interpolationspolynome haben die Eigenschaft, mit zunehmendem Grad (Anzahl der Stützstellen) eine immer stärker werdende Welligkeit zwischen den Stützpunkten zu zeigen. Daher werden in der Praxis Polynome vom Grad n > 5 möglichst vermieden.

Hier bedient man sich einer alten Methode zum Zeichnen möglichst „glatter“ Kurven. Im Schiffsbau und bei ähnlichen Blechkonstruktionen benutzt man ein bieg-

sames Kurvenlineal. Damit lässt sich durch vorgegebene Punkte eine Kurve ohne Knick bestimmen.

Eine Splinefunktion S(x) dritten Grades, daher auch als kubische Splinefunktion bezeichnet, ist in jedem Intervall $[x_i, x_{i+1}]$ für i=0(1)n-1 durch ein Polynom $P_i(x)$ dritten Grades bestimmt. Die Glätte wird dadurch erreicht, dass die einzelnen kubischen Parabeln sich nicht nur stetig, sondern mit stetiger zweiter Ableitung aneinander reihen. Die Berührungspunkte werden als Knoten bezeichnet. Man kann zeigen, dass kubische Splines die geringste Krümmung bei der Interpolation aufweisen.

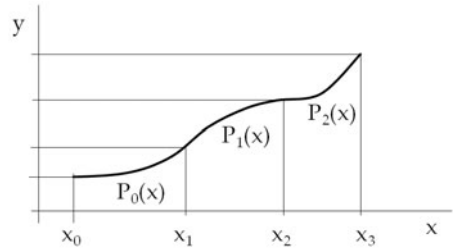


Abbildung 4-5: Aneinanderreihung kubischer Polynome

Die Konstruktion von S(x) erfolgt mit dem Ansatz

$$S(x) = P_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{4.1.8}$$

für alle $x \in [x_i, x_{i+1}]$ und n ≥ 2. Die Ableitungen ergeben sich zu

$$P_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \tag{4.1.9}$$

und

$$P_i''(x) = 2c_i + 6d_i(x - x_i). \tag{4.1.10}$$

Aus der Glättebedingung zwischen zwei Polynomen folgt

$$\begin{aligned} P_i''(x_i) &= P_{i-1}''(x_i) \\ 2c_i &= 2c_{i-1} + 6d_{i-1}(x_i - x_{i-1}) \end{aligned} \tag{4.1.11}$$

und daraus mit $h_i = x_i + 1 - x_i$

$$d_{i-1} = \frac{1}{3h_i}(c_i - c_{i-1}) \ . \tag{4.1.12}$$

Außerdem müssen die Polynome sich in jedem Knoten berühren, woraus folgt

$$\begin{aligned} P_i(x_i) &= P_{i-1}(x_i) \\ a_i &= a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \end{aligned}. \tag{4.1.13}$$

Durch Umstellung und Einsetzen von (4.1.12) folgt

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} - 2c_i). \tag{4.1.14}$$

Aus der Stetigkeit folgt weiterhin, dass auch die ersten Ableitungen im Knoten gleich sein müssen

$$P_i^{'}(x_i) = P_{i-1}^{'}(x_i)$$
$$b_i = b_{i-1} + 2c_{i-1}(x_i - x_{i-1}) + 3d_{i-1}(x_i - x_{i-1})^3 \qquad (4.1.15)$$

Durch Umstellung und Einsetzen ergibt sich ein lineares Gleichungssystem mit n-1 Gleichungen für n+1 Unbekannte bei bekannten a_i mit

$$h_{i-1}c_{i-1} + 2c_i(h_{i-1} + h_i) + h_i c_{i+1} =$$
$$\frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) \quad . \qquad (4.1.16)$$
$$i = 1(1)n-1$$

Fassen wir diesen Berechnungsalgorithmus in Struktogrammform zusammen. Der Algorithmus enthält einen Teil-Algorithmus zur Lösung eines Gleichungssystems, wie zuvor im Kapitel 3 unter Gauß Algorithmus beschrieben.

Als Beispiel zur Programmierung benutzen wir den zuvor behandelten Seilverlauf.

Tabelle 4-3: Ermittlung der Koeffizienten eines kubischen Splines

	i=0,1,n
	$a_i=y_i$
$c_0 = c_n = 0$	
	Löse Gleichungssystem für c_i i=1,1,n-1
	$h_i=x_{i+1}-x_i$
	$h_{i-1}c_{i-1} + 2c_i(h_{i-1} + h_i) + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$
	i=0,1,n-1
	$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} - 2c_i)$
	i=0,1,n-1
	$d_{i-1} = \frac{1}{3h_i}(c_i - c_{i-1})$
	i=0,1,n-1
	$P_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

Code 4-2: Interpolation mittels kubischer Splines

```
Option Explicit

Sub Newton_Leer()
   Call Verlauf_Entfernen
```

```
   ThisWorkbook.Worksheets("Newton").Cells.Clear
End Sub

Sub Newton_Testdaten()
   Call Newton_Leer
   Cells(1, 1) = 0: Cells(1, 2) = 30
   Cells(2, 1) = 10: Cells(2, 2) = 18
   Cells(3, 1) = 20: Cells(3, 2) = 11.5
   Cells(4, 1) = 30: Cells(4, 2) = 10
   Cells(5, 1) = 35: Cells(5, 2) = 10.5
   Cells(6, 1) = 40: Cells(6, 2) = 12.5
   Cells(7, 1) = 50: Cells(7, 2) = 20
End Sub

Private Sub Werte_Lesen(n, A)
   Dim i, j As Integer
'
'Bestimmung belegter Zeilen
'und Definition der notwendigen Datenfelder
   Cells(Rows.Count, 1).End(xlUp).Select
   n = ActiveCell.Row
   ReDim A(n, n) As Double
   For i = 1 To n
      For j = 1 To 2
         A(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Private Sub Steigungen(n, A)
   Dim i, j As Integer

   For j = 3 To n + 1
      For i = 1 To n - j + 2
         Cells(i, j) = (Cells(i + 1, j - 1) - _
         Cells(i, j - 1)) / (Cells(i + j - 2, 1) - _
         Cells(i, 1))
      Next i
   Next j
   Exit Sub
End Sub

Private Sub Verlauf(n, A)
   Dim v, b, x, y, z As Double
   Dim i, j, k, p As Integer

   p = 0
   v = Cells(1, 1)
   b = Cells(n, 1)
   For x = v To b Step 1
      y = Cells(1, 2)
      For j = 3 To n + 1
         z = Cells(1, j)
         For k = 1 To j - 2
            z = z * (x - Cells(k, 1))
         Next k
         y = y + z
```

```
      Next j
      p = p + 1
      Cells(n + 2 + p, 1) = x
      Cells(n + 2 + p, 2) = y
   Next x
End Sub

Sub Auswertung()
   ReDim A(1, 1) As Double
   Dim n As Integer

   Call Werte_Lesen(n, A)
   Call Steigungen(n, A)
   Call Verlauf(n, A)
End Sub

Sub Verlauf_Zeigen()
    Range("A10:B60").Select
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    ActiveChart.SetSourceData Source:= _
       Sheets("Newton").Range("A10:B60"), PlotBy:= _
       xlColumns
    ActiveChart.Location _
       Where:=xlLocationAsObject, Name:="Newton"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Seilverlauf"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, _
         xlPrimary).AxisTitle.Characters.Text = "Weite [m]"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, _
         xlPrimary).AxisTitle.Characters.Text = "Höhe [m]"
    End With
    ActiveChart.Legend.Select
    Selection.Delete
End Sub
Sub Verlauf_Entfernen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Newton").Shapes
      Shp.Delete
   Next
End Sub
```

Das Menü gestaltet sich analog zum vorherigen Beispiel. Die Auswertung der Testdaten finden Sie in Abbildung 4-7.

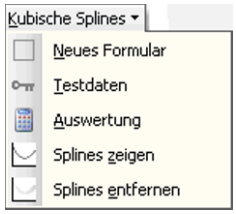


Abbildung 4-6: Menü zur Interpolation mittels kubischer Splines

	A	B	C	D	E	F	G	H	I
1	*x*	*y=a*	*b*	*c*	*d*			*x*	*y*
2	0,00	30,00	-1,32	0,00	0,00			0,00	30,00
3	10,00	18,00	-0,97	0,03	0,00			1,00	28,69
4	20,00	11,50	-0,36	0,03	0,00			2,00	27,38
5	30,00	10,00	0,01	0,01	0,00			3,00	26,08
6	35,00	10,50	0,23	0,04	0,00			4,00	24,81
7	40,00	12,50	0,56	0,03	0,00			5,00	23,57
8	50,00	20,00						6,00	22,36
9								7,00	21,19
10	*GI.Matrix*							8,00	20,07
11	40,00	10,00	0,00	0,00	0,00	1,65		9,00	19,00
12	10,00	40,00	10,00	0,00	0,00	1,50		10,00	18,00
13	0,00	10,00	30,00	5,00	0,00	0,75		11,00	17,07
14	0,00	0,00	5,00	20,00	5,00	0,90		12,00	16,20
15	0,00	0,00	0,00	5,00	30,00	1,05		13,00	15,40
16								14,00	14,66
17								15,00	13,99
18								16,00	13,38
19								17,00	12,82
20								18,00	12,33
21								19,00	11,89
22								20,00	11,50
23								21,00	11,17
24								22,00	10,88
25								23,00	10,64
26								24,00	10,45
27								25,00	10,29
28								26,00	10,17
29								27,00	10,09
30								28,00	10,03
31								29,00	10,00
32								30,00	10,00
33								31,00	10,02
34								32,00	10,07
35								33,00	10,16
36								34,00	10,30
37								35,00	10,50
38								36,00	10,77
39								37,00	11,11
40								38,00	11,51
41								39,00	11,97
42								40,00	12,50
43								41,00	13,08
44								42,00	13,72
45								43,00	14,40
46								44,00	15,13
47								45,00	15,89
48								46,00	16,67
49								47,00	17,48
50								48,00	18,31
51								49,00	19,15
52								50,00	20,00

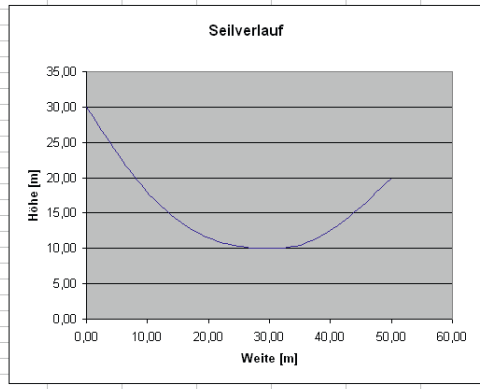


Abbildung 4-7: Durch kubische Splines ermittelter Seilverlauf

Übung 4.2: Vertauschung

Die Methode versagt, wenn der 1. Koeffizient in der 1. Gleichung, der 2. Koeffizient in der 2. Gleichung, usw. Null sind. Dieses Problem kann durch Vertauschung behoben werden, da, wenn es eine Lösung gibt, dies auch möglich ist. Der Algorithmus ist entsprechend zu ergänzen.

4.2 Approximation von Funktionen durch Polynome

Im Gegensatz zur Interpolation, wird bei der Approximation nicht verlangt, dass die Funktion in den Stützstellen den vorgegebenen Wert annimmt. Hier zählt vielmehr die bestmögliche Annäherung an den Funktionsverlauf nach einer definierten Methode.

Nachfolgend betrachten wir die Approximation nach der Methode der kleinsten Fehlerquadrate für lineare Gleichungen. Bei dieser Methode werden die Parameter so bestimmt, dass sich das Polynom mit dem kleinsten quadratischen Abstand an diese Werte anschmiegt.

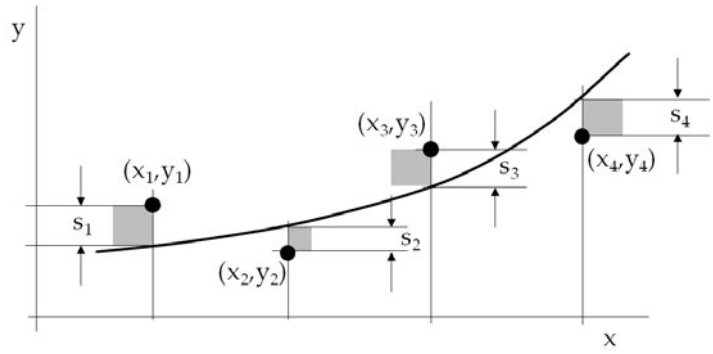


Abbildung 4-8: Grafische Darstellung der Methode der kleinsten Fehlerquadrate

Bei der linearen Approximation sind die Koeffizienten der Geraden

$$y = ax + b \tag{4.2.1}$$

gesucht, so dass die Summe der Abweichungen si zu jedem Messwert (x_i, y_i) im Quadrat aller Messwerte n ein Minimum annehmen.

Mathematisch lautet die Bedingung

$$\sum_{i=1}^{n}\left[y_i - \left(ax_i + b\right)\right]^2 = \sum_{1=1}^{n} s_i^2 = Minimum \ . \tag{4.2.2}$$

Wir erhalten ein lineares Gleichungssystem der Form

$$\begin{aligned} \alpha_{11} b + \alpha_{12} a &= \beta_1 \\ \alpha_{21} b + \alpha_{22} a &= \beta_2 \end{aligned} \ . \tag{4.2.3}$$

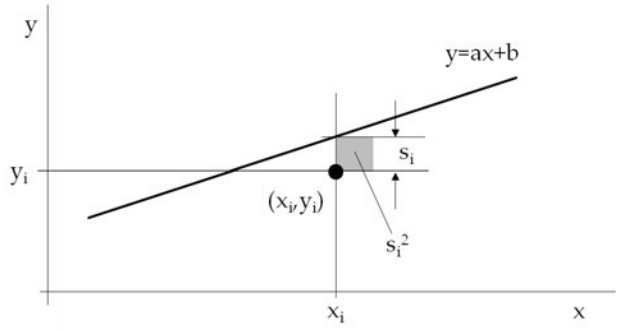


Abbildung 4-9: Lineare Approximation

In Matrizenschreibweise ergibt sich die Minimum-Bedingung

$$\underline{y} = A \cdot \underline{c} + \underline{s} \tag{4.2.4}$$

mit

$$\underline{y} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix},\ A = \begin{pmatrix} 1x_1 \\ \dots \\ 1x_n \end{pmatrix},\ \underline{c} = \begin{pmatrix} b \\ a \end{pmatrix} \text{ und } \underline{s} = \begin{pmatrix} s_1 \\ \dots \\ s_n \end{pmatrix}. \tag{4.2.5}$$

Wir erhalten die gesuchte Lösung

$$\underline{c} = (A^T \cdot A)^{-1} \cdot (A^T \cdot \underline{y}) . \tag{4.2.6}$$

Beispiel 4.2: Sensorkennlinie

Sensoren haben oft eine nichtlineare Ist-Kennlinie, die durch eine lineare Soll-Kennlinie beschrieben wird. Die Abweichung wird als Linearitätsfehler bezeichnet und wird auf zwei verschiedene Arten definiert.

Die erste Methode besteht darin, eine lineare Kennlinie durch Verbinden der beiden Endpunkte (x_{min}, y_{min}) und (x_{max}, y_{max}) zu bilden und wird Festpunktmethode genannt.

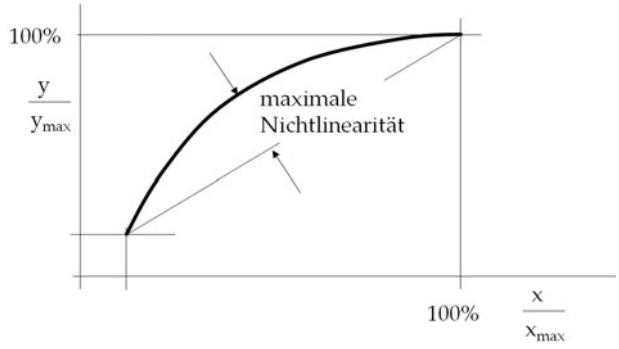


Abbildung 4-10: Lineare Approximation nach der Festpunktmethode

Die zweite Methode erstellt eine lineare Approximation nach dem Minimum der Fehlerquadrate. Diese Methode ergibt typischerweise die Hälfte des Linearitätsfehlers der Festpunktmethode.

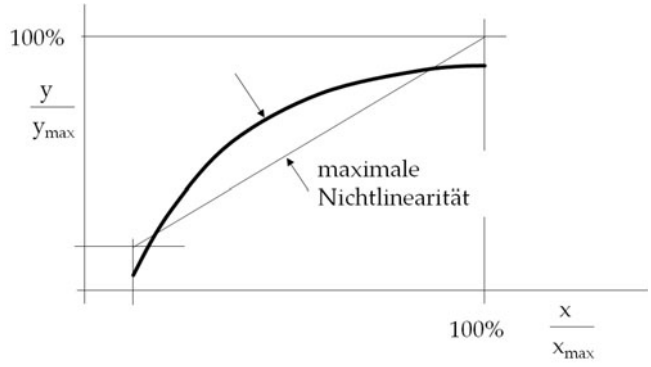


Abbildung 4-11: Lineare Approximation nach dem Minimum der Fehlerquadrate

Das nachfolgende Programm ermittelt beide Geraden nach Eingabe von beliebigen Messwerten in den ersten beiden Spalten der Tabelle.

Tabelle 4-4: Ermittlung der approximierenden Geraden

Einlesen der Daten i=1,1,n
x_i=Zelle(i,1) y_i=Zelle(i,2)
$\sum x = \sum x + x_i$
$\sum y = \sum y + y_i$
$\sum x^2 = \sum x^2 + x_i * x_i$
$\sum xy = \sum xy + x_i * y_i$
$p = \frac{y_n - y_1}{x_n - x_1}$
$q = \frac{y_n - (y_n - y_1)}{(x_n - x_1) \cdot x_n}$
Bestimmung der Geradenwerte i=1,1,n
$y_i = p \cdot x_i + q$
$p = \frac{n \cdot \sum xy - \sum x \cdot \sum y}{n \cdot \sum x^2 - \sum x \cdot \sum x}$
$q = \frac{\sum y \cdot \sum x^2 - \sum xy \cdot \sum x}{n \cdot \sum x^2 - \sum x \cdot \sum x}$
Bestimmung der Geradenwerte i=1,1,n
$y_i = p \cdot x_i + q$

Das Menü gestaltet sich analog zum vorherigen Beispiel. Die Auswertung der Testdaten finden Sie in Abbildung 4-13.

Code 4-3: Lineare Approximation

```
Option Explicit

Sub LinAppro_Leer()
   ThisWorkbook.Worksheets _
      ("Lineare Approximation").Cells.Clear
End Sub

Sub LinAppro_Testdaten()
```

```
   Call LinAppro_Leer
   Cells(1, 1) = 20: Cells(1, 2) = 20
   Cells(2, 1) = 30: Cells(2, 2) = 55
   Cells(3, 1) = 40: Cells(3, 2) = 70
   Cells(4, 1) = 50: Cells(4, 2) = 75
   Cells(5, 1) = 65: Cells(5, 2) = 82
   Cells(6, 1) = 70: Cells(6, 2) = 86
   Cells(7, 1) = 80: Cells(7, 2) = 89
   Cells(8, 1) = 90: Cells(8, 2) = 90
   Cells(9, 1) = 100: Cells(9, 2) = 91
End Sub

Private Sub Werte_Lesen(n, A)
   Dim i, j As Integer
'
'Bestimmung belegter Zeilen
'und Definition der notwendigen Datenfelder
   Cells(Rows.Count, 1).End(xlUp).Select
   n = ActiveCell.Row
   ReDim A(n, 2) As Double
   For i = 1 To 2
      For j = 1 To 2
         A(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Sub LinAppro_Auswertung()
   ReDim A(1, 1) As Double
   Dim p, q, sx, sy, sxx, sxy, y As Double
   Dim n, i As Integer
'
'Daten lesen
   Call Werte_Lesen(n, A)
'
'Festpunktmethode
   p = (Cells(n, 2) - Cells(1, 2)) / _
      (Cells(n, 1) - Cells(1, 1))
   q = Cells(n, 2) - (Cells(n, 2) - Cells(1, 2)) / _
      (Cells(n, 1) - Cells(1, 1)) * Cells(n, 1)
   Cells(1, 7) = "y ="
   Cells(1, 8) = p
   Cells(1, 9) = "x +"
   Cells(1, 10) = q
'
'Funktionswerte
   For i = 1 To n
      y = p * Cells(i, 1) + q
      Cells(i, 3) = y
   Next i
'
'Kleinste Fehlerquadrate
   For i = 1 To n
      sx = sx + Cells(i, 1)
      sxx = sxx + Cells(i, 1) * Cells(i, 1)
      sy = sy + Cells(i, 2)
      sxy = sxy + Cells(i, 1) * Cells(i, 2)
```

```
   Next i
   p = (n * sxy - sx * sy) / (n * sxx - sx * sx)
   q = (sy * sxx - sxy * sx) / (n * sxx - sx * sx)
   Cells(3, 7) = "y ="
   Cells(3, 8) = p
   Cells(3, 9) = "x +"
   Cells(3, 10) = q
'
'Funktionswerte
   For i = 1 To n
      y = p * Cells(i, 1) + q
      Cells(i, 4) = y
   Next i
End Sub

Sub LinAppro_Diagramm()

   Range("A1:D9").Select
   Charts.Add
   ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
   ActiveChart.SetSourceData Source:=_
      Sheets("Lineare Approximation").Range( "A1:D9"), _
      PlotBy:=xlColumns
   ActiveChart.SeriesCollection(1).Name = _
      "=""Sensorkennlinie"""
   ActiveChart.SeriesCollection(2).Name = _
      "=""Festpunktmethode"""
   ActiveChart.SeriesCollection(3).Name = _
      "=""Methode der kleinsten Fehlerquadrate"""
   ActiveChart.Location Where:=xlLocationAsObject, Name:= _
       "Lineare Approximation"
   With ActiveChart
       .HasTitle = True
       .ChartTitle.Characters.Text = _
          "Approximation einer Sensorkennlinie"
       .Axes(xlCategory, xlPrimary).HasTitle = True
       .Axes(xlCategory, _
           xlPrimary).AxisTitle.Characters.Text = "x/xmax %"
       .Axes(xlValue, xlPrimary).HasTitle = True
       .Axes(xlValue, _
           xlPrimary).AxisTitle.Characters.Text = "y/ymax %"
   End With
   ActiveWindow.Visible = False
End Sub

Sub LinAppro_Entfernen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Lineare Approxima-tion").Shapes
      Shp.Delete
   Next
End Sub
```

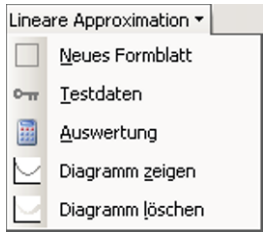
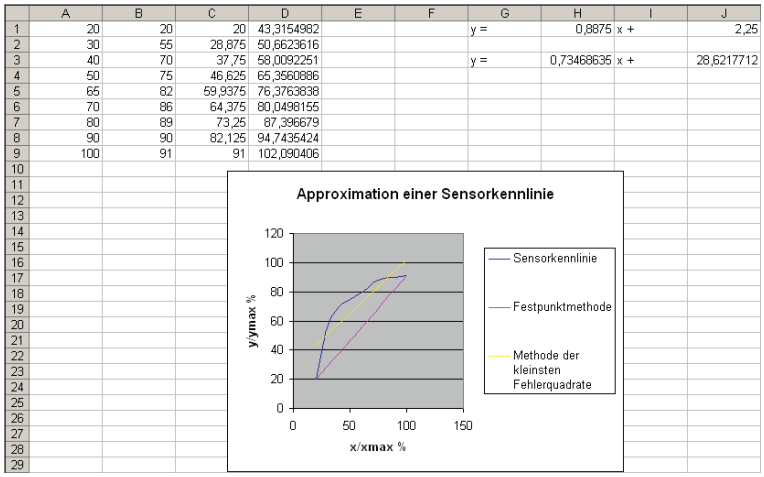


Abbildung 4-12: Menü zur linearen Approximation



	A	B	C	D	E	F	G	H	I	J
1	20	20	20	43,3154982			y =	0,8875	x +	2,25
2	30	55	28,875	50,6623616						
3	40	70	37,75	58,0092251			y =	0,73468635	x +	28,6217712
4	50	75	46,625	65,3560886						
5	65	82	59,9375	76,3763838						
6	70	86	64,375	80,0498155						
7	80	89	73,25	87,396679						
8	90	90	82,125	94,7435424						
9	100	91	91	102,090406						

Abbildung 4-13: Lineare Approximation einer Sensorkennlinie

Übung 4-3: Taylorreihe

Für komplizierte mathematische Funktionen verwendet man oft Taylorreihen, benannt nach dem Mathematiker Brook Taylor. So bereiten zum Beispiel die trigonometrischen Funktionen Probleme bei der Ableitung. Oft reichen wenige Glieder der Potenzreihe, um die Funktionen in der Umgebung bestimmter Punkte hinreichend genau zu approximieren.

Die allgemeine Form der Taylor-Reihe lautet:

$$T(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$
$$= f(a) + \frac{f'(a)}{1!}(x-a)^1 + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots \qquad (4.2.7)$$

Darin ist $f^{(n)}$ die n-te Ableitung von f(x). Oft setzt man a=0, wodurch die Formel etwas einfacher wird. Diesen Fall nennt man die MacLaurin-Reihe.

$$T(x) = f(0) + \frac{f'(0)}{1!}x^1 + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \ldots \qquad (4.2.8)$$

Um ein Taylor-Polynom vierten Grades von sin x in der Umgebung von null zu berechnen, benötigen wir alle Ableitungen $f^{(n)}(0)$.

$$
\begin{aligned}
&f(x) = \sin x, f(0) = 0\\
&f'(x) = \cos x, f'(0) = 1\\
&f''(x) = -\sin x, f''(0) = 0\\
&f'''(x) = -\cos x, f'''(0) = -1\\
&f''''(x) = \sin x, f''''(0) = 0
\end{aligned}
\tag{4.2.9}
$$

Damit ergibt sich die Taylorreihe

$$f(0) = f'(0)x + \frac{f''(0)}{2}x^2 + \frac{f'''(0)}{6}x^3 + \frac{f''''(0)}{24}x^4 + \frac{f'''''(0)}{120}x^5 + \cdots . \tag{4.2.10}$$

Ausgewertet

$$f(0) = x - \frac{x^3}{6} + \frac{x^5}{120} . \tag{4.2.11}$$

Schreiben Sie ein Programm, das die exakten und angenäherten Werte in der Umgebung von Null berechnet und stellen Sie deren Verlauf in einem Diagramm dar. Machen Sie sich mit dem Begriff des Konvergenzbereichs vertraut und bestimmen Sie den Konvergenzradius.

4.3 Numerische Integration

Die geometrische Deutung eines bestimmten Integrals ist die Fläche die zwischen x=a und x=b und von der Funktion y=f(x) und der x-Achse eingeschlossen wird.

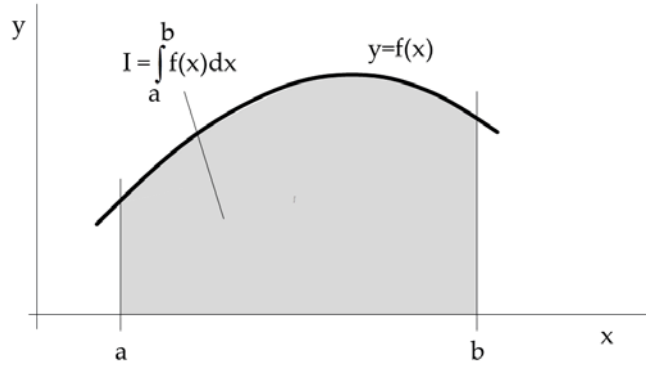


Abbildung 4-14: Bestimmtes Integral

In vielen Fällen ist zwar die Funktion y=f(x) bekannt, aber es gibt keinen analytischen Ansatz zur Integration. Hier bedient man sich der Trapezregel oder der Regel nach Simpson. Bei der Trapezregel wird das Intervall (a, b) in n gleich große Abschnitte unterteilt und die Kurvenstücke werden durch Gerade ersetzt.

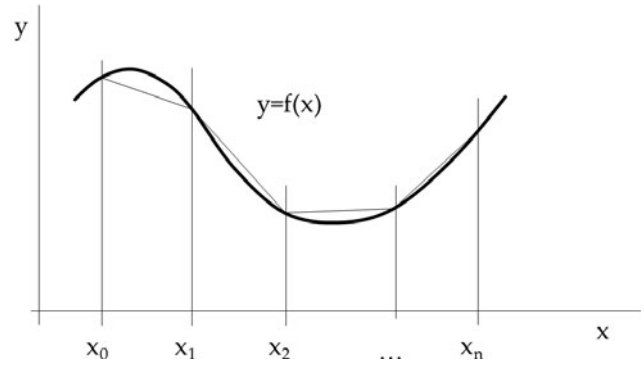


Abbildung 4-15: Einteilung nach der Trapezregel

Ist h die Breite eines Trapezes und n die Anzahl der Intervalle so gilt

$$I \approx \frac{h}{2}(y_0 + y_1) + \frac{h}{2}(y_1 + y_2) + \ldots \frac{h}{2}(y_{n-1} + y_n)$$
$$\approx \frac{h}{2}(f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)) \qquad (4.3.1)$$

mit

$$h = \frac{b-a}{n}. \qquad (4.3.2)$$

Tabelle 4-5: Numerische Integration nach der Trapezregel

Eingabe der erforderlichen Parameter x_0, x_n, n, f(x)
$h = \frac{b-a}{n}$
i=0 (1) n-1
$\sum y = \sum y + f(x_i) + f(x_{i+1})$
$I = \frac{h}{2}\sum y$

Die Trapezregel liefert umso genauere Werte, je kleiner h ist bzw. je mehr Intervalle n gesetzt werden. Die praktische Grenze liegt in den Rundungsfehlern der Berechnung. Bei der Methode nach Simpson werden die Geraden durch Kurvenstücke über mehrere Intervalle ersetzt. Bei dieser Methode erwartet man in der Regel auch eine höhere Genauigkeit.

Beispiel 4.3: Träger gleicher Zugfestigkeit

Gesucht ist das Profil eines Stabes, der nach Abbildung 4-16 einer Zugkraft unterliegt, die in jedem Querschnitt konstant sein soll.

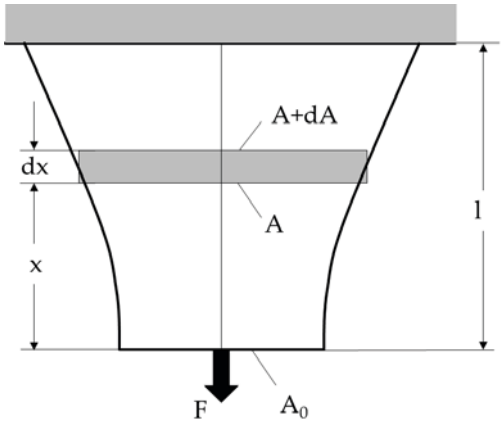


Abbildung 4-16: Träger mit gleicher Zugkraft

An einer beliebigen Stelle x mit dem Querschnitt A ergibt sich die Zugspannung

$$\sigma_x = \frac{F_x}{A} . \tag{4.3.3}$$

Darin ist F_x die Kraft an der Stelle x, die sich zusammensetzt aus der äußeren Kraft F und dem Gewicht der Masse unterhalb von x.

An der Stelle x+dx mit dem Querschnitt A+dA ergibt sich die Zugspannung

$$\sigma_{x+dx} = \frac{F_x + \rho \cdot A \cdot dx \cdot g}{A + dA} . \tag{4.3.4}$$

Darin ist ϱ die Materialdichte und g die Erdbeschleunigung. Da die Spannung über die ganze Trägerlänge l konstant sein soll, folgt aus der Gleichsetzung

$$\frac{dA}{A} = \frac{\rho \cdot g}{\sigma} \cdot dx . \tag{4.3.5}$$

Das Integral liefert

$$\int \frac{dA}{A} = \ln A \ , \tag{4.3.6}$$

so dass sich ein logarithmischer Verlauf ergibt. Wir wollen dieses Integral auf numerischem Wege lösen. Die einfachste Form bietet die Rechteckregel.

Tabelle 4-6: Träger gleicher Zugspannung

Eingabe der erforderlichen Parameter A_0, l, ρ, σ, n	
$\Delta x = \frac{l}{n}$	
	x=Δx,(Δx),l
	$\Delta A_x = \frac{\rho \cdot g}{\sigma} \cdot A_{x-1} \cdot \Delta x$
	$A_x = A_{x-1} + \Delta A_x$

Die Erstellung des Programms erfolgt nach dem üblichen Schema Formblatt-Testdaten-Auswertung-Grafik.

Code 4-4: Träger gleicher Zugspannung

```
Option Explicit

Sub Zugspannung_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Konstante Zugspan-nung").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Konstante" & _
      "Zugspannung").Cells.Clear

   Range("A1:E1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Träger mit konstanter Zugspannung"
   Range("A2:A16").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A2") = "Ao [m" & ChrW(178) & " ]"
   Range("A3") = "l [m]"
   Range("A4") = ChrW(961) & " [kg/m" & ChrW(179) & " ]"
   Range("A5") = ChrW(963) & " [N/m" & ChrW(178) & " ]"
   Range("A6") = "n"
   Range("B:B").ColumnWidth = "15"
   Range("C:C").ColumnWidth = "2"
   Range("D2") = "x [m]"
   Range("E2") = "A [m" & ChrW(178) & " ]"
   Range("D2:E2").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("B2").Select
End Sub

Sub Zugspannung_Testdaten()
   Cells(2, 2) = 0.2
   Cells(3, 2) = 0.5
   Cells(4, 2) = 0.00785
   Cells(5, 2) = 0.01
   Cells(6, 2) = 50
End Sub

Sub Zugspannung_Auswertung()
   Dim A0, l, r, S, n As Double
   Dim dx, x, dA, Ax As Double
   Dim i As Integer

   A0 = Cells(2, 2)
   l = Cells(3, 2)
   r = Cells(4, 2)
   S = Cells(5, 2)
   n = Cells(6, 2)
   dx = l / n
```

```
   Ax = A0
   i = 2
   For x = dx To l + dx Step dx
      dA = r * 9.81 / S * Ax * dx
      Ax = Ax + dA
      i = i + 1
      Cells(i, 4) = x
      Cells(i, 5) = Ax
   Next x
End Sub

Sub Zeige_Querschnittsverlauf()
   Range("D3:E52").Select
   Charts.Add
   ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
   ActiveChart.SetSourceData Source:= _
      Sheets("Konstante Zugspannung").Range("D3:E52"), _
         PlotBy:=xlColumns
   ActiveChart.SeriesCollection(1).Name = _
      "=""Querschnittsverlauf"""
   ActiveChart.Location Where:= _
      xlLocationAsObject, Name:="Konstante Zugspannung"
   With ActiveChart
       .HasTitle = True
       .ChartTitle.Characters.Text = _
           "Träger mit konstanter Spannung"
       .Axes(xlCategory, xlPrimary).HasTitle = True
       .Axes(xlCategory, _
           xlPrimary).AxisTitle.Characters.Text = "x [m]"
       .Axes(xlValue, xlPrimary).HasTitle = True
       .Axes(xlValue, _
           xlPrimary).AxisTitle.Characters.Text = "A [m^2]"
   End With
   ActiveChart.Legend.Select
   Selection.Left = 229
   Selection.Top = 274
   ActiveChart.Axes(xlValue).MajorGridlines.Select
   ActiveChart.PlotArea.Select
   Selection.Width = 314
   ActiveWindow.Visible = False
End Sub

Sub Lösche_Querschnittsverlauf()
   Dim Shp As Shape
   For Each Shp In Worksheets("Konstante Zugspan-nung").Shapes
      Shp.Delete
   Next
End Sub
```

Die Prozeduren werden wiederum durch Menüpunkte im Menü *Konstante Zugspannung* aufgerufen.

Abbildung 4-17: Menü zur konstanten Zugspannung

	A	B	C	D	E	F	G
1	*Träger mit konstanter Zugspannung*						
2	*Ao [m²]*	0,2		*x [m]*	*A [m²]*		
3	*l [m]*	0,5		0,01	0,2154017		
4	*ρ [kg/m³]*	0,00785		0,02	0,23198946		
5	*σ [N/m²]*	0,01		0,03	0,24985462		
6	*n*	50		0,04	0,26909555		
7				0,05	0,2898182		
8				0,06	0,31213666		
9				0,07	0,33617384		
10				0,08	0,36206208		
34				0,32	2,1480391		
35				0,33	2,31345637		
36				0,34	2,49161218		
37				0,35	2,6834875		

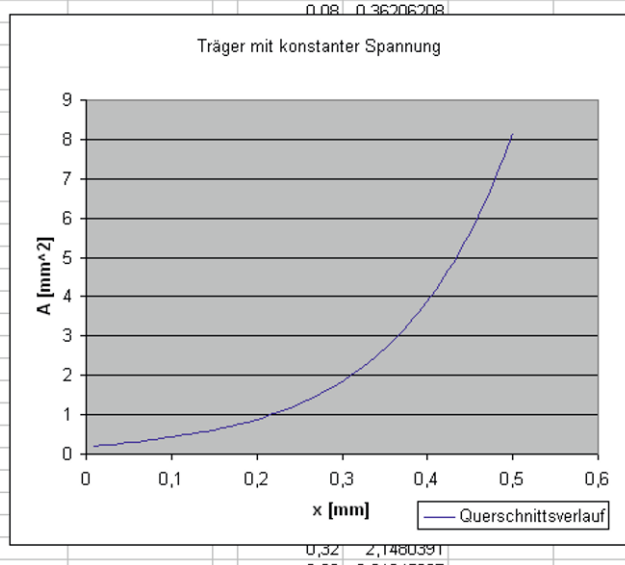


Abbildung 4-18: Auswertung der Testdaten

Abbildung 4-18 zeigt das Ergebnis aus den Testdaten. Daraus geht hervor, dass der Querschnitt des Stabes sich exponentiell verändern muss, um eine konstante Zugspannung zu gewährleisten.

Übung 4.3: Trapezregel

Der Algorithmus berechnet den Querschnittsverlauf nach der Rechteckregel. Zuvor haben wir schon die Trapezregel kennen gelernt. Ändern Sie die Berechnung in die nach der Trapezregel ab.

Der Querschnittsverlauf sagt noch nichts über die eigentliche Trägerform (rund, quadratisch, rechteckig, …) aus. Ergänzen Sie das Programm um diese Berechnungsmöglichkeiten.

Beispiel 4.4: Ausflusszeit von Flüssigkeiten

Wir betrachten einen Behälter, in dem sich Flüssigkeit mit der Höhe h befindet.

Nach dem italienischen Physiker Torricelli bestimmt sich die Ausflussgeschwindigkeit aus

$$v = \sqrt{2 \cdot g \cdot h} \ . \tag{4.3.7}$$

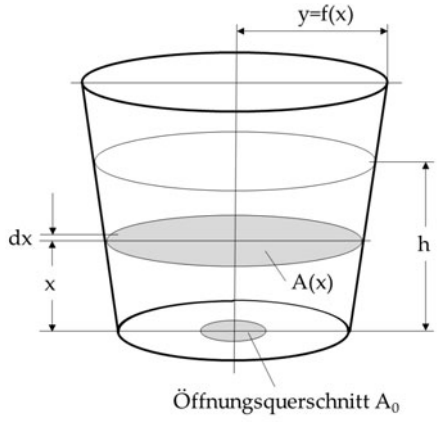


Abbildung 4-19: Ausfluss aus einem Behälter

Ebenso bestimmt sich die Ausflussmenge aus

$$Q = A \cdot v \ . \tag{4.3.8}$$

Der variable Gefäßquerschnitt ist durch die Funktion

$$y = f(x) \tag{4.3.9}$$

gegeben. Zum Zeitpunkt t beträgt die Flüssigkeitshöhe x. Für ein kleines Zeitintervall dt sinkt der Flüssigkeitsspiegel um dx und die austretende Menge berechnet sich, unter der Idealisierung, dass der Querschnitt im Zeitraum dt konstant ist, aus

$$A_O \cdot \sqrt{2 \cdot g \cdot x} \cdot dt = A(x) \cdot dx \ . \tag{4.3.10}$$

Umgestellt ergibt sich für das Zeitelement

$$dt = \frac{A(x) \cdot dx}{A_O \sqrt{2 \cdot g \cdot x}} \ . \tag{4.3.11}$$

Die Integration zum Zeitpunkt t, zu dem der Flüssigkeitsspiegel die Höhe x hat, liefert

$$\int_0^t dt = \int_h^x \frac{A(x) \cdot dx}{A_O \sqrt{2 \cdot g \cdot x}} \quad . \tag{4.3.12}$$

Die Lösung lautet

$$t = \frac{1}{A_O \sqrt{2 \cdot g}} \int_h^x \frac{A(x)}{\sqrt{x}} dx \quad . \tag{4.3.13}$$

Damit können wir den Berechnungsalgorithmus aufstellen. Diesmal verwenden wir einmal die Trapezregel zur Berechnung des Integrals und gleichzeitig bestimmen wir die Zeit aus dem Differenzenquotienten. Dazu gibt es nach Abbildung 4-20 drei Möglichkeiten.

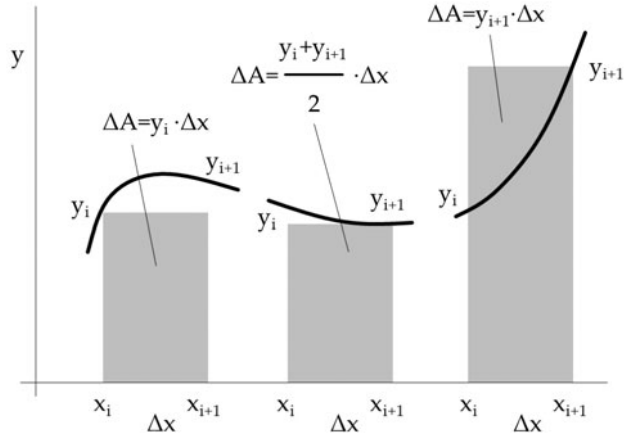


Abbildung 4-20: Vorschrittige, rückschrittige und zentrale Differenzen

Im Algorithmus werden die Differenzen in einer Programmschleife bestimmt, während dabei nur die Summe der Funktionen für die Berechnung nach der Trapezregel entsprechend der Darstellung in Tabelle 4-5 erfolgt. Nach der Programmschleife wird dann auch die Ausflusszeit nach der Trapezregel bestimmt.

Tabelle 4-7: Numerische Integration mit Trapezregel und Differenzen

Eingabe der erforderlichen Parameter A_O, h, u, n
$\Delta x = \frac{h-u}{n}$
$t_v = 0, t_r = 0, t_m = 0$
$k = \frac{1}{A_O \sqrt{2 \cdot g}}$
x_i = h (-Δx) u+Δx

	$y_i = \frac{A(x_i)}{\sqrt{x_i}}$
	$y_{i+1} = \frac{A(x_{i+1})}{\sqrt{x_{i+1}}}$
	$\sum y = y_i + y_{i+1}$
	$\Delta t_v = k \frac{A(x_{i)}}{\sqrt{x_i}} \Delta x$
	$t_v = t_v + \Delta t_v$
	$\Delta t_r = k \frac{A(x_{i+1)}}{\sqrt{x_{i+1}}} \Delta x$
	$t_r = t_r + \Delta t_r$
	$\Delta t_m = k \frac{A(x_i) + A(x_{i+1)}}{2\sqrt{\frac{x_i + x_{i+1}}{2}}} \Delta x$
	$t_m = t_m + \Delta t_m$
$t = \frac{\Delta x}{2} \sum y$	

Für die Form des Trichters wird eine eigene Prozedur geschrieben. So können auch andere Formen integriert werden.

Code 4-5: Bestimmung der Ausflusszeit aus einem Gefäß

```
Option Explicit

Sub Ausflusszeit_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Ausflusszeit").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Ausflusszeit").Cells.Clear

   Range("A1:E1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = _
      "Ausflusszeit bei abnehmendem Flüssigkeitsstand"

   Range("A2:A7").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A2") = "A [mm" & ChrW(178) & " ]"
   Range("A3") = "h [mm]"
   Range("A4") = "u [mm]"
   Range("A5") = "n"
   Range("A7") = "t (TR) [s]"
```

```
   Range("B:B").ColumnWidth = "15"
   Range("C:C").ColumnWidth = "2"
   Range("D2") = "x [mm]"
   Range("E2") = "y1"
   Range("F2") = "y2"
   Range("G2") = "t1 [s]"
   Range("H2") = "t2 [s]"
   Range("I2") = "tm [s]"
   Range("J2") = "dtm [s]"
   Range("D2:J2").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True

   Range("B2").Select
End Sub

Sub Ausflusszeit_Testdaten()
   Cells(2, 2) = 10
   Cells(3, 2) = 400
   Cells(4, 2) = 10
   Cells(5, 2) = 50
End Sub

Sub Trichter_Form(h, x, Ax)
   Dim r As Double
   r = 100 + 100 / h * x
   Ax = r * r * 4 * Atn(1)
End Sub

Sub Ausflusszeit_Auswertung()
   Dim A, Ax, Ax1, Ax2, h, u, dx, x, y1, y2 As Double
   Dim k, dt, dt1, dt2, t, t1, t2, Su As Double
   Dim i, n As Integer

   A = Cells(2, 2)
   h = Cells(3, 2)
   u = Cells(4, 2)
   n = Cells(5, 2)
   i = 2

   dx = (h - u) / n
   t1 = 0: t2 = 0: t = 0
   k = 1 / (A * Sqr(2 * 9810))
   For x = h To u + dx Step -dx
      Call Trichter_Form(h, x, Ax1)
      Call Trichter_Form(h, x - dx, Ax2)

'Summation nach der Trapezregel
      y1 = Ax1 / Sqr(x)
      Su = Su + y1
      y2 = Ax2 / Sqr(x - dx)
      Su = Su + y2

'Bestimmung von dt aus dem Differenzenquotienten
      dt1 = k * Ax1 / Sqr(x) * dx
      t1 = t1 + dt1
      dt2 = k * Ax2 / Sqr(x - dx) * dx
```

```
      t2 = t2 + dt2
      Ax = (Ax1 + Ax2) / 2
      dt = k * Ax / Sqr(x - dx / 2) * dx
      t = t + dt

'Ausgabe
      i = i + 1
      Cells(i, 4) = x
      Cells(i, 5) = y1
      Cells(i, 6) = y2
      Cells(i, 7) = t1
      Cells(i, 8) = t2
      Cells(i, 9) = t
      Cells(i, 10) = dt
   Next x

'Bestimmung der Ausflusszeit nach der Trapezregel
   t = Su * dx / 2 / (A * Sqr(2 * 9810))
   Cells(7, 2) = t
End Sub

Sub Zeitdifferenzen_zeigen()
    Range("D3:D51").Select
    ActiveWindow.ScrollRow = 14
    ActiveWindow.ScrollRow = 13
    ActiveWindow.ScrollRow = 12
    ActiveWindow.ScrollRow = 11
    ActiveWindow.ScrollRow = 10
    ActiveWindow.ScrollRow = 9
    ActiveWindow.ScrollRow = 8
    ActiveWindow.ScrollRow = 7
    ActiveWindow.ScrollRow = 6
    ActiveWindow.ScrollRow = 5
    ActiveWindow.ScrollRow = 4
    ActiveWindow.ScrollRow = 3
    ActiveWindow.ScrollRow = 2
    ActiveWindow.ScrollRow = 1
    Range("D3:D51,J3:J51").Select
    Range("J3").Activate
    Charts.Add
    ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
    ActiveChart.SetSourceData Source:= _
        Sheets("Ausflusszeit").Range( _
        "D3:D51,J3:J51"), PlotBy:=xlColumns
    ActiveChart.Location Where:= _
        xlLocationAsObject, Name:="Ausflusszeit"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Zeitverhalten"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, _
            xlPrimary).AxisTitle.Characters.Text = "h [mm]"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, _
            xlPrimary).AxisTitle.Characters.Text = "dt [s]"
    End With
    ActiveChart.HasLegend = False
```

```
End Sub

Sub Zeitdifferenzen_löschen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Ausflusszeit").Shapes
      Shp.Delete
   Next
End Sub
```

Auch dieses Beispiel erhält wieder die klassische Aufteilung des Menüs.

Abbildung 4-21: Menü zu den Ausflusszeiten

Die Auswertung der Testdaten in Abbildung 4-22 zeigt einen interessanten Verlauf. Danach nimmt die Zeitdifferenz Δt für ein Volumenelement Δx ständig ab bis zu einer Höhe von 135 mm. Danach nimmt sie wieder zu, und dies liegt an der geringer werdenden Masse der Flüssigkeit.

Übung 4.4: Nullproblem

In der Berechnung kann als unterster Wert nicht Null eingegeben werden, weil sonst innerhalb der Berechnung eine Division durch Null erfolgt. Lösen Sie dieses Problem. Vielleicht hilft Ihnen die Lösung des Integrals weiter? Ergänzen Sie außerdem die Berechnung durch andere Behälterformen.

	A	B	C	D	E	F	G	H	I	J
1	*Ausflusszeit bei abnehmendem Flüssigkeitsstand*									
2	*A [mm²]*	10		*x [mm]*	*y1*	*y2*	*t1 [s]*	*t2 [s]*	*tm [s]*	*dtm [s]*
3	*h [mm]*	400		400	6283,18531	6222,22586	34,9884714	34,6490134	34,819156	34,819156
4	*u [mm]*	10		392,2	6222,22586	6161,8819	69,6374848	68,9619961	69,3005631	34,481407
5	*n*	50		384,4	6161,8819	6102,17284	103,950468	102,942484	103,447702	34,1471387
6				376,6	6102,17284	6043,11951	137,930955	136,594128	137,264165	33,8164631
7	*t (TR) [s]*	1480,425774		368,8	6043,11951	5984,74434	171,582599	169,920704	170,753666	33,4895006
8				361	5984,74434	5927,07149	204,909176	202,926124	203,920046	33,1663808
9				353,2	5927,07149	5870,12704	237,914596	235,614444	236,76729	32,8472434
10				345,4	5870,12704	5813,93922	270,602916	267,989878	269,299529	32,5322396
11								300,056808	301,521062	32,2215328
12							279	331,819803	333,436363	31,9153006
13							274	363,283633	365,050099	31,6137358
14							05	394,453288	396,367147	31,3170485
15							759	425,333996	427,392615	31,0254682
16							168	455,93125	458,131861	30,7392458
17							721	486,25083	488,590518	30,458657
18							301	516,298835	518,774523	30,1840045
19							806	546,081715	548,690145	29,9156227
20							87	575,60631	578,344026	29,653881
21							781	604,87989	607,743216	29,3991895
22							861	633,910209	636,89522	29,1520046
23							681	662,70556	665,808056	28,9128356
24							031	691,27484	694,49031	28,6822536
25							911	719,627629	722,95121	28,4609008
26							61	747,774277	751,200713	28,2495023
27							749	775,726015	779,249593	28,0488807
28							486	803,495072	807,109566	27,8599731
29							643	831,094829	834,793419	27,6838525
30							801	858,539996	862,315173	27,5217536
31							467	885,84682	889,690278	27,3751057
32							991	913,033352	916,935852	27,2455735
33				166	4882,12291	4864,14278	948,021823	940,11976	944,07096	27,1351084
34				158,2	4864,14278	4850,23546	975,108231	967,128724	971,116975	27,046015
35				150,4	4850,23546	4840,94007	1002,11719	994,085925	998,098011	26,9810364
36				142,6	4840,94007	4836,90704	1029,0744	1021,02067	1025,04148	26,9434667
37				134,8	4836,90704	4838,92911	1056,00914	1047,96667	1051,97878	26,9372995
38				127	4838,92911	4847,98347	1082,95514	1074,9631	1078,94621	26,967429
39				119,2	4847,98347	4865,29011	1109,95157	1102,05589	1105,98613	27,039926
40				111,4	4865,29011	4892,39434	1137,04436	1129,29962	1133,14856	27,1624224
41				103,6	4892,39434	4931,28588	1164,28809	1156,75992	1160,49322	27,3446623
42				95,8	4931,28588	4984,57572	1191,74839	1184,51697	1188,09252	27,5993071
43				88	4984,57572	5055,76661	1219,50544	1212,67045	1216,03567	27,9431489
44				80,2	5055,76661	5149,68168	1247,65893	1241,34691	1244,43467	28,398997
45				72,4	5149,68168	5273,17327	1276,33538	1270,71104	1273,4334	28,9987308
46				64,6	5273,17327	5436,3563	1305,69951	1300,98387	1303,22187	29,7884696
47				56,8	5436,3563	5654,89483	1335,97234	1332,47365	1334,05971	30,8378434
48				49	5654,89483	5954,59584	1367,46212	1365,63234	1366,31755	32,2578375
49				41,2	5954,59584	6381,67188	1400,62081	1401,16925	1400,556	34,2384481
50				33,4	6381,67188	7029,31727	1436,15772	1440,31262	1437,69465	37,1386512
51				25,6	7029,31727	8123,75393	1475,30109	1485,55046	1479,43821	41,7435587

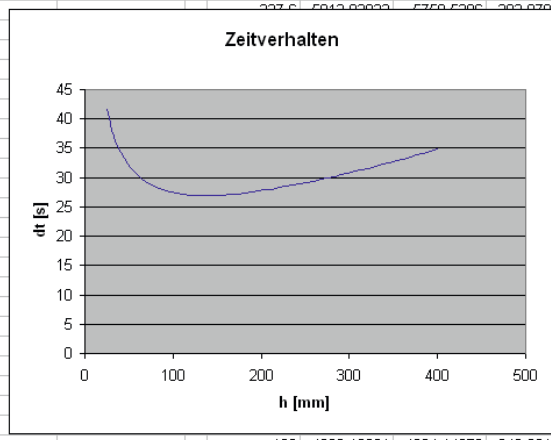


Abbildung 4-22: Bestimmung der Ausflusszeit des Testbeispiels

5 Differentialgleichungen

Bei den Differentialgleichungen zeigt sich die Anwendbarkeit der Mathematik in der Technik besonders deutlich. Bei Differentialgleichungen handelt es sich um Gleichungen, die zur Berechnung einer bestimmten Funktion dienen. Das wesentliche einer Differentialgleichung ist, dass neben der Funktion oder der unabhängig Veränderlichen auch mindestens eine Ableitung der gesuchten Funktion auftritt.

Die Beschäftigung mit Differentialgleichungen begann zeitgleich mit der Einführung der Differential- und Integralrechnung durch Newton und Leibniz zum Ausgang des 17. Jahrhunderts.

5.1 Gewöhnliche Differentialgleichungen

Die numerische Behandlung einer Differentialgleichung lässt sich nach vielen Methoden durchführen.

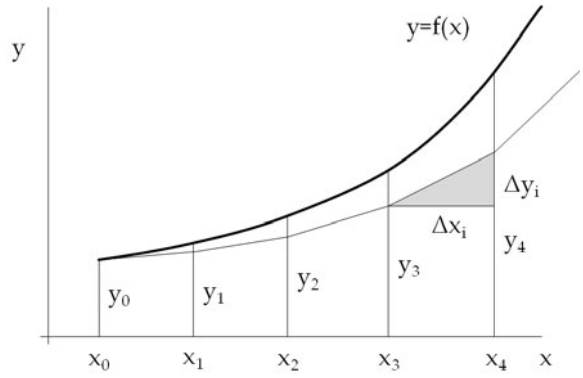


Abbildung 5-1: Näherung nach Euler-Cauchy

Das Euler-Cauchy-Verfahren ist eine einfache Methode und hat damit eine größere Fehlerrate gegenüber anderen Methoden. Da sie aber einfach zu handhaben ist, soll sie hier benutzt werden.

Es sei

$$y = f(x) \tag{5.1.1}$$

die analytische Lösung der Differentialgleichung

$$y' = f(x, y) \ . \tag{5.1.2}$$

Aus der Differentialgleichung folgt die Anfangsbedingung

$$y_0' = f(x_0, y_0) \tag{5.1.3}$$

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_5,

als bekannter Wert. Eine Veränderung des Abzissenwertes

$$x_1 = x_0 + \Delta x \tag{5.1.4}$$

ergibt den neuen Ordinatenwert

$$y_1 = y_0 + y_0' \cdot \Delta x \ . \tag{5.1.5}$$

Auf diese Weise erhält man einen Polygonzug, der der gesuchten Lösungsfunktion angenähert ist. Bei diesem Verfahren, wird also der Differentialquotient

$$\frac{dy}{dx} \tag{5.1.6}$$

durch den Differenzenquotienten

$$\frac{\Delta y}{\Delta x} \tag{5.1.7}$$

ersetzt.

Wählt man Δx genügend klein, kommt man der analytischen Lösung beliebig nahe. Der Nachteil wird aus der Darstellung ebenfalls recht deutlich. Mit zunehmenden Schritten entfernt sich die numerische Lösung immer mehr von der analytischen. Ein genaueres Verfahren ist z. B. das Runge-Kutta-Verfahren.

Die Betrachtung des Kräftegleichgewichts an beweglichen Massenpunkten führt unmittelbar zu einer Differentialgleichung. Ob dies nun einfache Bewegungsmodelle wie z.B. Rotationen sind oder komplexere Modelle wie z.B. Schwingungssysteme. Der Änderung des Bewegungszustandes setzt der Massenpunkt seine träge Masse entgegen

$$F = -m \cdot a \ . \tag{5.1.8}$$

Die Momentanbeschleunigung a ergibt sich als Differentialquotient

$$a = \frac{dv}{dt} = v' \ , \tag{5.1.9}$$

so dass die erste Anwendung des Euler-Cauchy-Verfahrens die während der Zeiteinheit dt auftretende Geschwindigkeitsänderung dv liefert

$$v = -\frac{F}{m} t \ . \tag{5.1.10}$$

Für die Momentangeschwindigkeit gilt weiterhin der Differentialquotient

$$v = \frac{ds}{dt} = s' \ , \tag{5.1.11}$$

so dass die zweite Anwendung des Euler-Cauchy-Verfahrens den während der Zeiteinheit dt zurückgelegten Weg ds annähernd beschreibt

$$v = \frac{ds}{dt} = s' \ . \tag{5.1.12}$$

Die Momentangeschwindigkeit wird dabei aus Einfachheitsgründen für die Berechnung durch die Anfangsgeschwindigkeit des betrachteten Zeitintervalls ersetzt.

Beispiel 5-1: Schubkurbeltrieb

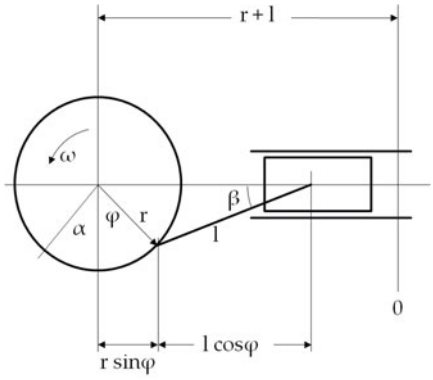


Abbildung 5-2: Schubkurbeltrieb

Der Kurbeltrieb dient zur Umwandlung von Schub- in Drehbewegung und umgekehrt. Der algebraische Ausdruck für die Kolbenbewegung ergibt sich, unter Betrachtung der Abbildung 5-2, aus

$$x = l + r - l \cdot \cos\beta - r \cdot \sin\varphi \ , \tag{5.1.13}$$

Weiterhin ist

$$\lambda = \frac{r}{l} = \frac{\sin\beta}{\cos\varphi} \ , \tag{5.1.14}$$

$$\cos\beta = \sqrt{1 - \lambda^2 \cos^2\varphi} \ , \tag{5.1.15}$$

$$x = r \cdot (1 - \sin\varphi) + l \cdot (1 - \sqrt{1 - \lambda^2 \cos^2(\varphi)} \ . \tag{5.1.16}$$

Bei allgemeiner Phasenverschiebung

$$x = r \cdot (1 - \sin\varphi) + l \cdot (1 - \sqrt{1 - \lambda^2 \cos^2(\varphi - \alpha)} \ . \tag{5.1.17}$$

Die Geschwindigkeit und Beschleunigung ergeben sich angenähert aus den Differenzenquotienten

$$v = \frac{\Delta x}{\Delta t} \tag{5.1.18}$$

und

$$a = \frac{\Delta v}{\Delta t} \ . \tag{5.1.19}$$

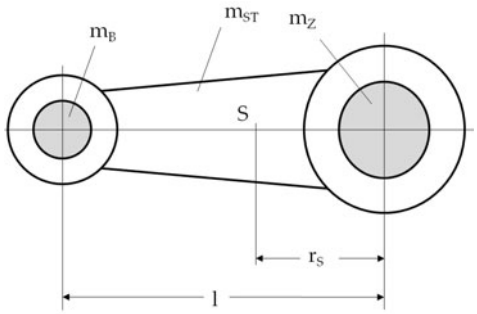


Abbildung 5-3: Schubstange und Bolzen

Entsprechend den Bewegungen der Triebwerksteile unterscheidet man oszillierende und rotierende Massen. Die oszillierende Masse bestimmt sich aus

$$m_O = m_{ST} \cdot \frac{r_{ST}}{l} + m_K + m_B \ . \tag{5.1.20}$$

Darin ist m_{ST} der Massenanteil der Schubstange, der durch den Faktor r_{ST}/l seinen oszillierenden Anteil hat, m_K die Kolbenmasse und m_B die Masse des Kolbenbolzens.

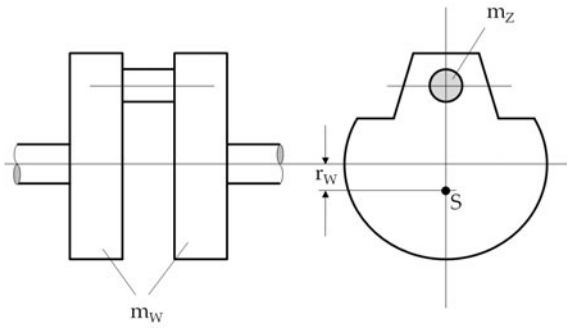


Abbildung 5-4: Kurbelwangen und Kurbelzapfen

Die rotierenden Massenanteile setzen sich aus

$$m_R = m_{ST} \cdot \frac{l - r_{ST}}{l} + m_W \cdot \frac{r_W}{r} + m_Z + m_N \tag{5.1.21}$$

zusammen. Darin ist $m_{ST}(l\text{-}r_{ST})/l$ der rotierende Massenanteil der Schubstange, m_W die Kurbelwangenmasse, die durch den Faktor r_W/r auf den Drehmittelpunkt reduziert werden muss, m_Z die Kurbelzapfenmasse und m_N die Nadellagermasse. Die oszillierende Massenkraft ergibt sich damit aus

$$F_O = m_O \cdot a_K \tag{5.1.22}$$

und die rotierende Massenkraft

$$F_R = m_R \cdot r \cdot \omega^2 \ . \tag{5.1.23}$$

Die durch die Zündung eines Gasgemisches auf den Kolben einwirkende Kraft, sorgt für eine Entspannungsbewegung des Systems, d. h. eine Vergrößerung des Zylinderraumes durch die Kolbenbewegung. Die Kraft liegt in der Regel indirekt als Indikatordiagramm vor.

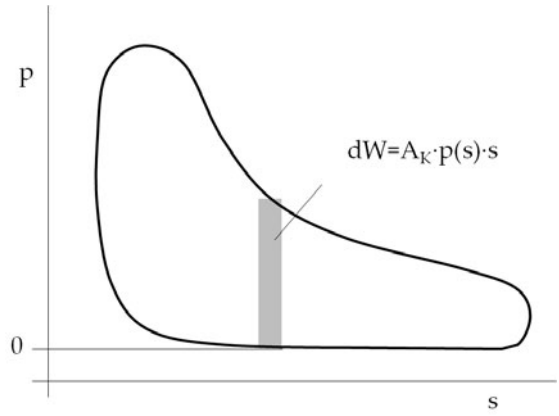


Abbildung 5-5: Indikatordiagramm

Diese praktische Messwertaufnahme zeigt den Zylinderdruck über dem Weg. Die obere Kurve stellt die Entspannungsphase und die untere die Kompressionsphase dar. Die eingeschlossene Fläche ist ein Maß für die geleistete Arbeit. Die Kolbenkraft ergibt sich über die Kolbenfläche und den indizierten Druck zu

$$F_K = \frac{\pi \cdot d^2}{4} \cdot p \ . \qquad (5.1.24)$$

In der Kompressionsphase wird die in einem Schwungrad bei der Entspannungsphase gespeicherte Energie übernommen.

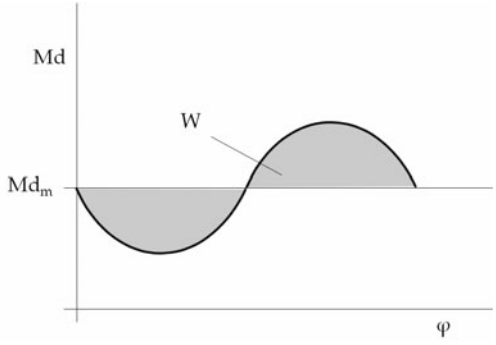


Abbildung 5-6: Drehmomentverlauf

Das Schwungrad ist für die Laufruhe eines Motors von entscheidender Bedeutung. Durch die Triebwerksbewegung und durch die Veränderung des indizierten Drucks ergeben sich wechselnde Drehmomentverläufe.

Daraus resultiert ein mittleres Drehmoment Md_M. Die Abweichungen von diesem kennzeichnen das Arbeitsvermögen W. Dieses wiederum bestimmt das Trägheitsmoment der Schwungscheibe. Aus der vorhandenen Winkelgeschwindigkeit und einem angenommenen Ungleichförmigkeitsgrad δ ergibt sich das Trägheitsmoment aus der Gleichung

$$Id = \frac{W}{\delta \cdot \omega^2} \quad . \tag{5.1.25}$$

Der Ungleichförmigkeitsgrad ist das Verhältnis der Differenzen der größten und kleinsten Winkelgeschwindigkeit der Schwungmassen ω_{max} und ω_{min} zu ihrem Mittelwert. Er wird aus Erfahrung bestimmt.

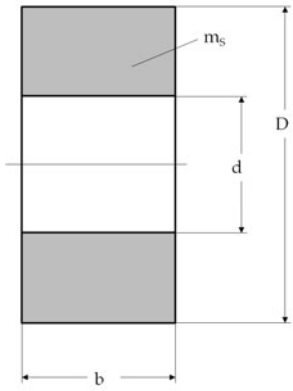


Abbildung 5-7: Schwungscheibe

Der Durchmesser der Schwungscheibe ergibt sich aus der Ableitung

$$Id = \frac{\pi}{32} \cdot \frac{m_S}{g} \cdot (D^4 - d^4) \cdot b \tag{5.1.26}$$

zu

$$D = \sqrt[4]{\frac{32 \cdot Id}{\frac{m_S}{g} \cdot \pi \cdot b} + d^4} \quad . \tag{5.1.27}$$

Tabelle 5-1: Algorithmus zur Bestimmung der Schubkurbelbewegung

Eingabe der Schubkurbeldaten d, l, r, r_{ST}, m_{ST}, m_B, m_Z, m_K, r_W, m_W, m_N, m_O, m_R
Eingabe Indikatordiagramm: für alle Winkel φ die zugehörigen p-Werte mit einer Schrittweite von 10 Grad
Bestimmung der Massenaufteilung $m_O = m_{ST} \cdot \frac{r_{ST}}{l} + m_K + m_B$

$m_R = m_{ST} \cdot \frac{l - r_{ST}}{l} + m_W \cdot \frac{r_W}{r} + m_Z + m_N$
$\lambda = \frac{r}{l}$
x_a=0, t=0, v_a=0
Für alle Winkel φ = -90, 10, 270
p = f(φ)
$x = r \cdot (1 - \sin\varphi) + l \cdot (1 - \sqrt{1 - \lambda^2 \cos^2(\varphi)}$
$\Delta x = x - x_a \,;\; x_a = x$
$F_K = \frac{\pi \cdot d^2}{4} \cdot p$
$\beta = \arcsin\left(\frac{r \cdot \cos\varphi}{l}\right)$
$F_{ST} = \frac{F_K}{\cos\beta}$
$F_R = F_{ST} \cdot \cos(90 - \varphi - \beta)$
$\omega = \sqrt{\frac{F_R}{m_R \cdot r}}$
$\Delta t = \frac{\Delta\varphi}{\omega}$
$t = t + \Delta t$
$v = \frac{\Delta x}{\Delta t}$
$\Delta v = v - v_a \,;\; v_a = v$
$a = \frac{\Delta v}{\Delta t}$
$F_O = m_O \cdot a$
$F_T = F_{ST} \cdot \sin(90 - \varphi - \beta)$
$M_d = F_T \cdot r$
$W = W + M_d \cdot \Delta\varphi$

Durch Einfügen eines Tabellenblattes *tblSchubkurbeltrieb*, kann der Algorithmus darin programmiert werden.

Code 5-1: Bewegung eines Schubkurbeltriebs

```
Option Explicit

Sub Schubkurbel_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Schubkurbeltrieb").Shapes
```

```
    Shp.Delete
Next
ThisWorkbook.Worksheets("Schubkurbeltrieb").Cells.Clear
Range("A1:B1").Select
Selection.MergeCells = True
Selection.Font.Bold = True
Selection.Font.Italic = True
Selection.Value = "Schubkurbeltrieb"

Range("C:C").ColumnWidth = "2"
Range("D1:E1").Select
Selection.MergeCells = True
Selection.Font.Bold = True
Selection.Font.Italic = True
Selection.Value = "Indikatordiagramm"
Range("D2") = ChrW(966) & " [Grad]"
Range("E2") = "p [N/m" & ChrW(178) & "]"
Range("D2:E2").Select
Selection.Font.Bold = True
Selection.Font.Italic = True

Range("F:F").ColumnWidth = "2"
Range("G1:H1").Select
Selection.MergeCells = True
Selection.Font.Bold = True
Selection.Font.Italic = True
Selection.Value = "Auswertung"
Range("G2") = "x [mm]"
Range("H2") = "FK [N]"
Range("I2") = ChrW(946) & " [Grad]"
Range("J2") = "FST [N]"
Range("K2") = "FR [N]"
Range("L2") = ChrW(969) & " [1/s]"
Range("M2") = ChrW(916) & "t [s]"
Range("N2") = "t [s]"
Range("O2") = "v [mm/s]"
Range("P2") = "a [mm/s" & ChrW(178) & "]"
Range("Q2") = "FO [N]"
Range("R2") = "FT [N]"
Range("S2") = "Md [Nmm]"
Range("T2") = "W [Nm/s]"
Range("G2:T2").Select
Selection.Font.Bold = True
Selection.Font.Italic = True
Range("G:T").Select
Selection.NumberFormat = "0.00"

Range("A2:A16").Select
Selection.Font.Bold = True
Selection.Font.Italic = True
Range("A2") = "d [mm]"
Range("A3") = "l [mm]"
Range("A4") = "r [mm]"
Range("A5") = "rST [mm]"
Range("A6") = "mST [kg]"
Range("A7") = "mB [kg]"
```

```
    Range("A8") = "mZ [kg]"
    Range("A9") = "mK [kg]"
    Range("A10") = "rW [mm]"
    Range("A11") = "mW [kg]"
    Range("A12") = "mN [kg]"
    Range("A15") = "mO [kg]"
    Range("A16") = "mR [kg]"
    Range("B2:B20").Select
    Selection.NumberFormat = "0.00"
    Range("B2").Select
End Sub

Sub Schubkurbel_Testdaten()
    Cells(2, 2) = 100
    Cells(3, 2) = 300
    Cells(4, 2) = 50
    Cells(5, 2) = 124
    Cells(6, 2) = 60
    Cells(7, 2) = 23.2
    Cells(8, 2) = 6
    Cells(9, 2) = 32
    Cells(10, 2) = 12
    Cells(11, 2) = 61.66
    Cells(12, 2) = 4
    Cells(3, 4) = -90: Cells(3, 5) = 100000
    Cells(4, 4) = -80: Cells(4, 5) = 70000
    Cells(5, 4) = -70: Cells(5, 5) = 45000
    Cells(6, 4) = -60: Cells(6, 5) = 25000
    Cells(7, 4) = -50: Cells(7, 5) = 20000
    Cells(8, 4) = -40: Cells(8, 5) = 20000
    Cells(9, 4) = -30: Cells(9, 5) = 20000
    Cells(10, 4) = -20: Cells(10, 5) = 20000
    Cells(11, 4) = -10: Cells(11, 5) = 20000
    Cells(12, 4) = 0: Cells(12, 5) = 20000
    Cells(13, 4) = 10: Cells(13, 5) = 20000
    Cells(14, 4) = 20: Cells(14, 5) = 20000
    Cells(15, 4) = 30: Cells(15, 5) = 20000
    Cells(16, 4) = 40: Cells(16, 5) = 20000
    Cells(17, 4) = 50: Cells(17, 5) = 20000
    Cells(18, 4) = 60: Cells(18, 5) = 55000
    Cells(19, 4) = 70: Cells(19, 5) = 100000
    Cells(20, 4) = 80: Cells(20, 5) = 190000
    Cells(21, 4) = 90: Cells(21, 5) = 600000
    Cells(22, 4) = 100: Cells(22, 5) = 630000
    Cells(23, 4) = 110: Cells(23, 5) = 700000
    Cells(24, 4) = 120: Cells(24, 5) = 735000
    Cells(25, 4) = 130: Cells(25, 5) = 750000
    Cells(26, 4) = 140: Cells(26, 5) = 720000
    Cells(27, 4) = 150: Cells(27, 5) = 650000
    Cells(28, 4) = 160: Cells(28, 5) = 560000
    Cells(29, 4) = 170: Cells(29, 5) = 480000
    Cells(30, 4) = 180: Cells(30, 5) = 400000
    Cells(31, 4) = 190: Cells(31, 5) = 350000
    Cells(32, 4) = 200: Cells(32, 5) = 305000
    Cells(33, 4) = 210: Cells(33, 5) = 280000
    Cells(34, 4) = 220: Cells(34, 5) = 250000
```

```
   Cells(35, 4) = 230: Cells(35, 5) = 230000
   Cells(36, 4) = 240: Cells(36, 5) = 200000
   Cells(37, 4) = 250: Cells(37, 5) = 180000
   Cells(38, 4) = 260: Cells(38, 5) = 135000
   Cells(39, 4) = 270: Cells(39, 5) = 100000
End Sub

Sub Schubkurbel_Auswertung()
   Dim d, l, r, rST, mST, mB, mZ, mK, rW, mW, mN As Double
   Dim mO, mR, ph, x, la, FK, p, be, FST, FR, z As Double
   Dim w, dx, xa, dph, dt, t, v, va, dv, aK, FO As Double
   Dim FT, Md, WA As Double
   Dim i As Integer
   d = Cells(2, 2)
   l = Cells(3, 2)
   r = Cells(4, 2)
   rST = Cells(5, 2)
   mST = Cells(6, 2)
   mB = Cells(7, 2)
   mZ = Cells(8, 2)
   mK = Cells(9, 2)
   rW = Cells(10, 2)
   mW = Cells(11, 2)
   mN = Cells(12, 2)
'Massenaufteilung
   mO = mST * rST / l + mK + mB
   mR = mST * (l - rST) / l + mW * rW / r + mZ + mN
   Cells(15, 2) = mO
   Cells(16, 2) = mR
'Bewegung
'Konstante: Atn(1)=pi/4
   la = r / l
   xa = 0
   dph = 10
   t = 0
   va = 0
   WA = 0
   For i = 3 To 39
      ph = Cells(i, 4)
      p = Cells(i, 5)
      ph = ph / 45 * Atn(1)
      x = r * (1 - Sin(ph)) + _
          l * (1 - Sqr(1 - (la * Cos(ph)) ^ 2))
      If i = 3 Then xa = x
      dx = x - xa
      xa = x
      Cells(i, 7) = x
      FK = d ^ 2 * Atn(1) * p / 1000000
      Cells(i, 8) = FK
      be = r * Cos(ph) / l
      If 1 - be * be >= 0 Then
         be = Atn(be / Sqr(1 - be * be))
      Else
         be = 0
      End If
      Cells(i, 9) = be / Atn(1) * 45
```

```
      FST = FK / Cos(be)
      Cells(i, 10) = FST
      FR = FST * Cos(2 * Atn(1) - ph - be)
      Cells(i, 11) = FR
      w = Sqr(Abs(FR / (mR * r)) * 9810)
      Cells(i, 12) = w
      If Not w = 0 Then
         dt = Abs(dph / w)
      Else
         dt = 0
      End If
      Cells(i, 13) = dt
      t = t + dt
      Cells(i, 14) = t
      If dt > 0 Then
         v = dx / dt
      Else
         v = 0
      End If
      Cells(i, 15) = v
      If i = 3 Then va = v
      dv = v - va
      va = v
      If dt > 0 Then
         aK = dv / dt
      Else
         aK = 0
      End If
      Cells(i, 16) = aK
      FO = mO * aK / 9810
      Cells(i, 17) = FO
      FT = FST * Sin(2 * Atn(1) - ph - be)
      Cells(i, 18) = FT
      Md = FT * r
      Cells(i, 19) = Md
      WA = WA + Md * dph / 1000
      Cells(i, 20) = WA
   Next i
End Sub
```

Die Programmliste enthält diesmal keine Diagrammdarstellung. Somit enthält die Symbolleiste nur drei Menüpunkte.

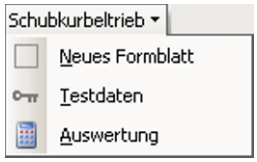


Abbildung 5-8: Menü Schubkurbeltrieb

Mit Hilfe der eingebauten Testdaten ergibt sich eine umfassende Darstellung.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Schubkurbeltrieb			Indikatordiagramm			Auswertung					
2	d [mm]	100,00		φ [Grad]	p [N/m²]		x [mm]	FK [N]	β [Grad]	FST [N]	FR [N]	ω [1/s]
3	l [mm]	300,00		-90	100000		100,00	785,40	0,00	785,40	-785,40	50,68
4	r [mm]	50,00		-80	70000		99,37	549,78	1,66	550,01	-538,66	41,97
5	rST [mm]	124,00		-70	46000		97,47	353,43	3,27	354,00	-325,21	32,61
6	mST [kg]	60,00		-60	25000		94,34	196,35	4,78	197,03	-161,83	23,00
7	mD [kg]	23,20		-50	20000		90,03	157,08	6,15	157,99	-109,45	18,92
8	mZ [kg]	6,00		-40	20000		84,59	157,08	7,34	158,38	-95,48	16,72
9	mK [kg]	32,00		-30	20000		78,14	157,08	8,30	158,74	-58,70	13,85
10	rW [mm]	12,00		-20	20000		70,80	157,08	9,01	159,04	-30,32	9,96
11	mW [kg]	61,66		-10	20000		62,75	157,08	9,45	159,24	-1,54	2,24
12	mN [kg]	4,00		0	20000		54,20	157,08	9,59	159,31	26,55	9,32
13				10	20000		45,39	157,08	9,45	159,24	53,02	13,17
14				20	20000		36,60	157,08	9,01	159,04	77,13	15,88
15	mO [kg]	80,00		30	20000		28,14	157,08	8,30	158,74	98,38	17,94
16	mR [kg]	60,00		40	20000		20,32	157,08	7,34	158,38	116,46	19,51
17				50	20000		13,42	157,08	6,15	157,99	131,21	20,71
18				60	66000		7,74	431,97	4,78	433,48	392,16	35,81
19				70	100000		3,50	785,40	3,27	786,68	753,37	49,63
20				80	190000		0,89	1492,26	1,66	1492,88	1477,09	69,50
21				90	600000		0,00	4712,39	0,00	4712,39	4712,39	124,14
22				100	630000		0,89	4948,01	-1,66	4950,08	4897,71	126,55
23				110	700000		3,50	5497,79	-3,27	5506,74	5273,59	131,32
24				120	736000		7,74	5772,68	-4,78	5792,83	5240,65	130,91
25				130	750000		13,42	5890,49	-6,15	5924,58	4920,36	126,85
26				140	720000		20,32	5654,87	-7,34	5701,53	4192,51	117,09
27				150	650000		28,14	5105,09	-8,30	5159,11	3197,43	102,25
28				160	560000		36,60	4398,23	-9,01	4453,18	2159,66	84,04
29				170	480000		45,39	3769,91	-9,45	3821,74	1272,39	64,50
30				180	400000		54,20	3141,59	-9,59	3186,16	531,03	41,67
31				190	350000		62,75	2748,89	-9,45	2786,69	-26,90	9,38
32				200	305000		70,80	2395,46	-9,01	2425,39	-462,35	38,88
33				210	280000		78,14	2199,11	-8,30	2222,39	-821,76	51,84
34				220	250000		84,59	1963,50	-7,34	1979,70	-1068,49	59,11
35				230	230000		90,03	1806,42	-6,15	1816,87	-1258,68	64,16
36				240	200000		94,34	1570,80	-4,78	1576,28	-1294,67	65,07
37				250	180000		97,47	1413,72	-3,27	1416,02	-1300,85	65,22
38				260	135000		99,37	1060,29	-1,66	1060,73	-1038,85	58,28
39				270	100000		100,00	785,40	0,00	785,40	-785,40	50,68

Abbildung 5-9: Auswertung der Testdaten Teil 1

M	N	O	P	Q	R	S	T
Δt [s]	t [s]	v [mm/s]	a [mm/s²]	FO [N]	FT [N]	Md [Nmm]	W [Nm/s]
0,20	0,20	0,00	0,00	0,00	0,00	0,00	0,00
0,24	0,44	-2,66	-11,17	-0,09	111,14	5557,21	55,57
0,31	0,74	-6,18	-11,46	-0,09	139,84	6992,12	125,49
0,43	1,18	-7,20	-2,35	-0,02	112,39	5619,73	181,69
0,53	1,71	-8,17	-1,84	-0,01	113,93	5696,73	238,66
0,60	2,30	-9,09	-1,54	-0,01	133,33	6666,37	305,32
0,72	3,03	-8,94	0,20	0,00	147,49	7374,56	379,07
1,00	4,03	-7,31	1,63	0,01	156,13	7806,29	457,13
4,46	8,49	-1,81	1,23	0,01	159,23	7961,59	536,75
1,07	9,56	-7,97	-5,75	-0,05	157,08	7853,98	615,29
0,76	10,32	-11,60	-4,78	-0,04	150,15	7507,73	690,36
0,63	10,95	-13,95	-3,74	-0,03	139,09	6954,37	759,91
0,56	11,51	-15,17	-2,19	-0,02	124,58	6228,94	822,20
0,51	12,02	-15,27	-0,19	0,00	107,33	5366,63	875,86
0,48	12,51	-14,27	2,06	0,02	88,00	4400,16	919,86
0,28	12,78	-20,35	-21,75	-0,18	184,70	9235,05	1012,21
0,20	12,99	-21,04	-3,44	-0,03	226,48	11324,16	1125,46
0,14	13,13	-18,19	19,78	0,16	216,58	10828,90	1233,75
0,08	13,21	-10,99	89,44	0,73	0,00	0,00	1233,75
0,08	13,29	11,20	280,86	2,29	-718,13	-35906,35	874,68
0,08	13,37	34,38	304,33	2,48	-1585,38	-79269,09	81,99
0,08	13,44	55,49	276,41	2,25	-2468,28	-123413,85	-1152,15
0,08	13,52	72,08	210,34	1,72	-3300,12	-165005,85	-2802,21
0,09	13,61	80,69	100,89	0,82	-3863,97	-193198,51	-4734,19
0,10	13,70	80,02	-6,86	-0,06	-4048,81	-202440,46	-6758,60
0,12	13,82	71,09	-75,03	-0,61	-3894,45	-194722,32	-8705,82
0,16	13,98	56,67	-93,05	-0,76	-3603,71	-180185,59	-10507,68
0,24	14,22	36,71	-83,16	-0,68	-3141,59	-157079,63	-12078,47
1,07	15,28	8,02	-26,91	-0,22	-2786,56	-139327,85	-13471,75
0,26	15,54	31,31	90,54	0,74	-2380,92	-119045,90	-14662,21
0,19	15,73	38,04	34,89	0,28	-2064,88	-103243,81	-15694,65
0,17	15,90	38,14	0,61	0,00	-1666,59	-83329,66	-16527,94
0,16	16,06	34,86	-21,05	-0,17	-1310,25	-65512,37	-17183,07
0,15	16,21	28,08	-44,12	-0,36	-899,16	-44957,82	-17632,65
0,15	16,37	20,40	-50,11	-0,41	-559,37	-27968,48	-17912,33
0,17	16,54	11,04	-54,57	-0,45	-214,35	-10717,48	-18019,51
0,20	16,74	3,21	-39,65	-0,32	0,00	0,00	-18019,51

Abbildung 5-10: Auswertung der Testdaten Teil 2

Übung 5-1: Diagramme

Die Erstellung der Diagramme, und hier gibt es einige, überlasse ich dem Leser. Sie können dies über die Funktion Einfügen/Diagramm tun und die entsprechenden Spalten für die x- und y-Achse auswählen. Schalten Sie vorher den Makrorecorder ein, so erhalten Sie den Quellcode in einem Modul und können diesen dem Programm hinzufügen. Die nachfolgende Darstellung (Abbildung 5-11) zeigt einige Beispiele. Wie können Sie mehrere Kurven in ein Diagramm zusammenfassen?

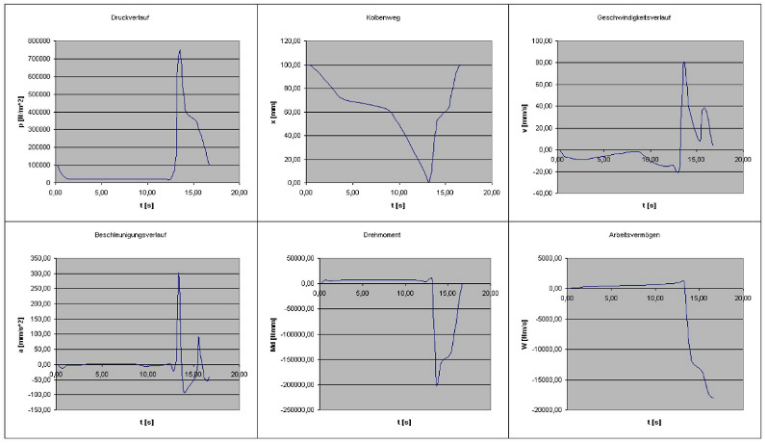


Abbildung 5-11: Diagramme zur Auswertung

Übung 5-2: Schwungscheibe

Ergänzen Sie außerdem das Programm um die Berechnung der Schwungscheibe. Die Formeln dazu habe ich Ihnen bereits geliefert.

Beispiel 5-2: Drehschwingungen

Ein Torsionspendel nach Abbildung 5-12 erfährt bei Auslenkung um den Winkel φ das rückstellende Moment

$$M_t = \frac{G \cdot I_p}{l} \varphi \quad . \tag{5.1.28}$$

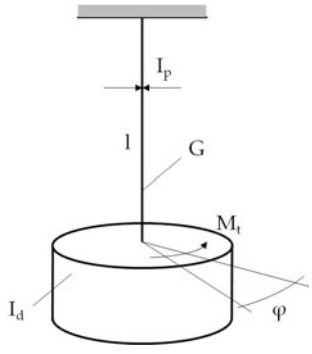


Abbildung 5-12: Torsionspendel

G ist der Gleitmodul des Fadens und I_P sein polares Flächenträgheitsmoment. Daraus folgt als Bewegungsgleichung für freie Drehschwingungen die Differentialgleichung

$$I_d \cdot \ddot{\varphi} = -\frac{G \cdot I_p}{l} \varphi \quad . \tag{5.1.29}$$

Umgestellt

$$\ddot{\varphi} = -\frac{G \cdot I_p}{l \cdot I_d} \varphi \tag{5.1.30}$$

und mittels Differentialquotienten

$$\ddot{\varphi} = \frac{d\omega}{dt} \tag{5.1.31}$$

folgt

$$d\omega = -\frac{G \cdot I_p}{l \cdot I_d} \varphi \cdot dt \quad . \tag{5.1.32}$$

Angenähert durch den Differenzenquotienten folgt

$$\Delta\omega = -\frac{G \cdot I_p}{l \cdot I_d} \varphi \cdot \Delta t \quad . \tag{5.1.33}$$

Den Algorithmus für eine Drehschwingungsberechnung gibt das nachfolgende Struktogramm wieder.

Tabelle 5-2: Algorithmus zur Bestimmung einer Drehschwingung

Eingabe G, I_P, l, I_d, φ_0, ω_0, t_0, Δt, t_{max}
So lange $t<t_{max}$
$t_i = t_{i-1} + \Delta t$
$\Delta\omega_i = -\frac{G \cdot I_p}{l \cdot I_d} \varphi_{i-1} \cdot \Delta t$
$\Delta\varphi_i = \omega_i \cdot \Delta t$
$\varphi_i = \varphi_{i-1} + \Delta\varphi_i$

In einer neuen Tabelle Drehschwingungen wird der Algorithmus programmiert.

Code 5-2: Drehschwingung

```
Option Explicit

Sub Drehschwingung_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Drehschwingung").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Drehschwingung").Cells.Clear
```

```
   Range("A1:B1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Drehschwingung"

   Range("A2:A16").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A2") = "G [N/m" & ChrW(178) & "]"
   Range("A3") = "Ip [m^4]"
   Range("A4") = "l [m]"
   Range("A5") = "Id [kgm]"
   Range("A6") = ChrW(966) & "[kg]"
   Range("A7") = ChrW(969) & " [1/s]"
   Range("A8") = "t0 [s]"
   Range("A9") = ChrW(916) & "t [s]"
   Range("A10") = "tmax [s]"
   Range("B:B").ColumnWidth = "15"

   Range("C:C").ColumnWidth = "2"
   Range("D1:F1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Auswertung"
   Range("D2") = "t [s]"
   Range("E2") = ChrW(969) & " [1/s]"
   Range("F2") = ChrW(966) & " [Grad]"
   Range("D2:F2").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True

   Range("B2").Select
End Sub

Sub Drehschwingung_Testdaten()
   Cells(2, 2) = 80000000000#
   Cells(3, 2) = 0.0000002
   Cells(4, 2) = 0.6
   Cells(5, 2) = 4.5
   Cells(6, 2) = 4
   Cells(7, 2) = 0
   Cells(8, 2) = 0
   Cells(9, 2) = 0.005
   Cells(10, 2) = 0.2
End Sub

Sub Drehschwingung_Auswertung()
   Dim G, Ip, l, Id, p0, w0, t0, dt, tm As Double
   Dim t, p, dp, dw, w As Double
   Dim i As Integer

   G = Cells(2, 2)
   Ip = Cells(3, 2)
```

```
   l = Cells(4, 2)
   Id = Cells(5, 2)
   p0 = Cells(6, 2)
   w0 = Cells(7, 2)
   t0 = Cells(8, 2)
   dt = Cells(9, 2)
   tm = Cells(10, 2)
   t = t0
   p = p0
   w = w0
   i = 2
   Do
      i = i + 1
      t = t + dt
      Cells(i, 4) = t
      dw = -G * Ip / l / Id * p * dt
      w = w + dw
      Cells(i, 5) = w
      dp = w * dt
      p = p + dp
      Cells(i, 6) = p
   Loop While t < tm
End Sub

Sub Drehschwingung_Zeigen()
   Range("D3:D42,F3:F42").Select
   Range("F3").Activate
   Charts.Add
   ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
   ActiveChart.SetSourceData _
      Source:=Sheets("Drehschwingung").Range( _
      "D3:D42,F3:F42"), PlotBy:=xlColumns
   ActiveChart.SeriesCollection(1).Name = _
      "=""Auslenkung"""
   ActiveChart.Location Where:=xlLocationAsObject, _
      Name:="Drehschwingung"
   With ActiveChart
      .HasTitle = True
      .ChartTitle.Characters.Text = "Drehschwingung"
      .Axes(xlCategory, xlPrimary).HasTitle = True
      .Axes(xlCategory, _
         xlPrimary).AxisTitle.Characters.Text = "t [s]"
      .Axes(xlValue, xlPrimary).HasTitle = True
      .Axes(xlValue, xlPrimary).AxisTitle.Characters. _
          Text = "Winkel [Grad]"
   End With
   ActiveWindow.Visible = False
   Windows("Kapitel 05.xls").Activate
End Sub

Sub Drehschwingung_Löschen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Drehschwingung").Shapes
      Shp.Delete
   Next
End Sub
```

Das Menü hat wieder die üblichen Funktionen. Der Schwingungsverlauf kann aufgerufen und wieder gelöscht werden. Für eine Neuberechnung wird die Funktion Neues Formular aufgerufen. Die Testdaten beruhen auf einem nachfolgend dargestellten Beispiel.

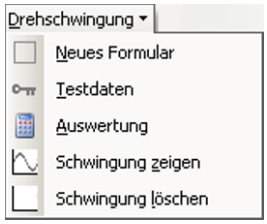


Abbildung 5-13: Menü Drehschwingungen

Als Testbeispiel wird ein Getriebe benutzt, das nachfolgend schematisch dargestellt ist.

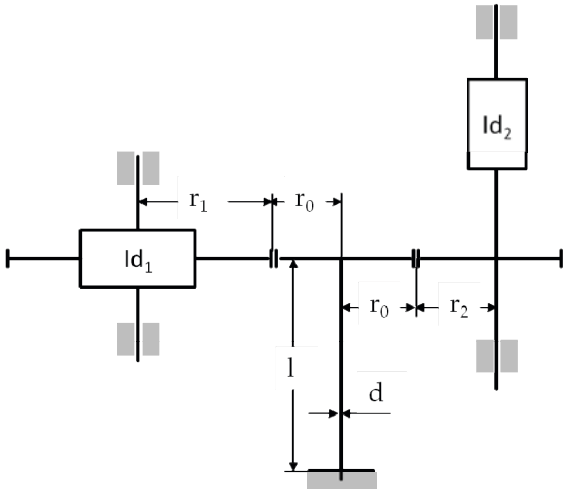


Abbildung 5-14: Schwingungssystem

Das Schwingungssystem hat folgende Daten:

$$r_1 = 2 \cdot r_0 ; r_2 = r_1$$

$$Id_1 = 6kgm; Id_2 = 3kgm$$

$$d = 0{,}04m; l = 0{,}6m$$

$$G = 80.000.000.000 \frac{N}{m^2}$$

$$Ip = 20000m^4$$

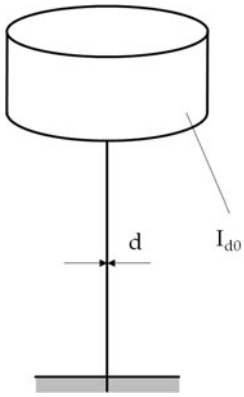


Abbildung 5-15: Ersatzsystem

Zur Betrachtung werden die Massen der Drehwellen auf ein Ersatzsystem nach Abbildung 5-15 reduziert.

Es folgt

$$Id_{10} = Id_1\left(\frac{r_0}{r_1}\right)^2 = \frac{1}{4}Id_1 \tag{5.1.34}$$

und

$$Id_{20} = Id_2\left(\frac{r_0}{r_2}\right)^2 = Id_2 \ . \tag{5.1.35}$$

So dass gilt

$$Id_0 = Id_{10} + Id_{20} = 4{,}5kgm \ . \tag{5.1.36}$$

Die Auswertung des Systems mit Daten und Diagramm sehen Sie in Abbildung 5-16.

Übung 5-3: Torsionspendel

Ergänzen Sie das Programm um eine allgemeine Berechnungsprozedur zur Bestimmung reduzierter Massen.

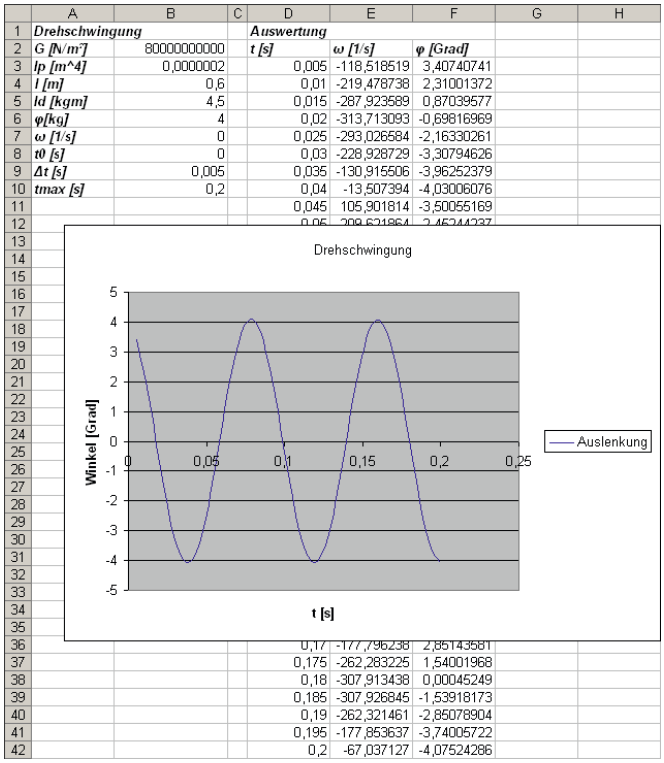
In der Praxis wird ein Torsionspendel mitunter zur Feststellung des Massenträgheitsmoments eines beliebigen Körpers benutzt. Den Versuchsaufbau zeigt Abbildung 5-17. Der Vorteil liegt darin, dass lediglich die Massen m_1, der Radius r_1 und die Schwingzeiten des Systems bekannt sein müssen

Die Zeit ohne aufgesetzte Massen beträgt

$$T = 2\sqrt{\frac{l \cdot Id}{G \cdot Ip}} \ . \tag{5.1.37}$$

Mit aufgesetzten Massen ergibt sich nach dem Steinerschen Satz

$$T_1 = 2\sqrt{\frac{l \cdot (Id + 2 \cdot m_1 \cdot r_1^2)}{G \cdot Ip}} \quad . \tag{5.1.38}$$



	A	B	C	D	E	F	G	H
1	*Drehschwingung*			*Auswertung*				
2	*G [N/m²]*	80000000000		*t [s]*	*ω [1/s]*	*φ [Grad]*		
3	*Ip [m^4]*	0,0000002		0,005	-118,518519	3,40740741		
4	*l [m]*	0,6		0,01	-219,478738	2,31001372		
5	*Id [kgm]*	4,5		0,015	-287,923589	0,87039577		
6	*φ[kg]*	4		0,02	-313,713093	-0,69816969		
7	*ω [1/s]*	0		0,025	-293,026584	-2,16330261		
8	*t0 [s]*	0		0,03	-228,928729	-3,30794626		
9	*Δt [s]*	0,005		0,035	-130,915506	-3,96252379		
10	*tmax [s]*	0,2		0,04	-13,507394	-4,03006076		
11				0,045	105,901814	-3,50055169		
12				0,05	209,621864	-2,46244237		
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								
34								
35								
36				0,17	-177,796238	2,85143581		
37				0,175	-262,283225	1,54001968		
38				0,18	-307,913438	0,00045249		
39				0,185	-307,926845	-1,53918173		
40				0,19	-262,321461	-2,85078904		
41				0,195	-177,853637	-3,74005722		
42				0,2	-67,037127	-4,07524286		

Abbildung 5-16: Auswertung des Schwingungssystems

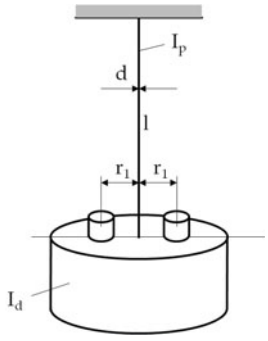


Abbildung 5-17: Torsionspendel zur Feststellung eines Id

Daraus folgt durch Gleichsetzung und Umstellung

$$Id = \frac{2 \cdot m_1 \cdot r_1^2}{\left(\frac{T_1}{T}\right)^2 - 1} \quad . \tag{5.1.39}$$

Schreiben Sie eine zusätzliche Prozedur, die durch Eingabe der Schwingzeiten das zugehörige Massenträgheitsmoment Id bestimmt.

5.2 Partielle Differentialgleichungen

In gewöhnlichen Differentialgleichungen treten nur Funktionen mit einer unabhängig Veränderlichen auf. Dagegen spricht man von einer partiellen Differentialgleichung, wenn die gesuchte Funktion

$$y = y(x_1, x_2, ..., x_n) \tag{5.2.1}$$

von mehreren Veränderlichen x_1, x_2, … , x_n abhängt und in der Gleichung partielle Ableitungen der Form

$$\frac{\partial y}{\partial x_i} \ , \ \frac{\partial^2 y}{\partial x_i \partial x_j} \text{, usw.} \tag{5.2.2}$$

auftreten. Um deren Lösung numerisch zu bestimmen, überzieht man die x, y-Ebene mit einem zweidimensionalen Gitter der Maschenweite h.

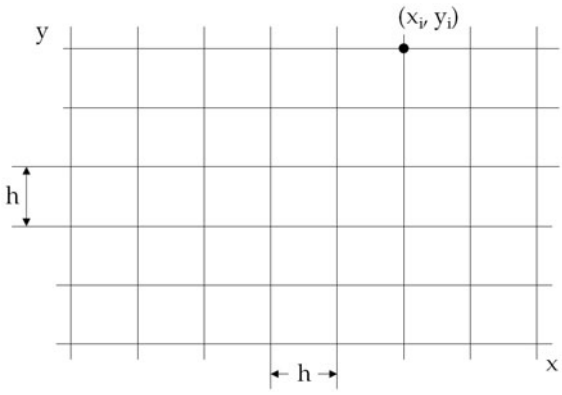


Abbildung 5-18: Gitterpunkte

Die Gitterpunkte bestimmen sich durch

$$x_i = x_0 + i \cdot h \ \text{ und } \ y_j = y_0 + j \cdot h \ . \tag{5.2.3}$$

Außerdem wollen wir folgende Abkürzung

$$u_{i,j} = u(x_i, y_j) \tag{5.2.4}$$

verwenden.

Ähnlich wie zuvor werden auch hier partielle Ableitungen erster und höherer Ordnung durch Differenzenquotienten approximiert (diskretisiert). So ergibt sich

$$\frac{\partial u}{\partial x}(x_i, y_j) = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + O(h^2) \tag{5.2.5}$$

und

$$\frac{\partial u}{\partial y}(x_i, y_j) = \frac{u_{i,j+1} - u_{i,j-1}}{2h} + O(h^2) \ . \tag{5.2.6}$$

Ein häufig auftretender Differentialoperator ist der Laplace-Operator Δ

$$\Delta u := \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{5.2.7}$$

mit der Differenzenapproximation

$$\Delta u(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} . \tag{5.2.8}$$

Dies lässt sich symbolisch und anschaulich durch folgenden Berechnungsoperator in der Abbildung 5-19 darstellen.

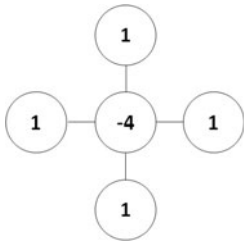


Abbildung 5-19: Berechnungsoperator

Beispiel 5-3: Eingespannte Membran

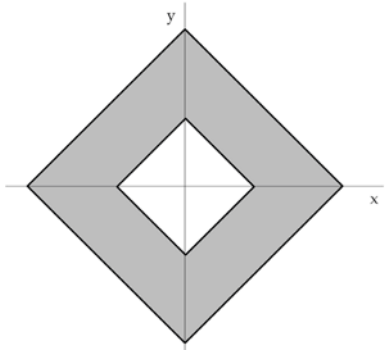


Abbildung 5-20: Eingespannte Membranform

Die in Abbildung 5-20 dargestellte elastische Membrane ist an den Rändern fest eingespannt. Sie erfüllt die Differentialgleichung nach Laplace

$$\frac{\vartheta^2 u}{\vartheta x^2}+\frac{\vartheta^2 u}{\vartheta y^2}=0\ . \qquad (5.2.9)$$

Dabei ist u die Höhe der Membrane über der (x,y)-Ebene. Die Randwerte seien u=0 für den äußeren und u=1 für den inneren Rand. Aus der Symmetrieeigenschaft des Laplace-Operators genügt die Betrachtung eines Viertelstücks.

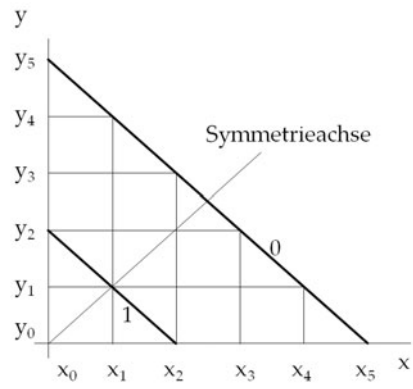


Abbildung 5-21: Membranausschnitt

Durch Anwendung des Operators ergeben sich nachfolgende Differenzen.

Verbesserung der Zeilenwerte (von Schritt n nach n+1):

j=0, i=3:

$$-u_{2,0}^{(n+1)}+4u_{3,0}^{(n+1)}-u_{4,0}^{(n+1)}=u_{3,-1}^{(n)}+u_{3,1}^{(n)}$$

daraus folgt:

$$4u_{3,0}^{(n+1)}-u_{4,0}^{(n+1)}=2u_{3,1}^{(n)}+1 \text{ weil } u_{2,0}^{(n+1)}=1.$$

j=0, i=4:

$$-u_{3,0}^{(n+1)}+4u_{4,0}^{(n+1)}-u_{5,0}^{(n+1)}=u_{4,-1}^{(n)}+u_{4,1}^{(n)}$$

daraus folgt:

$$-u_{3,0}^{(n+1)}+4u_{4,0}^{(n+1)}=0 \text{ weil } u_{5,0}^{(n+1)}=0.$$

j=1, i=2:

$$-u_{1,1}^{(n+1)}+4u_{2,1}^{(n+1)}-u_{3,1}^{(n+1)}=u_{2,0}^{(n)}+u_{2,2}^{(n)}$$

daraus folgt:

$$4u_{2,1}^{(n+1)}-u_{3,1}^{(n+1)}=u_{2,2}^{(n)}+2 \text{ weil } u_{1,1}^{(n+1)}=u_{2,0}^{(n)}=1.$$

j=1, i=3:

$$-u_{2,1}^{(n+1)}+4u_{3,1}^{(n+1)}-u_{4,1}^{(n+1)}=u_{3,0}^{(n)}+u_{3,2}^{(n)}$$

daraus folgt:

$-u_{2,1}^{(n+1)}+4u_{3,1}^{(n+1)}=u_{3,0}^{(n)}$ weil $u_{4,1}^{(n+1)}=u_{3,2}^{(n)}=0$.

j=2, i=2:

$-u_{1,2}^{(n+1)}+4u_{2,2}^{(n+1)}-u_{3,2}^{(n+1)}=u_{2,1}^{(n)}+u_{2,3}^{(n)}$

daraus folgt:

$-u_{2,1}^{(n+1)}+4u_{2,2}^{(n+1)}=u_{2,1}^{(n)}$

weil $u_{3,2}^{(n+1)}=u_{2,3}^{(n)}=0$ und $u_{1,2}^{(n+1)}=u_{2,1}^{(n+1)}$ symmetrisch.

Verbesserung der Spaltenwerte (von Schritt n+1 nach n+2):

i=2, j=1:

$-u_{2,0}^{(n+2)}+4u_{2,1}^{(n+2)}-u_{2,2}^{(n+2)}=u_{1,1}^{(n+1)}+u_{3,1}^{(n+1)}$

daraus folgt:

$4u_{2,1}^{(n+2)}-u_{2,2}^{(n+2)}=u_{3,1}^{(n+1)}+2$ weil $u_{2,0}=u_{1,1}=1$.

i=2, j=2:

$-u_{2,1}^{(n+2)}+4u_{2,2}^{(n+2)}-u_{2,3}^{(n+2)}=u_{1,2}^{(n+1)}+u_{3,2}^{(n+1)}$

daraus folgt:

$-u_{2,1}^{(n+2)}+4u_{2,2}^{(n+2)}=u_{2,1}^{(n+1)}$

weil $u_{2,3}=u_{3,2}=0$ und $u_{1,2}=u_{2,1}$ symmetrisch.

i=3, j=0:

$-u_{3,-1}^{(n+2)}+4u_{3,0}^{(n+2)}-u_{3,1}^{(n+2)}=u_{2,0}^{(n+1)}+u_{4,0}^{(n+1)}$

daraus folgt:

$4u_{3,0}^{(n+2)}-2u_{3,1}^{(n+2)}=u_{4,0}^{(n+1)}+1$

weil $u_{2,0}=1$ und $u_{3,-1}=u_{3,1}$ symmetrisch.

i=3, j=1:

$-u_{3,0}^{(n+2)}+4u_{3,1}^{(n+2)}-u_{3,2}^{(n+2)}=u_{2,1}^{(n+1)}+u_{4,1}^{(n+1)}$

daraus folgt:

$-u_{3,0}^{(n+2)}+4u_{3,1}^{(n+2)}=u_{2,1}^{(n+1)}$ weil $u_{3,2}=u_{4,1}=0$.

i=4, j=0:

$-u_{4,-1}^{(n+2)}+4u_{4,0}^{(n+2)}-u_{4,1}^{(n+2)}=u_{3,0}^{(n+1)}+u_{5,0}^{(n+1)}$

daraus folgt:

$4u_{4,0}^{(n+2)}=u_{3,0}^{(n+1)}$

weil $u_{4,1}=u_{5,0}=0$ und $u_{4,-1}=u_{4,1}$ symmetrisch.

Durch Umstellung ergeben sich die 10 Iterationsgleichungen:

$$u_{4,0}^{(n+1)} = \frac{1}{15}(2u_{3,1}^{(n)} + 1)$$

$$u_{3,0}^{(n+1)} = 4u_{4,0}^{(n)}$$

$$u_{3,1}^{(n+1)} = \frac{1}{15}(u_{2,2}^{(n)} + 4u_{2,2}^{(n)} + 1)$$

$$u_{2,1}^{(n+1)} = 4u_{3,1}^{(n+1)} - u_{3,0}^{(n)}$$

$$u_{2,2}^{(n+1)} = \frac{1}{4}(u_{2,1}^{(n+1)} + u_{2,1}^{(n)})$$

$$u_{2,2}^{(n+2)} = \frac{1}{15}(4u_{2,1}^{(n+1)} + u_{3,1}^{(n+1)} + 2)$$

$$u_{2,1}^{(n+2)} = 4u_{2,2}^{(n+1)} - u_{2,1}^{(n+1)}$$

$$u_{3,1}^{(n+2)} = \frac{1}{14}(4u_{2,1}^{(n+1)} + u_{4,0}^{(n+1)} + 1)$$

$$u_{3,0}^{(n+2)} = 4u_{3,1}^{(n+2)} - u_{2,1}^{(n+1)}$$

$$u_{4,0}^{(n+2)} = \frac{1}{4}u_{3,0}^{(n+1)}$$

Tabelle 5-3: Algorithmus zur Bestimmung einer eingespannten Membran

Eingabe Startwerte $u_{4,0}^{(0)} = 0; u_{3,0}^{(0)} = 0; u_{3,1}^{(0)} = 0; u_{2,1}^{(0)} = 0; u_{2,2}^{(0)} = 0$
n Iterationsschritte
$u_{4,0}^{(n+1)} = \frac{1}{15}(2u_{3,1}^{(n)} + 1)$
$u_{3,0}^{(n+1)} = 4u_{4,0}^{(n+1)}$
$u_{3,1}^{(n+1)} = \frac{1}{15}(u_{2,2}^{(n)} + 4u_{3,0}^{(n)} + 2)$
$u_{2,1}^{(n+1)} = 4u_{3,1}^{(n+1)} - u_{3,0}^{(n)}$
$u_{2,2}^{(n+1)} = \frac{1}{4}(u_{2,1}^{(n+1)} + u_{2,1}^{(n)})$
$u_{2,2}^{(n+2)} = \frac{1}{15}(4u_{2,1}^{(n+1)} + u_{3,1}^{(n+1)} + 2)$
$u_{2,1}^{(n+2)} = 4u_{2,2}^{(n+2)} - u_{2,1}^{(n+1)}$
$u_{3,1}^{(n+2)} = \frac{1}{14}(4u_{2,1}^{(n+1)} + u_{4,0}^{(n+1)} + 1)$

	$u_{3,0}^{(n+2)} = 4u_{3,1}^{(n+2)} - u_{2,1}^{(n+1)}$
	$u_{4,0}^{(n+2)} = \frac{1}{4}u_{3,0}^{(n+1)}$

Das Programm enthält in der Reihenfolge die Iterationsgleichungen. Es ist jedoch darauf zu achten, welcher Iterationswert in die jeweilige Gleichung eingeht. Ich habe dies durch die Variablenfolge u, v, w gekennzeichnet. Erst nach Beendigung der beiden Iterationsschritte erfolgt eine Verschiebung und ein neuer Durchlauf kann beginnen.

Code 5-3: Bestimmung einer Membranform

```
Option Explicit

Sub Membran_Formular()
   Dim Shp As Shape
   For Each Shp In Worksheets("Membran").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Membran").Cells.Clear

   Cells(1, 6) = 0
   Cells(2, 5) = 0
   Cells(2, 7) = 0
   Cells(3, 4) = 0
   Cells(3, 8) = 0
   Cells(4, 3) = 0
   Cells(4, 9) = 0
   Cells(5, 2) = 0
   Cells(5, 10) = 0
   Cells(6, 1) = 0
   Cells(6, 11) = 0
   Cells(7, 2) = 0
   Cells(7, 10) = 0
   Cells(8, 3) = 0
   Cells(8, 9) = 0
   Cells(9, 4) = 0
   Cells(9, 8) = 0
   Cells(10, 5) = 0
   Cells(10, 7) = 0
   Cells(11, 6) = 0

   Cells(4, 6) = 1
   Cells(5, 5) = 1
   Cells(5, 7) = 1
   Cells(6, 4) = 1
   Cells(6, 8) = 1
   Cells(7, 5) = 1
   Cells(7, 7) = 1
   Cells(8, 6) = 1

   Range("I6:J6").Select
   With Selection.Interior
```

```
        .ColorIndex = 15
        .Pattern = xlSolid
    End With
    Range("H5:I5").Select
    With Selection.Interior
        .ColorIndex = 15
        .Pattern = xlSolid
    End With
    Range("H4").Select
    With Selection.Interior
        .ColorIndex = 15
        .Pattern = xlSolid
    End With
    Range("A12").Select
    With Selection.Interior
        .ColorIndex = 15
        .Pattern = xlSolid
    End With
End Sub

Sub Membran_Auswertung()
    Dim u40, u30, u31, u21, u22 As Double
    Dim v40, v30, v31, v21, v22 As Double
    Dim w40, w30, w31, w21, w22 As Double
    Dim i, n As Integer
    n = Cells(12, 1)
    u40 = Cells(6, 10)
    u30 = Cells(6, 9)
    u31 = Cells(5, 9)
    u21 = Cells(5, 8)
    u22 = Cells(4, 8)

    For i = 1 To n
       v40 = 1 / 15 * (2 * u31 + 1)
       v30 = 4 * v40
       v31 = 1 / 15 * (u22 + 4 * u30 + 2)
       v21 = 4 * v31 - u30
       v22 = 1 / 4 * (v21 + u21)

       w22 = 1 / 15 * (4 * v21 + v31 + 2)
       w21 = 4 * w22 - v21
       w31 = 1 / 14 * (4 * v21 + v40 + 1)
       w30 = 4 * w31 - v21
       w40 = 1 / 4 * v30

       u40 = w40
       u30 = w30
       u31 = w31
       u21 = w21
       u22 = w22

       Cells(6, 10) = u40
       Cells(6, 9) = u30
       Cells(5, 9) = u31
       Cells(5, 8) = u21
       Cells(4, 8) = u22
    Next i
```

```
'
'Übertragung durch Symmetrie
   Cells(7, 9) = Cells(5, 9)
   Cells(7, 8) = Cells(5, 8)
   Cells(8, 8) = Cells(4, 8)
   Cells(8, 7) = Cells(4, 7)
   Cells(9, 7) = Cells(3, 7)
   Cells(9, 6) = Cells(3, 6)
   Cells(10, 6) = Cells(2, 6)
   Cells(2, 6) = Cells(6, 10)
   Cells(3, 6) = Cells(6, 9)
   Cells(3, 7) = Cells(5, 9)
   Cells(4, 7) = Cells(5, 8)
   Cells(3, 5) = Cells(3, 7)
   Cells(4, 5) = Cells(4, 7)
   Cells(8, 5) = Cells(8, 7)
   Cells(9, 5) = Cells(9, 7)
   Cells(4, 4) = Cells(4, 8)
   Cells(5, 4) = Cells(5, 8)
   Cells(7, 4) = Cells(7, 8)
   Cells(8, 4) = Cells(8, 8)
   Cells(5, 3) = Cells(5, 9)
   Cells(6, 3) = Cells(6, 9)
   Cells(7, 3) = Cells(7, 9)
   Cells(6, 2) = Cells(6, 10)
End Sub
```

Da es sich um ein spezielles Beispiel handelt, sind nur zwei Prozeduren vorgegeben. Zuerst der Aufbau des Formblatts mit den Daten.

Nach der manuellen Eingabe der Schleifendurchläufe in Zelle A12 müssen noch die Startwerte in den grauen Feldern eingetragen werden. Für die vorliegende Berechnung wurden diese Felder mit Null belegt. Aber auch andere Startwerte führen zum gleichen Ergebnis.

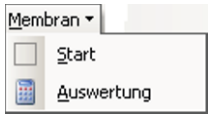


Abbildung 5-22: Menü Membran

Mit dem Start werden die Gitterpunkte der Membran berechnet und nach der Symmetrie übertragen.

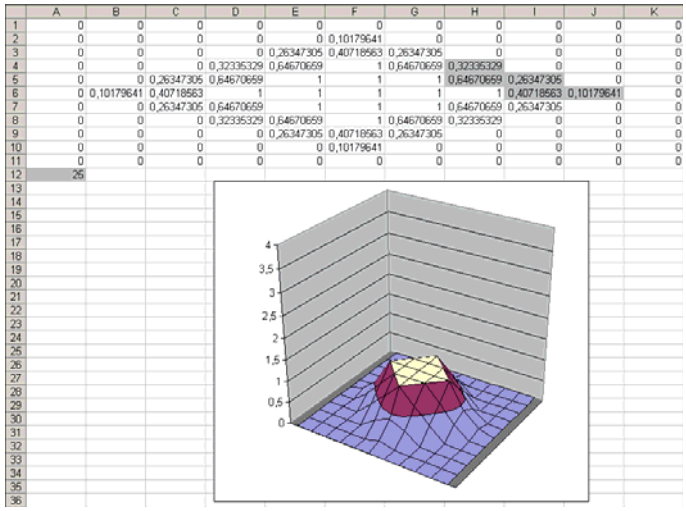


Abbildung 5-23: Auswertung zum Membranausschnitt

Die grafische Darstellung wurde manuell erstellt, da zur besseren Wiedergabe Korrekturen an den Einstellungen vorgenommen wurden. Ebenso wurden leeren Zellen mit angrenzenden Werten gefüllt.

Übung 5-4: Elektronenröhre

Eine Elektronenröhre hat den in Abbildung 5-24 dargestellten Aufbau.

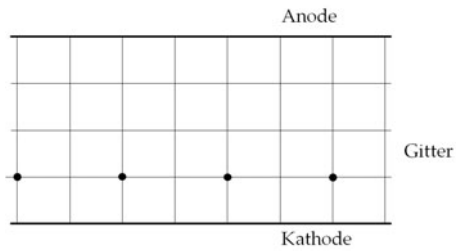


Abbildung 5-24: Schematischer Aufbau einer Elektronenröhre

Wir gehen von der vereinfachten Annahme aus, dass die Elektroden nach links und rechts unendlich fortgeführt sind. Dann genügt die Potentialverteilung im Raum zwischen den Elektroden der Laplace'schen Differentialgleichung (5.2.9).

Die Kathode hat das Potential 0 Volt und das Gitter -1 Volt. Wie groß muss die Anodenspannung gewählt werden, damit das Potential zwischen den Gitterfäden nicht negativ wird?

Benutzen Sie die dargestellte Aufteilung unter Berücksichtigung symmetrischer Verhältnisse.

6 Vektoren und Matrizen

Matrizen werden heutzutage in allen Bereichen immer dann eingesetzt, wenn es gilt große Datenmengen zu verarbeiten. Nicht zuletzt deswegen ist MS Excel mit seiner Matrixstruktur als ein überaus hilfreiches Entwicklungstool zu nennen. Die Informationsaufbereitung mit Matrizen ist ausgesprochen anschaulich. Dennoch bedarf es einiger wichtiger Matrizendefinitionen, bevor man sich mit deren Anwendung beschäftigt. Dabei werde ich nicht auf die Matrizenfunktionen von Excel eingehen.

6.1 Matrizendefinitionen

Eine Matrix ist per Definition ein rechteckiges Zahlenschema mit in der Regel mehreren Zeilen und Spalten.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{6.1.1}$$

Eine Matrix wird mit einem Großbuchstaben beschrieben und die Elemente einer Matrix mit dem zugehörigen Kleinbuchstaben. Am Index eines Elementes kann man erkennen, in welcher Zeile und Spalte sich das Element befindet.

Für die Schreibweise einer Matrix gibt es auch eine Kurzform

$$A = (a_{ik}), i = 1,...,m, k = 1,...,n \ , \tag{6.1.2}$$

für eine Matrix mit m Zeilen und n Spalten, oder noch kürzer

$$A = (a_{ik})_{(m,n)} \ . \tag{6.1.3}$$

Man sagt auch, dass die Matrix A vom Typ (m, n) ist. Die Zeilen einer Matrix heißen auch Zeilenvektoren. Analog bezeichnet man die Spalten auch als Spaltenvektoren. Eine besondere Form der Matrix ist dann gegeben, wenn m = n ist. Es liegt dann eine quadratische Matrix vor. Sind nur die diagonalen Elemente ungleich Null, so spricht man von einer Diagonalmatrix. Haben alle diese diagonalen Elemente zusätzlich den Wert 1, so heißt sie Einheitsmatrix.

Durch Vertauschung von Zeilen und Spalten einer Matrix, entsteht eine neue Matrix. Diese heißt die Transponierte der Matrix A.

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_6,

Eine Matrix der Form

$$A = \begin{pmatrix} a_{11}\, a_{12} \\ a_{21}\, a_{22} \\ a_{31}\, a_{32} \end{pmatrix}, \tag{6.1.4}$$

hat als Transponierte die Form

$$A^T = \begin{pmatrix} a_{11}\, a_{21}\, a_{31} \\ a_{12}\, a_{22}\, a_{32} \end{pmatrix}. \tag{6.1.5}$$

Eine Matrix vom Typ (m, n) hat eine Transponierte vom Typ (n, m). Die Transponierte einer Transponierten ergibt die ursprüngliche Matrix. Als Formel

$$(A^T)^T = A \ . \tag{6.1.6}$$

Für die Elemente einer Matrix bedeutet dies

$$a_{ik}^T = a_{ki} \ . \tag{6.1.7}$$

Der Algorithmus zur Bestimmung der Transponierten besteht also einfach in der Vertauschung der Indizes (Tabelle 6-1).

Beginnen wir nun mit einer Sammlung von Matrizenoperationen in einer eigenen Arbeitsmappe, da Matrizenoperationen immer wieder verwendet werden. Wir erstellen eine erste Tabelle mit dem Namen Matrix A. Eine zweite Tabelle mit dem Namen Matrix B wird automatisch als Transponierte benutzt. Allerdings lässt sie sich auch manuell neu erstellen über eine zusätzliche Prozedur. Ebenso eine dritte Tabelle Matrix C.

Tabelle 6-1: Bestimmung einer Transponierten

Eingabe der Elemente a_{ik} einer Matrix A
Bestimmung der Transponierten durch Vertauschung der Indizes i = 1 (1) m
k = i (1) n
$b_{ki} = a_{ik}$
Ausgabe der Matrix B als Transponierte

Code 6-1: Prozeduren zur den Matrizenoperationen

```
Option Explicit

'Prozedur Matrix A neu
Sub Matrix_A_Neu()
   Dim Blatt As Worksheet
   Dim Name As String

   Name = "Matrix A"
```

```
   On Error GoTo Matrix_A_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear
   Exit Sub
Matrix_A_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub

'Prozedur Matrix B neu
Sub Matrix_B_Neu()
   Dim Blatt As Worksheet
   Dim Name As String

   Name = "Matrix B"
   On Error GoTo Matrix_B_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear
   Exit Sub
Matrix_B_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub

'Prozedur Matrix C neu
Sub Matrix_C_Neu()
   Dim Blatt As Worksheet
   Dim Name As String

   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear
   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub

Sub Matrix_Transponierte()
   Dim Blatt As Worksheet
   Dim Name As String
   Dim i, k, m, Zeilen, Spalten As Integer
   Dim A() As Double


   Name = "Matrix A"
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Zeilen = Blatt.UsedRange.Rows.Count
   Spalten = Blatt.UsedRange.Columns.Count
```

```
'Matrix A lesen
   ReDim A(Zeilen, Spalten)
   For i = 1 To Zeilen
      For k = 1 To Spalten
         A(i, k) = Blatt.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob Matrix B vorhanden
   Name = "Matrix B"
   m = 0
   For Each Blatt In Sheets
      If Name = Blatt.Name Then
         m = 1
      End If
   Next

'Matrix B anlegen
   If m = 0 Then
      Set Blatt = Worksheets.Add
      Blatt.Name = Name
   End If
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear

'Bestimmung der Transponierten
   For i = 1 To Zeilen
      For k = 1 To Spalten
         Blatt.Cells(k, i) = A(i, k)
      Next k
   Next i
End Sub
```

Alle Prozeduren werden durch einen Menüpunkt aufgerufen.

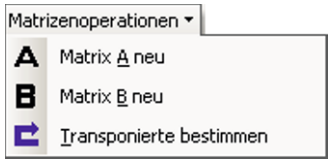


Abbildung 6-1: Erster Aufbau des Menüs Matrizenoperationen

Zwei Matrizen werden addiert, indem die Elemente mit gleichem Index addiert werden.

$$A+B=\begin{pmatrix} a_{11}\ a_{12} \\ a_{21}\ a_{22} \\ a_{31}\ a_{32} \end{pmatrix}+\begin{pmatrix} b_{11}\ b_{12} \\ b_{21}\ b_{22} \\ b_{31}\ b_{32} \end{pmatrix}=\begin{pmatrix} a_{11}+b_{11}\ a_{12}+b_{12} \\ a_{21}+b_{21}\ a_{22}+b_{22} \\ a_{31}+b_{31}\ a_{32}+b_{32} \end{pmatrix}. \qquad (6.1.8)$$

Es ist ersichtlich, dass beide Matrizen vom gleichen Typ sein müssen. Die Matrizenaddition ist sowohl kommutativ als auch assoziativ

$$\begin{aligned} A+B &= B+A \\ A+(B+C) &= (A+B)+C \end{aligned} \quad . \tag{6.1.9}$$

Für die Matrizenaddition erweitern wir den Quellcode um die nachfolgenden Zeilen. Dabei setzen wir die Matrix A und die Matrix B als Ausgangstabellen voraus.

Code 6-2: Matrizenaddition

```
Sub Matrix_Addition()
   Dim Blatt1, Blatt2, Blatt As Worksheet
   Dim Name1, Name2, Name As String
   Dim i, k, m, Z1, Z2, S1, S2 As Integer
   Dim A(), B() As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count

'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob Matrix B vorhanden
   Name2 = "Matrix B"
   Set Blatt2 = ThisWorkbook.Worksheets(Name2)
   m = 0
   For Each Blatt In Sheets
      If Name2 = Blatt2.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix B fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z2 = Blatt2.UsedRange.Rows.Count
   S2 = Blatt2.UsedRange.Columns.Count
'Matrix B lesen
   ReDim B(Z2, S2)
   For i = 1 To Z2
      For k = 1 To S2
```

```
         B(i, k) = Blatt2.Cells(i, k)
      Next k
   Next i

'Überprüfung der Typen
   If Z1 <> Z2 Or S1 <> S2 Then
      MsgBox "Matrixtypen unterschiedlich!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If

'Überprüfung, ob Matrix C vorhanden
   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear

'Addition
   For i = 1 To Z1
      For k = 1 To S1
         Blatt.Cells(i, k) = A(i, k) + B(i, k)
      Next k
   Next i

   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub
```

Analog zur Matrizenaddition definiert sich die Matrizensubtraktion

$$A-B=\begin{pmatrix} a_{11}\ a_{12} \\ a_{21}\ a_{22} \\ a_{31}\ a_{32} \end{pmatrix}-\begin{pmatrix} b_{11}\ b_{12} \\ b_{21}\ b_{22} \\ b_{31}\ b_{32} \end{pmatrix}=\begin{pmatrix} a_{11}-b_{11}\ a_{12}-b_{12} \\ a_{21}-b_{21}\ a_{22}-b_{22} \\ a_{31}-b_{31}\ a_{32}-b_{32} \end{pmatrix}. \tag{6.1.10}$$

Für die Matrizensubtraktion gelten weder das Kommutativ- noch das Assoziativgesetz. Die Prozedur ist ähnlich der der Addition.

Code 6-3: Matrizensubtraktion

```
Sub Matrix_Subtraktion()
   Dim Blatt1, Blatt2, Blatt As Worksheet
   Dim Name1, Name2, Name As String
   Dim i, k, m, Z1, Z2, S1, S2 As Integer
   Dim A(), B() As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
```

```
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count

'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob Matrix B vorhanden
   Name2 = "Matrix B"
   Set Blatt2 = ThisWorkbook.Worksheets(Name2)
   m = 0
   For Each Blatt In Sheets
      If Name2 = Blatt2.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix B fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z2 = Blatt2.UsedRange.Rows.Count
   S2 = Blatt2.UsedRange.Columns.Count
'Matrix B lesen
   ReDim B(Z2, S2)
   For i = 1 To Z2
      For k = 1 To S2
         B(i, k) = Blatt2.Cells(i, k)
      Next k
   Next i

'Überprüfung der Typen
   If Z1 <> Z2 Or S1 <> S2 Then
      MsgBox "Matrixtypen unterschiedlich!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If
'Überprüfung, ob Matrix C vorhanden
   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear
'Addition
   For i = 1 To Z1
      For k = 1 To S1
         Blatt.Cells(i, k) = A(i, k) - B(i, k)
```

```
      Next k
   Next i
   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub
```

Die Multiplikation einer Matrix mit einer Zahl (Skalar) erfolgt in der Form, dass jedes Element der Matrix mit dem Skalar multipliziert wird.

$$\alpha \cdot A = \alpha \begin{pmatrix} a_{11}\ a_{12} \\ a_{21}\ a_{22} \\ a_{31}\ a_{32} \end{pmatrix} = \begin{pmatrix} \alpha \cdot a_{11}\ \alpha \cdot a_{12} \\ \alpha \cdot a_{21}\ \alpha \cdot a_{22} \\ \alpha \cdot a_{31}\ \alpha \cdot a_{32} \end{pmatrix}. \tag{6.1.11}$$

Für eine Multiplikation eines Skalars mit einer Matrix gelten nachfolgende Gesetze:

Assoziativgesetz

$$\alpha_1 \cdot (\alpha_2 \cdot A) = (\alpha_1 \cdot \alpha_2) \cdot A \tag{6.1.12}$$

Distributivgesetz

$$\begin{aligned} (\alpha_1 \cdot \alpha_2) \cdot A &= \alpha_1 \cdot A + \alpha_2 \cdot A \\ \alpha \cdot (A + B) &= \alpha \cdot A + \alpha \cdot B \end{aligned} \tag{6.1.13}$$

Beim Aufruf dieser Prozedur wird außer der Matrix A der Skalar in Matrix B in Zelle(1,1) vorausgesetzt. Das Skalarprodukt steht dann in Matrix C.

Code 6-4: Skalarprodukt

```
Sub Matrix_Skalarprodukt()
   Dim Blatt1, Blatt2, Blatt As Worksheet
   Dim Name1, Name2, Name As String
   Dim i, k, m, Z1, S1 As Integer
   Dim A() As Double
   Dim Skalar As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count
```

```
'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob Matrix B vorhanden
   Name2 = "Matrix B"
   Set Blatt2 = ThisWorkbook.Worksheets(Name2)
   m = 0
   For Each Blatt In Sheets
      If Name2 = Blatt2.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Skalar in Matrix B fehlt!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If
   Skalar = Blatt2.Cells(1, 1)

'Überprüfung, ob Matrix C vorhanden
   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear

'Addition
   For i = 1 To Z1
      For k = 1 To S1
         Blatt.Cells(i, k) = A(i, k) * Skalar
      Next k
   Next i
   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub
```

Das Produkt einer Matrix A mit einer Matrix B ist ebenfalls wieder eine Matrix

$$C = A \cdot B \quad . \tag{6.1.14}$$

Das Element c_{ik} ist das Skalarprodukt des i-ten Zeilenvektors der Matrix A mit dem k-ten Spaltenvektor der Matrix B.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} \end{pmatrix} . \quad (6.1.15)$$

Das Matrizenprodukt ist nur definiert für Matrizen vom Typ (m, n), die multipliziert werden mit Matrizen vom Typ (n, r).

Code 6-5: Matrizenprodukt

```
Sub Matrix_Produkt()
   Dim Blatt1, Blatt2, Blatt As Worksheet
   Dim Name1, Name2, Name As String
   Dim i, j, k, m, Z1, Z2, S1, S2 As Integer
   Dim A(), B()  As Double
   Dim Sum As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count

'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob Matrix B vorhanden
   Name2 = "Matrix B"
   Set Blatt2 = ThisWorkbook.Worksheets(Name2)
   m = 0
   For Each Blatt In Sheets
      If Name2 = Blatt2.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix B fehlt!", vbOKOnly, "ACHTUNG"
```

```
      Exit Sub
   End If
   Z2 = Blatt2.UsedRange.Rows.Count
   S2 = Blatt2.UsedRange.Columns.Count

'Matrix B lesen
   ReDim B(Z2, S2)
   For i = 1 To Z2
      For k = 1 To S2
         B(i, k) = Blatt2.Cells(i, k)
      Next k
   Next i

'Überprüfung ob Multiplikation zulässig
   If S1 <> Z2 Then
      MsgBox "Multiplikation nicht möglich!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If

'Überprüfung, ob Matrix C vorhanden
   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear

'Produktbildung
   For j = 1 To S2
      For i = 1 To Z1
         Sum = 0
         For k = 1 To S1
            Sum = Sum + A(i, k) * B(k, j)
         Next k
         Blatt.Cells(i, j) = Str(Sum)
      Next i
   Next j
   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub
```

Während für die Matrizenmultiplikation das Kommutativgesetz im Allgemeinen nicht gilt, gelten das Assoziativgesetz

$$(A \cdot B) \cdot C = A \cdot (B \cdot C) \tag{6.1.16}$$

und die Distributivgesetze

$$\begin{aligned} A \cdot (B + C) &= A \cdot B + A \cdot C \\ (A + B) \cdot C &= A \cdot C + B \cdot C \end{aligned} . \tag{6.1.17}$$

Jeder quadratischen Matrix kann man auf eine bestimmte Weise einen Skalar zuordnen, der als Determinante der Matrix bezeichnet wird. Die Schreibweise ist

$$|A| = \det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \quad . \tag{6.1.18}$$

Dabei dürfen die senkrechten Striche nicht mit der Klammer der Matrix verwechselt werden. Die Berechnung der Determinanten zeigt sich am besten bei einer Matrix vom Typ (3, 3).

$$\begin{aligned} \det A &= \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \\ &= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} \quad . \\ &\quad - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32} \end{aligned} \tag{6.1.19}$$

Tabelle 6-2: Bestimmung einer Determinanten

Eingabe der n x n Elemente a_{ik} der Matrix A			
$q = \frac{n}{2}$			
Ist q = Int(q) (also n gerade)			
Ja		Nein	
r = n – 1		r = n	
Summe=0			
	i = 1 (1) r		
	j = i		
	p = 1		
		k = 1 (1) n	
		p = p * a(k,j)	
		j = j + 1	
		Ist j > n	
		Ja	Nein
		j = j - n	./.
	Summe = Summe + p		
	i = 1 (1) r		
	j = n – i + 1		
		k = 1 (1) n	
		p = p * a(k,j)	
		j = j - 1	

		Ist j < 1	
		Ja	Nein
		j = j + n	./.
	Summe = Summe - p		
Ausgabe der Determinanten Summe in Matrix C			

Code 6-6: Bestimmung der Determinante einer Matrix

```
Sub Matrix_Determinante()
   Dim Blatt1, Blatt As Worksheet
   Dim Name1, Name As String
   Dim i, j, k, m, n, Z1, S1 As Integer
   Dim A() As Double
   Dim Sum, Pro, q As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count

'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung ob quadratische Matrix vorliegt
   If Z1 <> S1 Then
      MsgBox "Keine quadratische Matrix!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If

'Überprüfung, ob Matrix C vorhanden
   Name = "Matrix C"
   On Error GoTo Matrix_C_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear

'Determinante
   q = Z1 / 2
```

```
   If q = Int(q) Then
      n = Z1 - 1
   Else
      n = Z1
   End If
   Sum = 0
   For i = 1 To n
      j = i
      Pro = 1
      For k = 1 To Z1
         Pro = Pro * A(k, j)
         j = j + 1
         If j > Z1 Then j = j - Z1
      Next k
      Sum = Sum + Pro
   Next i
   For i = 1 To n
      j = Z1 - i + 1
      Pro = 1
      For k = 1 To Z1
         Pro = Pro * A(k, j)
         j = j - 1
         If j < 1 Then j = j + Z1
      Next k
      Sum = Sum - Pro
   Next i
   Blatt.Cells(1, 1) = Sum
   Exit Sub
Matrix_C_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub
```

Determinanten dienen zur Lösung linearer Gleichungssysteme und für die Theorie der Eigenwerte ebenso, wie zur Bestimmung der Inversen einer Matrix.

Unter einer Inversen einer Matrix A versteht man eine Matrix B in der Form, das gilt

$$A \cdot B = E \ , \tag{6.1.20}$$

mit E als Einheitsmatrix. Die inverse Matrix von A wird mit A^{-1} bezeichnet

$$A \cdot A^{-1} = E \ . \tag{6.1.21}$$

Die Berechnung einer Inversen ist etwas kompliziert. Eine Methode ist das Gaußsche Eliminationsverfahren, das im nachfolgenden Kapitel behandelt wird. Eine zweite Methode ist die Berechnung mit Hilfe der Determinanten

$$A^{-1} = \frac{1}{\det A} \cdot A^{+} \ . \tag{6.1.22}$$

Dies setzt jedoch voraus, dass diese ungleich Null ist und mit A^+ die komplementäre Matrix von A vorliegt. Eine Matrix, deren Determinante Null ist, wird als singulär bezeichnet. Singuläre Matrizen besitzen keine Inverse.

Die komplementäre Matrix (wird auch adjungierte genannt) A^+ einer quadratischen Matrix A bestimmt sich aus deren Unterdeterminanten. Die Unterdeterminante det A_{ij} bestimmt sich aus der Matrix A unter Streichung der Zeile i und Spalte j. Die komplementäre Matrix hat dann die Elemente

$$(-1)^{i+j} \cdot \det A_{ij} \ . \tag{6.1.23}$$

Die komplementäre Matrix wird auch als Matrix der Kofaktoren bezeichnet.

Code 6-7: Bestimmung der komplementären Matrix

```
Sub Matrix_Komplement()
   Dim Blatt1, Blatt As Worksheet
   Dim Name1, Name As String
   Dim i, j, k, m, n, Z1, S1 As Integer
   Dim j1, l, l1 As Integer
   Dim A(), B() As Double
   Dim Det, q As Double

'Überprüfung, ob Matrix A vorhanden
   Name1 = "Matrix A"
   Set Blatt1 = ThisWorkbook.Worksheets(Name1)
   m = 0
   For Each Blatt In Sheets
      If Name1 = Blatt1.Name Then
         m = 1
      End If
   Next
   If m = 0 Then
      MsgBox "Matrix A fehlt!", vbOKOnly, "ACHTUNG"
      Exit Sub
   End If
   Z1 = Blatt1.UsedRange.Rows.Count
   S1 = Blatt1.UsedRange.Columns.Count

'Matrix A lesen
   ReDim A(Z1, S1)
   For i = 1 To Z1
      For k = 1 To S1
         A(i, k) = Blatt1.Cells(i, k)
      Next k
   Next i

'Überprüfung, ob quadratische Matrix vorliegt
   If Z1 <> S1 Then
      MsgBox "Keine quadratische Matrix!", vbOKOnly, _
             "ACHTUNG"
      Exit Sub
   End If
```

```
'Überprüfung, ob Matrix B vorhanden
   Name = "Matrix B"
   On Error GoTo Matrix_B_Neu
   Set Blatt = ThisWorkbook.Worksheets(Name)
   Blatt.Activate
   Blatt.Cells.Clear
'Determinante
   q = Z1 / 2
   If q = Int(q) Then
      n = Z1 - 1
   Else
      n = Z1
   End If
   For i = 1 To n
      For k = 1 To n
         'Bildung der Untermatrix
         ReDim B(n - 1, n - 1)
         For j = 1 To n
            For l = 1 To n
               If j <> i And l <> k Then
                  j1 = j
                  If j1 > i Then j1 = j1 - 1
                  l1 = l
                  If l1 > k Then l1 = l1 - 1
                  B(j1, l1) = A(j, l)
               End If
            Next l
         Next j
         Call Matrix_Kofaktoren(B, n - 1, Det)
         Blatt.Cells(i, k) = (-1) ^ (i + k) * Det
      Next k
   Next i
   Exit Sub
Matrix_B_Neu:
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Resume
End Sub

Sub Matrix_Kofaktoren(A, n, Det)
   Dim i, j, k, r As Integer
   Dim Pro, q As Double

   Det = 0
   q = n / 2
   If q = Int(q) Then
      r = n - 1
   Else
      r = n
   End If
   For i = 1 To r
      j = i
      Pro = 1
      For k = 1 To n
         Pro = Pro * A(k, j)
         j = j + 1
         If j > n Then j = j - n
```

```
      Next k
      Det = Det + Pro
   Next i

   For i = 1 To r
      j = n - i + 1
      Pro = 1
      For k = 1 To n
         Pro = Pro * A(k, j)
         j = j - 1
         If j < 1 Then j = j + n
      Next k
      Det = Det - Pro
   Next i
End Sub
```

Inzwischen haben sich einige weitere Menüpunkte angesammelt.

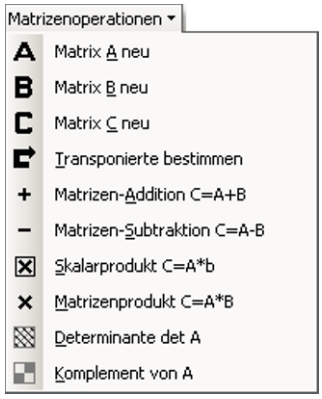


Abbildung 6-2: Menü Matrizenoperationen

An einem einfachen Beispiel soll noch einmal der Berechnungsvorgang verdeutlicht werden. Gegeben sei die Matrix

$$A = \begin{pmatrix} 1 & 3 & -2 \\ 0 & -2 & 3 \\ 2 & -1 & 3 \end{pmatrix}.$$

Dann ergibt die Berechnung als Determinante den Wert 7. Die Berechnung des Komplements von A erbringt

$$A^{+} = \begin{pmatrix} -3 & 6 & 4 \\ -7 & 7 & 7 \\ 5 & -3 & -2 \end{pmatrix}.$$

Ebenso bestimmt sich die transponierte Matrix zu

$$A^{T} = \begin{pmatrix} 1 & 0 & 2 \\ 3 & -2 & -1 \\ -2 & 3 & 3 \end{pmatrix}.$$

Das Matrizenprodukt $A^T A^{-1}$ liefert wiederum

$$A^T \cdot A^+ = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{pmatrix} = \det A \cdot E \ .$$

Dies ist auch die Aussage des Laplace'schen Entwicklungssatzes. Nach diesem lässt sich die inverse Matrix bestimmen (6.1.22)

$$\frac{1}{\det A} \cdot (A^+)^T = A^{-1} = \begin{pmatrix} -0{,}428... & -1 & 0{,}714... \\ 0{,}857 & 1 & -0{,}428 \\ 0{,}571... & 1 & -0{,}285... \end{pmatrix} .$$

Nach (6.1.21) ergibt sich dann auch

$$A \cdot A^{-1} = E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} .$$

Übung 6-1: Matrizenumformungen

Fassen Sie sich wiederholende Anweisungen, und da gibt es eine Menge, zu eigenen Prozeduren zusammen. Schreiben Sie ergänzende Prozeduren zur Matrizenumformung.

Eine n-reihige Determinante lässt sich auf (n-1)-reihige Determinanten zurückführen, (n-1)-reihige Determinanten auf (n-2)-reihige Determinanten und so weiter. Schreiben Sie eine entsprechend rekursive Prozedur.

Übung 6-2: Namen für Matrizen verwenden

Zellen oder Zellbereichen einen Namen geben zu können, ist eine der wesentlichen Stärken von Excel. Namen sind Objekte, die der Mappe zugeordnet werden. Ihr Container-Objekt ist *Names*. Im Namensmanager kann einem Namen jederzeit wieder ein neuer Bereich zugeordnet werden. Unabhängig von der Existenz eines Namens, und damit eines Bereichs, können Prozeduren zur Verwendung von Namen geschrieben werden.

Betrachten wir dazu die Addition von Matrizen, die wir ja bereits abgehandelt haben. Als Namen für die Matrizen legen wir Matrix_A und Matrix_B fest. In unserer Excel-Mappe existiert eine Tabelle *Matrizen* und eine Tabelle *Addition*. Auf der Tabelle Matrizen werden gleich große Bereiche mit den Namen *Matrix_A* und *Matrix_B* angelegt. Diese Bereiche können sich auch überschneiden. Die Prozedur *Matrizenaddition* liefert in der Tabelle *Addition* das Ergebnis.

Code 6-8: Matrizenaddition

```
Option Explicit

Sub Matrizenaddition()
   Dim objBook    As Object
   Dim objTab1    As Object
   Dim objTab2    As Object
   Dim objMatA    As Object
   Dim objMatB    As Object
   Dim objA       As Range
   Dim objB       As Range
   Dim iRowA      As Integer
   Dim iRowsA     As Integer
   Dim iColA      As Integer
   Dim iColsA     As Integer
   Dim iRowB      As Integer
   Dim iRowsB     As Integer
   Dim iColB      As Integer
   Dim iColsB     As Integer
   Dim iRow       As Integer
   Dim iCol       As Integer
   Dim iM()       As Integer

'Objekte
   Set objBook = ThisWorkbook
   Set objTab1 = objBook.Worksheets("Matrizen")
   Set objTab2 = objBook.Worksheets("Addition")
   Set objMatA = objBook.Names("Matrix_A")
   Set objMatB = objBook.Names("Matrix_B")
   Set objA = Range(objMatA)
   Set objB = Range(objMatB)

'Grenzen
   With objA
      iRowA = .Row
      iRowsA = .Rows.Count
      iColA = .Column
      iColsA = .Columns.Count
   End With
   With objB
      iRowB = .Row
      iRowsB = .Rows.Count
      iColB = .Column
      iColsB = .Columns.Count
   End With
   If Not iRowsA = iRowsB Or _
      Not iColsA = iColsB Then
      MsgBox "Falsche Grenzen!"
      Exit Sub
   End If
   ReDim iM(iRowsA, iColsA) As Integer

   For iRow = iRowA To iRowA + iRowsA - 1
      For iCol = iColA To iColA + iColsA - 1
         iM(iRow - iRowA + 1, iCol - iColA + 1) = objTab1.Cells(iRow, iCol)
      Next iCol
```

```
    Next iRow

    For iRow = iRowB To iRowB + iRowsB - 1
       For iCol = iColB To iColB + iColsB - 1
          iM(iRow - iRowB + 1, iCol - iColB + 1) = _
             iM(iRow - iRowB + 1, iCol - iColB + 1) + objTab1.Cells(iRow, iCol)
       Next iCol
    Next iRow

    objTab2.Select
    For iRow = 1 To iRowsA
       For iCol = 1 To iColsA
          objTab2.Cells(iRow, iCol) = iM(iRow, iCol)
       Next iCol
    Next iRow
End Sub
```

Schreiben Sie weitere Prozeduren für die anderen Matrizenumformungen.

6.2 Lösungen von Gleichungssystemen

Die zuvor behandelten Gleichungssysteme lassen sich mit Hilfe von Algorithmen auf Matrizen effektiver realisieren.

Ein allgemeines Gleichungssystem der Form

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = y_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = y_2 \\ &\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = y_m \end{aligned} \tag{6.2.1}$$

wird symbolisch in der Form

$$A \cdot x = b \tag{6.2.2}$$

geschrieben. Darin ist A eine Matrix mit m Zeilen und n Spalten

$$A = \begin{bmatrix} a_{11}a_{12}\ldots a_{1n} \\ a_{21}a_{22}\ldots a_{2n} \\ \ldots \\ a_{m1}a_{m2}\ldots a_{mn} \end{bmatrix} \tag{6.2.3}$$

und x ein n-dimensionaler Vektor

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} \tag{6.2.4}$$

sowie y ein m-dimensionaler Vektor

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} . \tag{6.2.5}$$

Die Rechenregeln für Matrizen und Vektoren haben wir im vorherigen Kapitel behandelt. Ebenso einfache Umformungsregeln, wie z. B. das Erstellen einer transponierten Matrix. Begriffe wie symmetrische Matrix setze ich voraus. Ich komme daher direkt zu einer Anwendung, die wir bereits kennen. Damit meine ich den Gauß-Algorithmus zur Auflösung eines linearen Gleichungssystems

$$A \cdot x = b \ , \ (\det A \neq 0) . \tag{6.2.6}$$

Wie wir es bereits praktiziert haben, gelangt man zur Lösung, wenn in den Gleichungen Unbekannte so eliminiert werden, dass zum Schluss eine Unbekannte bestimmt werden kann und durch Rückwärtseinsetzen ebenfalls alle anderen.

Dies erreicht man in der Matrizenanwendung durch fortgesetzte Linearkombinationen jeweils zweier Gleichungen zu einem gestaffelten System

$$R \cdot x = b^* \tag{6.2.7}$$

mit der Rechtsdreiecksmatrix R. Man spricht von Triangularisierung. Damit lassen sich dann die Unbekannten x_n, x_{n-1}, x_{n-2}, … , x_1 durch Rückwärtseinsetzen berechnen. Also nichts Neues, sondern nur eine andere Schreibweise.

Zur Ermittlung von R aus A sind n-1 Eliminationsschritte erforderlich. Diese sind wie folgt durchzuführen. Im k-ten Schritt wird die k-te Zeile (Pivotzeile) nacheinander mit den Faktoren

$$c_{ik} = \frac{a^*_{ik}}{a^*_{kk}} \ , \ (i = k+1,\dots, n) \tag{6.2.8}$$

multipliziert und von den Zeilen i = k+1, …, n subtrahiert, wodurch alle Elemente a_{ik}* unterhalb der Diagonalen zu Null werden. Nach n-1 Schritten ergibt sich so die Matrix R. Die mit einem * gekennzeichneten Größen, sind die durch die bereits vorangegangenen k-1 Schritte veränderten Elemente der Matrix A.

Zur Verringerung von Rundungsfehlern ist es zweckmäßig, vor jedem Eliminationsschritt einen Zeilentausch so durchzuführen, dass das betragsgrößte Element aller a_{ik}* (i = k+1, …, n) zum Pivotelement wird (Spaltenpivotisierung).

Der Betrag der Determinante von A wird dadurch nicht geändert, doch es wechselt das Vorzeichen bei jedem Zeilentausch. Mit einer Anzahl p von Zeilenvertauschungen gilt

$$\det A = (-1)^p \det R \ . \tag{6.2.9}$$

Der Gauß Algorithmus in Matrizenform ändert sich unwesentlich zum Kapitel 3.

Tabelle 6-3: Algorithmus der Gauß-Elimination

Eingabe der Koeffizienten des Gleichungssystems
Elimination i = 1 (1) m-1
j = i+1 (1) m
$c = \frac{a_{ji}}{a_{ii}}$
k = 1 (1) n
$a_{jk} = a_{jk} - c \cdot a_{ik}$
$y_j = y_j - c \cdot y_1$
Rückwärtsrechnung i= n (-1) 1
$\sum a = y_i$
k = i + 1 (1) n
$\sum a = \sum a - a_{ik} \cdot x_k$
$x_i = \frac{\sum a}{a_{ii}}$

Dafür hat das Programm eine etwas andere Struktur, denn die einzelnen Schritte wurden als Prozeduren ausgelagert.

Code 6-9: Gauß-Elimination in Matrizenform

```
Option Explicit
Dim A(), y(), x() As Double
Dim m, n As Integer

Sub Gauss_Leer()
   ThisWorkbook.Worksheets("Gauss").Cells.Clear
   Range("A1").Select
End Sub

Sub Gauss_Testdaten()
   Dim i, j As Integer
   For i = 1 To 15
      For j = 1 To 15
         Cells(i, j) = 0
         If i = j Then Cells(i, j) = 4
      Next j
   Next i
   Cells(1, 2) = -1: Cells(1, 6) = -1
   Cells(2, 1) = -1: Cells(2, 3) = -1: Cells(2, 7) = -1
   Cells(3, 2) = -1: Cells(3, 4) = -1: Cells(3, 8) = -1
   Cells(4, 3) = -1: Cells(4, 5) = -1: Cells(4, 9) = -1
   Cells(5, 4) = -1: Cells(5, 10) = -1
   Cells(6, 1) = -1: Cells(6, 7) = -1: Cells(6, 11) = -1
```

```
   Cells(7, 2) = -1: Cells(7, 6) = -1
   Cells(7, 8) = -1: Cells(7, 12) = -1
   Cells(8, 3) = -1: Cells(8, 7) = -1
   Cells(8, 9) = -1: Cells(8, 13) = -1
   Cells(9, 4) = -1: Cells(9, 8) = -1
   Cells(9, 10) = -1: Cells(9, 14) = -1
   Cells(10, 5) = -1: Cells(10, 9) = -1
   Cells(10, 15) = -1
   Cells(11, 6) = -1: Cells(11, 12) = -1
   Cells(12, 7) = -1: Cells(12, 11) = -1
   Cells(12, 13) = -1
   Cells(13, 8) = -1: Cells(13, 12) = -1
   Cells(13, 14) = -1
   Cells(14, 9) = -1: Cells(14, 13) = -1
   Cells(14, 15) = -1
   Cells(15, 10) = -1: Cells(15, 14) = -1
   Cells(1, 17) = 24
   Cells(2, 17) = 18
   Cells(3, 17) = 24
   Cells(4, 17) = 36
   Cells(5, 17) = 84
   Cells(6, 17) = 22
   Cells(7, 17) = 0
   Cells(8, 17) = 0
   Cells(9, 17) = 0
   Cells(10, 17) = 30
   Cells(11, 17) = 34
   Cells(12, 17) = 24
   Cells(13, 17) = 32
   Cells(14, 17) = 24
   Cells(15, 17) = 32
End Sub

Sub Gauss_Daten_lesen()
   Dim MyDoc As Object
   Dim nRows, nCols, i, j As Integer

   Set MyDoc = ThisWorkbook.Worksheets("Gauss")
   nRows = MyDoc.UsedRange.Rows.Count
   nCols = MyDoc.UsedRange.Columns.Count
   m = nRows
   n = nCols - 2
   ReDim A(m, n), y(m), x(n)

   For i = 1 To m
      y(i) = Cells(i, nCols)
   Next i
   For i = 1 To m
      For j = 1 To n
         A(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Sub Gauss_Elimination()
   Dim c As Double
   Dim i, j, k As Integer
```

```
   For i = 1 To m - 1
      For j = i + 1 To m
         c = A(j, i) / A(i, i)
         For k = 1 To n
            A(j, k) = A(j, k) - c * A(i, k)
         Next k
         y(j) = y(j) - c * y(i)
      Next j
   Next i
End Sub

Sub Gauss_Rückwärtsrechnung()
   Dim i, j, k As Integer
   Dim s As Double

   For i = n To 1 Step -1
      s = y(i)
      For k = i + 1 To n
         s = s - A(i, k) * x(k)
      Next k
      x(i) = s / A(i, i)
   Next i
End Sub

Sub Gauss_Auswertung()
   Dim i, j As Integer
   Dim s As Double

   Call Gauss_Daten_lesen

   Call Gauss_Elimination

   Call Gauss_Rückwärtsrechnung

'Ausgabe
   For i = 1 To m
      For j = 1 To n
         Cells(m + 1 + i, j) = A(i, j)
      Next j
      Cells(m + 1 + i, n + 2) = y(i)
   Next i
   For j = 1 To n
      Cells(2 * m + 3, j) = x(j)
   Next j

'Testrechnung
   s = 0
   For j = 1 To n
      s = s + A(1, j) * x(j)
   Next j
   Cells(2 * m + 3, n + 2) = s
End Sub
```

Nach der Installation des Menüs

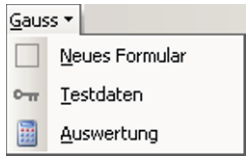


Abbildung 6-3: Menü zum Gauß-Algorithmus

ist das Ergebnis mit den Testdaten das Gleiche. Daher erspare ich mir eine nochmalige grafische Darstellung.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	4	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0		24
2	-1	4	-1	0	0	0	-1	0	0	0	0	0	0	0	0		18
3	0	-1	4	-1	0	0	0	-1	0	0	0	0	0	0	0		24
4	0	0	-1	4	-1	0	0	0	-1	0	0	0	0	0	0		36
5	0	0	0	-1	4	0	0	0	0	-1	0	0	0	0	0		84
6	-1	0	0	0	0	4	-1	0	0	0	-1	0	0	0	0		22
7	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0	0	0		0
8	0	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0	0		0
9	0	0	0	-1	0	0	0	-1	4	-1	0	0	0	-1	0		0
10	0	0	0	0	-1	0	0	0	-1	4	0	0	0	0	-1		30
11	0	0	0	0	0	-1	0	0	0	0	4	-1	0	0	0		34
12	0	0	0	0	0	0	-1	0	0	0	-1	4	-1	0	0		24
13	0	0	0	0	0	0	0	-1	0	0	0	-1	4	-1	0		32
14	0	0	0	0	0	0	0	0	-1	0	0	0	-1	4	-1		24
15	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	4		32
16																	
17	4	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0		24
18	0	3,75	-1	0	0	-0,25	-1	0	0	0	0	0	0	0	0		24
19	0	0	3,7333333	-1	0	-0,0666667	-0,2666667	-1	0	0	0	0	0	0	0		30,4
20	0	0	0	3,7321429	-1	-0,0178571	-0,0714286	-0,2678571	-1	0	0	0	0	0	0		44,1428571
21	0	0	0	0	3,732057	-0,0047847	-0,0191388	-0,0717703	-0,2679426	-1	0	0	0	0	0		95,8277512
22	0	0	0	0	0	3,7320513	-1,0717949	-0,0192308	-0,0051282	-0,0012821	-1	0	0	0	0		30,4769231
23	0	0	0	0	0	0	3,4050154	-1,0824459	-0,0219856	-0,0054964	-0,2871865	-1	0	0	0		18,6602542
24	0	0	0	0	0	0	0	3,3673325	-1,0839386	-0,0209847	-0,0964487	-0,3178975	-1	0	0		19,2429379
25	0	0	0	0	0	0	0	0	3,3637534	-1,0785871	-0,0342751	-0,1087875	-0,3218983	-1	0		25,0643257
26	0	0	0	0	0	0	0	0	0	3,3860624	-0,0123985	-0,0384779	-0,1094484	-0,3206499	-1		63,8743053
27	0	0	0	0	0	0	0	0	0	0	3,7046717	-1,0948697	-0,0323232	-0,0113636	-0,0036616		44,7805556
28	0	0	0	0	0	0	0	0	0	0	0	3,3488754	-1,1156118	-0,0393427	-0,0124456		46,0656823
29	0	0	0	0	0	0	0	0	0	0	0	0	3,2967608	-1,119266	-0,0365012		57,9143283
30	0	0	0	0	0	0	0	0	0	0	0	0	0	3,2918553	-1,1072467		57,8407296
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3,3317848		71,1758408
32																	
33	16,0313341	20,3130166	25,1700181	31,1461428	35,96011	19,8123199	22,050714	25,2209131	27,4544407	28,6943075	19,1672316	22,8566063	26,2084797	24,7563993	21,3626767		24

Abbildung 6-4: Testdatenauswertung (siehe auch Kapitel 3)

Übung 6-3: Zeilentausch

Dem Algorithmus fehlt der zuvor beschriebene Zeilentausch zur Verringerung von Rundungsfehlern. Ergänzen Sie diese Anweisungen. Ebenso werden Gültigkeitsbedingungen nicht beachtet. Diese sind ebenfalls zu ergänzen.

6.3 Differenzenverfahren für gewöhnliche Differentialgleichungen

Das Ersetzen von Differentialquotienten durch Differenzenquotienten haben wir bereits in den vorangegangenen Kapiteln praktiziert.

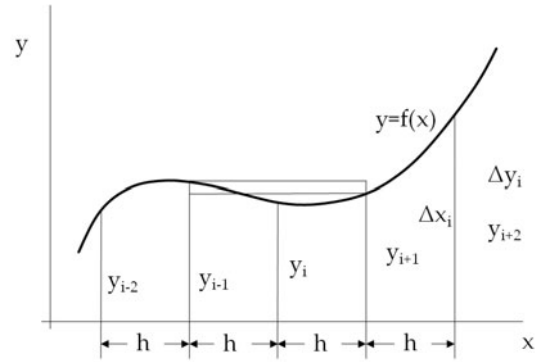


Abbildung 6-5: Differenzen

Der Differentialquotient

$$\frac{dy}{dx} = y' \tag{6.3.1}$$

wird durch den hier zentralen Differenzenquotienten

$$y_i' = \left(\frac{\Delta y}{\Delta x}\right)_i = \frac{y_{i+1} - y_{i-1}}{2h} \tag{6.3.2}$$

an einer beliebigen Stelle i approximiert. Auf die gleiche Weise können ebenso die höheren Ableitungen approximiert werden.

So gilt

$$y_i'' = \left(\frac{\Delta^2 y}{\Delta x^2}\right)_i = \left(\frac{\Delta}{\Delta x}\left(\frac{\Delta y}{\Delta x}\right)\right)_i = \frac{1}{h}\left(\frac{y_{i+1} - y_i}{h} - \frac{y_i - y_{i-1}}{h}\right). \tag{6.3.3}$$

Die zentralen Differenzenformeln für die ersten vier Ableitungen lauten

$$\begin{aligned}
y_i' &= \frac{1}{2 \cdot h}\left(y_{i+1} - y_{i-1}\right) \\
y_i'' &= \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right) \\
y_i''' &= \frac{1}{2 \cdot h^3}\left(y_{i+2} - 2y_{i+1} + 2y_{i-1} - y_{i-2}\right) \\
y_i'''' &= \frac{1}{h^4}\left(y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}\right)
\end{aligned} \tag{6.3.4}$$

Beispiel 6-1: Einseitig eingespannter Biegeträger

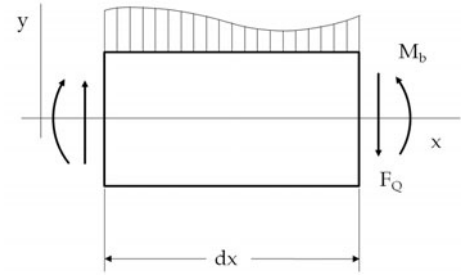


Abbildung 6-6: Gleichgewicht eines finiten Elements unter Biegebelastung

Damit ein finites Trägerelement sich im Gleichgewicht befindet, müssen folgende Bedingungen erfüllt sein.

$$\frac{d^2}{dx^2}\left(E\cdot I(x)\frac{d^2y(x)}{dx^2}\right)=q(x)$$

$$M(x)=-E\cdot I(x)\frac{d^2y(x)}{dx^2} \qquad (6.3.5)$$

$$F_Q(x)=\frac{dM(x)}{dx}=-\frac{d}{dx}\left(E\cdot I(x)\frac{d^2y(x)}{dx^2}\right)$$

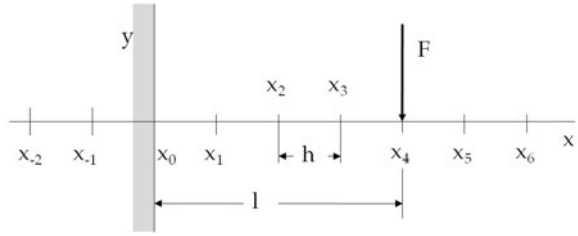


Abbildung 6-7: Einseitig eingespannter Biegeträger

Für einen einseitig eingespannten Biegeträger gelten zusätzlich die Randbedingungen

$$\begin{aligned} &y(0)=0\\ &y'(0)=0\\ &y''(l)=0, M(l)=0\\ &y'''(l)=-\frac{F}{E\cdot I}, F_Q(l)=F\\ &y''''=0 \end{aligned} \qquad (6.3.6)$$

Zur Vereinfachung wurde der Träger in vier Teile gleicher Länge unterteilt, und es ergeben sich die Differenzen

$$y_{i-2}-4y_{i-1}+6y_i-4y_{i+1}+y_{i+2}=0, i=0,...,4 \quad . \qquad (6.3.7)$$

Tabelle 6-4: Gleichungssystem

y_{-2}	$-4y_{-1}$	$+6y_0$	$-4y_1$	$+y_2$				$= 0$
	y_{-1}	$-4y_0$	$+6y_1$	$-4y_2$	$+ y_3$			$= 0$
		y_0	$-4y_1$	$+6y_2$	$-4y_3$	$+ y_4$		$= 0$
			y_1	$-4y_2$	$+6y_3$	$-4y_4$	$+y_5$	$= 0$
				y_2	$-4y_3$	$+6y_4$	$-4y_5$	$+ y_6$ $= 0$

Wir erhalten ein lineares Gleichungssystem mit 9 Unbekannten. Aus den Randbedingungen

$$\begin{aligned} & y_0 = 0 \\ & -y_{-1} + y_1 = 0 \\ & y_3 - 2 \cdot y_4 + y_5 = 0 \\ & -y_2 + 2 \cdot y_3 - 2 \cdot y_5 + y_6 = -\frac{F \cdot l^3}{32 \cdot E \cdot I} \end{aligned} \qquad (6.3.8)$$

folgt durch Umstellung

$$\begin{aligned} & -y_{-1} = y_1 \\ & y_5 = -y_3 + 2 \cdot y_4 \\ & y_6 = +y_2 - 2 \cdot y_3 + 2 \cdot y_5 - \frac{F \cdot l^3}{32 \cdot E \cdot I} \end{aligned} \quad . \qquad (6.3.9)$$

und eingesetzt

Tabelle 6-5: Umgeformtes Gleichungssystem

y_{-2}	$-4y_{-1}$	$+6y_0$	$-4y_1$	$+y_2$		$= 0$
		$-4y_0$	$+7y_1$	$-4y_2$	$+ y_3$	$= 0$
		y_0	$-4y_1$	$+6y_2$	$-4y_3$	$+ y_4$ $= 0$
			y_1	$-4y_2$	$+5y_3$	$-2y_4$ $= 0$
				$2y_2$	$-4y_3$	$+2y_4$ $= c$

mit

$$c = \frac{F \cdot l^3}{32 \cdot E \cdot I} \qquad (6.3.10)$$

Die erste Gleichung werden wir nicht verwenden, weil so y-2 raus fällt. Was bleibt ist die Matrizengleichung mit der obigen Matrix

$$A \cdot x = b \quad , \qquad (6.3.11)$$

die wir direkt mittels des zuvor erstellten Gaus-Algorithmus lösen können.

Mit den Werten

$$E_{Stahl} = 21000 \frac{N}{mm^2}$$

$$I = \frac{b \cdot h^3}{12} = \frac{50 \cdot 200^3}{12} = 33.333.333 mm^4$$

$$l = 875 mm$$

$$F = 1000 N$$

ergibt sich für die Konstante c als gerundeten Wert 0,03.

	A	B	C	D	E	F
1	7	-4	1	0		0
2	-4	6	-4	1		0
3	1	-4	5	-2		0
4	0	2	-4	2		0,03
5						
6	7	-4	1	0		0
7	0	3,71428571	-3,42857143	1		0
8	0	0	1,69230769	-1,07692308		0
9	0	0	0	0,09090909		0,03
10						
11	0,03	0,105	0,21	0,33		0
12						

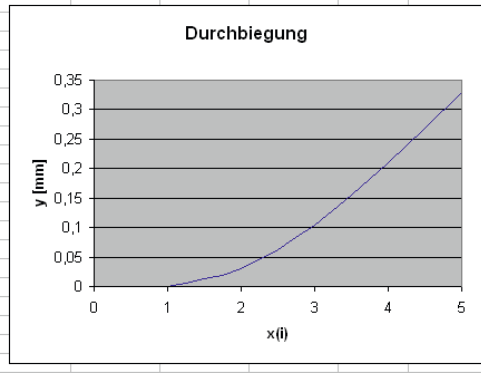


Abbildung 6-8: Auswertung nach der Gauß-Methode

Übung 6-4: Taylor-Reihenentwicklung

Die approximierenden Differenzen nach (6.3.4) lassen sich über eine Taylor-Reihenentwicklung von y(x) noch verbessern. So gilt an der Stelle xi

$$y_{i+1} = y(x_i + h) = \sum_{n=0}^{\infty} \frac{h^n}{n!} \tag{6.3.12}$$

und für die erste Ableitung

$$\left(\frac{\Delta y}{\Delta x}\right)_i = \frac{1}{2 \cdot h}\left(-\sum_{n=0}^{\infty} \frac{(-h)^n}{n!} y^{(n)}(x_i) + \sum_{n=0}^{\infty} \frac{(h)^n}{n!} y^{(n)}(x_i) + \right) \tag{6.3.13}$$

$$= y_i' + \frac{h^2}{6} y_i''' + \frac{h^4}{120} y_i^{(5)} + \ldots$$

6.4 Eigenwertprobleme

In der Mechanik führen viele Probleme auf die Lösung von Eigenwertproblemen der Form

$$(A - \lambda \cdot B) \cdot x = 0 \ . \tag{6.4.1}$$

zurück. Darin sind A und B quadratische Matrizen. Gesucht sind die Eigenwerte λ und der zugehörige Eigenvektor x. In der Regel kann das allgemeine Eigenwertproblem überführt werden in das spezielle Eigenwertproblem

$$(A - \lambda \cdot E)x = 0 \ . \tag{6.4.2}$$

Darin ist E die Einheitsmatrix. Diese Schreibweise stellt ein homogenes Gleichungssystem dar, dass nur dann eine nichttriviale Lösung besitzt, wenn gilt

$$\det(A - \lambda \cdot E) = 0 \, , \tag{6.4.3}$$

wenn also die Determinante Null ist.

Eigenwertprobleme treten z. B. immer da auf, wo etwas schwingt oder wo Schwingungen verhindert werden sollen. Beispiele sind Membranen, Platten, Tragwerke, usw.

Beispiel 6-2: Freie Biegeschwingung eines geraden Balkens

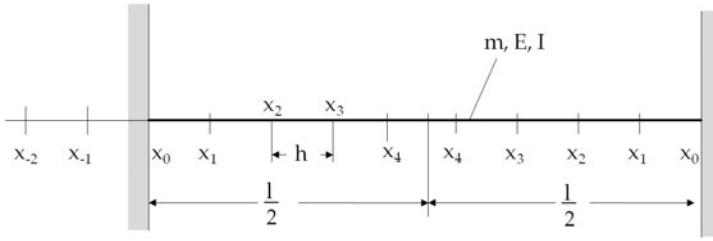


Abbildung 6-9: Beidseitig eingespannter Träger

Die Differentialgleichung für die freie Eigenschwingung eines geraden Trägers lautet

$$\frac{d^2}{dx^2}\left(E \cdot I(x) \frac{d^2 y(x)}{dx^2} \right) - \omega^2 \cdot m(x) \cdot y(x) = 0 \tag{6.4.4}$$

oder umgestellt

$$y^{(4)} - \frac{\omega^2 \cdot m}{E \cdot I} = 0 \ . \tag{6.4.5}$$

Eine Approximation durch einfache Differenzen liefert

$$y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2} - \lambda y_i = 0 \ , \tag{6.4.6}$$

mit

$$\lambda = \frac{m \cdot h^4}{E \cdot I} \omega^2 \quad . \tag{6.4.7}$$

Wie zuvor erhalten wir ein Gleichungssystem, diesmal mit 6 Unbekannten.

Tabelle 6-6: Gleichungssystem

y_{-1}	$-4y_0$	$+6y_1$	$-4y_2$	$+ y_3$			$= 0$
	y_0	$-4y_1$	$+6y_2$	$-4y_3$	$+ y_4$		$= 0$
		y_1	$-4y_2$	$+6y_3$	$-4y_4$	$+y_4$	$= 0$
			y_2	$-4y_3$	$+6y_4$	$-4y_4$	$+ y_3$ $= 0$

Mit den Randbedingungen

$$\begin{aligned} y_0 &= 0 \\ y_0' &= 0 \\ y_{-1} &= y_1 \end{aligned} \tag{6.4.8}$$

ergibt sich die endgültige Matrix nach Tabelle 6-5 für die Gleichung (6.4.2).

Tabelle 6-7: Gleichungssystem

$7y_1$	$-4y_2$	$+ y_3$		$= 0$
$-4y_1$	$+6y_2$	$-4y_3$	$+ y_4$	$= 0$
y_1	$-4y_2$	$+6y_3$	$-3y_4$	$= 0$
	y_2	$-3y_3$	$+2y_4$	$= 0$

Die numerischen Verfahren zur Berechnung der Eigenwerte einer Matrix unterscheidet man nach direkten und indirekten Methoden. Die direkten Methoden liefern die Wertetabelle des charakteristischen Polynoms und dessen Ableitung. Die indirekten Methoden umgehen die Aufstellung des charakteristischen Polynoms und versuchen die Eigenwerte und Eigenvektoren sukzessive anzunähern.

Das Iterations-Verfahren nach von Mises ist eine indirekte Methode zur Bestimmung des betragsgrößten Eigenwertes und des zugehörigen Eigenvektors. Ausgehend von einem beliebigen Startvektor x(0), der die Basisdarstellung

$$x^{(0)} = c_1 \cdot z^1 + c_2 \cdot z^2 + \ldots + c_n \cdot z^n \tag{6.4.9}$$

besitzt und so gewählt ist, dass alle

$$c_k \neq 0, k = 1,\ldots, n \tag{6.4.10}$$

folgt der erste Schritt

$$x^{(1)} = A \cdot x^{(0)} \tag{6.4.11}$$

und allgemein

$$x^{(m+1)} = A \cdot x^{(m)} \quad . \tag{6.4.12}$$

Äquivalent dazu gilt

$$x^{(m)} = A^m \cdot x^{(0)} \ . \tag{6.4.13}$$

Wegen dieser Äquivalenz auch oft als Potenzmethode bezeichnet. Es gilt weiterhin

$$x^{(m)} = A^m \cdot x^{(0)} = \lambda_1^m \cdot \left(c_1 \cdot z^1 + \sum_{i=2}^{n} c_i \cdot \left(\frac{\lambda_i}{\lambda_1} \right)^m \cdot z^i \right) \tag{6.4.14}$$

für den Fall das A reell ist. Und daraus bestimmt sich der zugehörige Eigenvektor aus

$$\frac{1}{\lambda_1^m} \cdot x^{(m)} \to c_1 \cdot z^1 \text{ , wenn } m \to \infty \ . \tag{6.4.15}$$

Dieses Verfahren lässt sich mit Hilfe der Prozeduren unter 6.1 leicht ausrechnen.

$$A = \begin{pmatrix} 7 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -3 \\ 0 & 1 & -3 & 2 \end{pmatrix}, x^{(0)} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ 4 \end{pmatrix}$$

$$x^{(1)} = A \cdot x^{(0)} = \begin{pmatrix} -2 \\ 6 \\ -5 \\ 2 \end{pmatrix}, \ x^{(2)} = A \cdot x^{(1)} = \begin{pmatrix} -43 \\ 66 \\ -62 \\ 25 \end{pmatrix},$$

$$x^{(3)} = A \cdot x^{(2)} = \begin{pmatrix} -627 \\ 841 \\ -754 \\ 302 \end{pmatrix}, \ x^{(4)} = A \cdot x^{(3)} = \begin{pmatrix} -8507 \\ 10872 \\ -9421 \\ 3707 \end{pmatrix},$$

$$x^{(5)} = A \cdot x^{(4)} = \begin{pmatrix} -112458 \\ 140651 \\ -119642 \\ 46549 \end{pmatrix}.$$

Für die Quotienten

$$\frac{x_k^{(m+1)}}{x_k^{(m)}}$$

ergeben sich die Werte in Tabelle 6-8.

Tabelle 6-8: Quotientenwerte

m\k	1	2	3	4
0	-2	2	-1,67	0,5
1	21,5	11	12,4	12,5
2	14,58	12,74	12,16	12,08
3	13,58	12,93	12,49	12,27
4	13,22	12,94	12,7	12,56

Es ist ersichtlich, dass der betragsgrößte Eigenwert

$$\lambda_1 \approx 13$$

ist. Der dazugehörige Eigenvektor bestimmt sich nach (6.4.15) zu

$$x^{(5)} = \begin{pmatrix} -0{,}3 \\ 0{,}38 \\ -0{,}32 \\ 0{,}13 \end{pmatrix} .$$

Die manuelle Nachprüfung mit den Prozeduren von Kapitel 6.1 ergibt:

$$\begin{pmatrix} 7 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -3 \\ 0 & 1 & -3 & 2 \end{pmatrix} - \lambda \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -6 & -4 & 1 & 0 \\ -4 & -7 & -4 & 1 \\ 1 & -4 & -7 & -3 \\ 0 & 1 & -3 & -11 \end{pmatrix}$$

$$\begin{pmatrix} -6 & -4 & 1 & 0 \\ -4 & -7 & -4 & 1 \\ 1 & -4 & -7 & -3 \\ 0 & 1 & -3 & -11 \end{pmatrix} \cdot \begin{pmatrix} -0{,}3 \\ 0{,}38 \\ -0{,}32 \\ 0{,}13 \end{pmatrix} = \begin{pmatrix} -0{,}04 \\ 0{,}05 \\ -0{,}03 \\ -0{,}09 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Nachfolgend wollen wir noch eine Prozedur für die Iterationsmethode von Mises schreiben.

Code 6-10: Verfahren nach von Mises

```
Option Explicit
Dim a(), x() As Double
Dim n As Integer
Sub vMises_Leer()
   ThisWorkbook.Worksheets("vMises").Cells.Clear
   Range("A1").Select
End Sub

Sub vMises_Testdaten()
   Cells(1, 1) = 7
   Cells(1, 2) = -4
   Cells(1, 3) = 1
   Cells(1, 4) = 0
   Cells(2, 1) = -4
   Cells(2, 2) = 6
```

```
   Cells(2, 3) = -4
   Cells(2, 4) = 1
   Cells(3, 1) = 1
   Cells(3, 2) = -4
   Cells(3, 3) = 6
   Cells(3, 4) = -3
   Cells(4, 1) = 0
   Cells(4, 2) = 1
   Cells(4, 3) = -3
   Cells(4, 4) = 2

   Cells(1, 6) = 1
   Cells(2, 6) = 3
   Cells(3, 6) = 3
   Cells(4, 6) = 4
End Sub

Sub vMises_Daten_lesen()
   Dim MyDoc As Object
   Dim nRows, nCols, i, j As Integer

   Set MyDoc = ThisWorkbook.Worksheets("vMises")
   nRows = MyDoc.UsedRange.Rows.Count
   n = nRows
   ReDim a(n, n), x(n)

   For i = 1 To n
      x(i) = Cells(i, n + 2)
   Next i
   For i = 1 To n
      For j = 1 To n
         a(i, j) = Cells(i, j)
      Next j
   Next i
End Sub

Sub vMises_Auswertung()
   Dim y(), z() As Double
   Dim i, j, k, m As Integer
   Dim L, p As Double

   Call vMises_Daten_lesen
   ReDim y(n), z(n)
   m = 10

'Iterationen
   For i = 1 To m
      For j = 1 To n
         y(j) = 0
         For k = 1 To n
            y(j) = y(j) + a(j, k) * x(k)
         Next k
      Next j

      'Quotient
      For j = 1 To n
         If x(j) = 0 Then
```

```
            Cells(i, n + 5 + j) = ""
         Else
            Cells(i, n + 5 + j) = y(j) / x(j)
         End If
      Next j

      For j = 1 To n
         x(j) = y(j)
         Cells(j, n + 2) = y(j)
      Next j
   Next i

'Mittelwert bilden
   L = 0
   For i = 1 To n
      L = L + Cells(m, n + 5 + i)
   Next i
   L = L / n
   Cells(m + 1, n + 5 + n) = L

'Eigenvektor
   For i = 1 To n
      y(i) = 1 / L ^ m * x(i)
      Cells(i, n + 3) = y(i)
   Next i
'Überprüfung
   For i = 1 To n
      z(i) = 0
      For j = 1 To n
         p = a(i, j)
         If i = j Then p = p - L
         z(i) = z(i) + p * y(j)
      Next j
   Next i
   For i = 1 To n
      Cells(i, n + 4) = z(i)
   Next i
End Sub
```

Die Prozeduren werden durch die nachfolgenden Menüpunkte aufgerufen.

Abbildung 6-10: Menü zur Methode von Mises

Die Testdaten zum vorherigen Beispiel sehen wir in den Abbildungen 6-11 und 6-12.

Im Bereich A1:D4 ist die Matrix A wiedergegeben und im Bereich F1:F4 zunächst der Startvektor.

	A	B	C	D	E	F	G	H
1	7	-4	1	0		-4,141E+10	-0,31923025	-0,00219309
2	-4	6	-4	1		5,077E+10	0,39138755	0,00110565
3	1	-4	6	-3		-4,2397E+10	-0,32683999	0,00033301
4	0	1	-3	2		1,63E+10	0,12566058	-0,00044534

Abbildung 6-11: Teil 1 der Auswertung zum Testbeispiel

In den gleichen Bereich werden nacheinander die ermittelten Vektoren gespeichert. Die zugehörigen Quotienten finden Sie für die 10 Iterationsschritte im Bereich J1:M10. In M11 steht der Mittelwert der letzten λ-Werte, der für die Bestimmung des Eigenvektors genommen wird. Dieser Eigenvektor steht im Bereich G1:G4.

J	K	L	M
-2	2	-1,66666667	0,5
21,5	11	12,4	12,5
14,5813953	12,7424242	12,1612903	12,08
13,5677831	12,9274673	12,494695	12,2748344
13,2194663	12,9369941	12,6995011	12,5570542
13,0666738	12,9316891	12,8095568	12,7322821
12,9940583	12,9278348	12,8662174	12,8264023
12,9585643	12,9257681	12,8948891	12,8747519
12,9410233	12,9247202	12,9092872	12,8991796
12,9323133	12,9241957	12,9164915	12,9114354
			12,921109

Abbildung 6-12: Teil 2 der Auswertung zum Testbeispiel

Die Überprüfung der Daten nach der Formel (6.4.2) zeigt im Bereich H1:H4 den angenäherten Nullvektor.

Es bleibt zum Schluss noch nach zu halten, dass die Eigenkreisfrequenz des Trägers durch Umstellung von (6.4.7) lautet

$$\omega = \sqrt{\lambda \cdot \frac{E \cdot I}{m \cdot h^4}} = 3{,}595 \sqrt{\frac{E \cdot I}{m \cdot h^4}} \quad . \tag{6.4.16}$$

Übung 6-5: Grenzwertbetrachtung

Nicht immer konvergiert die Methode von Mises gegen einen Grenzwert.

Beispiel:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 1 \end{pmatrix}, x^{(0)} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}.$$

Untersuchen Sie den Fall und finden Sie die Lösung.

Programmieren Sie ein weiteres Verfahren zur Bestimmung von Eigenwerten.

7 Pseudozufallszahlen

Mit Computern ist es zwar nicht möglich, echte Zufallszahlen zu erzeugen, aber man kann Zahlenfolgen erhalten, die im halboffenen Intervall [0,1) hinreichend gleichmäßig verteilt sind. Diese bezeichnet man als Pseudozufallszahlen. Hinreichend gleichmäßig verteilt bedeutet, dass bei einer ausreichenden Anzahl von Pseudozufallszahlen und bei jeder Unterteilung des Intervalls, in allen Teilintervallen die gleiche Anzahl Treffer liegt.

7.1 Integration nach der Monte-Carlo-Methode

Die Gleichverteilung der Pseudozufallszahlen nutzt man in vielen Bereichen aus, um Verhältnisse auszudrücken. So ist auch die Integration als Verhältnis von bekannter zu unbekannter Fläche mittels Pseudozufallszahlen möglich.

Dieses Verfahren wird als Monte-Carlo-Methode bezeichnet.

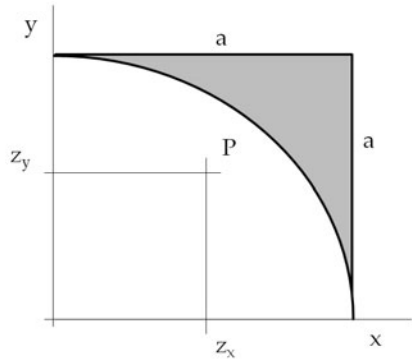


Abbildung 7-1: Viertelkreis im Quadrat

Die Integration einer Fläche geht von der Überlegung aus, wenn eine Gleichverteilung auf einem Intervall vorliegt, dann kann man mit zwei Intervallen eine Ebene aufspannen, auf der alle Punkte dieser Intervalle in der Ebene ebenfalls gleich verteilt sind.

Betrachten wir dazu in Abbildung 7-1 die Darstellung eines Viertelkreises und ein Quadrats. Nehmen wir weiterhin an, der Flächeninhalt des Viertelkreises sei unbekannt und der des Quadrats gegeben mit $A = a^2$. Die Überlegung ist nun sehr einfach. Erzeugt man jetzt hinreichend viele Zufallspunkte im Quadrat, so müsste das Verhältnis der Punkte im Quadrat zu den Punkten im Viertelkreis im gleichen Verhältnis stehen wie die Flächen

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_7,

$$\frac{n_K}{n_Q} = \frac{A_K}{A_Q} \ . \tag{7.1.1}$$

Ein Treffer im Viertelkreis liegt dann vor, wenn gilt

$$z_x^2 + z_y^2 < a^2 \ . \tag{7.1.2}$$

Tabelle 7-1: Algorithmus zur Monte-Carlo-Methode

<table>
<tr><td colspan="3">Eingabe der Anzahl Punkte n
Der Einfachheit wegen wird a=1 gesetzt und damit ist auch A_Q=1.</td></tr>
<tr><td colspan="3">Randomize</td></tr>
<tr><td colspan="3">x=0</td></tr>
<tr><td colspan="3">m=0</td></tr>
<tr><td></td><td colspan="2">i=1 (1) n</td></tr>
<tr><td></td><td colspan="2">x=Rnd(x)</td></tr>
<tr><td></td><td colspan="2">y=Rnd(x)</td></tr>
<tr><td></td><td colspan="2">Ist x*x+y*y < 1</td></tr>
<tr><td></td><td>ja</td><td>nein</td></tr>
<tr><td></td><td>Treffer
m=m+1</td><td>keine Anweisung</td></tr>
<tr><td colspan="3">$A_K = A_Q \frac{m}{n}$</td></tr>
<tr><td colspan="3">Ausgabe
A_K (der exakte Wert lautet $\frac{\pi}{4} = 0{,}78539816$)</td></tr>
</table>

Wir schreiben nun eine Prozedur, die in zwanzig Programmschleifen den Inhalt eines Viertelkreises berechnet.

Code 7-1: Flächenberechnung eines Viertelkreises nach der Monte-Carlo-Methode

```
Option Explicit

Sub MonteCarlo_Test()
   Dim x, y, z As Double
   Dim n, m As Long
   Dim i, j As Long

   ThisWorkbook.Worksheets("MonteCarlo").Cells.Clear
   n = 1000000
   Randomize
   Cells(1, 2) = Atn(1)
   z = Timer

   For i = 1 To 20
      m = 0
      For j = 1 To n
         x = Rnd(z)
```

```
            y = Rnd(z)
            If x * x + y * y < 1 Then
               m = m + 1
            End If
         Next j
         Cells(i, 1) = Str(m / n)
      Next i
End Sub
```

Obwohl es sich nur um eine einfache Berechnung handelt, wollen wir diese auch über ein Menü aufrufbar machen.

Abbildung 7-2: Menü zur Monte-Carlo-Methode

Ein Testlauf mit n=1.000.000 lieferte bei 20 Durchläufen eine Genauigkeit nur bis zur zweiten Nachkommastelle. Rechts oben (Spalte B) zum Vergleich der exakte Wert aus ATN(1).

	A	B
1	0,785752	0,78539816
2	0,784976	
3	0,785409	
4	0,78513	
5	0,7847	
6	0,785687	
7	0,785885	
8	0,786046	
9	0,785073	
10	0,785124	
11	0,785627	
12	0,785349	
13	0,784491	
14	0,785694	
15	0,785642	
16	0,785825	
17	0,785279	
18	0,785514	
19	0,78552	
20	0,785119	

Abbildung 7-3: Testergebnisse zur Monte-Carlo-Methode

Auch das liegt in der Eigenschaft der Pseudozufallszahlen. Es kommt nicht immer das gleiche Ergebnis heraus, wenn die Anzahl der Versuch nicht hinreichend ist. So gibt es immer wieder „Ausreißer“.

Bevor wir nun zu einem Beispiel kommen, müssen wir uns noch mit dem Zufallszahlen-Generator befassen.

Die Rnd-Funktion liefert einen Wert aus dem halboffenen Intervall [0,1).

```
Rnd(Zahl)
```

Der Wert von Zahl bestimmt, wie die Zufallszahl generiert wird. Für jeden gegebenen Standardwert wird dieselbe Zufallszahlenfolge generiert, daher ist auch eine Pseudozufallszahlenfolge alles andere als zufällig. Denn bei jedem nachfolgenden Aufruf der Rnd-Funktion, dient die vorherige Zahl als Startwert für die nächste Zahl in der Folge.

Damit nicht immer die gleichen Pseudozufallszahlen erzeugt werden, gibt es die Randomize-Funktion. Sie muss vor dem ersten Aufruf der Rnd-Funktion aufgerufen werden.

```
Randomize
```

Sie wird ohne Argument ausgeführt und initialisiert den Zufallszahlengenerator auf der Basis der Systemzeit.

Oft wird ein Intervall [a, b] mit Zufallszahlen benötigt. Die Erzeugung von Pseudozufallszahlen bei beliebiger [a, b]-Gleichverteilung erhält man, in dem man die Verteilungsfunktion

$$y = F(x) = (x - a)/(b - a) \tag{7.1.3}$$

nach x auflöst.

Die nachfolgende Anweisung erzeugt Pseudozufallszahlen im Intervall [a, b]:

```
(b - a + ε) * Rnd(x) + a
```

Darin ist x die letzte Pseudozufallszahl und ε die kleinste darstellbare Zahl, z.B. 1E-308 für eine Double-Fließkomma-Zahl. Sie ist notwendig, damit aus einem halboffenen Intervall ein geschlossenes wird.

Beispiel 7-1: Bestimmung der Fläche eines Blechteils

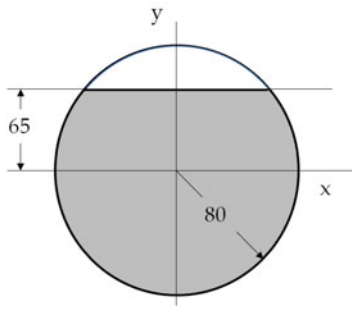


Abbildung 7-4: Fläche eines Blechteils

Für das in Abbildung 7-4 dargestellte Blechteil ist der Flächeninhalt gesucht. Nun gibt es dafür zwar Formeln, aber wir wollen den Flächeninhalt mit der Monte-Carlo-Methode bestimmen.

Tabelle 7-2: Flächenberechnung des Blechteils

Eingabe der Anzahl Punkte n	
Randomize	
$z=z_0$	
m=0	
A=160*160	
	i=1 (1) n
	x=(160+1E-308)*Rnd(z)-80
	y=(160+1E-308)*Rnd(z)-80

	Ist Sqr(x*x+y*y) <= 80		
	Ja		nein
	Ist y<=65		./.
	Ja	nein	
	m=m+1 (Treffer)	./.	
$A_x = A\frac{m}{n}$			
Ausgabe A_x			

Als Vorlage können wir hier den Testalgorithmus nehmen.

Code 7-2: Flächenberechnung des Blechteils nach der Monte-Carlo-Methode

```
Option Explicit

Sub MonteCarlo_Blechteil()
   Dim x, y, z, e, A As Double
   Dim n, m As Long
   Dim i, j As Long

   ThisWorkbook.Worksheets("Blechteil").Cells.Clear
   n = InputBox("Anzahl Durchläufe:")
   Randomize
   A = 160 * 160
   z = Timer
   e = 1E-308
   Cells(1, 1) = A * Atn(1)
   For i = 1 To 20
      m = 0
      For j = 1 To n
         x = (160 + e) * Rnd(z) - 80
         y = (160 + e) * Rnd(z) - 80
         If Sqr(x * x + y * y) <= 80 Then
            If y <= 65 Then
               m = m + 1
            End If
         End If
      Next j
      Cells(i, 2) = Str(A * m / n)
   Next i
End Sub
```

Auch diese Berechnung bekommt einen Menüaufruf.

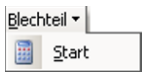


Abbildung 7-5: Menü zur Flächenberechnung

Die Auswertung (Abbildung 7-6) zeigt in der Spalte A den berechneten Flächeninhalt der gesamten Kreisfläche und in Spalte B die Bestimmungen des Blechteils nach der Monte-Carlo-Methode.

Übung 7-1: Unterschiedliche Laufwerte

Schreiben Sie die Prozeduren so um, dass auch Testergebnisse mit unterschiedlichen Laufwerten n ausgegeben werden. Fügen Sie beim Blechteil die exakte Berechnung hinzu.

	A	B
1	20106,193	19147,6992
2		19162,2656
3		19154,0992
4		19155,6608
5		19146,24
6		19149,0048
7		19160,7808
8		19150,5152
9		19164,0832
10		19150,592
11		19162,112
12		19159,0144
13		19143,7568
14		19146,624
15		19151,1296
16		19154,0736
17		19164,8256
18		19154,1504
19		19164,928
20		19147,9552

Abbildung 7-6: Blechteilfläche nach der Monte-Carlo-Methode

7.2 Probabilistische Simulation

Was wir zuvor über die Eigenschaft von Pseudozufallszahlen gesagt haben, dass sie bei hinreichender Anzahl eine Gleichverteilung auf einem Intervall garantieren, lässt sich auch in anderen Bereichen anwenden. Ein Bereich ist die Simulation von Wahrscheinlichkeitsmodellen. Man spricht hier von Probabilistischer Simulation.

Eine der ersten Anwendungen war die Simulation der Diffusion von Neutronen durch die Bleiwände von Kernreaktoren. Dann kamen Warteschlangen- und Optimierungsprobleme hinzu. Heute gibt es Simulationen in allen Bereichen.

Beispiel 7-2: Warteschlangenproblem Maschinenwartung

Grundlage für eine solche Simulation sind Wahrscheinlichkeitsbetrachtungen. Nehmen wir an, eine Person bedient einige Maschinen. Dann kann sie sich beim Ausfall mehrerer Maschinen nur um eine kümmern. Erst wenn diese wieder funktioniert, kann sie sich um eine andere ausgefallene Maschine kümmern. Bedient diese Person nun zu viele Maschinen, so kann es zu einem erheblichen Produktionsausfall kommen. Umgekehrt ist die Person bei zu wenigen Maschinen nicht vollbeschäftigt.

Der Ausfall von Maschinen ist zufällig. Mit Langzeitstudien kann man jedoch eine gewisse Wahrscheinlichkeitsverteilung angeben. Nehmen wir an, es sei eine Anzahl n Maschinen gegeben, die von einer Person bedient werden. Wir betrachten einen längeren Zeitraum in Zeitschritten Δt. Wir nehmen weiterhin an, dass mit einer Wahrscheinlichkeit von $x \cdot \Delta t$ eine Maschine in Δt ausfällt. Ebenso sei die

Wahrscheinlichkeit y · Δt gegeben, dass innerhalb von Δt die Reparatur wieder beendet ist. Die letzte Wahrscheinlichkeit w steht dafür, dass zu einem bestimmten Zeitpunkt t sich m Maschinen in der Warteschlange befinden. Setzt man

$$\lim_{t\to\infty} w(t) = w_m \ , \tag{7.2.1}$$

so ergibt sich ein System von Differenzengleichungen

$$\begin{aligned} y \cdot w_1 &= m \cdot x \cdot w_0 \\ y \cdot w_{m+1} &= \left[x(n-m)+y\right]w_m - x(n-m+1)w_{m-1} \end{aligned} \tag{7.2.2}$$

für m=2,3,4,….

Erkennbar ist die Rekursionsformel

$$y \cdot w_{m+1} = x(n-m)w_m \ . \tag{7.2.3}$$

Da diese mit Fakultät wächst, kann es zu einem erheblichen Rechenaufwand kommen, den man bei der Probabilistischen Simulation umgeht.

Tabelle 7-3: Lösungsalgorithmus Maschinenwartung

<table>
<tr><td colspan="4">Eingabe der erforderlichen Parameter:
Δt als Zeitintervall in Sekunden
t_{max} als Betrachtungszeitraum in Minuten
x∈[0,1) Wahrscheinlichkeit für den Ausfall einer Maschine in Δt
y∈[0,1) Wahrscheinlichkeit für die Reparatur einer Maschine in Δt</td></tr>
<tr><td colspan="4">Randomize</td></tr>
<tr><td colspan="4">Startbedingung w=0, z=0</td></tr>
<tr><td colspan="4">t=1 (1) t_{max}</td></tr>
<tr><td colspan="4">Ausfallwahrscheinlichkeit
z_1=Rnd(z)</td></tr>
<tr><td colspan="4">Ist eine Maschine ausgefallen?
Ist z_1<=x</td></tr>
<tr><td colspan="3">Ja</td><td>Nein</td></tr>
<tr><td colspan="3">Maschine gelangt in die Warteschlange
w=w+1</td><td>./.</td></tr>
<tr><td colspan="4">Sind Maschinen in der Warteschlange?
w>0</td></tr>
<tr><td colspan="3">Ja</td><td>Nein</td></tr>
<tr><td colspan="3">Reparaturwahrscheinlichkeit
z_2=Rnd(z)</td><td rowspan="3">./.</td></tr>
<tr><td colspan="3">Ist eine Maschine repariert?
z_2<=y</td></tr>
<tr><td>Ja</td><td colspan="2">Nein</td></tr>
<tr><td>w=w-1</td><td colspan="2">./.</td></tr>
<tr><td colspan="4">Ausgabe t, w</td></tr>
</table>

Der Algorithmus ist offensichtlich einfach zu programmieren.

Code 7-3: Einfaches Modell einer Maschinenwartung

```
Option Explicit

Sub Maschinenwartung_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Maschinenwartung").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Maschinenwartung").Cells.Clear
   Range("A1:B1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Maschinenwartung"

   Range("A:B").ColumnWidth = "10"
   Range("C:C").ColumnWidth = "2"
   Range("A2:A16").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A2") = ChrW(916) & "t [s]"
   Range("A3") = "tmax [s]"
   Range("A4") = "x"
   Range("A5") = "y"

   Range("D1:E1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Auswertung"
   Range("D2:E2").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("D2") = "t [s]"
   Range("E2") = "w"
   Selection.NumberFormat = "0"
   Range("B2").Select
End Sub

Sub Maschinenwartung_Testdaten()
   Cells(2, 2) = 1
   Cells(3, 2) = 360
   Cells(4, 2) = 0.3
   Cells(5, 2) = 0.4
End Sub

Sub Maschinenwartung_Auswertung()
   Dim w, t, dt, tm, i As Long
   Dim x, y, z As Double

   dt = Cells(2, 2)
   tm = Cells(3, 2)
   x = Cells(4, 2)
   y = Cells(5, 2)
```

```
   Randomize
   z = Timer
   i = 2
   w = 0
   For t = 0 To tm Step dt
      z = Rnd(z)
      If z <= x Then
         w = w + 1
      End If
      If w > 0 Then
         z = Rnd(z)
         If z <= y Then
            w = w - 1
         End If
      End If
      i = i + 1
      Cells(i, 4) = t
      Cells(i, 5) = w
   Next t
End Sub

Sub Warteschlange_zeigen()
    Range("D3:E363").Select
    Charts.Add
    ActiveChart.ChartType = xlXYScatterLinesNoMarkers
    ActiveChart.SetSourceData Source:= _
       Sheets("Maschinenwartung").Range("D3:E363"), _
       PlotBy:=xlColumns
    ActiveChart.SeriesCollection(1).Name = _
       "=""Warteschlange"""
    ActiveChart.Location Where:= _
       xlLocationAsObject, Name:="Maschinenwartung"
    With ActiveChart
       .HasTitle = True
       .ChartTitle.Characters.Text = "Warteschlange"
       .Axes(xlCategory, xlPrimary).HasTitle = True
       .Axes(xlCategory, _
           xlPrimary).AxisTitle.Characters.Text = _
           "t [Sek.]"
       .Axes(xlValue, xlPrimary).HasTitle = True
       .Axes(xlValue, _
           xlPrimary).AxisTitle.Characters.Text = _
           "Anzahl Maschinen"
    End With
    ActiveWindow.Visible = False
End Sub

Sub Grafik_löschen()
   Dim Shp As Shape
   For Each Shp In Worksheets("Maschinenwartung").Shapes
      Shp.Delete
   Next
End Sub
```

Die Symbolleiste erhält das in der nachfolgenden Abbildung dargestellte Menü.

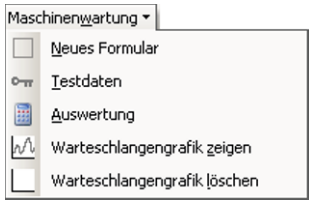


Abbildung 7-7: Menü zur Maschinenwartung

Bei der Auswertung der Testdaten ergeben sich ähnliche Daten wie in Abbildung 7-8 dargestellt.

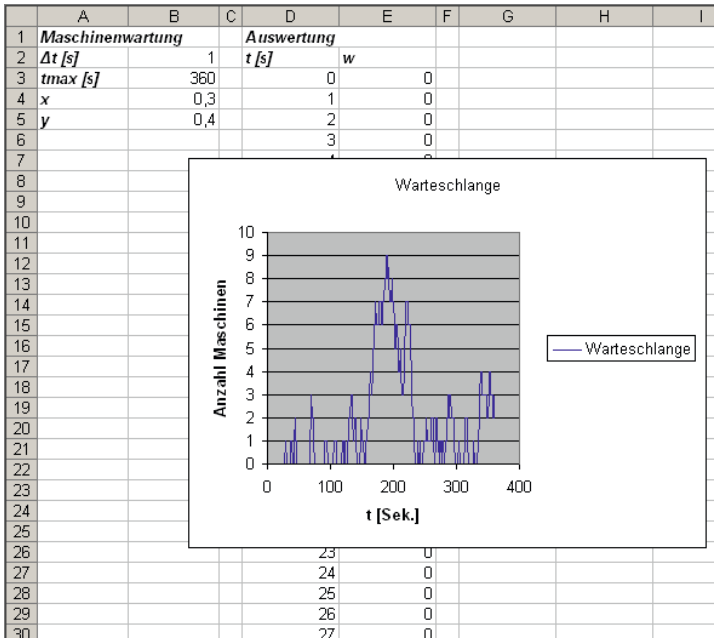


Abbildung 7-8: Maschinenwartung mit Testdaten

Übung 7-2: Mehrere Personen

Interessant ist der Wirkzusammenhang zwischen den Wahrscheinlichkeiten x und y. Er lässt sich an diesem einfachen Beispiel anschaulich studieren.

Natürlich können n Maschinen auch von p Personen bedient werden, die wiederum nur zu bestimmten Zeiten (Pausen, Arbeitszeit) zu Verfügung stehen. Sie sehen, mit steigender Komplexität wächst die Realitätsnähe.

Beispiel 7-3: Ermittlung der Lebensdauer von Pumpenventilen

Ein Unternehmen, z. B. der chemischen Industrie, setzt in der Produktion Pumpen ein. Die Pumpen besitzen drei Einlass- und drei Auslass-Ventile. Wird nur ein Ventil unbrauchbar, muss die Pumpe zur Instandsetzung außer Betrieb gesetzt werden. Jede Ventilgruppe (Einlass oder Auslass) bildet eine Montageeinheit.

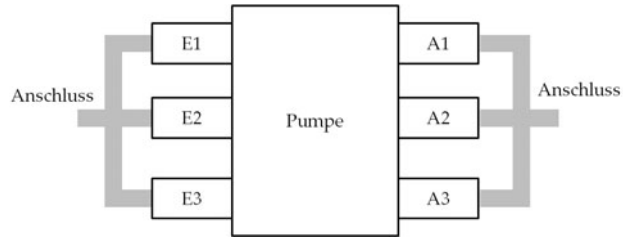


Abbildung 7-9: Schema der Pumpeneinheit

Für die Instandhaltung, die nach der besten Lösung sucht, ergeben sich unterschiedliche Verhaltensweisen.

1. Methode: Nur Ventile die defekt sind werden ausgetauscht.

2. Methode: Ist ein Ventil einer Gruppe defekt, wird die gesamte Gruppe überholt. Dabei wird das defekte Ventil ersetzt und alle anderen, deren durchschnittliche Nutzungszeit überschritten ist.

3. Methode: Ist ein Ventil einer Gruppe defekt, werden alle Ventile der Gruppe ausgetauscht.

Die anfallenden Kosten sind bekannt und lassen sich in Montagestunden ausdrücken. So ergeben sich als Kostengruppen:

- Pumpe stilllegen und Instandhaltung vorbereiten
- Anschlüsse demontieren
- Ventilgruppe demontieren
- Ventil demontieren
- Ventil reparieren
- Ventil montieren
- Ventilgruppe montieren
- Anschlüsse montieren
- Pumpe in Betrieb nehmen

Die durchschnittliche Lebensdauer eines Ventils ist aus jahrelangen Untersuchungen ebenfalls bekannt, so dass sich die Ausfallwahrscheinlichkeit vereinfacht durch eine Gerade darstellen lässt.

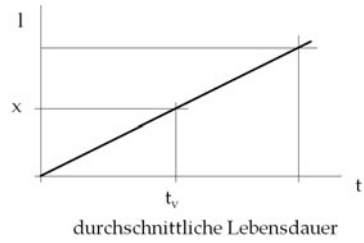


Abbildung 7-10: Wahrscheinlichkeitsfunktion der durchschnittlichen Lebensdauer

Entwerfen wir zunächst den Algorithmus für die erste Methode.

Tabelle 7-4: Lösungsalgorithmus Ersatzproblem

Eingabe der erforderlichen Parameter: Schrittweite Δt in 100h t_{max} als Betrachtungszeitraum in 100h Maximale Nutzzeit in 100h Zeiten für die einzelnen Montagen: Z1 = Pumpe ausschalten und vorbereiten Z2 = Anschluss demontieren Z3 = Ventilgruppe demontieren Z4 = Ventil demontieren Z5 = Ventil reparieren Z6 = Ventil montieren Z7 = Ventilgruppe montieren Z8 = Anschlüsse montieren Z9 = Pumpe einschalten		
Randomize		
z=Timer, Startwert Zufallszahl		
t = Δt (Δt) t_{max}		
Ausfall eines Einlassventils e=0		
j = 1 (1) 3		
z=Rnd(z)		
x=ev(j)/tn		
Ist z <= x		
Ja	Nein	
e = e + 1	./.	
∑ev=∑ev+ev(j)		
∑ae=∑ae+1		
ev(j)=0		
Ausfall eines Auslassventils a = 0		
j = 1 (1) 3		
z=Rnd(z)		
x=ev(j)/tn		
Ist z <= x		
Ja	Nein	
a = a + 1	./.	
∑av=∑av+av(j)		
∑aa=∑aa+1		
av(j)=0		
Ist e > 0 oder a > 0		

	Ja	Nein
	Pumpe aus- und einschalten zm = zm + z1 + z9	./.
	Za = za z1+z2+z3+z4+z5+z6+z7+z8+z9	
	Ist e > 0	
	Ja	Nein
	Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8	./.
	Ventil demontieren, reparieren und montieren zm = zm + e(z4 + z5 +z6)	
	Ist a > 0	
	Ja	Nein
	Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8	./.
	Ventil demontieren, reparieren und montieren zm = zm + a(z4 + z5 +z6)	
Ausgabe zm, za, Durchschnitte		

Für das Programm erstellen wir wieder ein Formblatt, und die Berechnung mit Testdaten.

Code 7-4: Ersatzproblem nach der 1. Methode

```
Option Explicit

Sub Ersatzproblem_Leer()
   Dim Shp As Shape
   For Each Shp In Worksheets("Ersatzproblem").Shapes
      Shp.Delete
   Next
   ThisWorkbook.Worksheets("Ersatzproblem").Cells.Clear

   Range("A1:B1").Select
   Selection.MergeCells = True
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Selection.Value = "Instandhaltungsdaten"

   Range("A:A").ColumnWidth = "30"
   Range("B:B").ColumnWidth = "10"
   Range("C:C").ColumnWidth = "2"
   Range("D:I").ColumnWidth = "10"
   Range("A2:A20").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
```

```
    Range("A2") = ChrW(916) & "t [100h]"
    Range("A3") = "tmax [100h]"
    Range("A4") = "tNutz [100h]"
    Range("A5") = "Pumpe ausschalten [h]"
    Range("A6") = "Anschlüsse demontieren [h]"
    Range("A7") = "Gruppe demontieren [h]"
    Range("A8") = "Ventil demontieren [h]"
    Range("A9") = "Ventil reparieren [h]"
    Range("A10") = "Ventil montieren [h]"
    Range("A11") = "Gruppe montieren [h]"
    Range("A12") = "Anschlüsse montieren [h]"
    Range("A13") = "Pumpe einschalten [h]"
    Range("A15") = "Auswertung"
    Range("A16") = "Montagezeit [h]"
    Range("A17") = "Ausfallzeit [h]"
    Range("A18") = "Standzeit Einlass [100h]"
    Range("A19") = "Standzeit Auslass [100h]"
    Range("B2:B20").Select
    Selection.NumberFormat = "0.0"

    Range("D1:I1").Select
    Selection.MergeCells = True
    Selection.Font.Bold = True
    Selection.Font.Italic = True
    Selection.Value = "Auswertung"
    Range("D2:j2").Select
    Selection.Font.Bold = True
    Selection.Font.Italic = True
    Selection.NumberFormat = "0"
    Range("D2") = "E1  [100h]"
    Range("E2") = "E2  [100h]"
    Range("F2") = "E3  [100h]"
    Range("G2") = "A1  [100h]"
    Range("H2") = "A2  [100h]"
    Range("I2") = "A3  [100h]"
    Range("B2").Select
End Sub

Sub Ersatzproblem_Testdaten()
    Cells(2, 2) = 1
    Cells(3, 2) = 1000
    Cells(4, 2) = 10
    Cells(5, 2) = 0.5
    Cells(6, 2) = 0.7
    Cells(7, 2) = 0.4
    Cells(8, 2) = 0.8
    Cells(9, 2) = 0.3
    Cells(10, 2) = 0.6
    Cells(11, 2) = 0.4
    Cells(12, 2) = 0.6
    Cells(13, 2) = 0.2
End Sub

Sub Ersatzproblem_Auswertung_1()
    Dim dt, tm, x, tn As Double
    Dim z1, z2, z3, z4, z5, z6, z7, z8, z9 As Double
```

```
   Dim t, z, zm, za As Double
   Dim i, j, e, a As Long
   Dim EV(3), Ez(3), AV(3), Az(3) As Long
   Dim Se, Ae, Sa, Aa

   dt = Cells(2, 2)
   tm = Cells(3, 2)
   tn = Cells(4, 2)
   z1 = Cells(5, 2)
   z2 = Cells(6, 2)
   z3 = Cells(7, 2)
   z4 = Cells(8, 2)
   z5 = Cells(9, 2)
   z6 = Cells(10, 2)
   z7 = Cells(11, 2)
   z8 = Cells(12, 2)
   z9 = Cells(13, 2)

   For j = 1 To 3
      Ez(j) = 0
      Az(j) = 0
   Next j
   Se = 0: Ae = 0
   Sa = 0: Aa = 0
   zm = 0
   i = 2
   z = Timer
   Randomize
   For t = dt To tm Step dt
'
'Einlassventile
      e = 0
      For j = 1 To 3
         z = Rnd(z)
         x = EV(j) / tn
         If z <= x Then
            e = e + 1
            Ez(j) = Ez(j) + 1
            Cells(Ez(j) + 2, 3 + j) = EV(j)
            Se = Se + EV(j)
            Ae = Ae + 1
            EV(j) = 0
         End If
      Next j
'
'Auslassventile
      a = 0
      For j = 1 To 3
         z = Rnd(z)
         x = AV(j) / tn
         If z <= x Then
            a = a + 1
            Az(j) = Az(j) + 1
            Cells(Az(j) + 2, 6 + j) = AV(j)
            Sa = Sa + AV(j)
            Aa = Aa + 1
            AV(j) = 0
```

```
         End If
      Next j
'
'Montage- und Ausfallzeiten
      If e > 0 Or a > 0 Then
         zm = zm + z1 + z9
         za = za + z1 + z2 + z3 + z4 + z5 + z6 + z7 + _
              z8 + z9
      End If

      If e > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         zm = zm + e * (z4 + z5 + z6)
      End If
      If a > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         zm = zm + a * (z4 + z5 + z6)
      End If
'
'Zeitschritt
      For j = 1 To 3
         EV(j) = EV(j) + dt
         AV(j) = AV(j) + dt
      Next j
   Next t
   Cells(16, 2) = zm
   Cells(17, 2) = za
   Cells(18, 2) = Se / Ae
   Cells(19, 2) = Sa / Aa
End Sub
```

Die 2. Methode ersetzt auch nicht defekte Ventile, wenn die Hälfte der Lebensdauer erreicht ist.

Tabelle 7-5: Lösungsalgorithmus Ersatzproblem 2. Methode

Randomize			
z=Timer, Startwert Zufallszahl			
	$t = \Delta t\ (\Delta t)\ t_{max}$		
	Ausfall eines Einlassventils e=0		
		j = 1 (1) 3	
		z=Rnd(z)	
		x=ev(j)/tn	
		Ist z <= x	
		Ja	Nein
		e = e + 1	
		∑ev=∑ev+ev(j)	
		∑ae=∑ae+1	
		ev(j)=0	
	Ausfall eines Auslassventils		

a = 0	
j = 1 (1) 3	
z=Rnd(z)	
x=ev(j)/tn	
Ist z <= x	
Ja	Nein
a = a + 1	
∑av=∑av+av(j)	
∑aa=∑aa+1	
av(j)=0	
Ist e > 0 oder a > 0	
Ja	Nein
Pumpe aus- und einschalten zm = zm + z1 + z9	
za = za+z1+z2+z3+z4+z5+z6+z7+z8+z9	
Ist e > 0	
Ja	Nein
Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8	
Ventil demontieren, reparieren und montieren zm = zm + e(z4 + z5 +z6)	
k = 1 (1) 3	
Ist ev(k) >= tn/2	
Ja	Nein
zm = zm + z4+z5+z6	
ev(k)=0	
Ist a > 0	
Ja	Nein
Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8	
Ventil demontieren, reparieren und montieren zm = zm + a(z4 + z5 +z6)	
k = 1 (1) 3	
Ist av(k) >= tn/2	
Ja	Nein
zm = zm + z4+z5+z6	
av(k)=0	

Für die zweite Methode wird lediglich eine weitere Prozedur benötigt.

Code 7-5: Ersatzproblem 2. Methode

```
Sub Ersatzproblem_Auswertung_2()
   Dim dt, tm, x, tn As Double
   Dim z1, z2, z3, z4, z5, z6, z7, z8, z9 As Double
   Dim t, z, zm, za As Double
   Dim i, j, k, e, a As Long
   Dim EV(3), Ez(3), AV(3), Az(3) As Long
   Dim Se, Ae, Sa, Aa

   dt = Cells(2, 2)
   tm = Cells(3, 2)
   tn = Cells(4, 2)
   z1 = Cells(5, 2)
   z2 = Cells(6, 2)
   z3 = Cells(7, 2)
   z4 = Cells(8, 2)
   z5 = Cells(9, 2)
   z6 = Cells(10, 2)
   z7 = Cells(11, 2)
   z8 = Cells(12, 2)
   z9 = Cells(13, 2)

   For j = 1 To 3
      Ez(j) = 0
      Az(j) = 0
   Next j
   Se = 0: Ae = 0
   Sa = 0: Aa = 0
   zm = 0: za = 0
   i = 2
   z = Timer
   Randomize
   For t = dt To tm Step dt
'
'Einlassventile
      e = 0
      For j = 1 To 3
         z = Rnd(z)
         x = EV(j) / tn
         If z <= x Then
            e = e + 1
            Ez(j) = Ez(j) + 1
            Cells(Ez(j) + 2, 3 + j) = EV(j)
            Se = Se + EV(j)
            Ae = Ae + 1
            EV(j) = 0
         End If
      Next j
'
'Auslassventile
      a = 0
      For j = 1 To 3
         z = Rnd(z)
         x = AV(j) / tn
```

```
         If z <= x Then
            a = a + 1
            Az(j) = Az(j) + 1
            Cells(Az(j) + 2, 6 + j) = AV(j)
            Sa = Sa + AV(j)
            Aa = Aa + 1
            AV(j) = 0
         End If
      Next j
'
'Montage- und Ausfallzeiten
      If e > 0 Or a > 0 Then
         zm = zm + z1 + z9
         za = za + z1 + z2 + z3 + z4 + z5 + z6 + z7 + _
              z8 + z9
      End If
      If e > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         zm = zm + e * (z4 + z5 + z6)
         For k = 1 To 3
            If EV(k) >= tn / 2 Then
               zm = zm + z4 + z5 + z6
               Ez(k) = Ez(k) + 1
               Cells(Ez(k) + 2, 3 + k) = EV(k)
               Se = Se + EV(k)
               Ae = Ae + 1
               EV(k) = 0
            End If
         Next k
      End If
      If a > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         zm = zm + a * (z4 + z5 + z6)
         For k = 1 To 3
            If AV(k) >= tn / 2 Then
               zm = zm + z4 + z5 + z6
               Az(k) = Az(k) + 1
               Cells(Az(k) + 2, 6 + k) = AV(k)
               Sa = Sa + AV(k)
               Aa = Aa + 1
               AV(k) = 0
            End If
         Next k
      End If
'
'Zeitschritt
      For j = 1 To 3
         EV(j) = EV(j) + dt
         AV(j) = AV(j) + dt
      Next j
   Next t
   Cells(16, 2) = zm
   Cells(17, 2) = za
   Cells(18, 2) = Se / Ae
   Cells(19, 2) = Sa / Aa
End Sub
```

Die 3. Methode ersetzt auch nicht defekte Ventile einer Gruppe.

Tabelle 7-6: Lösungsalgorithmus Ersatzproblem 3. Methode

Randomize			
z=Timer, Startwert Zufallszahl			
t = Δt (Δt) t_{max}			
	Ausfall eines Einlassventils e=0		
		j = 1 (1) 3	
		z=Rnd(z)	
		x=ev(j)/tn	
		Ist z <= x	
		Ja	Nein
		e = e + 1	./.
		∑ev=∑ev+ev(j)	
		∑ae=∑ae+1	
		ev(j)=0	
	Ausfall eines Auslassventils a = 0		
		j = 1 (1) 3	
		z=Rnd(z)	
		x=ev(j)/tn	
		Ist z <= x	
		Ja	Nein
		a = a + 1	./.
		∑av=∑av+av(j)	
		∑aa=∑aa+1	
		av(j)=0	
	Ist e > 0 oder a > 0		
	Ja		Nein
	Pumpe aus- und einschalten zm = zm + z1 + z9		./.
	za = za+z1+z2+z3+z4+z5+z6+z7+z8+z9		
	Ist e > 0		
	Ja		Nein
	Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8		./.
		k = 1 (1) 3	
		zm = zm + z4+z5+z6	
		ev(k)=0	

	Ist a > 0		
	Ja		Nein
	Anschlüsse und Gruppe demontieren und montieren zm = zm + z2 + z3 + z7 + z8		./.
	k = 1 (1) 3		
	zm = zm + z4+z5+z6		
	av(k)=0		

Und auch die dritte Methode wird als eigenständige Prozedur integriert.

Code 7-6: Ersatzproblem 3. Methode

```
Sub Ersatzproblem_Auswertung_3()
   Dim dt, tm, x, tn As Double
   Dim z1, z2, z3, z4, z5, z6, z7, z8, z9 As Double
   Dim t, z, zm, za As Double
   Dim i, j, k, e, a As Long
   Dim EV(3), Ez(3), AV(3), Az(3) As Long
   Dim Se, Ae, Sa, Aa

   dt = Cells(2, 2)
   tm = Cells(3, 2)
   tn = Cells(4, 2)
   z1 = Cells(5, 2)
   z2 = Cells(6, 2)
   z3 = Cells(7, 2)
   z4 = Cells(8, 2)
   z5 = Cells(9, 2)
   z6 = Cells(10, 2)
   z7 = Cells(11, 2)
   z8 = Cells(12, 2)
   z9 = Cells(13, 2)

   For j = 1 To 3
      Ez(j) = 0
      Az(j) = 0
   Next j
   Se = 0: Ae = 0
   Sa = 0: Aa = 0
   zm = 0: za = 0
   i = 2
   z = Timer
   Randomize
   For t = dt To tm Step dt
'
'Einlassventile
      e = 0
      For j = 1 To 3
         z = Rnd(z)
         x = EV(j) / tn
         If z <= x Then
            e = e + 1
            Ez(j) = Ez(j) + 1
```

```
            Cells(Ez(j) + 2, 3 + j) = EV(j)
            Se = Se + EV(j)
            Ae = Ae + 1
            EV(j) = 0
         End If
      Next j
'
'Auslassventile
      a = 0
      For j = 1 To 3
         z = Rnd(z)
         x = AV(j) / tn
         If z <= x Then
            a = a + 1
            Az(j) = Az(j) + 1
            Cells(Az(j) + 2, 6 + j) = AV(j)
            Sa = Sa + AV(j)
            Aa = Aa + 1
            AV(j) = 0
         End If
      Next j
'
'Montage- und Ausfallzeiten
      If e > 0 Or a > 0 Then
         zm = zm + z1 + z9
         za = za + z1 + z2 + z3 + z4 + z5 + z6 + z7 + _
              z8 + z9
      End If

      If e > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         For k = 1 To 3
            zm = zm + z4 + z5 + z6
            Ez(k) = Ez(k) + 1
            Cells(Ez(k) + 2, 3 + k) = EV(k)
            Se = Se + EV(k)
            Ae = Ae + 1
            EV(k) = 0
         Next k
      End If

      If a > 0 Then
         zm = zm + z2 + z3 + z7 + z8
         For k = 1 To 3
            zm = zm + z4 + z5 + z6
            Az(k) = Az(k) + 1
            Cells(Az(k) + 2, 6 + k) = AV(k)
            Sa = Sa + AV(k)
            Aa = Aa + 1
            AV(k) = 0
         Next k
      End If
'
'Zeitschritt
      For j = 1 To 3
         EV(j) = EV(j) + dt
         AV(j) = AV(j) + dt
```

```
      Next j
   Next t
   Cells(16, 2) = zm
   Cells(17, 2) = za
   Cells(18, 2) = Se / Ae
   Cells(19, 2) = Sa / Aa
End Sub
```

Die mittleren Werte mehrerer Auswertungen der Testdaten sehen Sie in Tabelle 7-7.

Es ergeben sich für die erste und zweite Methode keine nennenswerten Unterschiede. Zwar ist die Ausfallzeit bei der zweiten Methode etwas geringer, dafür aber auch die Standzeit. Die dritte Methode hat deutlich geringere Ausfallzeiten, aber ebenso deutlich geringere Standzeiten.

Tabelle 7-7: Auswertungsdaten im Mittel

	1. Methode	2. Methode	3. Methode
Montagezeit [h]	5960	5956	6795
Ausfallzeit [h]	3815	3745	3063
Standzeit [100h]	3,65	3,3	1,6

Übung 7-3: Weitere Methoden

Überlegen und programmieren Sie weitere Methoden.

Beispiel 7-4: Neutronendiffusion durch eine Reaktorwand

Eine der ersten Anwendungen der probabilistischen Simulation fand sich bei der Auslegung von Kernreaktoren. Es stellte sich die Frage, wie viele Neutronen die Bleihülle des Reaktors durchdringen. Eine klassische Aufgabenstellung für eine Simulation.

Gehen wir von der vereinfachten Annahme aus, dass die Neutronen senkrecht in die Wand des Kernreaktors eindringen. Als weitere Vereinfachung nehmen wir an, das die Dicke der Wand dass Maß 5s besitzt. Ein Neutron legt nun die Strecke s zurück, bevor es auf ein Bleiatom trifft und dabei in eine zufällige Richtung abgelenkt wird. Bis zum nächsten Zusammenstoß mit einem Bleiatom legt es wieder die Strecke s zurück, und so weiter. Nach 10 Kollisionen wird das Neutron absorbiert, vorausgesetzt, es tritt nicht vorher aus der Wand aus. Für unsere Simulation ergeben sich nun folgende Fragestellungen:

- Wie viele Neutronen durchdringen die Wand
- Wie viele Neutronen werden von der Wand absorbiert
- Wie viele Neutronen gelangen zurück in den Reaktor

Die Abbildung 7-11 zeigt den Sachverhalt noch einmal in einem Schema.

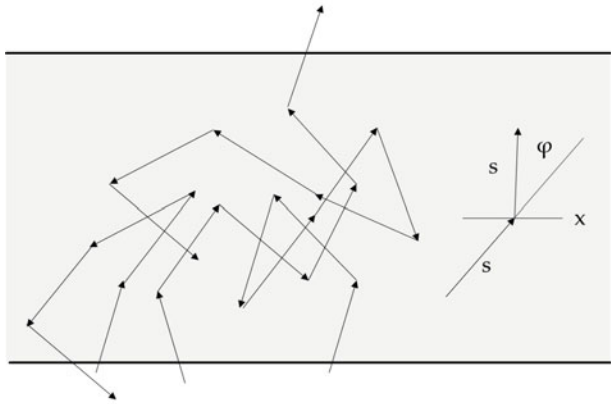


Abbildung 7-11: Bewegung der Neutronen durch die Reaktorwand

Einen einfachen Algorithmus zeigt die Tabelle 7-8. Schreiben Sie entsprechend diesem Algorithmus eine Prozedur, die die Anzahlen ermittelt. Erweitern Sie die Prozedur in der Form, dass die Werte mehrerer Simulationen automatisch in eine Tabelle geschrieben werden. Ermitteln Sie daraus abschließend den Mittelwert.

Tabelle 7-8: Neutronenbewegung in einer Reaktorwand

Setze nd = 0, na = 0, nr = 0		
i = 1(1)100000		
n=0, m=false		
n = n + 1		
Erzeuge Zufallszahl z ε[0,1)		
x = x + cos(2πz)		
Ist x > = 5		
Ja	Nein	
nd = nd +1, m = true	./.	
Ist x < 0		
Ja	Nein	
nr = nr + 1, m = true	./.	
Ist n = 10		
Ja	Nein	
na = na + 1, m = true	./.	
solange m = false		
Ausgabe nd, na, nr		

8 Algorithmen auf Datenstrukturen

Datenstrukturen realisieren in der einfachsten Form einen Datentyp. Ein Datentyp ist eine Zusammenfassung von Objekten, einschließlich der darauf zulässigen Operationen. Grundlegende Datentypen sind Felder, Arrays, Listen, Stacks, Queues, Bäume und Graphen. Dazu kommen noch zusammengesetzte Datentypen.

8.1 Permutationen

Jede vollständige Zusammenstellung einer endlichen Anzahl von Elementen in beliebiger Reihenfolge heißt Permutation. Aus der Mathematik ergeben sich für n Elemente n! Permutationen. Da n! eine sehr schnell wachsende Funktion ist, lassen sich Berechnungen auf Permutationen nur im unteren Zahlenbereich sinnvoll auf Rechenanlagen einsetzen.

Bevor wir zu einem Anwendungsbeispiel kommen, wollen wir uns zunächst mit der Bestimmung von Permutationen befassen. Ähnlich, wie wir n! rekursiv auf (n-1)! zurückgeführt haben, lässt sich dies auch bei den Permutationen bewerkstelligen. Setzt man die Permutationen n-1 voraus, so erhält man n Permutationen, indem die Zahl n an jede mögliche Stelle eingefügt wird.

Betrachten wir diesen Algorithmus in Struktogrammform. Dazu benutzen wir der Einfachheit halber die natürlichen Zahlen.

Tabelle 8-1: Dateneingabe zur Erzeugung von Permutationen

Eingabe der Anzahl n	
	i=1 (1) n
	x(i)=i
Permutation(1)	

Zunächst wird nach der Eingabe der Anzahl n ein Vektor x() definiert und mit natürlichen Zahlen von 1 bis n gefüllt.

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_8,

Tabelle 8-2: Rekursiver Algorithmus zur Erzeugung von Permutationen

Permutation (k)	
y = x(k)	
i=k (1) n	
x(k)=x(i)	
x(i)=y	
Ist k < n	
Ja	Nein
Permutation (k+1)	Ausgabe von x(i)
x(i)=x(k)	
x(k)=y	

Schon für diese Berechnung legen wir ein Tabellenblatt Permutationen an und programmieren diesen Algorithmus.

Code 8-1: Erzeugung von Permutationen natürlicher Zahlen

```
Option Explicit

Dim n, x(), j As Integer

Sub Permut_Start()
   Dim i As Integer

   ThisWorkbook.Worksheets("Permutationen").Cells.Clear
   n = InputBox("Anzahl")
   ReDim x(n)
   j = 0
   For i = 1 To n
      x(i) = i
   Next i
   Call Permut(1)
End Sub

Sub Permut(k As Integer)
   Dim i, y As Integer

   y = x(k)
   For i = k To n
      x(k) = x(i)
      x(i) = y
      If k < n Then
         Call Permut(k + 1)
      Else
         Call Permut_Ausgabe
      End If
      x(i) = x(k)
   Next i
   x(k) = y
```

```
End Sub

Sub Permut_Ausgabe()
   Dim i As Integer

   j = j + 1
   For i = 1 To n
      Cells(j, i) = x(i)
   Next i
End Sub
```

Aufgerufen wird die Prozedur *Permut_Start* über einen Menüpunkt.

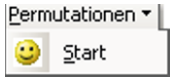


Abbildung 8-1: Menü Permutationen

Das Programm liefert für n = 5 genau 5! = 120 Permutationen. Abbildung 8-2 zeigt die ersten Zeilen der Berechnung.

	A	B	C	D	E
1	1	2	3	4	5
2	1	2	3	5	4
3	1	2	4	3	5
4	1	2	4	5	3
5	1	2	5	4	3
6	1	2	5	3	4
7	1	3	2	4	5
8	1	3	2	5	4
9	1	3	4	2	5
10	1	3	4	5	2

Abbildung 8-2: Permutationen von n = 5

Wenden wir uns nun einem Anwendungsbeispiel zu.

Beispiel 8-1: Fließbandarbeit

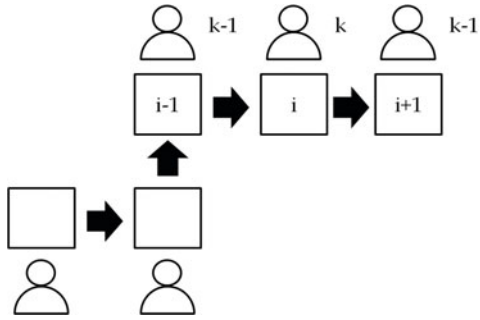


Abbildung 8-3: Schema einer Fließbandarbeit

Ein Fließband hat n Stationen zur Bearbeitung. Der auf der Station i postierte Arbeiter k, übernimmt das Werkstück von der Station i-1 und übergibt sie nach der Bearbeitung an die Station i+1. Die eingesetzten Arbeiter haben an den Stationen ein unterschiedliches Arbeitsvermögen. Allgemein hat ein Arbeiter k an der Station i das Arbeitsvermögen a_{ik}. Dieses wird zum Beispiel in Stückzahl/Zeiteinheit ausgedrückt.

Das Problem besteht nun darin, die Arbeiter so den einzelnen Stationen zuzuordnen, dass der kleinste Ausstoß einer einzelnen Station maximiert wird. Dieser Wert entspricht dem Gesamtausstoß und kann auch durch erhöhte Beschickung des Fließbandes nicht überschritten werden.

Gesucht ist also die Permutation $(p_1,\ldots,p_n)$ der Arbeitsvermögen der Arbeiter $(1,\ldots,n)$, so dass gilt

$$\min_{i=1,\ldots,n} a_{ip_i} \rightarrow Maximum \quad . \tag{8.1.1}$$

Entsprechend müssen wir den vorherigen Algorithmus abändern.

Tabelle 8-3: Dateneingabe zum Engpassproblem

Bestimmung der Anzahl n
i=1 (1) n
x(i)=i
k=1 (1) n
a(i,k)=Zelle(i,k)
Permutation(1)

Tabelle 8-4: Auswertung des Engpassproblems

Permutation (k)	
y = x(k)	
i=k (1) n	
x(k)=x(i)	
x(i)=y	
Ist k < n	
Ja	Nein
Permutation (k+1)	Ausgabe von x(i)
x(i)=x(k)	
x(k)=y	

Tabelle 8-5: Ausgabe der berechneten Daten

∑=0	
i=i (1) n	
∑=∑+a(i,x(i))	
Ist ∑ > Max	
Ja	Nein
Max=∑	
i=1 (1) n	
Zelle(n+2,i)=x(i)	
Zelle(n+3,i)=a(i,x(i))	
Zelle(n+3,n+1)=∑	

In das neue Tabellenblatt Engpass können die Prozeduren zunächst übernommen und dann ergänzt werden.

Code 8-2: Prozeduren zum Engpassproblem

```
Option Explicit

Dim MyDoc As Object
Dim n, x() As Long
Dim a(), Max As Double

Sub Engpass_Leer()
   ThisWorkbook.Worksheets("Engpass").Cells.Clear
End Sub
Sub Engpass_Test()
   Cells(1, 1) = 7
   Cells(1, 2) = 5
   Cells(1, 3) = 11
   Cells(1, 4) = 13
   Cells(1, 5) = 14
   Cells(1, 6) = 9

   Cells(2, 1) = 3
   Cells(2, 2) = 6
   Cells(2, 3) = 15
   Cells(2, 4) = 6
   Cells(2, 5) = 3
   Cells(2, 6) = 11

   Cells(3, 1) = 10
   Cells(3, 2) = 6
   Cells(3, 3) = 3
   Cells(3, 4) = 1
   Cells(3, 5) = 2
   Cells(3, 6) = 7

   Cells(4, 1) = 4
   Cells(4, 2) = 7
   Cells(4, 3) = 5
   Cells(4, 4) = 1
   Cells(4, 5) = 6
   Cells(4, 6) = 8

   Cells(5, 1) = 5
   Cells(5, 2) = 7
   Cells(5, 3) = 8
   Cells(5, 4) = 3
   Cells(5, 5) = 10
   Cells(5, 6) = 6

   Cells(6, 1) = 4
   Cells(6, 2) = 5
   Cells(6, 3) = 3
   Cells(6, 4) = 6
   Cells(6, 5) = 12
   Cells(6, 6) = 9
End Sub
```

```
Sub Enpass_Start()
   Dim i, k As Long
   Set MyDoc = ThisWorkbook.Worksheets("Engpass")
   n = MyDoc.UsedRange.Rows.Count
   Max = 0

   ReDim x(n), a(n, n)

   For i = 1 To n
      x(i) = i
      For k = 1 To n
         a(i, k) = Cells(i, k)
      Next k
   Next i

   Call Permut(1)
End Sub

Sub Permut(k As Integer)
   Dim i, y As Long

   y = x(k)
   For i = k To n
      x(k) = x(i)
      x(i) = y
      If k < n Then
         Call Permut(k + 1)
      Else
         Call Permut_Ausgabe
      End If
      x(i) = x(k)
   Next i
   x(k) = y
End Sub

Sub Permut_Ausgabe()
   Dim i As Integer
   Dim z As Double

   z = 0
   For i = 1 To n
      z = z + a(i, x(i))
   Next i

   If z > Max Then
      Max = z
      For i = 1 To n
         Cells(n + 2, i) = x(i)
         Cells(n + 3, i) = a(i, x(i))
      Next i
      Cells(n + 3, n + 1) = z
   End If
End Sub
```

Damit freie Daten ins Formblatt eingetragen werden können, muss zunächst der alte Inhalt gelöscht werden. Dafür sorgt die Prozedur *Engpass_Leer*. Testdaten und Auswertung sind nach dem bisherigen Schema aufrufbar.

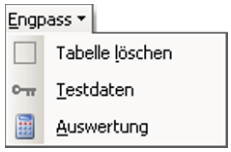


Abbildung 8-4: Menü zur Fließbandarbeit

Als Beispieldaten sind sechs Arbeitsstationen mit sechs Arbeitern besetzt, deren Arbeitsvermögen an den einzelnen Stationen durch die Testdaten ausgedrückt werden.

	A	B	C	D	E	F	G
1	7	5	11	13	14	9	
2	3	6	15	6	3	11	
3	10	6	3	1	2	7	
4	4	7	5	1	6	8	
5	5	7	8	3	10	6	
6	4	5	3	6	12	9	
7							
8	4	3	1	6	2	5	
9	13	15	10	8	7	12	65

Abbildung 8-5: Auswertung der Testdaten

Der Maximalwert von 65 wird erreicht, wenn in der ersten Zeile der 4. Spaltenwert, nämlich 13 genommen wird. Dann in der zweiten Zeile und der 3. Spalte der Wert 15, usw. Von allen Werten 13, 15, 10, 8, 7 und 12 ist 7 der Minimalwert. Folglich ist der Arbeitswert $a_{5,2}$ der Engpass.

Übung 8-1: Parallele Arbeiten

Ist die Reihenfolge der auszuführenden Arbeiten nicht fest vorgegeben und kann damit parallel erfolgen, so ist der Engpass derjenige Auftrag, der die längste Arbeitszeit erfordert. Mathematisch bedeutet dies

$$\max_{i=1,\ldots,n} a_{ip_i} \rightarrow Minimum \quad . \tag{8.1.2}$$

Das Problem ist mit dem gleichen Programm lösbar, wenn die Werte negativ eingegeben werden. Prüfen Sie dies nach.

8.2 Regression und Korrelation

Regressions- und Korrelationsanalyse befassen sich mit der Aufdeckung und Beschreibung der Abhängigkeit von Merkmalen (Zufallsgrößen). Während sich die Regressionsanalyse mit der Art des Zusammenhangs zwischen den Merkmalen beschäftigt, ist es die Aufgabe der Korrelationsanalyse, den Grad dieses Zusammenhangs zu bestimmen.

Oft ergeben sich in der Praxis Messwerte, bei denen man in der einfachsten Form einen linearen Zusammenhang vermutet. In der Regel sind n Messwertepaare gegeben, wobei einer unabhängigen Variablen x_i (i=1,…,n) genauso viele abhängige Werte y_i zugeordnet werden. Der funktionale Zusammenhang kann nicht direkt angegeben werden, da die Messwerte von einer Störgröße ε überlagert werden.

$$y = a \cdot x + b + \varepsilon \tag{8.2.1}$$

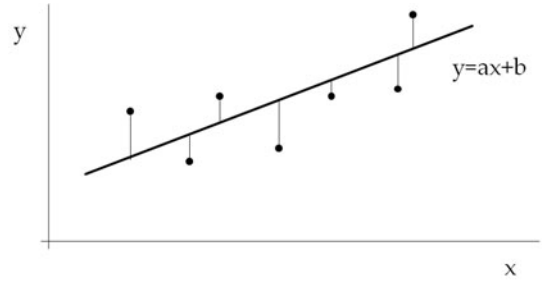


Abbildung 8-6: Messwerte und Regressionsgerade

Es gibt verschiedene Möglichkeiten, die Gerade zu bestimmen. Eine der herkömmlichsten Methoden ist die uns bereits bekannte Methode der kleinsten Fehlerquadrate, die wir bereits bei der Approximation verwendet haben. Dabei ist das Ziel

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - \left(a \cdot x_i + b\right)\right)^2 \rightarrow Minimum \ , \tag{8.2.2}$$

die Minimierung der summierten Quadrate der Residuen. Aus dieser Forderung ergibt sich

$$a = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{8.2.3}$$

und

$$b = \bar{y} - a \cdot \bar{x} \ . \tag{8.2.4}$$

Darin sind $\bar{x}$ und $\bar{y}$ die aus den x_i und y_i gebildeten Mittelwerte

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{8.2.5}$$

und

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \ . \tag{8.2.6}$$

Mit Hilfe der Koeffizienten a und b haben wir somit zwar eine Gerade, aber noch immer keine Aussage, ob es wirklich einen linearen Zusammenhang gibt. Eine Aussage darüber bekommt man durch die Bestimmung des Korrelationskoeffizienten aus der Gleichung

$$r = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \cdot \sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} = \frac{Kov(x,y)}{\sqrt{Var(x)} \cdot \sqrt{Var(y)}} \quad . \tag{8.2.7}$$

Während man den Ausdruck im Zähler als Kovarianz bezeichnet, heißen die Ausdrücke unter der Wurzel Varianzen. Der Korrelationskoeffizient r kann nur Werte zwischen +1 und -1 annehmen. Bei einem Wert von +1 besteht ein vollständig positiver linearer Zusammenhang. Bei -1 ein entsprechend negativer. Ist der Wert Null, dann besteht kein linearer Zusammenhang. Es kann dann aber immer noch ein nichtlinearer Zusammenhang bestehen.

Schreiben wir also zunächst ein Programm zur Bestimmung einer Regressionsgeraden und suchen wir uns dann einen Anwendungsfall.

Tabelle 8-6: Bestimmung der Regressionsgeraden

Einlesen und Auswertung der Daten
i=i (1) n
x_i =Zelle(i,1)
y_i =Zelle(i,2)
$\sum x_i = \sum x_i + x_i$
$\sum y_i = \sum y_i + y_i$
$\bar{x} = \frac{1}{n}\sum x_i$
$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
i=1 (1) n
$\sum (x_i - \bar{x}) = \sum (x_i - \bar{x}) + (x_i - \bar{x})$
$\sum (y_i - \bar{y}) = \sum (y_i - \bar{y}) + (y_i - \bar{y})$
$\sum (x_i - \bar{x})^2 = \sum (x_i - \bar{x})^2 + (x_i - \bar{x})^2$
$\sum (x_i - \bar{x})(y_i - \bar{y}) = \sum (x_i - \bar{x})(y_i - \bar{y}) + (x_i - \bar{x})(y_i - \bar{y})$

$$a = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - b \cdot \bar{x}$$

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Zusätzlich werden wir in der Prozedur noch die Werte der Regressionsgeraden berechnen und zusammen mit den Messwerten darstellen.

Code 8-3: Bestimmung einer linearen Regression

```
Option Explicit


Sub Regression_Leer()
   ThisWorkbook.Worksheets("Regression").Cells.Clear
   Call Regression_Grafik_löschen
End Sub
Sub Regression_Test()
   Dim i As Integer
   For i = 1 To 10
      Cells(i, 1) = i
   Next i
   Cells(1, 2) = 1.57
   Cells(2, 2) = 2.22
   Cells(3, 2) = 2.03
   Cells(4, 2) = 2.41
   Cells(5, 2) = 2.35
   Cells(6, 2) = 2.38
   Cells(7, 2) = 2.58
   Cells(8, 2) = 2.49
   Cells(9, 2) = 2.78
   Cells(10, 2) = 2.92
End Sub

Sub Regression_Start()
   Dim myDoc As Object
   Dim i As Long
   Dim a, b, r, sx, sy, xq, yq As Double
   Dim sxx, syy, sx2, sy2, sxy As Double
   Set MyDoc = ThisWorkbook.Worksheets("Regression")
   n = MyDoc.UsedRange.Rows.Count

   ReDim x(n), y(n)

   For i = 1 To n
      x(i) = Cells(i, 1)
```

```
      y(i) = Cells(i, 2)
      sx = sx + x(i)
      sy = sy + y(i)
   Next i
   xq = sx / n
   yq = sy / n
   For i = 1 To n
      sxx = sxx + (x(i) - xq)
      syy = syy + (y(i) - yq)
      sx2 = sx2 + (x(i) - xq) ^ 2
      sy2 = sy2 + (y(i) - yq) ^ 2
      sxy = sxy + (x(i) - xq) * (y(i) - yq)
   Next i
   a = sxy / sx2
   b = yq - a * xq
   r = sxy / Sqr(sx2) / Sqr(sy2)
   Cells(1, 5) = a
   Cells(2, 5) = b
   Cells(3, 5) = r
   For i = 1 To n
      Cells(i, 3) = a * x(i) + b
   Next i
End Sub

Sub Regression_Grafik()
   Charts.Add
   ActiveChart.ChartType = xlXYScatterSmoothNoMarkers
   ActiveChart.SetSourceData Source:= _
      Sheets("Regression").Range("C15")
   ActiveChart.SeriesCollection.NewSeries
   ActiveChart.SeriesCollection.NewSeries
   ActiveChart.SeriesCollection(1).XValues = _
      "=Regression!R1C1:R10C1"
   ActiveChart.SeriesCollection(1).Values = _
      "=Regression!R1C2:R10C2"
   ActiveChart.SeriesCollection(1).Name = "=""Messwerte"""
   ActiveChart.SeriesCollection(2).XValues = _
      "=Regression!R1C1:R10C1"
   ActiveChart.SeriesCollection(2).Values = _
      "=Regression!R1C3:R10C3"
   ActiveChart.SeriesCollection(2).Name = _
      "=""Regressionsgerade"""
   ActiveChart.Location Where:=xlLocationAsObject, _
      Name:="Regression"
   ActiveChart.Axes(xlValue).Select
   With ActiveChart.Axes(xlValue)
      .MinimumScale = 1.5
      .MaximumScale = 3
      .MinorUnitIsAuto = True
      .MajorUnitIsAuto = True
      .Crosses = xlAutomatic
      .ReversePlotOrder = False
      .ScaleType = xlLinear
      .DisplayUnit = xlNone
   End With
   ActiveChart.Axes(xlCategory).Select
   With ActiveChart.Axes(xlCategory)
```

```
        .MinimumScaleIsAuto = True
        .MaximumScale = 10
        .MinorUnitIsAuto = True
        .MajorUnitIsAuto = True
        .Crosses = xlAutomatic
        .ReversePlotOrder = False
        .ScaleType = xlLinear
        .DisplayUnit = xlNone
    End With
End Sub
Sub Regression_Grafik_löschen()
    Dim Shp As Shape
    For Each Shp In Worksheets("Regression").Shapes
        Shp.Delete
    Next
End Sub
```

Die Symbolleiste bekommt wieder einen kompletten Eintrag.

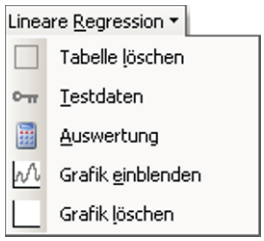


Abbildung 8-7: Menü zur Bestimmung der Regressionsgeraden

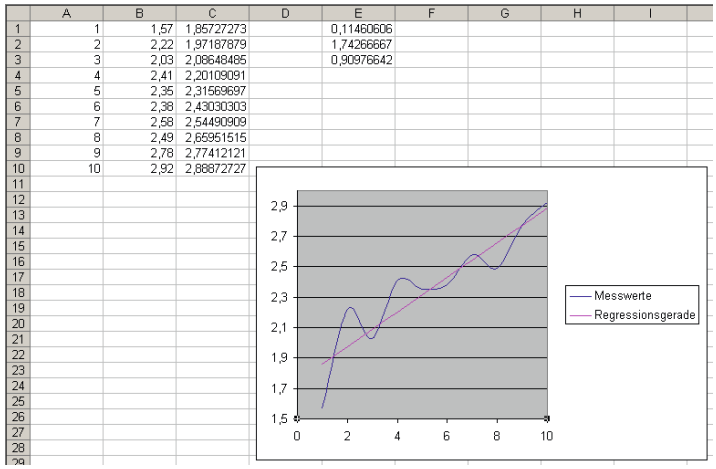


Abbildung 8-8: Auswertung der Testdaten

Die Auswertung der Testdaten ergibt einen Korrelationskoeffizienten von 0,91, und damit ist ein linearer Zusammenhang gegeben. Die Regressionsgerade berechnet sich angenähert mit

$$y = 0{,}1146 \cdot x + 1{,}7427 \; .$$

Beispiel 8-1: Experimentelle Bestimmung einer Feder

Dieses elementare Verfahren der Messtechnik bestimmt die Federkonstante und die effektive Federmasse einer unbekannten Feder. Die Versuchsanordnung ist in Abbildung 8-9 schematisch dargestellt. Danach werden verschiedene Zusatzmassen aufgelegt und die jeweilige Auslenkung festgehalten. Allgemein gilt für die Kraft F die zur Auslenkung s einer Feder notwendig ist

$$F = c \cdot s \quad . \tag{8.2.8}$$

Der Proportionalitätsfaktor c wird als Federkonstante bezeichnet und bestimmt sich durch Umstellung aus

$$c = \frac{F}{s} \quad . \tag{8.2.9}$$

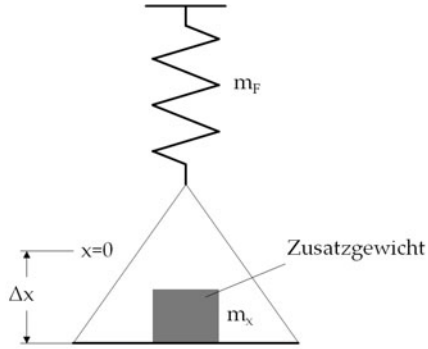


Abbildung 8-9: Federpendel

Eine Beispielfeder hat folgende Daten:

Masse [kg]	Gewichtskraft [N]	Auslenkung [cm]
0,5	4,905	1,92
1,0	9,81	3,84
1,5	14,715	5,55
2,0	19,62	7,34
2,5	24,525	9,21

Diese in das Programm eingegeben, liefern einen starken linearen Zusammenhang (0,999…) und die Federkonstante ist 0,3686 N/cm (Abbildung 8-10).

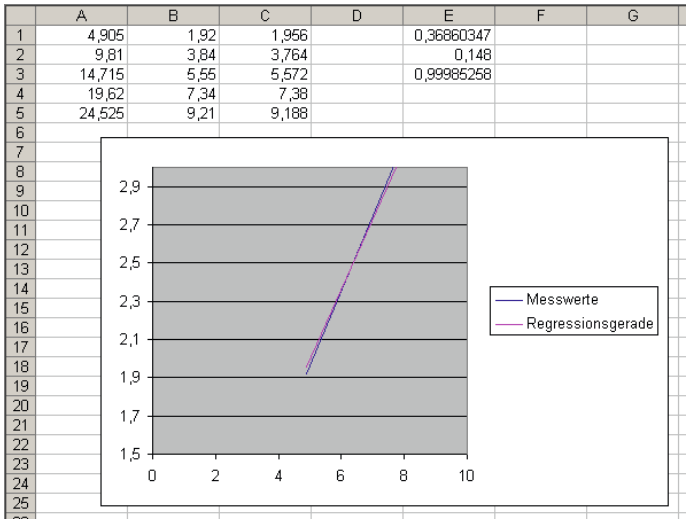


Abbildung 8-10: Bestimmung der Federkonstanten aus den Beispieldaten

Übung 8-2: Anzahl der Messwerte

Zur Darstellung der Grafik ist eine feste Größe von 10 Messwerten programmiert. Ändern Sie dies auf die tatsächliche Anzahl vorhandener Messwerte ab.

Befassen Sie sich auch mit der nichtlinearen Regression und schreiben Sie Prozeduren für eine quadratische und kubische Regression.

8.3 Arrays und Datenfelder

Arrays sind eine Sammlung von gleichen Datentypen unter einem Namen. Vergleichbar den Matrizen in der Mathematik. Arrays sind außerdem n-dimensionale Gebilde und besitzen für jede Dimension einen Index (Abbildung 8-11).

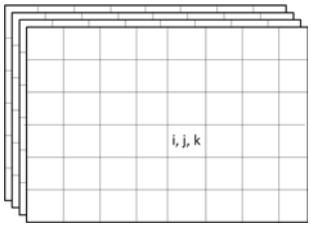


Abbildung 8-11: Aufbau eines Arrays

Arrays werden auch als Datenfeldern, Matrizen, indizierte Variable, etc. bezeichnet. Arrays mit nur einem Index werden oft als Vektoren bezeichnet.

Diese Datenstruktur wurde bereits unter 3.1 zur Lösung eines linearen Gleichungssystems nach dem Gauß Algorithmus benutzt. Neben der Verwaltung von Zahlenmengen lassen sich mit ihnen auch logische Operationen durchführen. Im Prinzip ist eine Excel-Tabelle ein zweidimensionales Array.

Arrays tauchen in allen Beispielen dieses Buches auf und unterstreichen daher die Wichtigkeit dieser Datenstruktur. Wir wollen uns nachfolgend mit einem kleinen Beispiel begnügen, in dem Text und Werte in einem Array verwaltet werden.

Beispiel 8-2: Nutzwertanalyse

Die Nutzwertanalyse wurde in den USA unter dem Begriff utility analysis entwickelt und seit den 70er Jahren auch in Deutschland eingesetzt.

Die Nutzwertanalyse priorisiert verschiedene Lösungen. Dies geschieht durch deren Gewichtung im Hinblick zur Erreichung eines oder mehrerer Ziele.

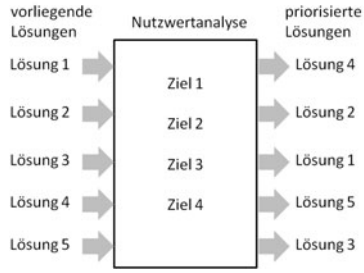


Abbildung 8-12: Schema einer Nutzwertanalyse

Zunächst gilt es festzustellen, welche Kriterien für eine Projektentscheidung wichtig und maßgeblich sein sollen. In den meisten Fällen können schon im ersten Schritt KO-Kriterien formuliert werden, die zwingend erfüllt werden müssen. Lösungen, die diese Bedingung nicht erfüllen, scheiden sofort aus. Diese Muss-Kriterien können durch Soll-Kriterien ergänzt werden, deren Erfüllung erwünscht, aber nicht notwendig ist.

In einem zweiten Schritt müssen nun die einzelnen Soll-Ziele in eine Ordnung gebracht werden. Möglich ist eine Systematisierung in Form von Oberzielen und dazugehörigen Unterzielen. Den einzelnen Zielen werden Gewichtungsfaktoren zugeordnet. Das kann in der Form von Multiplikatoren von 1 (wenig wichtig) bis 5 (sehr wichtig) oder als Prozentangaben geschehen.

Lösungen	Teilnutzen 1 x Ziel 1 (Faktor 2)	Teilnutzen 2 x Ziel 2 (Faktor 4)	Teilnutzen 3 x Ziel 3 (Faktor 3)	
A	3 x 2 = 6	2 x 4 = 8	4 x 3 = 12	26
B	3 x 2 = 6	3 x 4 = 12	2 x 3 = 6	24
C	2 x 2 = 4	2 x 4 = 8	1 x 3 = 3	15

Abbildung 8-13: Beispiel einer Nutzwertanalyse

Unter den verschiedenen Formen wählen wir die in dem Beispiel von Abbildung 8-13 dargestellte Form. Ein Struktogramm halte ich für diesen einfachen Algorithmus nicht für notwendig. Vielmehr betrachten wir gleich den Quellcode.

Code 8-4: Nutzwertanalyse

```
Option Explicit
   Dim m, n As Integer

Sub Nutzwert_Start()
   m = 0
   n = 0
   Call Nutzwert_Formblatt(m, n)
End Sub

Sub Nutzwert_Formblatt(m, n)
   Dim i, j As Integer
   Dim r, c As String

   ThisWorkbook.Worksheets("Nutzwertanalyse").Cells.Clear

   'Eingaben erforderlich?
   If m = 0 Then
      m = InputBox("Anzahl der Lösungen")
   End If
   If n = 0 Then
      n = InputBox("Anzahl der Kriterien")
   End If
   If m = 0 Or n = 0 Then
      MsgBox "Eingabefehler!"
      Exit Sub
   End If

   'Spaltenbeschriftung
   Columns("A:A").ColumnWidth = 30
   Range("A1").Value = "Lösungen"
   For i = 1 To n
      c = Chr(64 + i * 2) & "1"
      Range(c).Select
      With Selection
         .Value = Str(i) & ". Krit."
      End With
   Next i

   'Spaltenbreite
   For i = 1 To n
      c = Chr(64 + i * 2) & ":" & Chr(64 + i * 2 + 1)
      Columns(c).Select
      With Selection
         .ColumnWidth = 6
      End With
   Next i

   c = Chr(64 + (n + 1) * 2) & "1"
   Range(c).Value = "Nutzen"

   'Farbkennung
```

```
   For i = 1 To n
      r = Chr(64 + i * 2 + 1) & "2:" & _
          Chr(64 + i * 2 + 1) & LTrim(Str(1 + m))
      Range(r).Select
      With Selection.Interior
         .ColorIndex = 15
         .Pattern = xlSolid
      End With
   Next i

   r = Chr(64 + n * 2 + 2) & "2:" & _
       Chr(64 + n * 2 + 2) & LTrim(Str(1 + m))
   Range(r).Select
   With Selection.Interior
      .ColorIndex = 15
      .Pattern = xlSolid
   End With

   'Formeln setzen
   For i = 1 To n
      c = Chr(64 + i * 2 + 1)
      For j = 1 To m
         r = c & LTrim(Str(1 + j))
         Range(r).Select
         ActiveCell.FormulaR1C1 = "=RC[-1]*R[-" & _
                                  LTrim(Str(j)) & "]C"
      Next j
   Next i

   'Summen bestimmen
   For j = 1 To m
      r = Chr(64 + n * 2 + 2) & LTrim(Str(1 + j))
      c = "="
      For i = 1 To n
         c = c + "RC[-" & LTrim(Str(2 * i - 1)) & "]"
         If i < n Then c = c & "+"
      Next i
      Range(r).Select
      ActiveCell.FormulaR1C1 = c
   Next j
End Sub

Sub Nutzwert_Beispiel()
   m = 3
   n = 3
   Call Nutzwert_Formblatt(m, n)

   Cells(2, 1) = "A"
   Cells(3, 1) = "B"
   Cells(4, 1) = "C"

   Cells(1, 3) = 2
   Cells(1, 5) = 4
   Cells(1, 7) = 3

   Cells(2, 2) = 3
   Cells(3, 2) = 3
```

```
    Cells(4, 2) = 2

    Cells(2, 4) = 2
    Cells(3, 4) = 3
    Cells(4, 4) = 2

    Cells(2, 6) = 4
    Cells(3, 6) = 2
    Cells(4, 6) = 1
End Sub
```

In diesem Beispiel werden die indirekte Adressierung und die Berechnung der direkten und indirekten Adressierung gezeigt. Aufgerufen werden beide Prozeduren wieder über einen Menüpunkt.

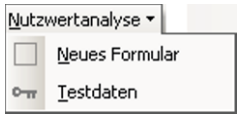


Abbildung 8-14: Menü zur Nutzwertanalyse

Das Testbeispiel liefert das in Abbildung 8-15 dargestellte Ergebnis.

	A	B	C	D	E	F	G	H
1	Lösungen	1. Krit.	2	2. Krit.	4	3. Krit.	3	Nutzen
2	A	3	6	2	8	4	12	26
3	B	3	6	3	12	2	6	24
4	C	2	4	2	8	1	3	15

Abbildung 8-15: Ergebnis des Testbeispiels

Übung 8-3: Formblatt

Entwerfen Sie ein Formblatt, in dem Ober- und Unterziele definiert werden können.

In VBA gibt es die Möglichkeit Bereiche (Range) durch einen Variablen-Namen zu deklarieren. So kann zum Beispiel der Bereich G1:H6 eines Tabellenblattes *Test* den Namen Feld durch folgende Anweisung erhalten:

```
ActiveWorkbook.Names.Add "Feld" & Format(i, "_00"), _
RefersTo:="=Test!$G$1:$H$6"
```

8.4 Arbeiten auf Listenstrukturen

Daten in Listen zu sammeln dient oft dem Zweck, einer bestimmten Gruppe von Benutzern schnell brauchbare Informationen zu liefern. Wie diese zu organisieren sind, hängt von der Art der Nutzung ab. Ebenso gibt es eine Vielzahl von Suchmethoden zum Finden der Informationen. Siehe dazu auch Kapitel 9.1.

Zunächst betrachten wir den im Prinzip schnellsten Sortieralgorithmus auf Listen.

Beispiel 8-3: Quicksort

Quicksort ist ein schneller rekursiver Sortieralgorithmus, der nach dem Prinzip Teile und Herrsche (engl. divide and conquer) arbeitet (siehe 9.1). Er wurde 1960 von C. Antony R. Hoare in seiner Grundform entwickelt und seither von anderen weiter verfeinert. Der Algorithmus hat den Vorteil, dass er über eine sehr kurze innere Schleife verfügt (was die Ausführungsgeschwindigkeit stark erhöht) und ohne zusätzlichen Speicherplatz auskommt (abgesehen von dem für die Rekursion zusätzlichen benötigten Platz auf dem Aufruf-Stack).

QuickSort wählt ein Element aus der zu sortierenden Liste aus (Pivotelement) und zerlegt die Liste in zwei Teillisten, eine obere und eine untere, von denen die eine alle Elemente enthält, die größer sind als das Pivotelement, die andere den Rest. Dazu wird zunächst ein Element von unten gesucht, das größer als das Pivotelement und damit für die untere Liste zu groß ist. Entsprechend wird von oben ein kleineres (oder gleichgroßes) Element als das Pivotelement gesucht. Die beiden Elemente werden dann vertauscht und landen damit in der jeweils richtigen Liste. Der Vorgang wird fortgesetzt, bis sich die untere und obere Suche treffen. Damit sind die oben erwähnten Teillisten in einem einzigen Durchlauf entstanden. Suche und Vertauschung können *in place* durchgeführt werden, d. h. es ist keine zweite Liste erforderlich.

Die noch unsortierten Teillisten werden über denselben Algorithmus in noch kleinere Teillisten zerlegt (z.B. mittels Rekursion) und, sobald nur noch Listen mit je einem Element vorhanden sind, wieder zusammengesetzt. Die Sortierung ist damit abgeschlossen.

Tabelle 8-7: QuickSort

Quicksort (von, bis)	
i = von (linker Zeiger = Anfang)	
j = bis (rechter Zeiger = Ende)	
p=(i+j)/2 (Intervall-Halbierung)	
x=a(p)	
So lange a(i) < x	
i=i+1	
So lange a(j) > x	
j=j-1	
Ist i <= j	
Ja	Nein
Tausche (a(i) mit a(j)	./.
i=i+1	
j=j-1	

So lange j < i	
Ist von < j	
Ja	Nein
QuickSort (von, j)	./.
Ist i < bis	
Ja	Nein
Quicksort (i, bis)	./.

Das Pivotelement wird immer in der Mitte des Intervalls bestimmt. Optimierungen suchen das Pivotelement nach geeigneten Kriterien aus.

Code 8-5: Quicksort

```
Option Explicit

Sub Quicksort_Test()
   Dim x As Double
   Dim i As Integer

   x = Timer
   Randomize (x)
   For i = 1 To 100
      Cells(i, 1) = Rnd(x)
   Next i
End Sub

Sub Quicksort_Start()
   Dim MyDoc As Object
   Dim von, bis As Integer

   Set MyDoc = ThisWorkbook.Worksheets("Quicksort")
   von = 1
   bis = MyDoc.UsedRange.Rows.Count
   Call QuickSort(von, bis)
End Sub

Sub QuickSort(v, b)
   Dim i, j, p As Integer
   Dim x, y As Double

   i = v
   j = b
   p = Int((v + b) / 2)
   x = Cells(p, 1)

   'Array aufteilen
   Do
      While (Cells(i, 1) < x)
         i = i + 1
      Wend
      While (Cells(j, 1) > x)
         j = j - 1
      Wend
```

```
      If i <= j Then
         'tauschen
         y = Cells(i, 1)
         Cells(i, 1) = Cells(j, 1)
         Cells(j, 1) = y
         i = i + 1
         j = j - 1
      End If
   Loop Until i > j

   ' rekursive Aufrufe
   If v < j Then
      Call QuickSort(v, j)
   End If
   If i < b Then
      Call QuickSort(i, b)
   End If
End Sub
```

Eine Testprozedur erzeugt zunächst im Tabellenblatt QuickSort 100 Zufallszahlen. Danach werden sie mittels QuickSort sortiert.

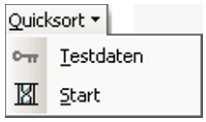


Abbildung 8-16: Menü zum QuickSort

Übung 8-4: BubbleSort

Schreiben Sie einen alternativen Sortieralgorithmus (z. B. den BubbleSort) und vergleichen Sie Sortierzeiten und Durchläufe.

Beispiel 8-4: Stücklistenorganisation

Stücklisten zusammen mit Fertigungsplänen stellen ebenfalls einen Algorithmus dar, nämlich den der Herstellung eines Bauteils, einer Baugruppe oder einer Maschine. Sie sind mit die wichtigsten Datenstrukturen eines produzierenden Unternehmens.

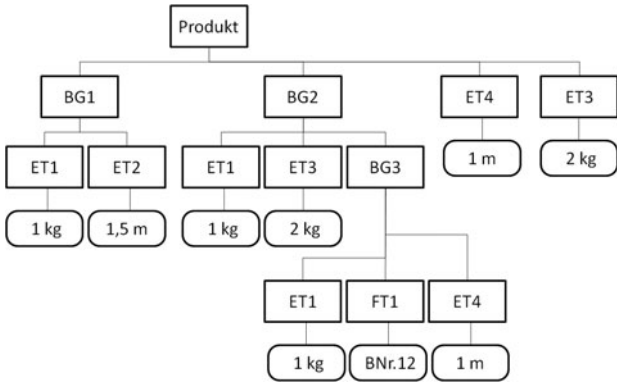


Abbildung 8-17: Beispiel einer Produktstruktur

Wir betrachten ein Produkt, das sich aus Baugruppen und Elementen zusammensetzt.

In unserem Beispiel besteht das Produkt aus zwei Baugruppen und zwei Einzelteilen. Die erste Baugruppe besteht aus zwei Einzelteilen, die zweite aus zwei Einzelteilen und einer weiteren Baugruppe. Diese Baugruppe enthält zwei Einzelteile und ein Fertigteil, dass bei einem anderen Hersteller bestellt werden muss.

Die übliche Stückliste ist die so genannte Baugruppen-Stückliste. Sie ist einfach zu verwalten und aus ihr können alle anderen Dokumente abgeleitet werden. Wir gehen von der ersten Ebene der Produktdarstellung aus und erzeugen eine leere Stückliste.

Code 8-6: Stücklistenorganisation

```
Option Explicit
   Dim Blatt As Worksheet

Sub Stückliste_Neu()
   Dim Name As String
   Dim Index As Integer

   Name = InputBox("Stücklisten-Kennung eingeben!")

   'Überprüfung, ob Stückliste vorhanden
   For Each Blatt In Sheets
      If Name = Blatt.Name Then
         MsgBox "Stückliste bereits vorhanden", vbOKOnly
         Exit Sub
      End If
   Next

   'Stückliste neu erzeugen
   Set Blatt = Worksheets.Add
   Blatt.Name = Name
   Call Stückliste_Formblatt
End Sub

Sub Stückliste_Löschen()
   Dim Name, t As String

   t = ThisWorkbook.ActiveSheet.Name
   Name = InputBox("Zu löschende Stücklisten-Kennung eingeben!", "Löschung", t)

   'Überprüfung, ob Stückliste vorhanden
   For Each Blatt In Sheets
      If Name = Blatt.Name Then
         If MsgBox("Stückliste wirklich löschen?", _
            vbYesNo, "ACHTUNG!") = vbNo Then
            Exit Sub
         End If
         Blatt.Delete
         Exit Sub
      End If
   Next
```

```
End Sub

Sub Stückliste_Formblatt()
   Dim MyDoc As Object

   Set MyDoc = ThisWorkbook.ActiveSheet
   MyDoc.Activate
   MyDoc.Cells.Clear

   MyDoc.Range("A1") = MyDoc.Name & "- Baugruppe"
   MyDoc.Range("A1:F1").Select
   With Selection
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlBottom
      .WrapText = False
      .Orientation = 0
      .AddIndent = False
      .IndentLevel = 0
      .ShrinkToFit = False
      .ReadingOrder = xlContext
      .MergeCells = True
   End With
   With Selection.Font
      .Name = "Arial"
      .Size = 14
      .Strikethrough = False
      .Superscript = False
      .Subscript = False
      .OutlineFont = False
      .Shadow = False
      .Underline = xlUnderlineStyleNone
      .ColorIndex = xlAutomatic
   End With
   Selection.Font.Bold = True

   MyDoc.Range("A2") = "Stufe"
   MyDoc.Range("B2") = "PosNr."
   MyDoc.Range("C2") = "Menge"
   MyDoc.Range("D2") = "Bezeichnung"
   MyDoc.Range("E2") = "Ident-Nr."
   MyDoc.Range("F2") = "Material"
   MyDoc.Range("A2:A2").Select
   With Selection
      .Value = "Stufe"
      .ColumnWidth = 10
   End With
   MyDoc.Range("B2").Select
   With Selection
      .Value = "Pos.Nr."
      .ColumnWidth = 7
   End With
   MyDoc.Range("C2").Select
   With Selection
      .Value = "Menge"
      .ColumnWidth = 8
   End With
   MyDoc.Range("D2").Select
```

```
   With Selection
      .Value = "Bezeichnung"
      .ColumnWidth = 30
   End With
   MyDoc.Range("E2").Select
   With Selection
      .Value = "Baugruppe"
      .ColumnWidth = 10
   End With
   MyDoc.Range("F2").Select
   With Selection
      .Value = "Material"
      .ColumnWidth = 20
   End With

   MyDoc.Columns("A:A").Select
   With Selection
      .NumberFormat = "@"
      .HorizontalAlignment = xlLeft
      .VerticalAlignment = xlBottom
   End With
   MyDoc.Columns("B:C").Select
   With Selection
      .NumberFormat = "0"
      .HorizontalAlignment = xlRight
      .VerticalAlignment = xlBottom
   End With
   MyDoc.Columns("D:F").Select
   With Selection
      .NumberFormat = "@"
      .HorizontalAlignment = xlLeft
      .VerticalAlignment = xlBottom
   End With

   MyDoc.Range("A3").Select
End Sub

Sub Stückliste_Struktur()
   Dim MyDoc, MyDocNew As Object
   Dim Min, Max, MaxU, x As Integer
   Dim Name, t As String
   Dim i, j, k, s As Integer

   Set MyDoc = ThisWorkbook.ActiveSheet
   MyDoc.Activate

   Max = MyDoc.UsedRange.Rows.Count - 2
   Name = "S-" & MyDoc.Name

   'Überprüfung, ob Stückliste vorhanden
   For Each Blatt In Sheets
      If Name = Blatt.Name Then
         MsgBox "Stückliste bereits vorhanden", vbOKOnly
         Exit Sub
      End If
   Next
```

```
'Stückliste neu erzeugen
Set Blatt = Worksheets.Add
Blatt.Name = Name

'Mit neuer Stückliste arbeiten
Set MyDocNew = ThisWorkbook.ActiveSheet
MyDocNew.Activate
MyDocNew.Cells.Clear
Call Stückliste_Formblatt

'Übernahme der Daten
For i = 1 To Max
   For j = 1 To 6
      MyDocNew.Cells(i + 2, j) = MyDoc.Cells(i + 2, j)
   Next j
Next i

'Auflösung der Struktur
Min = 0
Do
   x = 0
   Max = MyDocNew.UsedRange.Rows.Count - 2
   For i = Min + 1 To Max
      Name = MyDocNew.Cells(i + 2, 5)
      If Not Name = "" Then
         Set MyDoc = ThisWorkbook.Worksheets(Name)
         MaxU = MyDoc.UsedRange.Rows.Count - 2
         For j = 1 To MaxU
            If Not MyDoc.Cells(j + 2, 1) = "" Then
               x = x + 1
               MyDocNew.Rows(i + 2 + x).Select
               Selection.Insert Shift:=xlDown
               MyDocNew.Cells(i + 2 + x, 1) = _
                  MyDocNew.Cells(i + 2, 1) + 1
               For k = 2 To 6
                  MyDocNew.Cells(i + 2 + x, k) = _
                     MyDoc.Cells(j + 2, k)
               Next k
            End If
         Next j
         Min = i
         i = Max
      End If
   Next i
Loop While x > 0
'Struktur ergänzen
Max = MyDocNew.UsedRange.Rows.Count
For i = 3 To Max
   j = Val(MyDocNew.Cells(i, 1))
   If j > 1 Then
      t = ""
      For k = 2 To j
         t = t & "."
      Next k
      t = t & LTrim(Str(j))
      MyDocNew.Cells(i, 1) = t
   End If
```

```
    Next i
End Sub
```

Mit Hilfe dieser Prozeduren, die durch das nachfolgende Menü aufrufbar sind, erstellen wir die Stücklisten für das Beispiel in Abbildung 8-17.

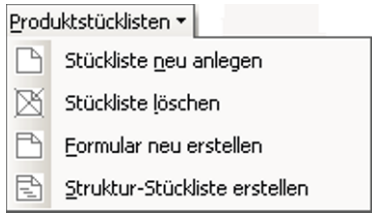


Abbildung 8-18: Menü zur Produktstückliste

Dazu werden die Stücklisten P001 bis P004 neu angelegt und mit den entsprechenden Daten gefüllt.

	A	B	C	D	E	F
1	**P001- Baugruppe**					
2	Stufe	Pos.Nr.	Menge	Bezeichnung	Baugruppe	Material
3	1	1	1	BG1	P002	
4	1	2	1	BG2	P003	
5	1	3	2	ET3		1 m
6	1	4	1	ET4		2 kg

	A	B	C	D	E	F
1	**P002- Baugruppe**					
2	Stufe	Pos.Nr.	Menge	Bezeichnung	Baugruppe	Material
3	1	1	1	ET1		1 kg
4	1	2	2	ET2		1,5 m

	A	B	C	D	E	F
1	**P003- Baugruppe**					
2	Stufe	Pos.Nr.	Menge	Bezeichnung	Baugruppe	Material
3	1	1	2	ET1		1 kg
4	1	2	3	ET3		2 kg
5	1	3	1	BG3	P004	

	A	B	C	D	E	F
1	**P004- Baugruppe**					
2	Stufe	Pos.Nr.	Menge	Bezeichnung	Baugruppe	Material
3	1	1	1	ET1		1 kg
4	1	2	4	FT1		Best.Nr. 12345
5	1	3	2	ET4		1 m

Abbildung 8-19: Baugruppen-Stücklisten zum Beispiel

Die Prozedur *Stückliste_Struktur* erzeugt aus diesen dann eine Strukturstückliste.

	A	B	C	D	E	F
1	**S-P001- Baugruppe**					
2	Stufe	Pos.Nr.	Menge	Bezeichnung	Baugruppe	Material
3	1	1	1	BG1	P002	
4	.2	1	1	ET1		1 kg
5	.2	2	2	ET2		1,5 m
6	1	2	1	BG2	P003	
7	.2	1	2	ET1		1 kg
8	.2	2	3	ET3		2 kg
9	.2	3	1	BG3	P004	
10	..3	1	1	ET1		1 kg
11	..3	2	4	FT1		Best.Nr. 12345
12	..3	3	2	ET4		1 m
13	1	3	2	ET3		1 m
14	1	4	1	ET4		2 kg

Abbildung 8-20: Struktur-Stückliste zum Beispiel

Zur Erzeugung aktiviert man die entsprechende Baugruppen-Stückliste (hier P001) und ruft dann den Menüpunkt Struktur-Stückliste erstellen auf. Die Strukturstückliste erhält den gleichen Namen wie die Baugruppenstückliste, lediglich ein S- wird vorangestellt.

Übung 8-5: Rezepturen

Schreiben Sie Prozeduren zur Verwaltung von Rezepturen. Anders als Stücklisten dienen sie zur Herstellung von Produkten, die aus Mischungen, Mischungsreihenfolgen und deren Verarbeitung entstehen.

Übung 8-6: Arbeitspläne

Genauso wichtig wie Stücklisten sind Arbeitspläne. Verwalten Sie Arbeitspläne zu vorhandenen Stücklisten.

8.5 Arbeiten auf Baumstrukturen und Graphen

Die Graphentheorie ist eigentlich eine mathematische Disziplin. Allerdings ist sie auch ein wichtiges Werkzeug der Informatik, um bestimmte Zusammenhänge zu beschreiben. Ein besonderer Vorteil für den Anwender ist, dass sich die Sachverhalte leicht graphisch darstellen lassen.

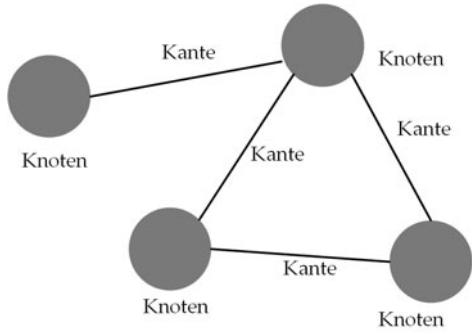


Abbildung 8-21: Beispiel eines Graphen

Ein einfacher ungerichteter Graph besteht aus Knoten und Kanten. Die Kanten verbinden die Knoten. Sind die Kanten gerichtet (Pfeil), so spricht man von einem gerichteten Graphen. Haben die Kanten noch Wertigkeiten, wie z. B. die Entfernungen von Orten untereinander, so spricht man von einem gewichteten Graphen, ansonsten von ungewichteten Graphen.

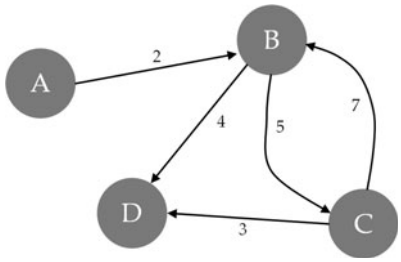


Abbildung 8-22: Beispiel eines gerichteten und gewichteten Graphen

Eine spezielle Form von Graphen werden auch Bäume genannt.

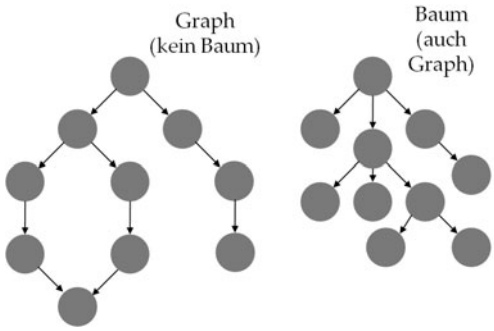


Abbildung 8-23: Vergleich Graph und Baum

Aus der Abbildung 8-23 geht hervor, welche Eigenschaft Bäume haben. Alle Wege gehen von einem Knoten, der Wurzel aus. Vorgänger heißen die Knoten auf dem Weg zur Wurzel und Nachfolger die davon weg. Der unmittelbare Vorgänger heißt Vater und alle Knoten mit demselben Vater heißen Brüder. Ein Knoten ohne Sohn heißt Blatt und mit mindestens einem Sohn innerer Knoten.

Einen typischen Baum haben wir im vorangegangenen Kapitel kennen gelernt. In Abbildung 8-17 wird die Produktstruktur als Baum dargestellt. Die Betrachtung dieses Sachverhalts überlasse ich gerne dem Leser. Ein Beispiel für das Suchen in Bäumen finden Sie im Kapitel 9.3.

Einen weiteren gerichteten Graphen in dem Zusammenhang erhält man, wenn man für ein Produkt den Gozintographen erstellt. Dieser Graph beschreibt, aus welchen verschiedenen Teilen ein Produkt besteht. Die Teile bilden die Knoten und die Kanten geben als Gewichtung die Stückzahl des Einzelteils für ein Produkt an.

Der Name des Graphen ist auf den Mathematiker Andrew Vazsonyi zurückzuführen, der als Vater des Graphen den Mathematiker Zepartzat Gozinto nannte. Dieser Name ist jedoch eine Verballhornung des Begriffs „The Part that goes into“.

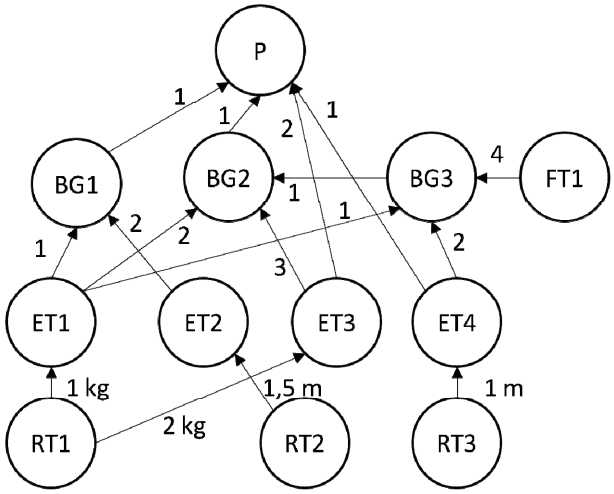


Abbildung 8-24: Gozintograph zum Stücklistenbeispiel

Weitere Betrachtungen überlasse ich dem Leser und wir wollen uns einem klassischen Beispiel für die Anwendung von Graphen zuwenden, der Netzplantechnik.

Beispiel 8-5: Netzplantechnik

Die Netzplantechnik unterstützt und kontrolliert die Durchführung von Projekten. Weiterentwickelte Systeme erlauben sogar die Kapazitäts- und Kostenplanung. Diese einfache Graphenmethode ist im Projektmanagement die Erfolgsmethode schlechthin.

Netzpläne sind gerichtete und gewichtete Graphen, die nach bestimmten Regeln erstellt werden. Die zwei bekanntesten Techniken heißen PERT (Program Evaluation an Review Technic) und CPM (Critical Path Method). Nicht nur, dass Netzpläne den Ablauf von Projekten anschaulich darstellen, sie zwingen auch zum Durchdenken des Projekts. Außerdem fördern sie die Kommunikation und das Verständnis für alle Beteiligten.

Wir betrachten nachfolgend die Methode PERT. Knoten sind darin Ereignisse und Kanten Aktivitäten.

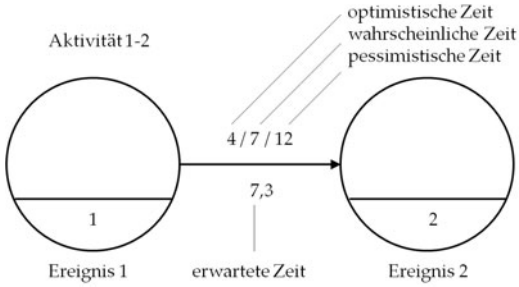


Abbildung 8-25: Elemente in PERT

Ereignisse (Events) sind genau definierte Zeitpunkte. Dabei kommt dem Anfangs- und Endereignis eine besondere Bedeutung zu. Aktivitäten (Activitys) liegen immer zwischen zwei Ereignissen und verbrauchen Zeit. Sie werden als Pfeil dargestellt. Scheinaktivitäten sind Hilfsgrößen zur besseren Darstellung des Sachverhalts. Sie verbrauchen keine Zeit.

Für die Dauer einer Aktivität werden drei Zeitschätzungen erstellt. Eine optimistische, eine wahrscheinliche und eine pessimistische Zeitschätzung. Für die optimistische Zeit t_o gilt, eine kürzere Zeit ist nicht möglich und auch nicht denkbar. Die pessimistische Zeit t_p darf auf keinen Fall überschritten werden. Die wahrscheinliche Zeit t_w gibt die Zeit an, die man angeben würde, wäre nur eine Zeitangabe erlaubt.

Die erwartete Zeit berechnet sich aus der Formel

$$t_e = \frac{t_o + 4 \cdot t_w + t_p}{6} . \tag{8.5.1}$$

Die erwartete Zeit TE für eine Aktivität ist die Summe aller t_e vom Startereignis an

$$T_E = \sum_{i=1}^{j} t_{ei} . \tag{8.5.2}$$

Die spätmöglichste Zeit TL zu der eine Aktivität abgeschlossen sein muss, ist die Summe aller t_e vom Endergebnis bis zur betrachteten Aktivität

$$T_L = \sum_{i=j}^{n} t_{ei} \quad . \tag{8.5.3}$$

Als Schlupf bezeichnet man die Differenz

$$S = T_L - T_E \quad . \tag{8.5.4}$$

Die Varianz, ist ein Maß für die Bewertung der der Unsicherheit bei der Angabe der Vorgangsdauer und bestimmt sich aus

$$\delta^2 = \sum_{i} \left(\frac{t_p - t_o}{6} \right)^2 \tag{8.5.5}$$

Als Standardabweichung δ wird die Wurzel der Varianz bezeichnet. Sie ist ein Maß für die Größe der Abweichung.

Jede Aktivität beginnt mit einem Ereignis und endet mit einem Ereignis. Die Ausnahme bilden das Start- und Endereignis. Eine Aktivität kann erst beginnen, wenn das vorherige Ereignis erzielt wurde.

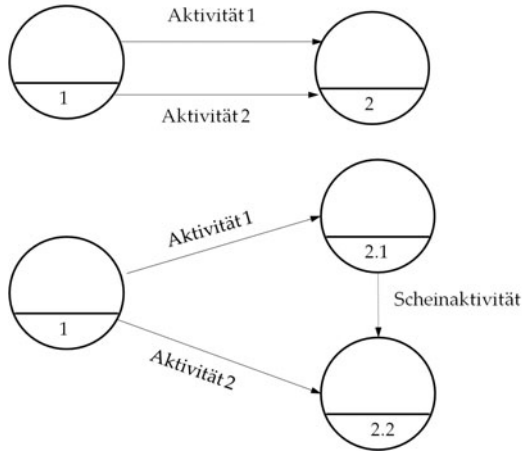


Abbildung 8-26: Scheinaktivität

Parallele Aktivitäten, die von einem Ereignis ausgehen und im gleichen Folgeereignis enden, werden mit Hilfe von Scheinaktivitäten aufgelöst. Scheinaktivitäten benötigen keine Zeit.

Besonders zu beachten ist, dass in einem PERT Netzplan keine Schleifen entstehen.

Als konkretes Beispiel betrachten wir die Daten in der nachfolgenden Tabelle 8-8. Es spielt für die Programmierung keine Rolle, um welche Ereignisse und Aktivitäten es sich handelt. Lediglich die entsprechenden Daten eines Projektes sind für uns von Wichtigkeit.

Die Tätigkeit Nr. 5 ist eine Scheintätigkeit und erhält somit keine Zeiten. Mit Hilfe dieser Daten lässt sich der Netzplan bereits zeichnen. Siehe Abbildung 8-27. Ein Ergebnis ist bereits eingetragen. Die stärker gezeichneten Pfeile kennzeichnen den kritischen Pfad. Der kritische Pfad oder Weg ist nach DIN 69900 der Weg in einem Netzplan vom Anfangs- zum Endknoten, bei dem die Pufferzeiten minimal sind.

Tabelle 8-8: Beispiel von Netzplandaten

Nr	Tätigkeit	Zeit opti.	Zeit wahr.	Zeit pess.
1	1-2	2	3	5
2	1–3	3	4	5
3	1–4	4	6	9
4	2–5	2	4	7
5	3–6	0	0	0
6	4–6	3	5	8,5
7	5-7	1,5	4	7
8	6–7	1	3	7
9	7–8	2	5	9

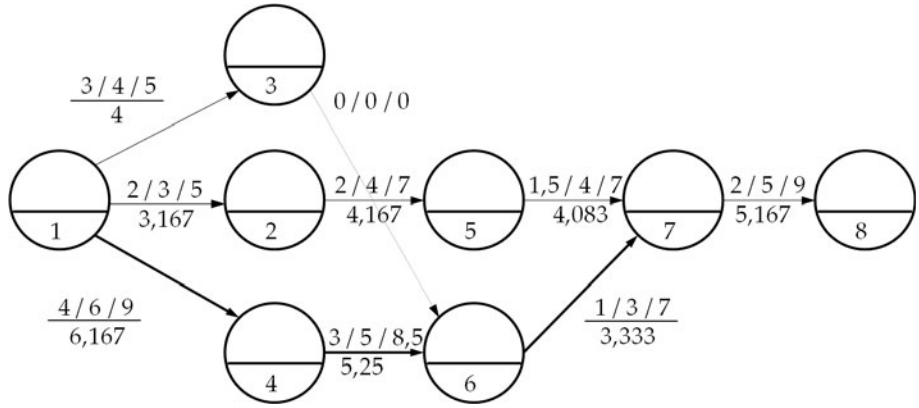


Abbildung 8-27: Netzplan zum Beispiel

Nachdem wir nun alle Begriffe eingeführt haben und bereits an einem Beispiel die Begriffe geübt haben, kommen wir nun zum Algorithmus.

Tabelle 8-9: PERT-Netzplan

	i = 1 (1) Max (über alle Einträge)	
	Bestimme die Knoten a(i,1) und a(i,2)	
	Erwartete Zeit $t_e = \frac{t_o + 4 \cdot t_w + t_p}{6}$ = e(i,1)	
	Varianzanteil $\delta^2 = \left(\frac{t_p - t_o}{6}\right)^2$ = e(i,2)	
	S(i)=0	
	F(i)=0	
	finde früheste Startzeiten i = 1 (1) Max	
	Ist S(a(i,2))<S(a(I,1))+e(i,1)	
	Ja	Nein
	S(a(i,2))=S(a(i,1))+e(i,1)	./.
	finde späteste Endzeiten i = Max (-1) 1	
	Ist F(a(i,2))=0 oder F(a(i,1))>F(a(I,2)-e(i,1)	
	Ja	Nein
	F(a(i,1))=F(a(i,2)-e(i,1)	./.
L=0		
V=0		
	i = 1 (1) Max	

<table>
<tr><td></td><td colspan="3">Schlupf
SL = F(a(i,2))-S(a(i,1))-e(i,1)</td></tr>
<tr><td></td><td colspan="3">Ist SL<=0</td></tr>
<tr><td></td><td>Ja</td><td colspan="2">Nein</td></tr>
<tr><td></td><td>Ausgabe: Kritisch!</td><td colspan="2">./.</td></tr>
<tr><td></td><td colspan="3">Ausgabe: Erwartete Dauer e(i,1)</td></tr>
<tr><td></td><td colspan="3">Ausgabe: Abweichung √(e(i,2))</td></tr>
<tr><td></td><td colspan="3">Ist SL<=0</td></tr>
<tr><td></td><td colspan="2">Ja</td><td>Nein</td></tr>
<tr><td></td><td colspan="2">Ausgabe:
Spätestens starten S(a(i,1))</td><td>Ausgabe:
Frühester Start S(a(i,1))</td></tr>
<tr><td></td><td colspan="2">Ausgabe:
Spätestens enden F(a(i,2))</td><td>Ausgabe:
Spätester Start F(a(i,2))-e(i,1)</td></tr>
<tr><td></td><td colspan="2">Ist L<F(a(i,2))</td><td>Ausgabe:
Frühestes Ende S(a(i,1))+e(i,1)</td></tr>
<tr><td></td><td>Ja</td><td>Nein</td><td>Ausgabe:
Spätestes Ende F(a(i,2))</td></tr>
<tr><td></td><td>L=F(a(i,2)</td><td>./.</td><td rowspan="2">Ausgabe:
Schlupf SL</td></tr>
<tr><td></td><td colspan="2">V=V+e(i,2)</td></tr>
<tr><td colspan="4">Ausgabe: Kritischer Pfad L</td></tr>
<tr><td colspan="4">Ausgabe: Abweichung √(V)</td></tr>
</table>

Für die Programmausführung benötigen wir ein Formblatt, das Testbeispiel und die Auswertung.

Code 8-7: PERT-Netzplan

```
Sub Netzplan_Neu()
   ThisWorkbook.Worksheets("Netzplan").Cells.Clear
   Range("A1") = "Tätigkeiten"
   Range("B1") = "Optimis-" & vbLf & "tische Zeit"
   Range("C1") = "Wahrschein-" & vbLf & "liche Zeit"
   Range("D1") = "Pessimis-" & vbLf & "tische Zeit"
   Range("F1") = "Ereignis"
   Range("G1") = "Erwartete" & vbLf & "Dauer"
   Range("H1") = "Abweichung"
   Range("I1") = "Spätestens" & vbLf & "starten um"
   Range("J1") = "Spätestens" & vbLf & "enden um"
   Range("K1") = "Frühester" & vbLf & "Start um"
   Range("L1") = "Spätester" & vbLf & "Start um"
   Range("M1") = "Frühestes" & vbLf & "Ende um"
   Range("N1") = "Spätestes" & vbLf & "Ende um"
   Range("O1") = "Schlupf"
   Range("P1") = "Kritischer" & vbLf & "Pfad"
   Range("Q1") = "Abweichung"
   Range("A1:Q1").Select

   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Columns("A:A").Select
```

```
    Selection.NumberFormat = "@"
    Columns("B:D").Select
    Selection.NumberFormat = "#0.0"
    Columns("G:Q").Select
    Selection.NumberFormat = "#0.000"
    Range("A2").Select
End Sub

Sub Netzplan_Test()
    Cells(2, 1) = "1-2"
    Cells(3, 1) = "1-3"
    Cells(4, 1) = "1-4"
    Cells(5, 1) = "2-5"
    Cells(6, 1) = "3-6"
    Cells(7, 1) = "4-6"
    Cells(8, 1) = "5-7"
    Cells(9, 1) = "6-7"
    Cells(10, 1) = "7-8"
    Cells(2, 2) = 2
    Cells(3, 2) = 3
    Cells(4, 2) = 4
    Cells(5, 2) = 2
    Cells(6, 2) = 0
    Cells(7, 2) = 3
    Cells(8, 2) = 1.5
    Cells(9, 2) = 1
    Cells(10, 2) = 2
    Cells(2, 3) = 3
    Cells(3, 3) = 4
    Cells(4, 3) = 6
    Cells(5, 3) = 4
    Cells(6, 3) = 0
    Cells(7, 3) = 5
    Cells(8, 3) = 4
    Cells(9, 3) = 3
    Cells(10, 3) = 5
    Cells(2, 4) = 5
    Cells(3, 4) = 5
    Cells(4, 4) = 9
    Cells(5, 4) = 7
    Cells(6, 4) = 0
    Cells(7, 4) = 8.5
    Cells(8, 4) = 7
    Cells(9, 4) = 7
    Cells(10, 4) = 9
End Sub

Sub Netzplan_Start()
    Dim MyDoc As Object
    Dim i, Max As Integer
    Dim tx, e1, e2 As String
    Dim S(), F(), a(), e() As Double
    Dim t1, t2, t3 As Double
    Dim V, L, SL As Double

    Set MyDoc = ThisWorkbook.ActiveSheet
    MyDoc.Activate
```

```
Max = MyDoc.UsedRange.Rows.Count - 1
If Max > 0 Then
   ReDim S(Max), F(Max), a(Max, 2), e(Max, 2)
   For i = 1 To Max
      tx = Cells(i + 1, 1)
      e1 = Left(tx, InStr(tx, "-") - 1)
      e2 = Right(tx, InStr(tx, "-") - 1)
      a(i, 1) = Val(e1)
      a(i, 2) = Val(e2)
      t1 = CDbl(Cells(i + 1, 2))
      t2 = CDbl(Cells(i + 1, 3))
      t3 = CDbl(Cells(i + 1, 4))
      e(i, 1) = (t1 + 4 * t2 + t3) / 6 'erwartete Dauer
      e(i, 2) = ((t3 - t1) / 6) ^ 2    'Varianz
      S(i) = 0
      F(i) = 0
   Next i
   'Auffinden der frühesten Startzeiten
   For i = 1 To Max
      If S(a(i, 2)) < S(a(i, 1)) + e(i, 1) Then
         S(a(i, 2)) = S(a(i, 1)) + e(i, 1)
      End If
   Next i
   F(a(Max, 2)) = S(a(Max, 2))
   'Auffinden der spätesten Endzeiten
   For i = Max To 1 Step -1
      If F(a(i, 1)) = 0 Or _
         F(a(i, 1)) > F(a(i, 2)) - e(i, 1) Then
         F(a(i, 1)) = F(a(i, 2)) - e(i, 1)
      End If
   Next i

   L = 0
   V = 0
   For i = 1 To Max
      SL = F(a(i, 2)) - S(a(i, 1)) - e(i, 1)
      If SL <= 0.1 Then
         Cells(i + 1, 6) = "Kritisch!"
      End If
      Cells(i + 1, 7) = e(i, 1)
      Cells(i + 1, 8) = Sqr(e(i, 2))
      If SL <= 0.1 Then
         Cells(i + 1, 9) = S(a(i, 1))
         Cells(i + 1, 10) = F(a(i, 2))
         'Aufsummierung von Pfadlänge und Varianz
         If L < F(a(i, 2)) Then L = F(a(i, 2))
         V = V + e(i, 2)
      Else
         Cells(i + 1, 11) = S(a(i, 1))
         Cells(i + 1, 12) = F(a(i, 2)) - e(i, 1)
         Cells(i + 1, 13) = S(a(i, 1)) + e(i, 1)
         Cells(i + 1, 14) = F(a(i, 2))
         Cells(i + 1, 15) = SL
      End If
   Next i
End If
```

```
    Cells(2, 16) = L
    Cells(2, 17) = Sqr(V)
End Sub
```

Aufgerufen werden die Prozeduren wieder durch entsprechende Menüpunkte.

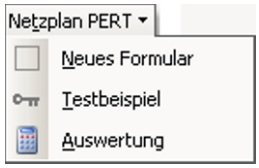


Abbildung 8-28: Menü zum Netzplan PERT

Die Ausführung des Testbeispiels liefert den kritischen Pfad.

	A	B	C	D	E	F	G	H
1	*Tätigkeiten*	*Optimis-tische Zeit*	*Wahrschein-liche Zeit*	*Pessimis-tische Zeit*		*Ereignis*	*Erwartete Dauer*	*Abweichung*
2	1-2	2,0	3,0	5,0			3,167	0,500
3	1-3	3,0	4,0	5,0			4,000	0,333
4	1-4	4,0	6,0	9,0		Kritisch!	6,167	0,833
5	2-5	2,0	4,0	7,0			4,167	0,833
6	3-6	0,0	0,0	0,0			0,000	0,000
7	4-6	3,0	5,0	8,5		Kritisch!	5,250	0,917
8	5-7	1,5	4,0	7,0			4,083	0,917
9	6-7	1,0	3,0	7,0		Kritisch!	3,333	1,000
10	7-8	2,0	5,0	9,0		Kritisch!	5,167	1,167

Abbildung 8-29: Teil 1 – Eingabedaten und Auswertungsteil

	A	I	J	K	L	M	N	O	P	Q
1	*Tätigkeiten*	*Spätestens starten um*	*Spätestens enden um*	*Frühester Start um*	*Spätester Start um*	*Frühestes Ende um*	*Spätestes Ende um*	*Schlupf*	*Kritischer Pfad*	*Abweichung*
2	1-2			0,000	3,333	3,167	6,500	3,333	19,917	1,974
3	1-3			0,000	7,417	4,000	11,417	7,417		
4	1-4	0,000	6,167							
5	2-5			3,167	6,500	7,333	10,667	3,333		
6	3-6			4,000	11,417	4,000	11,417	7,417		
7	4-6	6,167	11,417							
8	5-7			7,333	10,667	11,417	14,750	3,333		
9	6-7	11,417	14,750							
10	7-8	14,750	19,917							
11										

Abbildung 8-30: Teil 2 – restliche Auswertung

Übung 8-7: Strukturstückliste

Erstellen Sie ein Programm, das nach der Strukturstückliste des vorangegangenen Kapitels einen Baum erstellt.

Grundsätzlich ist die Strukturstückliste eine Form, die auf der untersten Ebene nur Rohstoffe aufweist, jedenfalls theoretisch. Praktisch gehören auch Zukaufteile zum Produkt. Ein Problem ist die Ermittlung des Materialbedarfs, wenn gleiche Materialien auf unterschiedlichen Stufen vorkommen. Schreiben Sie dafür ebenfalls eine Prozedur.

Erweitern Sie ihre Strukturstückliste um die Maße und Dichte der Bauteile und bestimmen sie das Gesamtgewicht. Erweitern Sie ihre Strukturstückliste um die Kosten der Materialien und Zukaufsteile und bestimmen sie die einzelnen Summen. Erstellen Sie außerdem ein Programm, das ebenfalls zur Strukturstückliste einen Gozintographen erstellt.

9 Verhaltens-Algorithmen

9.1 Das Prinzip Teile und Herrsche

Die Strategie „Teile und Herrsche“ (lat.: divide et impera) besteht darin, ein gegebenes Problem in verschiedene Teilprobleme aufzuspalten. Diese Teilprobleme müssen gelöst werden. Dann muss eine Methode gefunden werden, die Teillösungen zu einer Gesamtlösung zusammen zu fügen. Sind die Teilprobleme immer noch relativ groß, dann muss die „Teile und Herrsche“-Strategie möglicherweise noch einmal angewendet werden. Oft sind die entstandenen Teilprobleme vom gleichen Typ wie das Hauptproblem. In solchen Fällen formuliert man die wiederholte Anwendung des „Teile und Herrsche“-Prinzips als rekursive Prozedur. Damit werden immer kleinere Teilprobleme erzeugt die letztlich nicht weiter zergliedert werden müssen.

Beispiel 9-1: Suchen nach der Bisektionsmethode

Bei der Bisektionsmethode geht man von geordneten Datenstrukturen aus. Als Beispiel wählen wir das Inhaltsverzeichnis dieses Buches. Es soll als geordnete Liste auf einem Tabellenblatt stehen. Diese Liste kann noch durch weitere Buchinhalte ergänzt werden, so dass ein brauchbares Suchverzeichnis entsteht. Jeder benutzt eine überschaubare Menge Fachbücher, die so effektiver genutzt werden können.

Die Methode ist auch vom Suchen in Telefonbüchern her bekannt und man geht wie folgt vor. Zunächst wird in der Mitte des Telefonbuches nachgesehen. Dann wird bestimmt, in welcher der beiden Hälften des Buches der Begriff nun liegt. Diese Hälfte wird wieder in der Mitte unterteilt und man geht für diese Hälfte genauso vor. So gelangt man zu immer kleineren Datenmengen, bis letztlich ein Gefunden oder Nichtvorhanden als Ergebnis vorliegt.

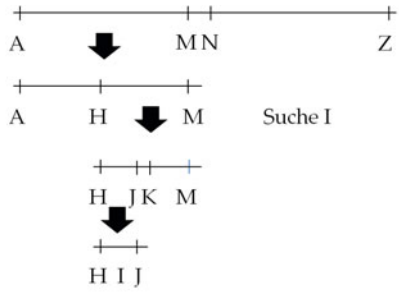


Abbildung 9-1: Schema einer Bisektionsmethode

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_9,

Der Algorithmus besteht im Prinzip aus der Teilung von Datenstrukturen derart, dass immer das zu suchende Element in der Teilmenge ist. Dies muss zwangsläufig zum zu suchenden Element führen.

Tabelle 9-1: Suchen nach der Bisektionsmethode

Eingabe des Suchbegriffs		
v=1		
b=Anzahl Einträge		
So lange b>=v		
$m = Int(\frac{v+b}{2})$		
Ist t > Listeneintrag in Zeile m		
Ja	Nein	
v=m+1	b=m-1	
i=v		
Ausgabe Listeneintrag in Zeile i		

Der Algorithmus ist genauso leicht zu programmieren wie es das Struktogramm darstellt. Das einzige Problem ist eine sichere Terminierung des Suchprozesses. Und hier genügt in der Tat die Abfrage ob b>=v.

Code 9-1: Suchen mittels Bisektionsmethode in einer geordneten Liste

```
Option Explicit

Sub SuchListe()
   Dim myDoc As Object
   Dim t As String
   Dim v, b, m, i As Long

   Set myDoc = ThisWorkbook.Worksheets("Suchliste")

   'Eingabe
   t = InputBox("Suchbegriff angeben!")

   'Start-Suchbereich
   'von 1 bis Anzahl Einträge
   v = 1
   b = myDoc.UsedRange.Rows.Count

   'Suchschleife
   Do While b >= v
      m = Int((v + b) / 2)
      If t > Cells(m, 1) Then
         v = m + 1
      Else
         b = m - 1
      End If
      i = v
   Loop
```

```
    'Ausgabe
    Cells(i, 1).Activate
    MsgBox "Zeile:" & Str(i) & vbLf & Cells(i, 1), _
       vbOKOnly, "Suchergebnis von: " & t
End Sub
```

Zum Aufruf der Suchfunktion genügt ein einziger Menüaufruf.

Abbildung 9-2: Menü zur Suchprozedur

Die Prozedur liefert immer eine Ausgabe, egal ob der Suchbegriff gefunden wurde oder nicht.

Übung 9-1: Hash-Methode

Will man eine solche Liste noch pflegen, dann kommen zwei weitere Prozeduren hinzu, nämlich das Einfügen und Entfernen von Einträgen. Schreiben Sie diese Prozeduren und benutzen Sie ein Formular zur Verwaltung.

Nicht immer liegen Listen in geordneter Form vor. Eine sehr effiziente Methode, in diesen Listen Informationen zu finden, ist die Hash-Methode. Die Methode wurde in den fünfziger Jahren bei IBM unter dem Begriff gestreute Speicherung entwickelt. Befassen Sie sich mit dem Thema und schreiben Sie eine entsprechende Prozedur.

9.2 Die Greedy-Methode

Die Greedy-Methode (engl.: greedy = gierig, geizig) beruht auf einer einfachen Entwurfstechnik. Außerdem kann sie auf eine Vielzahl von Problemen angewandt werden. Fast alle diese Probleme erfordern die Bestimmung einer Teilmenge, die bestimmten Bedingungen genügt. Diese Teilmengen nennt man dann eine mögliche Lösung. Ziel ist es dann eine Teillösung zu finden, die eine gegebene Zielfunktion maximiert oder minimiert. Dies ist dann eine optimale Lösung. Gewöhnlich gibt es klare Anweisungen zur Bestimmung einer Lösung, aber nicht notwendiger weise einer optimalen Lösung.

Beispiel 9-2: Auftragsfolgenproblem

Gegeben sei eine Menge von n Aufträgen a_i. Jeder Auftrag hat einen Endtermin $e_i >= 0$ und einen Gewinn $g_i >= 0$. Dabei wird der Gewinn nur erzielt, wenn der Auftrag bis zum Endtermin ausgeführt ist.

Gehen wir weiter davon aus, dass für alle Aufträge nur eine Maschine zur Verfügung steht, und jeder Auftrag diese Maschinen für eine bestimmte Zeiteinheit z_i belegt.

Eine Lösung dieses Problems ist eine Teilmenge der Aufträge, die alle bis zum Ende erledigt werden können. Der Wert dieser Lösung w_j ist die Summe der Gewinne

$$w_j = \sum g_i \quad . \tag{9.2.1}$$

Folglich ist eine Lösung mit maximalem Wert eine optimale Lösung.

Eine Greedy-Methode ist es nun, ein Optimierungsmaß für die Zusammenstellung der Gewinne zu finden. Ausgehend von einem beliebigen Auftrag a_i, werden immer nur die Gewinne hinzuaddiert, die den größtmöglichen Gewinn versprechen und dabei den Endtermin einhalten. Aus den so gewonnenen möglichen Lösungen, wird die beste ausgewählt. Sie ist allerdings kein Garant für die optimale Lösung.

Betrachten wir dies an den konkreten Zahlen eines Beispiels in Tabelle 9-2. Neben dem Wert und der belegten Zeit für die Maschinen ist außerdem die Zeit angegeben, die bis zur Erledigung des Auftrags verbleibt. Zur Vereinfachung betrachten wir die Zeiten in Stunden. Zur Ermittlung gehen wir von allen Aufträgen aus und suchen die dazu passenden Folgeaufträge nach der zuvor erläuterten Greedy-Strategie. Diese ist in Tabelle 9-3 wiedergegeben.

Tabelle 9-2: Beispieldaten für Aufträge

Auf. Nr.	Wert	Beleg. Zeit [h]	Zeitraum zur Erl. [h]
1	73	12	30
2	61	10	40
3	55	8	50
4	12	11	30
5	48	14	35
6	33	10	45

Tabelle 9-3: Algorithmus zur Greedy-Methode

Eingabe der Auftragswerte, Belegzeiten und des Zeitraums für die Ausführung			
	Für alle vorhandenen Aufträge i = 1 (1) n		
		Zeitsummierung Z=z(i)	
		Wertsummierung W=w(i)	
			Für alle verbleibenden Aufträge
			Suche größten Wert w(j)
			Ist Z+z(j) innerhalb der zulässigen Zeit
			Ja / Nein
			Z=Z+z(j) / ./.
			W=W+w(j)
Bestimme die Kombination mit dem größten Gewinn			

Das Programm bauen wir wieder wie gewohnt auf, so dass erst ein Formblatt erstellt wird. Dann können wahlweise eigene Daten eingegeben oder die Testdaten aufgerufen werden.

Code 9-2: Suchen einer Lösung für ein Aufgabenfolgeproblem nach der Greedy-Methode

```
Option Explicit

Sub Greedy_Neu()
   ThisWorkbook.Worksheets("Greedy").Cells.Clear

   Range("A1") = "Auf.Nr."
   Range("B1") = "Wert"
   Range("C1") = "Beleg." & vbLf & "Zeit" & vbLf & "[h]"
   Range("D1") = "Zeitraum" & vbLf & "zur Erl." & _
        vbLf & "[h]"
   Range("F1") = "Auf.Nrn."
   Range("G1") = "Ges.Wert"
   Range("A1:G1").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("E:E").ColumnWidth = 2
   Range("A2").Select
End Sub

Sub Greedy_Test()
   Dim i As Integer

   For i = 1 To 6
      Cells(i + 1, 1) = i
   Next i
   Cells(2, 2) = 73
   Cells(3, 2) = 61
   Cells(4, 2) = 55
   Cells(5, 2) = 12
   Cells(6, 2) = 48
   Cells(7, 2) = 33
   Cells(2, 3) = 12
   Cells(3, 3) = 10
   Cells(4, 3) = 8
   Cells(5, 3) = 11
   Cells(6, 3) = 14
   Cells(7, 3) = 10
   Cells(2, 4) = 30
   Cells(3, 4) = 40
   Cells(4, 4) = 50
   Cells(5, 4) = 30
   Cells(6, 4) = 35
   Cells(7, 4) = 45
End Sub

Sub Greedy_Start()
   Dim MyDoc As Object
   Dim W, Z, wi, zi, ei As Double
   Dim i, j, k, l, wMax, kMax, n As Integer
```

```
Dim a() As Integer
Dim t As String

Set MyDoc = ThisWorkbook.Worksheets("Greedy")
n = MyDoc.UsedRange.Rows.Count - 1
ReDim a(n, 4)

'Daten übernehmen
For i = 1 To n
   a(i, 2) = Cells(i + 1, 2)
   a(i, 3) = Cells(i + 1, 3)
   a(i, 4) = Cells(i + 1, 4)
Next i

For i = 1 To n
   'Merker löschen
   For j = 1 To n
      a(j, 1) = 0
   Next j
   'Ersten Wert setzen
   a(i, 1) = 1

   'Sortierung der Werte
   For j = 2 To n
      'grösster freier Wert
      W = 0
      l = 0
      For k = 1 To n
         If a(k, 1) = 0 Then
            If a(k, 2) > W Then
               W = a(k, 2)
               l = k
            End If
         End If
      Next k
      a(l, 1) = j
   Next j

   'Prüfung der Zulässigkeit und Ausgabe
   Z = 0
   W = 0
   t = ""
   For j = 1 To n
      For k = 1 To n
         If a(k, 1) = j Then
            If Z + a(k, 3) <= a(k, 4) Then
               Z = Z + a(k, 3)
               W = W + a(k, 2)
               If j = 1 Then
                  t = LTrim(Str(k))
               Else
                  t = t & "-" & LTrim(Str(k))
               End If
            End If
            k = n
         End If
      Next k
```

```
        Next j
        Cells(i + 1, 6) = t
        Cells(i + 1, 7) = W
    Next i
End Sub
```

Der Aufruf der Prozeduren über das Menü in Abbildung 9-3 liefert die in Abbildung 9-4 dargestellten Daten.

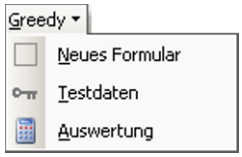


Abbildung 9-3: Menü zur Greedy-Methode

Danach ist die Lösung die Aufgabenfolge 5-1-2-3 mit einem Gesamtwert von 237. Auch für dieses Beispiel gilt, dass es nicht die optimale Lösung sein muss. Die Greedy-Methode liefert jedoch oft ein schnelles und hinreichend gutes Ergebnis.

	A	B	C	D	E	F	G
1	*Auf.Nr.*	*Wert*	*Beleg. Zeit [h]*	*Zeitraum zur Erl. [h]*		*Auf.Nrn.*	*Ges.Wert*
2	1	73	12	30		1-2-3-6	222
3	2	61	10	40		2-1-3-6	222
4	3	55	8	50		3-1-2-6	222
5	4	12	11	30		4-1-2-3	201
6	5	48	14	35		5-1-2-3	237
7	6	33	10	45		6-1-2-3	222

Abbildung 9-4: Ergebnis der Testdaten

Übung 9-2: Kriterien

Wählen Sie statt des Kriteriums Wert, das Kriterium

$$\frac{Wert}{Belegzeit}$$

und vergleichen Sie beide Ergebnisse. Insbesondere, wenn noch die Kosten für die Maschinennutzung dem Gewinn entgegen gesetzt werden.

Ändern Sie den Algorithmus so ab, dass auch Vertauschungen der Reihenfolgen untersucht werden.

9.3 Rückverfolgung oder Backtracking

Die Rückverfolgung stellt ein fundamentales algorithmisches Prinzip dar. Viele Probleme, bei denen man nach einer Menge von Lösungen bzw. nach einer optimalen Lösung sucht, können mit dieser Methode gelöst werden. Der Name Rückverfolgung (engl.: backtracking) schildert im Wort bereits anschaulich die Methode. Der Backtracking-Algorithmus ähnelt dem Suchen in Bäumen.

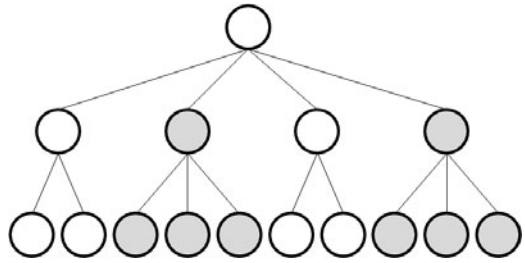


Abbildung 9-5: Baumstruktur

Man durchläuft dabei alle Zweige eines Baumes (Abbildung 9-5) bis in die Spitzen und kehrt dann bis zur vorherigen Abzweigung zurück, die man noch nicht durchlaufen hat. Von diesem Zurückgehen hat der Algorithmus seinen Namen. Beim Durchsuchen der Bäume verfährt man nach einer Strategie. Zum Beispiel orientiert man sich immer nach rechts zu gehen, soweit das möglich ist. Hat man die Spitze erreicht, geht man bis zur ersten Möglichkeit nach links zu gehen, die man noch nicht durchlaufen hat, wieder zurück. Danach orientiert man sich wieder rechts. Wir kennen dieses Verfahren auch zum Finden eines Ausgangs aus einem Labyrinth.

Ein klassisches Beispiel ist das 8-Damen Problem, das ich bereits im Jahre 1978 in der Wissenschaftszeitschrift Bild der Wissenschaft als Programmierbeispiel auf einem Taschenrechner beschrieben habe. Da wird nach den Positionen gesucht, die 8 Damen auf einem Schachbrett einnehmen können, ohne sich gegenseitig zu „schlagen“. Weitere typische Anwendungsbeispiele für den Backtracking-Algorithmus sind die Suche nach einem Ausweg aus einem Labyrinth, angrenzende Flächen mit unterschiedlichen Farben zu versehen und der Weg eines Handelsreisenden. Der Algorithmus spielt ebenfalls in der KI (Künstlichen Intelligenz) eine bedeutende Rolle. Wir wollen nachfolgend eine industrielle Anwendung betrachten.

Beispiel 9-3: Einschrittige Codes für die industrielle Wegmessung

Zur Wegmessung, ob inkremental oder absolut, wird in der Regel der Gray-Code benutzt. Er hat eine Eigenschaft, die der Binärcode nicht besitzt. Beim Wechsel von einer Zahl zur nachfolgenden oder zur vorhergehenden ändert sich nur ein Bit in der Darstellung.

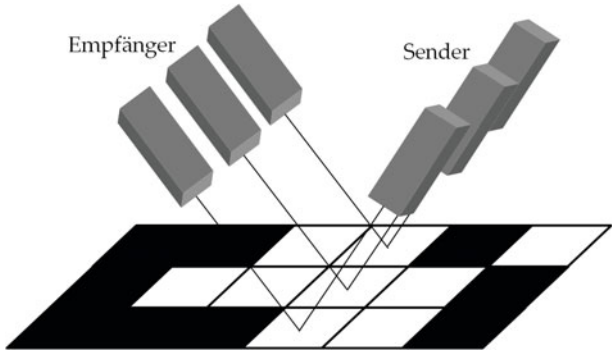


Abbildung 9-6: Industrielle Wegmessung

Nur dadurch, dass sich immer nur ein Bit ändert, sind fehlerfreie Messungen möglich. Würden sich zwei Bits ändern, käme es möglicherweise kurzzeitig zu einem falschen Wert, wenn diese sich nicht wirklich zeitgleich ändern.

Tabelle 9-4: Binär- und Gray-Code

i	Binär-Code	Gray-Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Wir wollen nachfolgend mittels Backtracking-Algorithmus untersuchen, ob es möglicherweise noch andere so genannte einschrittige Codes gibt.

Als Basis nehmen wir 4 Bit und für die Farben Weiß und Schwarz die Ziffern 0 und 1. Eine Ausgangskonstellation sei mit nur Nullen gegeben. Dann ergibt sich die nächste Konstellation durch Veränderung eines Bits (Abbildung 9-5). Ist eine solche Konstellation gefunden, dann beginnt die Veränderung wieder beim ersten Bit. Gibt es die Konstellation bereits, wird das zweite Bit verändert, usw.; so lange bis eine weitere Konstellation gefunden ist. Ausgang ist immer die zuletzt gefundene Konstellation. Die jeweilige Position der Bitveränderung wird registriert, da mit Erreichen der 16. Bitkonstellation oder schon früher, wieder ein Rückwärtsschritt erforderlich ist.

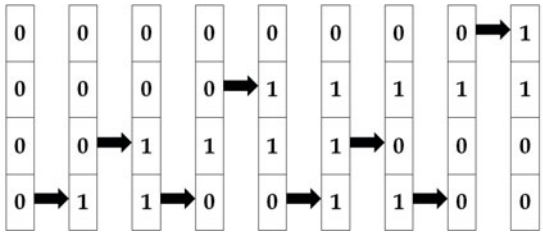


Abbildung 9-7: Erste Schritte des Algorithmus

Ein Rückwärtsschritt ist immer dann erforderlich, wenn mit Veränderung des 4. Bits keine neue Konstellation erreicht wird. Dann wird auf die vorherige gültige Konstellation zurückgegriffen, die noch kein 4. Bit verändert hat.

Damit ist der Algorithmus umfassend beschrieben und wir wollen ihn im nachfolgenden Schritt als Struktogramm definieren.

Tabelle 9-5: Bestimmung einschrittiger Codes nach der Backtracking-Methode

<table>
<tr><td colspan="5">Bestimmung der Ausgangskonfiguration</td></tr>
<tr><td colspan="5">j=1 (1) 4</td></tr>
<tr><td colspan="5">A(1,j)=0</td></tr>
<tr><td colspan="5">Merker auf Null setzen</td></tr>
<tr><td colspan="5">i=1 (1) 16</td></tr>
<tr><td colspan="5">M(i)=0</td></tr>
<tr><td colspan="5">i=0 Start</td></tr>
<tr><td colspan="5">j=1 (1) 4 Schritt vor</td></tr>
<tr><td colspan="5">U(j)=A(i,j)</td></tr>
<tr><td colspan="5">Ist M(i)<4</td></tr>
<tr><td colspan="4">Ja</td><td>Nein</td></tr>
<tr><td colspan="4">Bit erhöhen
M(i)=M(i)+1</td><td rowspan="3">Schritt zurück</td></tr>
<tr><td colspan="4">Bit ändern
U(M(i))=1-U(M(i))</td></tr>
<tr><td colspan="4">Konstellation bereits vorhanden?</td></tr>
<tr><td>Ja</td><td colspan="3">Nein</td><td rowspan="4">M(i)=0</td></tr>
<tr><td></td><td colspan="3">Neue Position
i=i+1</td></tr>
<tr><td></td><td colspan="3">Übernahme
J=1 (1) 4
A(i,j)=U(j)</td></tr>
<tr><td></td><td colspan="3">Ist die 16. Position erreicht?</td></tr>
<tr><td></td><td>Ja</td><td>Nein</td><td rowspan="2">i=i-1</td></tr>
<tr><td></td><td>Ist der Code in sich geschlossen?</td><td>./.</td></tr>
</table>

	Ja	Nein		
	Code anzeigen	./.		
	Schritt zurück			
	M(i)=0			
	i=i-1			
So lange nicht alle Möglichkeiten untersucht sind (i>0)				

Da eine lange Laufzeit zu erwarten ist, habe ich noch ein Abbruchkriterium eingebaut, so dass nach Erreichen einer vorgegebenen Anzahl Konstellationen der Prozess terminiert.

Code 9-3: Bestimmung einschrittiger Codes

```
Option Explicit
Sub EinCodes_Start()
   Dim A(16, 4), U(4), M(16) As Integer
   Dim i, j, k, l, n As Integer
   Dim p1, p2 As Integer
   Dim y, z, s As Integer
   Dim t As String
'Start
   ThisWorkbook.Worksheets("EinCodes").Cells.Clear
   Range("H:W").ColumnWidth = 1
   z = InputBox("Anzahl der Codes bis zum Stop angeben")
   y = 1

'Ausgangskonfiguration 0000
   For j = 1 To 4
      A(1, j) = 0
      Cells(1, j) = 0
   Next j

'Merker Ausgangswerte
   For i = 1 To 16
      M(i) = 0
   Next i

'Start
   i = 1

   Do
'Schritt vor
      For j = 1 To 4
         U(j) = A(i, j)
      Next j
      If M(i) < 4 Then
         M(i) = M(i) + 1
         U(M(i)) = 1 - U(M(i))
'Prüfung
         'Wird die gleiche Konstellation
         'noch einmal gefunden, ist p1=1
         p1 = 0
         For k = 1 To i
            p2 = 0
```

```
            For j = 1 To 4
               If Not U(j) = A(k, j) Then p2 = 1
            Next j
            If p2 = 0 Then p1 = 1
         Next k
         '
         'Neue Konstellation
         If p1 = 0 Then
            i = i + 1
            For j = 1 To 4
               A(i, j) = U(j)
               Cells(i, j) = U(j)
            Next j

            If i = 16 Then
               'Prüfung, ob der Code in sich geschlossen ist
               s = 0
               For j = 1 To 4
                  If Not A(1, j) = A(16, j) Then
                     s = s + 1
                  End If
               Next j
               If s = 1 Then
                  'Code brauchbar und wird registriert
                  For k = 1 To 16
                     t = ""
                     For j = 1 To 4
                        If A(k, j) = 1 Then
                           t = t + ChrW(9608)
                        Else
                           t = t + ChrW(8901)
                        End If
                        If j < 4 Then t = t + vbLf
                     Next j
                     Cells(y, 7 + k) = t
                  Next k
                  y = y + 1

                  'Merkerstand zeigen
                  For k = 1 To 16
                     Cells(k, 6) = M(k)
                  Next k

                  'Abbruch
                  If y = z + 1 Then
                     Exit Sub
                  End If

               End If

               'Schritt zurück
               M(i) = 0
               i = i - 1
            End If
         End If
      Else
         'Schritt zurück
```

```
            M(i) = 0
            i = i - 1
        End If
    Loop While i > 0
End Sub
```

Als Menü wird lediglich ein Start benötigt.

Abbildung 9-8: Menü zum Bestimmungsstart

Die ersten neun gefundenen Codes sehen Sie in Abbildung 9-7. Die Spalten A-D zeigen die vier Bit der 16 Positionen der letzten gefundenen gültigen Konstellation. In Spalte F sehen Sie den aktuellen Merker der 16 Positionen, und damit den Fortgang des Algorithmus. In den Spalten H-V eine grafische Darstellung aller gefundenen Konstellationen.

	A	B	C	D	E	F
1	0	0	0	0		1
2	1	0	0	0		2
3	1	1	0	0		1
4	0	1	0	0		3
5	0	1	1	0		2
6	0	0	1	0		1
7	1	0	1	0		2
8	1	1	1	0		4
9	1	1	1	1		2

Abbildung 9-9: Die ersten neun gefundenen einschrittigen Codes

Übung 9-3: Codeeigenschaften

Ändern Sie den Algorithmus so ab, dass mit ihm die Ziffern 0 bis 9 dargestellt werden können. Es ergeben sich dann zehn Positionen von möglichen sechzehn. Die restlichen sechs Positionen sind überflüssig und werden als Redundanz bezeichnet.

Schreiben Sie zusätzliche Prozeduren, die gefundene Codes z. B. auf Gewichtung oder andere Eigenschaften untersucht.

9.4 Rückwärtsrechnen oder rekursive Prozeduren

Man spricht von einer rekursiven Methode, wenn in einer Prozedur ein Aufruf von ihr selbst steht. So berechnet sich n! aus seinem Vorgänger (n-1)! Durch

$$n! = n \cdot (n-1)! \ . \tag{9.4.1}$$

Damit die Rekursion terminiert, muss ein Rekursionsanfang gegeben sein. Für Null-Fakultät gilt

$$0! = 1 \tag{9.4.2}$$

per Definition.

Ein sehr bekanntes Beispiel für das Rückwärtsrechnen ist das nachfolgend dargestellte Beispiel, in der Literatur auch als Jeep-Problem bekannt.

Beispiel 9-4: Das Jeep-Problem

Ein Fahrer möchte mit seinem Jeep eine Wüste durchqueren. An seinem Ausgangspunkt steht ihm ein unbegrenztes Tanklager zur Verfügung. Der Jeep verbraucht 10 Liter auf 100 km und kann für eine Fahrt immer nur 60 Liter Treibstoff laden. Damit kann der Fahrer 600 km fahren. Der Fahrer ist also gezwungen, in der Wüste ein Depot anzulegen, auf das er bei seiner Durchquerung zurückgreifen kann.

Die Lösung des Problems ergibt sich durch Rückwärtsrechnen, das anschaulich in Abbildung 9-10 dargestellt ist.

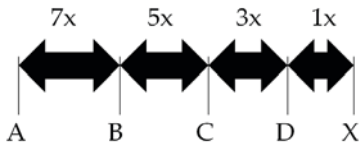


Abbildung 9-10: Die Methode des Rückwärtsrechnens

Um vom letzten Depot D zum Ziel X zu gelangen, ist eine einzige Tankfüllung nötig.

Um eine Tankfüllung vom vorletzten Depot C nach D zu transportieren und dann von D nach C zurückzukehren, muss der Jeep die Strecke CD dreimal fahren. Kommt er das erste Mal nach D kann er 1/3 der Tankladung zurücklassen. Beim zweiten Mal hat er noch 2/3 der Tankladung. C muss also 1/3 der Tankladung von D entfernt sein = 600/3 = 200 km.

Um vom drittletzten Depot B nach C zwei Tankladungen zu bringen, muss der Jeep fünfmal hin- und herfahren. Zweimal kann er eine 3/5 Tankladung deponieren und beim dritten Mal besitzt er noch 4/5 der Tankfüllung. C muss also von B entfernt sein = 600/5 = 120 km.

Allgemein betrachtet ergibt sich

$$600 \cdot \left(1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + ...\right) . \qquad (9.4.3)$$

Diese Summe divergiert und so lässt sich mit dieser Methode jede beliebige Strecke zurücklegen.

Tabelle 9-6: Algorithmus zum Jeep-Problem

Starteingaben v=Verbrauch in Liter/100 km t=Tankfüllung e=Entfernung
∑=0
i=0
i=i+1
$\sum = \sum + \frac{1}{(i-1)\cdot 2+1}$
So lange wie $\frac{t}{v}\sum < e$
Ausgabe der Anzahl Depots

Mit Hilfe variabler Eingabedaten sollen die Anzahl Depots bestimmt werden, die zur Überbrückung einer bestimmten Entfernung notwendig sind.

Code 9-4: Das Jeep-Problem

```
Option Explicit

Sub Jeep_Neu()
   ThisWorkbook.Worksheets("JeepProblem").Cells.Clear

   Range("A1") = "Tankfüllung [L]"
   Range("A2") = "Verbrauch [L/100km]"
   Range("A3") = "Entfernung [km]"
   Range("D1") = "Depot"
   Range("E1") = "Entfernung"
   Range("F1") = "Differenz"
   Range("A1:A3").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("D1:F1").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A:A").ColumnWidth = 20
   Range("C:C").ColumnWidth = 5
   Range("B1").Select
End Sub

Sub Jeep_Test()
   Cells(1, 2) = 80
```

```
   Cells(2, 2) = 10
   Cells(3, 2) = 1500
End Sub


Sub Jeep_Start()
   Dim v, t, e, s, u, w As Double
   Dim i As Integer

   t = Cells(1, 2)
   v = Cells(2, 2)
   e = Cells(3, 2)
   i = 0
   s = 0
   u = 0

   Do
      i = i + 1
      s = s + 1 / ((i - 1) * 2 + 1)
      Cells(i + 1, 4) = i
      w = t / v * 100 * s
      Cells(i + 1, 5) = w
      Cells(i + 1, 6) = w - u
      u = w
   Loop While t / v * 100 * s < e
End Sub
```

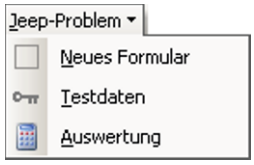


Abbildung 9-11: Menü zum Jeep-Problem

Die programmierten Testdaten liefern die nachfolgende Lösung.

	A	B	C	D	E	F
1	*Tankfüllung [L]*	80		*Depot*	*Entfernung*	*Differenz*
2	*Verbrauch [L/100km]*	10		1	800	800
3	*Entfernung [km]*	1500		2	1066,66667	266,666667
4				3	1226,66667	160
5				4	1340,95238	114,285714
6				5	1429,84127	88,8888889
7				6	1502,56854	72,7272727

Abbildung 9-12: Beispielrechnung zum Jeep-Problem

Übung 9-4: Reservetanks

Erweitern Sie die Berechnung so, dass zusätzliche Reservetanks mitgeführt werden können.

10 Bioalgorithmen

Die Bionik, ein Fachgebiet das die Fachgebiete Biologie und Technik einschließt, befasst sich mit in der Natur zu findender Lösungen, Strukturen und Prinzipien und wie weit sich diese für die Technik umsetzen lassen.

Die Natur bietet eine Fülle von Optimallösungen, von denen die nachfolgenden zwei Beispiele nur einen kleinen Einblick wiedergeben können.

10.1 Der Ameisenalgorithmus

Der Ameisenalgorithmus wurde in der Tat den Ameisen abgeschaut. Es war der italienische Wissenschaftler Marco Dorigo, der 1991 erstmals diesen Algorithmus einsetzte.

Die Ameisen errichten zwischen ihrem Ameisenhaufen und einer Futterquelle stets direkte Straßen. Doch wie machen sie das, wenn sie schlecht sehen können und jeder Grashalm für sie ein fast unüberwindbares Hindernis ist? Dafür hat ihnen die Natur eine wunderbare Einrichtung gegeben. Ameisen besitzen am Hinterleib eine Drüse, mit der sie den Lockstoff Pheromon produzieren können. Nachfolgende Ameisen orientieren sich an diesem Stoff und folgen mit hoher Wahrscheinlichkeit den am stärksten markierten Wegen.

Doch wie kommt es dabei in der Regel zu den kürzesten machbaren Wegen?

Betrachten wir als einfaches Modell (Abbildung 10-1) einen Ameisenhaufen und eine Futterquelle. Zwischen beiden liegt nun ein Hindernis, das die Ameisen zwingt einen Umweg zu nehmen. Für eine Gruppe von Ameisen, die die Futterquelle zum ersten Mal besucht, ergeben sich zwei Möglichkeiten, links oder rechts um das Hindernis herum.

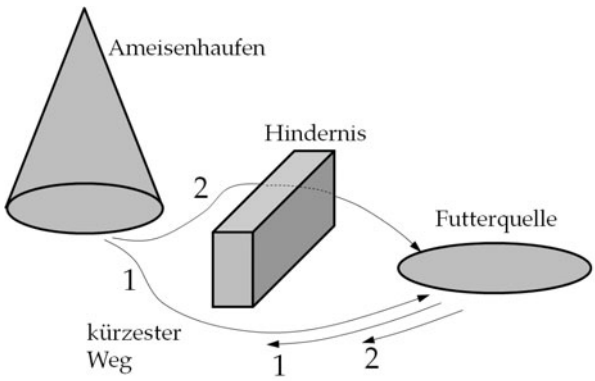


Abbildung 10-1: Kürzeste machbare Wege der Ameisen

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_10,

Während ihres Weges sondern die Ameisen einen Sexuallockstoff, also ein Pheromon ab. Da aber noch kein Pheromon vorliegt ist anzunehmen, dass sich die Gruppe teilt und die erste Gruppe nimmt den kürzeren Weg (1) und erreicht die Futterquelle zuerst. Sie wählt als Rückweg wieder den kürzeren Weg, da sie nun vom eigenen Pheromon des Hinwegs auch zurück geleitet wird. Die zweite Gruppe (2) mit dem längeren Weg kommt danach auch zur Futterquelle und nimmt nun, da der kürzere Weg bereits doppelt mit dem Pheromon der ersten Gruppe gekennzeichnet ist als der längere (nur einmal durch Gruppe 2), nun ebenfalls diesen Rückweg. So kommt es zu einer erneuten Verstärkung der Wegkennzeichnung. Für alle nachfolgenden Ameisen ist der Weg vorgegeben und wird immer stärker gekennzeichnet. Es entsteht eine Ameisenstraße. Sie sehen, eine einfache aber sehr wirksame Methode der Natur. Die Natur regelt dieses System aber noch weiter. Wind, Regen, Waldtiere und andere Einflüsse sorgen dafür, dass nur die aktuellen Wege gekennzeichnet bleiben.

Ein weiterer Störfaktor liegt darin, dass Ameisen sich durchaus auch einmal den längeren Weg aussuchen können, bzw. andere Wege suchen. Möglicherweise finden Sie so einen noch kürzeren Weg bzw. es wird verhindert, dass ein Weg zu dominant wird und das Finden besserer Lösungen verhindert. Hier wirkt ein Mechanismus, den wir bei den genetischen Algorithmen noch kennen lernen werden und der dort als Mutation bezeichnet wird. Alle diese Mechanismen ändern jedoch nichts am Grundprinzip.

Die Frage ist nun, wie kann dieses Prinzip im Computer nachgebildet werden. Eine Pseudo-Ameise verhält sich ebenso wie ihr natürliches Vorbild und macht ihre Entscheidung abhängig von der Pheromonausprägung aller möglichen Lösungen.

Das klassische Modell ist der Weg eines Handelsreisenden, der nacheinander verschiedene Orte besucht. Umgesetzt für unsere Pseudoameise wird die Stärke des Pheromons durch eine Zahl gekennzeichnet, die in einer Entfernungsmatrix alle Möglichkeiten beinhaltet. Mit dieser Matrix kann die Pseudoameise eine Auswahl treffen.

Tabelle 10-1: Beispiel einer Pheromonmatrix

Orte	1	2	3	4	5
1	-	0,2	0,5	0,4	0,5
2	0,2	-	0,3	0,5	0,6
3	0,2	0,5	-	0,4	0,6
4	0,3	0,4	0,3	-	0,6
5	0,5	0,3	0,2	0,4	-

Die Auswahlwahrscheinlichkeit wird auf das Intervall (0,1) abgebildet und mittels Monte-Carlo-Methode eine zufallsbedingte Auswahl getroffen. Die Gewichtung erfolgt also durch die Faktoren aus der Pheromonmatrix. Zusätzlich können diese

Faktoren noch durch eine Prioritätsregel verstärkt werden. Letztendlich ergibt sich aber immer eine prozentuale Verteilung wie sie ein Beispiel in Abbildung 10-2 zeigt.

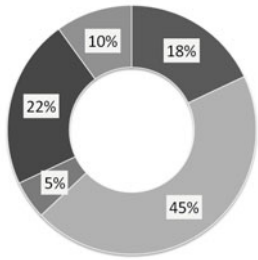


Abbildung 10-2: Beispiel für Wahrscheinlichkeiten einer Lösungsauswahl im Ameisenalgorithmus

Für eine Pseudoameise a_i, für die sich j=1,..,n Lösungsmöglichkeiten ergeben, die in der Pheromonmatrix die Pheromonfaktoren p_j besitzen und jeweils durch einen Wert w_j einer Prioritätsregel gewichtet werden, ergibt sich die Auswahlwahrscheinlichkeit

$$w_{ij} = \frac{g_{ij} \cdot p_{ij}}{\sum_{k=1}^{n} g_{ik} \cdot p_{ik}} \quad . \tag{10.1.1}$$

Nach der Wegsuche ist die Pheromonablage der nächste Schritt. Sie muss bei der Pseudoameise jedoch anders erfolgen als bei ihrem natürlichen Vorbild. In der Natur erfolgt die Ablage bei kürzeren Wegen schneller als bei längeren. Durch die diskrete Betrachtungsweise des Computers ist dies nicht möglich. Daher erfolgt die Kennzeichnung erst, nachdem eine Gruppe A von Pseudoameisen ihre Wege bestimmt hat und die Güte der Lösungen bewertet wurde. Diese Bewertung wird in die Pheromonmatrix eingetragen, die wiederum der Ausgang für die nächste Gruppe B ist. Hier kann auch, abweichend von der Natur, nur die beste Lösung in die Pheromonmatrix eingetragen werden. Dazu gibt es unterschiedliche Änderungen und Ergänzungen. Auch so genannte Verwitterungsfaktoren wurden eingeführt.

Eine Vertiefung des Stoffes in dieser Richtung überlasse ich dem Leser und möchte eine Konkretisierung in Richtung Optimierungsprobleme vornehmen. Der Ameisenalgorithmus kann auf viele Optimierungsprobleme übertragen werden. Etwa bei der Abstimmung von Produktionsabläufen in Fertigungsstraßen, bei der Produktion von Artikeln unter Beachtung der Rohstoffe, bei der Bewegung von Robotern, und vieles mehr. Der Ablauf für eine Optimierungsaufgabe hat aber in der einfachsten Form die nachfolgende Gestalt.

Tabelle 10-2: Wegsuche der Pseudoameisen ohne Gewichtungsfaktoren

Eingabe m = Anzahl Gruppen n = Anzahl Pseudoameisen einer Gruppe
Initialisiere Pheromonmatrix
i=1 (1) m
j=1 (1) n
Suche für alle Pseudoameisen a_j einen Weg in Abhängigkeit von der Pheromonmatrix
Beurteile die gefundenen Lösungen und trage die Beurteilung in die Pheromonmatrix ein.
Gib die beste Lösung an

Wir setzen den Algorithmus nachfolgend zur Optimierung einer Maschinenbelegung ein.

Beispiel 10-1: Maschinenbelegung

Dieses Beispiel habe ich meinem Buch [3] entnommen, in dem ich die Maschinenbelegung nach Johnson untersucht habe. In einem Produktionsbetrieb werden fünf Produkte auf drei Maschinen in gleicher Reihenfolge produziert. Die Belegung der Maschinen (M1, M2, M3) in Stunden je Produkt sind P1(5, 3, 8), P2(5, 3, 4), P3(12, 4, 2), P4(4, 2, 7) und P5(8, 2 ,9).

Konkretisieren wir den Algorithmus nach Tabelle 10-2 für diesen Fall.

Tabelle 10-3: Maschinenbelegungsoptimierung mit dem Ameisenalgorithmus

Eingabe g = Anzahl Gruppen n = Anzahl Pseudoameisen einer Gruppe p = Anzahl Produkte m = Anzahl Maschinen
i=1 (1) m
j=1 (1) n
Beurteilungsvektor-Produkte setzen k=1 (1) p
y(k,1)=k
y(k,2)=1
k=1 (1) p
q=0
Berücksichtigung der Pheromonanteile in y(k,2)
Bestimmung der Wahrscheinlichkeitsverteilung

				Erzeugung einer Zufallszahl	
				Ist die Konstellation zulässig?	
				Ja	Nein
				Speichere die Konstellation	./.
				q=1	
				So lange q=0	
			Bestimmung der Durchlaufzeit für die gefundene Konstellation		
		Beurteilung aller Konstellationen einer Gruppe dadurch, dass nur die minimalsten Werte in einer Pheromonmatrix eingetragen werden.			
Gib die beste Lösung an					

Zur Realisierung benutzen wir die ersten drei Spalten der Tabelle als Pheromonmatrix. In den nachfolgenden Spalten werden die Eingabedaten der Belegungszeiten für die Produkte auf den einzelnen Maschinen eingetragen. In den letzten beiden Spalten gebe ich die gefundenen Konstellationen einer Gruppe aus. Sie zeigen anschaulich die Funktion des Algorithmus. Mit einem Zwischenstopp des Programms können Sie die Ergebnisse jeder Gruppe beobachten.

Code 10-1: Prozeduren zur Bestimmung einer optimalen Maschinenbelegung

```
Option Explicit
   Dim Zeile As Integer

Sub Maschinen_Neu()
   Dim m, p, i As Integer
   Dim t As String

   ThisWorkbook.Worksheets("Maschinenbelegung").Cells.Clear
   Cells(1, 1) = "Folge"
   Cells(1, 2) = "Wert"
   Cells(1, 3) = "Pheromon"

   m = InputBox("Anzahl Maschinen")
   p = InputBox("Anzahl Produkte")

   Cells(1, 5 + m) = "Folge"
   Cells(1, 6 + m) = "Wert"
   t = "A1:" & Chr(64 + 6 + m) & "1"
   Range(t).Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   t = "D1:D" & LTrim(Str(p + 1))
   Range(t).Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True

   Range("D1").Select
   Selection.NumberFormat = "@"
   ActiveCell.FormulaR1C1 = LTrim(Str(p)) & "/" & LTrim(Str(m))
```

```
   For i = 1 To m
      Cells(1, 4 + i) = "M" & LTrim(Str(i))
   Next i
   For i = 1 To p
      Cells(1 + i, 4) = "P" & LTrim(Str(i))
   Next i
End Sub

Sub Maschinen_Test()
   ThisWorkbook.Worksheets("Maschinenbelegung").Cells.Clear
   Range("D1").Select
   Selection.NumberFormat = "@"
   ActiveCell.FormulaR1C1 = "5/3"
   Cells(1, 1) = "Folge"
   Cells(1, 2) = "Wert"
   Cells(1, 3) = "Pheromon"
   Cells(1, 5 + 3) = "Folge"
   Cells(1, 6 + 3) = "Wert"
   Range("A1:I1").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("D2:D6").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True

   Cells(2, 4) = "P1"
   Cells(3, 4) = "P2"
   Cells(4, 4) = "P3"
   Cells(5, 4) = "P4"
   Cells(6, 4) = "P5"
   Cells(1, 5) = "M1"
   Cells(1, 6) = "M2"
   Cells(1, 7) = "M3"

   Cells(2, 5) = 5
   Cells(2, 6) = 3
   Cells(2, 7) = 8
   Cells(3, 5) = 5
   Cells(3, 6) = 3
   Cells(3, 7) = 4
   Cells(4, 5) = 12
   Cells(4, 6) = 4
   Cells(4, 7) = 2
   Cells(5, 5) = 4
   Cells(5, 6) = 2
   Cells(5, 7) = 7
   Cells(6, 5) = 8
   Cells(6, 6) = 2
   Cells(6, 7) = 9

End Sub

Sub Maschinen_Start()
   Dim g, m, n, p, q, r, v, i, j, k, l, f() As Integer
   Dim t, ts, z() As String
   Dim w, x, u(), s(), y(), Min As Double
```

```
Zeile = 1
x = 0
Randomize

g = InputBox("Anzahl Gruppen")
n = InputBox("Anzahl Ameisen/Gruppe")
p = Left(Cells(1, 4), InStr(Cells(1, 4), "/") - 1)
m = Right(Cells(1, 4), InStr(Cells(1, 4), "/") - 1)

ReDim z(n), y(p, 2), f(p), u(m), s(n)

For i = 1 To g        'über alle Gruppen
   For j = 1 To n     'über alle Pseudoameisen
      For k = 1 To p
         y(k, 1) = k
         y(k, 2) = 1
      Next k

      'Auswahl
      For k = 1 To p  'über alle Produkte
         Do
            q = 0

            'Berücksichtigung der Pheromonanteile
            If k > 1 And Zeile > 1 Then
               For l = 1 To p
                  If y(l, 1) > 0 Then
                     y(l, 2) = 1
                     For r = 1 To Zeile
                        ts = t & "-" & LTrim(Str(l))
                        If ts = _
                        Left(Cells(r, 1), Len(ts)) Then
                           y(l, 2) = y(l, 2) + _
                                     Val(Cells(r, 3))
                        End If
                     Next r
                  End If
               Next l
            End If

            'Wahrscheinlichkeitsverteilung
            g = 0
            For l = 1 To p
               g = g + y(l, 2)
            Next l
            x = Rnd(x)
            w = Int(x * g + 1)
            g = 0
            For l = 1 To p
               g = g + y(l, 2)
               If g >= w Then
                  v = l
                  l = p
               End If
            Next l
            If y(v, 1) > 0 Then
```

```
            y(v, 1) = 0
            y(v, 2) = 0
            If k = 1 Then
               t = LTrim(Str(v))
            Else
               t = t & "-" & LTrim(Str(v))
            End If
            f(k) = v
            q = 1
         End If
      Loop While q = 0
   Next k
   Cells(j + 1, 5 + m) = t
   z(j) = t

   'Bestimmung der Durchlaufzeit
   For k = 1 To p
      For l = 1 To m
         If k = 1 Then
            u(l) = Cells(f(k) + 1, l + 4)
         Else
            If l = 1 Then
               u(l) = u(l) + Cells(f(k) + 1, l + 4)

            Else
               If u(l - 1) < u(l) Then
                  u(l) = u(l) + _
                         Cells(f(k) + 1, l + 4)
               Else
                  u(l) = u(l - 1) + _
                         Cells(f(k) + 1, l + 4)
               End If
            End If
         End If
      Next l
   Next
   Cells(j + 1, 6 + m) = u(m)
   s(j) = u(m)
Next j

'Beurteilung
'Nur die minimalsten Werte werden eingetragen
For j = 1 To n
   If j = 1 Then
      Min = s(j)
   Else
      If s(j) < Min Then
         Min = s(j)
      End If
   End If
Next j
For j = 1 To n
   If s(j) = Min Then
      q = 0
      For k = 1 To Zeile
         If Cells(k, 1) = z(j) Then
```

```
                Cells(k, 3) = Cells(k, 3) + 1
                q = 1
                k = Zeile
             End If
          Next k
          If q = 0 Then
             Zeile = Zeile + 1
             Cells(Zeile, 1) = z(j)
             Cells(Zeile, 2) = s(j)
             Cells(Zeile, 3) = 1
          End If
       End If
    Next j
 Next i
End Sub
```

Mit Hilfe eines Menüs werden die drei Prozeduren aufgerufen.

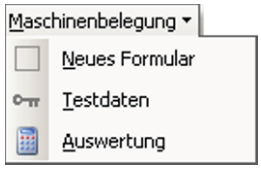


Abbildung 10-3: Menü zum Ameisenalorithmus

Die Testdaten liefern das bereits bekannte Ergebnis von 40 Stunden. Es werden gleich mehrere Lösungen ausgegeben, ohne dass deren Vollständigkeit gewährleistet ist.

	A	B	C	D	E	F	G	H	I
1	*Folge*	*Wert*	*Pheromon*	*5/3*	*M1*	*M2*	*M3*	*Folge*	*Wert*
2	2-1-5-4-3	40	27	*P1*	5	3	8	4-1-5-2-3	40
3	4-1-2-5-3	40	126	*P2*	5	3	4	1-4-2-5-3	40
4	5-2-1-4-3	40	18	*P3*	12	4	2	3-4-1-5-2	46
5	1-4-2-5-3	40	108	*P4*	4	2	7	4-1-2-5-3	40
6	5-4-2-1-3	40	49	*P5*	8	2	9	5-1-4-2-3	40
7	5-1-4-2-3	40	16					2-4-1-5-3	40
8	5-4-1-2-3	40	73					2-1-5-3-4	43
9	2-4-5-1-3	40	38					1-4-2-5-3	40
10	5-1-2-4-3	40	25					2-4-1-5-3	40
11	1-4-5-2-3	40	47					1-4-2-5-3	40
12	2-1-4-5-3	40	43						
13	2-4-1-5-3	40	64						
14	4-5-2-1-3	40	22						
15	5-2-4-1-3	40	4						
16	1-2-5-4-3	40	2						
17	4-1-5-2-3	40	3						
18	4-2-1-5-3	40	2						
19	4-2-5-1-3	40	2						

Abbildung 10-4: Ein mögliches Ergebnis der Testdaten

Bedingt durch die Pseudozufallszahlen, wird jedes Mal ein anderes Ergebnis erzeugt. Die Pheromonmatrix (links) und die Ergebnisse der letzten Gruppe (rechts) habe ich grau gekennzeichnet.

Übung 10-1: Gewichtung

Führen Sie eine Gewichtung der Produkte ein und berücksichtigen Sie diese bei der Pheromon-Auswertung.

Überlegen Sie, wie Sie einen Verwitterungsfaktor realisieren können (Pseudozufallszahlen).

10.2 Evolutionsstrategien

Die ersten Anwendungen der Evolutionsstrategie sind Ende der 60er Jahre zu finden. Es wurden ähnliche Theorien und Verfahren an verschiedenen Orten publiziert. In Deutschland ist es der Name Rechenberg, der mit der simulierten Evolution in Verbindung gebracht wird. Er lieferte die ersten wichtigen theoretischen Betrachtungen und sinnvollen Anwendungen. Als Beispiele sind die Optimierung eines Rohrkrümmers, einer Überschalldüse, eines pneumatischen Reglers, eines Stabtragwerkes, u. a. zu nennen.

Die Evolutionsstrategie ist eine universelle Methode zur Optimierung technischer Probleme. Sie wird vor allem bei Problemen eingesetzt, für die keine geschlossene Lösung vorliegt. Die Methode ähnelt dem Prinzip der biologischen Evolution und besteht meist aus Lösungsversuchen auf realen Parametersätzen, die dadurch besonders zur Lösung technischer Probleme geeignet sind. Im Gegensatz zu genetischen Algorithmen, bei denen eher ein genetischer Code benutzt wird, und die wir im Anschluss behandeln werden. Beide Methoden wurden unabhängig zur gleichen Zeit entwickelt.

Das Prinzip einer Evolutionsstrategie ist im nachfolgenden Struktogramm allgemein wiedergegeben.

Tabelle 10-4: Das Prinzip der Evolutionsstrategie

<table>
<tr><td colspan="3">Parametrisierung der Elemente des Gesamtsystems</td></tr>
<tr><td colspan="3">Initialisierung der Startparameter</td></tr>
<tr><td rowspan="6"></td><td colspan="2">Generierung neuer Elemente eines Gesamtsystems durch Rekombination oder Variation</td></tr>
<tr><td colspan="2">Zusammenfügen der neuen Elemente zu neuen Gesamtsystemen</td></tr>
<tr><td colspan="2">Beurteilung der neuen Gesamtsysteme in Bezug auf ein Kriterium oder mehrere Kriterien</td></tr>
<tr><td colspan="2">Verbesserung eines neuen Systems</td></tr>
<tr><td>Ja</td><td>Nein</td></tr>
<tr><td>Übernahme des verbesserten Systems als Startkriterium für weitere Generierungen</td><td>./.</td></tr>
<tr><td colspan="3">Wiederholung, bis ein Abbruchkriterium greift.</td></tr>
<tr><td colspan="3">Ausgabe der erzielten Lösung</td></tr>
</table>

Betrachten wir die Methode am nachfolgend einfachen Beispiel eines Stabwerks.

Beispiel 10-2: Stabwerkoptimierung

Das dargestellte Stabwerk unterliegt den angegebenen Kräften und soll mit dem geringsten Materialaufwand gestaltet werden. Die Daten sind a=100 mm, F1=1000 N, F2=500 N, F3=200 N.

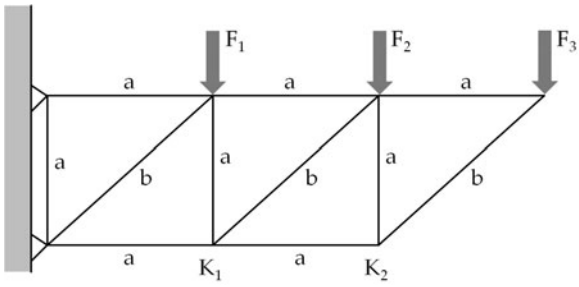


Abbildung 10-5: Stabwerk unter Belastung

Während der obere Gurt unverändert bleiben soll, können die stützenden Stäbe reduziert werden. Das bedeutet, dass die Knoten K1 und K2 verändert werden können. Wir legen in diese Knoten Koordinatensysteme und verändern deren Lage zufallsbedingt um 1 mm. Die Beziehungen ergeben sich aus den nachfolgenden Bildern.

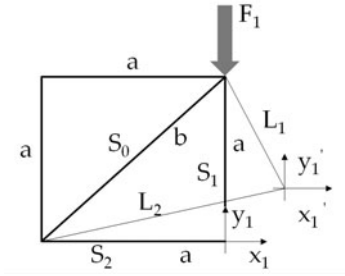


Abbildung 10-6: Verschiebung des 1. Knotens

$$L_1 = \sqrt{(a - y_1')^2 + x_1'^2} \tag{10.2.1}$$

$$L_2 = \sqrt{(a + x_1')^2 + y_1'^2} \tag{10.2.2}$$

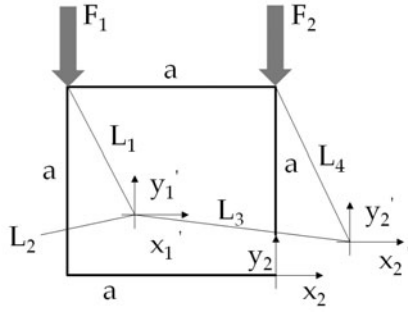


Abbildung 10-7: Verschiebung des 2. Knotens

$$L_3 = \sqrt{(a - x_1' + x_2')^2 + (y_1' - y_2')^2} \tag{10.2.3}$$

$$L_4 = \sqrt{(a - y_2')^2 + x_2'^2} \tag{10.2.4}$$

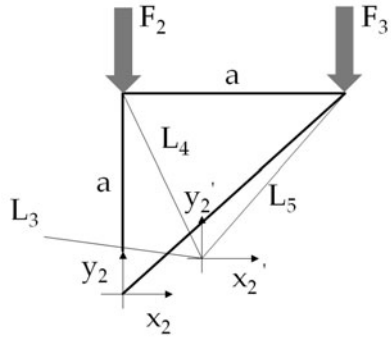


Abbildung 10-8: Der Endstab

$$L_5 = \sqrt{(a - x_2')^2 + (a - y_2')^2} \tag{10.2.5}$$

Nun können wir den Algorithmus in Struktogrammform aufstellen.

Tabelle 10-5: Optimierung eines Stabwerks nach der Evolutionsstrategie

Startbedingungen x1=0, y1=0, x2=0, y2=0 a=100 L1=a, L2=a, L3=a, L4=a, L5=a√2 m1=0, m2=0, m3=0		
	G1=L1+L2	
	Δx1=Int(Rnd(x)*3-1)	
	Δy1=Int(Rnd(x)*3-1)	
	x1′=x1+Δx1	
	y1′=y1+Δy1	
	$L_1 = \sqrt{(a - y_1')^2 + x_1'^2}$	
	$L_2 = \sqrt{(a + x_1')^2 + y_1'^2}$	
	m1=m1+1	
	Ist L1+L2<G1	
	Ja	Nein
	m1=0	./.
	G2=L3+L4	
	Δx2=Int(Rnd(x)*3-1)	
	Δy2=Int(Rnd(x)*3-1)	
	x2′=x2+Δx2	
	y2′=y2+Δy2	

	$L_3 = \sqrt{(a - x_1' + x_2')^2 + (y_1' - y_2')^2}$	
	$L_4 = \sqrt{(a - y_2')^2 + x_2'^2}$	
	m2=m2+1	
	Ist L3+L4<G2	
	Ja	Nein
	m2=0	./.
	$L_5 = \sqrt{(a - x_2')^2 + (a - y_2')^2}$	
	m3=m3+1	
	Ist m1=0 und m2=0	
	Ja	Nein
	Übernehme neue Koordinaten	./.
	So lange noch Veränderungen möglich, also m3<100	
Ausgabe der Lösung		

Da auch hier, bedingt durch die Pseudozufallszahlen, unterschiedliche Ergebnisse zu erwarten sind, wollen wir in einer Berechnungsschleife unter fünfzig Lösungen die beste aussuchen.

Code 10-2: Stabwerksoptimierung nach der Evolutionsstrategie

```
Option Explicit

Sub Träger_Optimierung()
   Dim x1a, y1a, x1n, y1n, dx1, dy1 As Integer
   Dim x2a, y2a, x2n, y2n, dx2, dy2 As Integer
   Dim i, j, m1, m2, m3 As Integer
   Dim x, a As Double
   Dim l1a, l2a, l3a, l4a, l5a As Double
   Dim l1n, l2n, l3n, l4n, l5n As Double
   Dim g12a, g12n, g34a, g34n As Double

   Randomize

   For j = 1 To 50
      x1a = 0
      y1a = 0
      x2a = 0
      y2a = 0
      a = 100
      m1 = 0: m2 = 0: m3 = 0
      l1a = a
      l2a = a
      l3a = a
      l4a = a
      l5a = Sqr(2) * a

      Do
```

```
'1. Knoten
g12a = l1a + l2a
x = Rnd(x)
dx1 = Int(x * 3 - 1)
x = Rnd(x)
dy1 = Int(x * 3 - 1)
x1n = x1a + dx1
y1n = y1a + dy1
l1n = Sqr((a - y1n) ^ 2 + x1n ^ 2)
l2n = Sqr((a + x1n) ^ 2 + y1n ^ 2)
g12n = l1n + l2n
m1 = m1 + 1
If g12n < g12a Then
   m1 = 0
End If
Cells(1, 1) = x1a
Cells(1, 2) = y1a
Cells(1, 3) = g12a

'2. Knoten
g34a = l3a + l4a
x = Rnd(x)
dx2 = Int(x * 3 - 1)
x = Rnd(x)
dy2 = Int(x * 3 - 1)
x2n = x2a + dx2
y2n = y2a + dy2
l3n = Sqr((a - x1n + x2n) ^ 2 + (y1n - y2n) ^ 2)
l4n = Sqr((a - y2n) ^ 2 + x2n ^ 2)
g34n = l3n + l4n
m2 = m2 + 1
If g34n < g34a Then
   m2 = 0
End If
Cells(2, 1) = x2a
Cells(2, 2) = y2a
Cells(2, 3) = g34a

'letzte Verbindung
l5n = Sqr((a - x2n) ^ 2 + (a - y2n) ^ 2)
Cells(3, 3) = l5n

m3 = m3 + 1
If m1 = 0 And m2 = 0 Then
   x1a = x1n
   y1a = y1n
   l1a = l1n
   l2a = l2n
   g12a = g12n
   x2a = x2n
   y2a = y2n
   l3a = l3n
   l4a = l4n
   g34a = g34n
   l5a = l5n
   m3 = 0
End If
```

```
      Loop While m3 < 100
      Cells(j, 5) = Cells(1, 1)
      Cells(j, 6) = Cells(1, 2)
      Cells(j, 7) = Cells(2, 1)
      Cells(j, 8) = Cells(2, 2)
      Cells(j, 9) = Cells(1, 3)
      Cells(j, 10) = Cells(2, 3)
      Cells(j, 11) = Cells(3, 3)
      Cells(j, 12) = Cells(1, 3) + Cells(2, 3) + Cells(3, 3)
   Next j
End Sub
```

Zum Aufruf der Prozedur benötigen wir lediglich einen Menüpunkt.

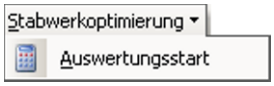


Abbildung 10-9: Menü zur Stabwerksoptimierung

Im vorderen Bereich A1:C3 werden die in der Berechnungsschleife ermittelten Lösungen mit Koordinaten und Stabwerkslängen angezeigt, während alle Lösungen als Tabelle im Bereich E1:L50 ausgegeben werden.

	A	B	C	D	E	F	G	H	I	J	K	L
1	-32	68	141,421356		-25	75	-40	74	141,421356	132,713324	143,377125	417,511805
2	-33	80	138,312184		-31	61	-52	66	141,917449	141,286968	156,51837	439,722786
3			135,340312		-28	72	-42	75	141,421356	134,929707	145,168867	421,519931
4					-29	71	-55	65	141,421356	139,434869	159,662143	440,518368
5					-28	72	-47	74	141,421356	134,736884	149,459024	425,617264
44					-28	72	-40	72	141,421356	136,826222	143,951381	422,198959
45					-24	76	-44	71	141,421356	132,853346	146,11297	420,387672
46					-24	76	-33	75	141,421356	132,405977	134,164079	407,991412
47					-26	74	-43	75	141,421356	132,745344	144,18391	418,350611
48					-25	75	-47	75	141,421356	131,235327	148,296999	420,953682
49					-28	72	-37	71	141,421356	138,016131	141,014184	420,451671
50					-32	68	-33	80	141,421356	138,312184	135,340312	415,073852

Abbildung 10-10: Ergebnisse aus den Testdaten

Hier ist es die 46. Lösung, die die minimalste Stabwerkslänge ausgibt. Sie beträgt jetzt 408 mm im Vergleich zur Ausgangslänge von 541 mm. Das bedeutet eine Einsparung von fast 25%. Allerdings ist dies eine nur geometrisch optimierte Lösung, bei der die Stäbe S0, S1 und S2 zusammenfallen.

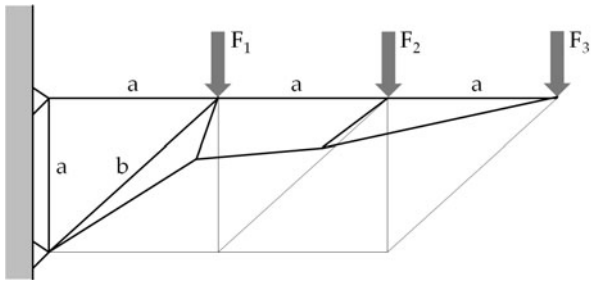


Abbildung 10-11: Das geometrisch optimierte Stabwerk

Übung 10-2: Grenzwerte

Legen Sie die maximalen Zug- und Druckkräfte aus dem Ausgangsstabwerk fest und berücksichtigen Sie diese als Grenzwerte für die Stabwerksoptimierung. Bekommt das Stabwerk im Beispiel dadurch eine andere Form?

Variieren Sie die Belastung und die Stabwerksform.

10.3 Genetische Algorithmen

Ebenfalls in den 60er Jahren war es John Holland, der die Mechanismen adaptiver Systeme durch so genannte productive plans auf Computern implementierte. Erst später kam der Begriff Genetischer Algorithmus auf. Die genetischen Algorithmen sind heute in Forschung und Anwendung die dominierenden Algorithmen des Bereichs evolutionärer Algorithmen.

Das Prinzip eines genetischen Algorithmus ist im nachfolgenden Struktogramm allgemein wiedergegeben.

Tabelle 10-6: Das Prinzip eines genetischen Algorithmus

Codierung der Individuen	
Initialisierung der Individuen der Startpopulation (oft zufallsbedingt)	
	Bewertung der Individuen nach einer Funktion (Fitness)
	Selection Auswahl der Elternpaare nach Fitness
	Kreuzung (Paarung) Erzeuge Nachkommen durch Rekombination, Variation oder aus einer Bibliothek
	Mutation Mutiere die erzeugten Nachkommen mit geringer Wahrscheinlichkeit einer Veränderung
	Nachkommen Ersetze die Individuen der aktuellen Generation nach einem Ersetzungsschema
	Wiederholung, bis ein Abbruchkriterium greift.
Ausgabe der erzielten Lösung	

Genetische Algorithmen imitieren evolutionäre Prozesse. Dabei liegt die Betrachtung bei den genetischen Prozessen. Ein Genetischer Algorithmus arbeitet mit einer Menge künstlicher Chromosomen, die meist Individuen oder Strings genannt werden. Jedes dieser Strings korrespondiert nun mit einer Variablen eines zu betrachtenden Optimierungsproblems. Dazu ist es erforderlich, die Variable auf einem zugeordneten String binär zu codieren (hier wird auch gerne der Gray-Code

benutzt) und ihnen Grenzwerte zu geben. In einer Initialisierungsphase wird eine Ausgangspopulation stochastisch erzeugt. Diese wird in einem zweiten Schritt mittels einer Fitnessfunktion bewertet und danach erfolgt eine fitnessproportionale Selektion. Da alle Individuen eine positive Selektionswahrscheinlichkeit haben, können sie potentiell Nachkommen zeugen. Entgegen diskriminierender Evolutionsstrategien bei denen Populationen keine Chance erhalten, Nachkommen zu haben. Dann erfolgt das Erzeugen von Nachkommen durch Partnerwahl und den Variationsoperatoren Crossover und Mutation. Die erzielten Nachkommen werden bewertet und ergänzen möglicherweise die neue Population. Dieses Verfahren wird so lange wiederholt, bis ein entsprechendes Abbruchkriterium greift.

Beispiel 10-3: Packproblem

Dieses Packproblem ist in der Literatur als 0/1-Rucksackproblem bekannt und lautet wie folgt. Eine Menge von Gegenständen soll so verpackt werden, dass die Gesamtmasse einen Grenzwert nicht übersteigt und der Wert der verpackten Gegenstände ein Maximum erzielt.

In diesem Fall wollen wir die einzelnen Schritte des genetischen Algorithmus auch einzeln betrachten. Zunächst erzeugen wir wieder ein Formblatt und fünfzig zufallsbedingte Testdaten mit Massen im Bereich von 1…50 und Werten im Bereich von 1…100. Die maximale Packmenge soll 200 betragen.

Code 10-3: Formblatt und Testdaten zum PP

```
Option Explicit

Sub PP_Neu()
   ThisWorkbook.Worksheets("Packproblem").Cells.Clear
   Cells(1, 1) = "Maximal"
   Cells(1, 3) = "Massen:"
   Cells(2, 3) = "Werte:"
   Cells(4, 1) = "Masse"
   Cells(4, 2) = "Wert"
   Cells(4, 3) = "Codes"
   Range("A1").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A4:D4").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("C1:C2").Select
   Selection.Font.Bold = True
   Selection.Font.Italic = True
   Range("A2").Select
End Sub
'Erzeugung von 50 zufallbedingten Gegenständen
Sub PP_Test()
   Dim i As Integer
   Dim x, m, w As Double

   Randomize
   For i = 1 To 50
```

```
      x = Rnd(x)
      m = Int(x * 50) + 1   'Massen im Bereich 1...50
      x = Rnd(x)
      w = Int(x * 100) + 1  'Werte im Bereich 1...100
      Cells(i + 4, 1) = m
      Cells(i + 4, 2) = w
   Next i
   Cells(2, 1) = 200
End Sub
```

In einem ersten Schritt erfolgt nun die Initialisierung der Individuen, so dass durch eine nachfolgende Selektion die Eltern bestimmt werden. Dazu erhalten die Spalten D bis G den genetischen Code in binärer Form (0/1). 1 bedeutet dabei die Auswahl des Gegenstandes und 0 seine Abwahl. Eine Spalte stellt somit das Chromosom eines Individuums dar, deren Gene die Auswahl bestimmen. Ähnlich einer DNA, deren Gene Erbinformationen speichern.

Allgemein ist immer darauf zu achten, dass die Codierung der Individuen in geeigneter Form auf die spätere Anwendung genetischer Operatoren zielt. Nur so kann ein genetischer Algorithmus erfolgreich arbeiten.

Code 10-4: Initialisierung der Eltern

```
'Erzeugung der Start-Eltern
Sub PP_Elter()
   Dim i, j, k, m, n As Integer
   Dim x, g, g1, w, w1, Max As Double

   Randomize
   Max = Cells(2, 1)
   n = ThisWorkbook.Worksheets("Packproblem"). _
       UsedRange.Rows.Count
   n = n - 4

   For k = 1 To 4
      For i = 1 To n 'Nullvektor
         Cells(i + 4, 3 + k) = 0
      Next i

      'Zufallsbedingte Auswahl von Genen
      g = 0
      w = 0
      m = 0
      Do
         x = Rnd(x)
         j = Int(x * n + 1)
         g1 = Cells(j + 4, 1)
         w1 = Cells(j + 4, 2)
         m = m + 1
         If g + g1 <= Max Then
            g = g + g1
            w = w + w1
            Cells(j + 4, 3 + k) = 1
            m = 0
         End If
      Loop While m < 50 'Abbruchkriterium
```

```
      Cells(1, 3 + k) = g
      Cells(2, 3 + k) = w
   Next k
End Sub
```

Jetzt erfolgt der erste Schritt der eigentlichen Iterationsschleife, die Selektion, auch Fitnesstest genannt. Es wird geprüft, ob die neue Generation die Optimierungskriterien besser erfüllt als die Eltern. Ist dies der Fall, so erfolgt ein Generationenwechsel.

Code 10-5: Selektion

```
'Vergleich der Chromosomen
'Erfüllungskriterium maximaler Wert
Sub PP_Selection()
   Dim i, j, k, n As Integer
   Dim Max As Double
   n = ThisWorkbook.Worksheets("Packproblem"). _
       UsedRange.Rows.Count
   n = n - 4
   Max = Cells(2, 1)
   For i = 1 To 2
      If Cells(1, 5 + i) <= Max Then
         For j = 1 To 2
            If Cells(2, 5 + i) > Cells(2, 3 + j) Then
               For k = 1 To n + 4
                  Cells(k, 3 + j) = Cells(k, 5 + i)
               Next k
               j = 2
               i = i + 1
            End If
         Next j
      End If
   Next i
   Range("F:G").Select
   Selection.Clear
   Range("A2").Select
End Sub
```

Eine neue Population wird durch Rekombination (Kreuzung, Paarung, etc.) erzielt. Diese spielt bei den genetischen Algorithmen die bedeutende Rolle, die die Mutation bei der Evolutionsstrategie spielt. Die Wahl der geeigneten Rekombinationsmethode ist wesentlich für den Algorithmus. Es existieren eine Fülle solcher Verfahren, von denen einige Kreuzungen (Crossover-Verfahren) sind. Wir befassen uns an dieser Stelle mit dem einfachsten Verfahren, der Ein-Punkt-Kreuzung (engl. one-point-crossover). Hier liegt auch das Problem unseres Algorithmus, wie Sie später beim Testen sicher noch feststellen werden.

Beim one-point-crossover ermittelt eine Pseudozufallszahl eine Bruchstelle in den Chromosomen der beiden Eltern und fügt die Bruchstücke gekreuzt zu neuen Chromosomen zusammen.

Code 10-6: Rekombination

```
'Kreuzung nach dem Ein-Punkt-Überkreuzverfahren
'(one-point-crossover)
Sub PP_Crossover()
   Dim i, j, n As Integer
   Dim x, m, m1, w, w1 As Double

   Randomize
   n = ThisWorkbook.Worksheets("Packproblem"). _
       UsedRange.Rows.Count
   n = n - 4

   x = Rnd(x)
   j = Int(x * n + 1) 'Vertauschungspunkt der Chromosomen

   For i = 1 To j
      Cells(i + 4, 6) = Cells(i + 4, 4)
      Cells(i + 4, 7) = Cells(i + 4, 5)
   Next i
   For i = j To n
      Cells(i + 4, 6) = Cells(i + 4, 5)
      Cells(i + 4, 7) = Cells(i + 4, 4)
   Next i

   'Ermittlung der neuen Werte
   For i = 1 To 2
      m = 0
      w = 0
      For j = 1 To n
         If Cells(j + 4, 5 + i) = 1 Then
            m1 = Cells(j + 4, 1)
            w1 = Cells(j + 4, 2)
            m = m + m1
            w = w + w1
         End If
      Next j
      Cells(1, 5 + i) = m
      Cells(2, 5 + i) = w
   Next i
End Sub
```

Im Gegensatz zur Evolutionsstrategie, spielt die Mutation im genetischen Algorithmus eine untergeordnete Rolle. Sie erfüllt hier auch einen anderen Zweck, sie soll die zunehmende Konvergenz von Generationen verhindern. Die Mutation beschränkt sich auf die Genänderung (Bitumwandlung) an einer zufallsbestimmten Stelle des Chromosoms. Eine Eins wird zur Null und umgekehrt. Die Mutation wird meist mit geringer Wahrscheinlichkeit ausgelöst. In unserem Fall liegt sie bei 20%.

Code 10-7: Mutation

```
Sub PP_Mutation()
   Dim i, j, n As Integer
   Dim x, m, m1, w, w1 As Double
```

```
    Randomize
    n = ThisWorkbook.Worksheets("Packproblem"). _
        UsedRange.Rows.Count
    n = n - 4

    For i = 1 To 2
       x = Rnd(x)
       If x < 0.2 Then '20% Wahrscheinlichkeit einer Mutation
          x = Rnd(x)
          j = Int(x * n + 1)
          If Cells(j + 4, 5 + i) = 0 Then
             Cells(j + 4, 5 + i) = 1
          Else
             Cells(j + 4, 5 + i) = 0
          End If
       End If
    Next i
    'Ermittlung der neuen Werte
    For i = 1 To 2
       m = 0
       w = 0
       For j = 1 To n
          If Cells(j + 4, 5 + i) = 1 Then
             m1 = Cells(j + 4, 1)
             w1 = Cells(j + 4, 2)
             m = m + m1
             w = w + w1
          End If
       Next j
       Cells(1, 5 + i) = m
       Cells(2, 5 + i) = w
    Next i
End Sub
```

Nach der Mutation beginnt der Algorithmus wieder bei der Selektion. Dieser Kreislauf wird solange durchgeführt wie Populationen gewünscht werden, bzw. bis ein Abbruchkriterium greift.

Abbildung 10-12: Menü zum genetischen Algorithmus

Mit Hilfe der einzelnen Menüpunkte können Sie den Verlauf der Stationen des genetischen Algorithmus genau verfolgen und beurteilen. Um mehrere Populationen hintereinander ablaufen zu lassen, benutzen Sie den Menüpunkt Populationen und die nachfolgende Prozedur.

Code 10-8: Populationen

```
Sub PP_Populationen()
   Dim i, n As Integer

   n = InputBox("Anzahl der Populationen eingeben!")

   For i = 1 To n
      Call PP_Selection
      Call PP_Crossover
      Call PP_Mutation
   Next i
End Sub
```

Die Auswertung der Testdaten liefert als Ergebnis Beispieldaten, wie sie in Abbildung 10-13 dargestellt sind.

	A	B	C	D	E	F	G
1	*Maximal*		*Massen:*	500	500	425	387
2	500		*Werte:*	1605	1448	1021	1194
3							
4	*Masse*	*Wert*	*Codes*				
5	1	51		1	1	1	1
6	34	36		0	1	0	1
7	8	92		1	1	1	1
8	9	94		0	1	0	1
9	25	95		1	0	1	0
10	10	79		1	0	1	0
11	21	97		1	0	1	0
12	46	46		1	0	1	0
13	6	4		0	0	0	0
14	27	48		1	0	1	0
15	40	6		0	0	0	0
16	29	20		0	0	0	0
17	11	27		0	0	0	0
18	35	85		0	1	0	1
19	26	93		0	0	0	0
20	21	89		0	0	0	0
21	5	27		0	0	0	0
22	16	58		0	0	0	0
23	1	72		0	1	0	1
24	25	80		0	0	0	0
25	24	32		1	0	1	0
26	4	12		1	1	1	1
27	25	47		0	0	0	0
28	11	18		0	0	0	0
29	3	83		0	0	0	0
30	11	19		1	1	1	1
31	15	58		0	0	0	0

Abbildung 10-13: Auswertung der Testdaten zum genetischen Algorithmus

Übung 10-3: Variationen

Sie werden bei der Beurteilung feststellen, dass die hier benutzte Rekombination sehr einfach ist. Programmieren Sie daher einen n-Point-Crossover und beurteilen Sie die Veränderungen. Überlegen Sie außerdem, wie die Rekombination derart verändert werden kann, dass immer brauchbare Populationen entstehen.

In diesem Beispiel werden nur zwei Eltern benutzt. Oftmals wählt man aber aus einem Pool die fittesten Paarungen aus. Erweitern Sie das Programm um diesen Aspekt.

11 Künstliche Intelligenz

Ähnlich wie zu den Algorithmen gibt es auch für die künstliche Intelligenz keine einheitliche Definition. 1950 hat sich bereits Alan Turing in „Computing machinery and intelligence“ mit maschineller Intelligenz beschäftigt.

Er schlägt einen empirischen Test vor, bei dem über Terminals ein Mensch und eine Maschine einen Dialog mit einer Prüfperson führen. Wenn diese Prüfperson nach dem Dialog nicht sagen kann, wer Mensch und wer Maschine ist, dann ist nach Turing diese Maschine intelligent.

11.1 Fuzzy Logic

Der Einstieg in die Fuzzy Logic wird oft mit folgender Betrachtung begonnen. Werden mehrere Personen gefragt, welche Temperatur sie als angenehm empfinden, dann sind die Antworten recht unterschiedlich. Für die eine Person sind dies 18 Grad Celsius und für eine andere 22 Grad Celsius. Im Sinne der klassischen Mengenlehre würde der Bereich von z.B. 18 bis 24 Grad als angenehm definiert werden können, der Bereich darunter als unangenehm, auch wenn es 17,9 Grad sind. Wir merken, ein scharfer Übergang von unangenehm zu angenehm entspricht nicht dem menschlichen Empfinden.

Im Jahre 1965 entwickelte Lotfi A. Zadeh an der Universität von Kalifornien die Theorie der unscharfen Mengen (fuzzy set theory) und ermöglichte es damit Computern mit unscharfen Mengen zu rechnen. Im Gegensatz zur klassischen Mengenlehre können Elemente bei Fuzzy Sets bis zu einem bestimmten Grad einer Menge angehören. Dieser Zugehörigkeitsgrad wird durch eine Zahl im Intervall [0,1] ausgedrückt. Dabei bedeutet 0 keine und 1 volle Zugehörigkeit.

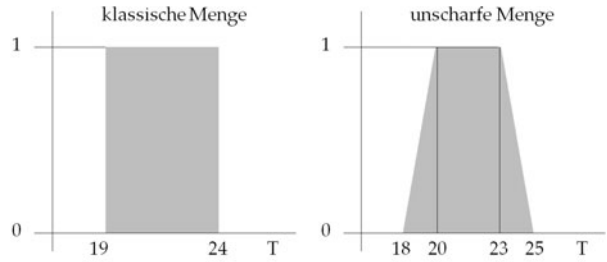


Abbildung 11-1: Menge der angenehmen Raumtemperatur

So lassen sich auch unscharfe Begriffe wie „angenehm“, „kalt“, „zu tief“, usw. durch unscharfe Mengen beschreiben. Sie werden als linguistische Werte bezeichnet. Einfache Formen von Fuzzy-Mengen eignen sich besonders gut zur Auswertung.

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2_11,

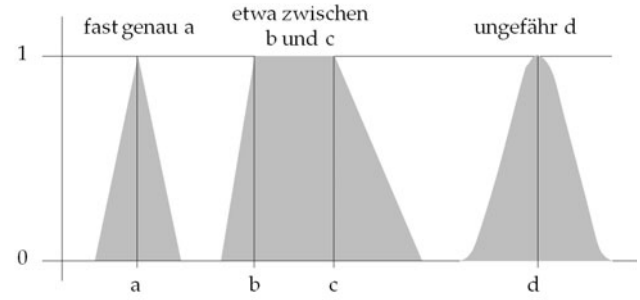


Abbildung 11-2: Formen von Fuzzy-Mengen

Damit die Variable einer Fuzzy-Anwendung korrekt arbeiten kann, muss jeder mögliche Wert zumindest einer unscharfen Menge angehören. Der Wertebereich wird daher in linguistische Werte aufgeteilt, die sich teilweise überdecken. Diese Aufteilung bezeichnet man als unscharfe Fuzzy-Zerlegung und die Mengen als Fuzzy-Sets.

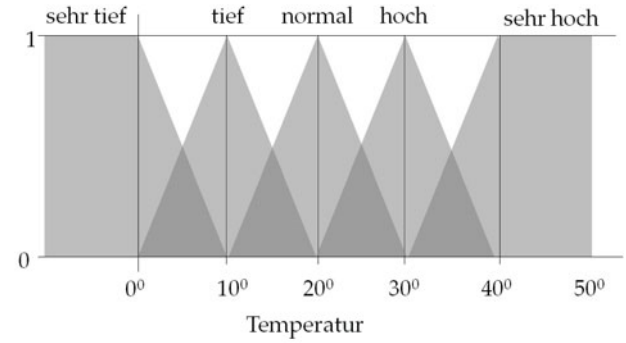


Abbildung 11-3: Beispiel einer Fuzzy-Zerlegung für die Raumtemperatur

Operationen mit Fuzzy-Mengen verlaufen genauso wie mit normalen Mengen. Teilmenge, Nullmenge und gleiche Menge definieren sich genauso. Mengenoperationen sind wie folgt definiert.

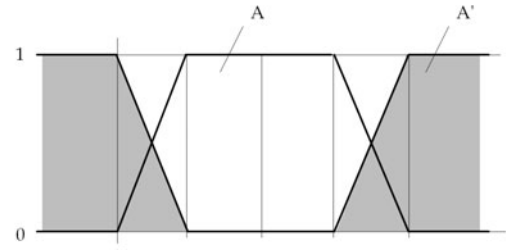


Abbildung 11-4: Das Komplement A' einer Fuzzy-Menge A

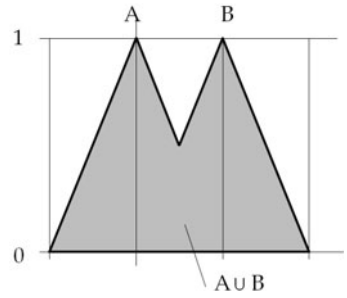


Abbildung 11-5: Die Vereinigungsmenge der Fuzzy-Mengen A und B

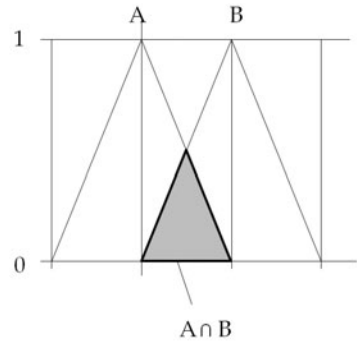


Abbildung 11-6: Die Durchschnittsmenge der Fuzzy-Mengen A und B

Ein Fuzzy-Regelsystem besteht nun im Wesentlichen aus drei Schritten:

- Fuzzyfizierung der Eingangsgrößen
- Inferenz und Komposition der Regeln
- Defuzzyfizierung der Ausgangsgrößen

Eine Eingangsgröße erhält über die Fuzzy-Menge einen Zugehörigkeitsgrad, und damit einen Wert aus dem Intervall [0,1]. Dieser Vorgang wird als Fuzzyfizierung bezeichnet.

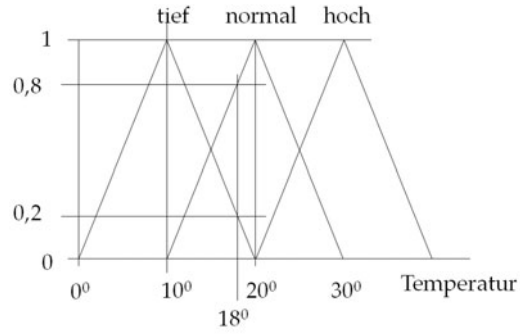


Abbildung 11-7: Beispiel einer Fuzzyfizierung

Als Beispiel ist die Raumtemperatur von 18 Grad nach Abbildung 11-7 mit einer Wahrscheinlichkeit von 0,8 eine normale Temperatur und mit einer Wahrscheinlichkeit von 0,2 eine tiefe Temperatur.

In den Bedingungen für eine Regelung werden linguistische Werte oft durch logische Operatoren miteinander verknüpft. Dieser, als Inferenz bezeichnete Vorgang, wird aus dem Zugehörigkeitsgrad der unterschiedlichen Fuzzy-Mengen gebildet.

Verknüpfung linguistischer Werte

Bei einer UND-Verknüpfung wird der gemeinsame Zugehörigkeitsgrad aus dem Minimum der einzelnen Zugehörigkeitsgrade gebildet.

Nach Abbildung 11-8 ist somit

$$C = A \vee B = Min\{0{,}6;0{,}3\} = 0{,}3 \qquad (11.1.1)$$

und

$$C = A \wedge B = Max\{0{,}6;0{,}3\} = 0{,}6 \ . \qquad (11.1.2)$$

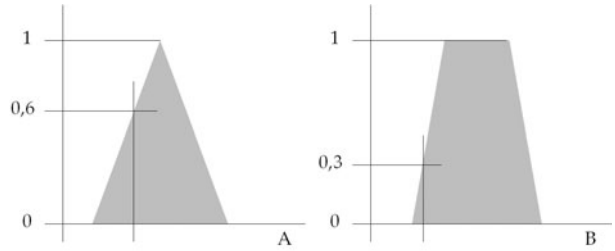


Abbildung 11-8: Verknüpfung linguistischer Werte

Bei einer ODER-Verknüpfung wird der gemeinsame Zugehörigkeitsgrad aus dem Maximum der einzelnen Zugehörigkeitsgrade gebildet.

In Fuzzy-Reglern werden oft mehrere Inferenzen benötigt, so dass diese wiederum zu einer Gesamtfunktion zusammengefasst werden müssen. Diesen Vorgang bezeichnet man als Komposition. Es gibt mehrere solcher Methode, aber die gebräuchlichste wird als Maximum-Methode bezeichnet. Dabei ergibt sich die Gesamtfunktion als Summe aller Zugehörigkeitsfunktionen.

Als letzten Schritt erfolgt eine Defuzzyfizierung zur Ermittlung der konkreten Ausgabegröße. Auch hier gibt es unterschiedliche Methoden. Die gebräuchlichste davon ist die Schwerpunkt-Methode. Zunächst wird der Schwerpunkt der Gesamtfunktionsfläche bestimmt, dessen Abszissenwert dann die Ausgangsgröße ist. Die nachfolgende Abbildung 11-9 zeigt noch einmal anschaulich den Ablauf einer Fuzzy-Regelung.

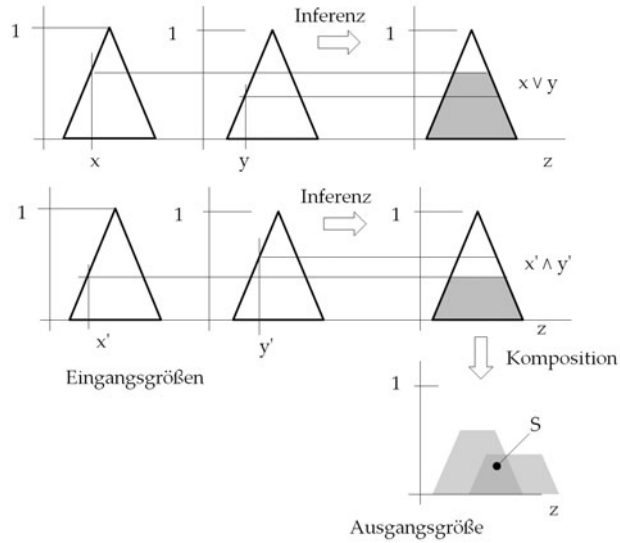


Abbildung 11-9: Das Schema einer Fuzzy-Regelung

Beispiel 11-1: Fuzzy-Regelung eines Industrieofens

Gesucht ist das Regelfeld für die Brennkammer eines Industrieofens. Zu regeln sind die Temperatur im Brennraum und der Druck, mit dem das Brenngas zugeleitet wird. Die notwendigen Daten ergeben sich aus den nachfolgenden Bildern. Dieses Beispiel ist aus den Übungen meines Buches [3] entnommen.

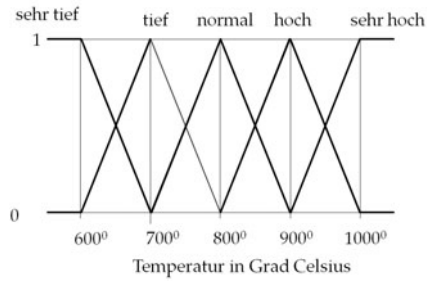


Abbildung 11-10: Fuzzy-Set Temperatur

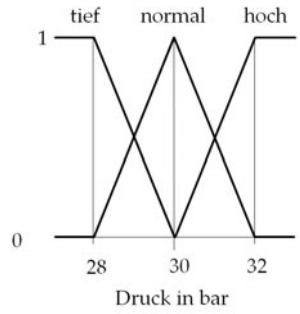


Abbildung 11-11: Fuzzy-Set Druck

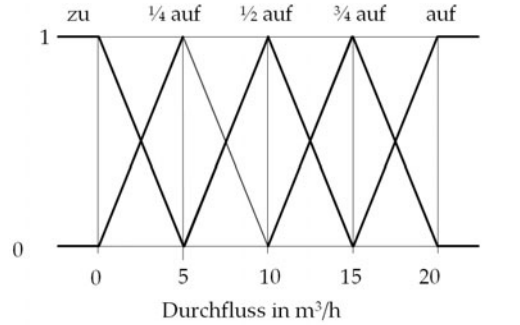


Abbildung 11-12: Ventilstellung

Für diese Wahrscheinlichkeiten definieren wir jetzt die Regeln in Form einer Inferenzen-Matrix.

Tabelle 11-1: Inferenzen-Matrix zur Ventileinstellung

Druck \ Temp.	sehr tief	tief	normal	hoch	sehr hoch
tief	auf	auf	¾ auf	½ auf	¼ auf
normal	auf	¾ auf	½ auf	½ auf	zu
hoch	¾ auf	½ auf	¼ auf	zu	zu

Tabelle 11-2: Algorithmus zur Berechnung des Fuzzy-Reglers

Für alle Temperaturen T von 600 bis 1000 Grad Celsius jeweils um 50 Grad verändert
Für alle Drücke von tief bis hoch jeweils um ¼ -Anteil verändert
Fuzzyfizierung der Temperatur wT(i)=f(Temperatur-Set), i=0,…,4
Fuzzyfizierung des Drucks wD(i)=f(Drücke-Set), i=0,…,2
Eintragung der Wahrheitswerte in eine Matrix M(0,i+1)=wT(i), i=0,…,4 M(i+1,0)=wD(i), i=0,…,4
Auswertung der Matrix, Inferenzen M(i,j)=Minimum(M(i,0),M(0,j)), i=1,..,3, j=1,…,5
Normierung der Summen gleicher Ventilstellungen auf 1
$v(i)=\sum M_v(j,k)$, i=0,…,4, j=1,…,3, k=1,…,5
$s=\sum v(i)$, i=0,…,4
$v_n(i)=v(i)/s$
Bestimmung des Gesamtschwerpunktes und damit der Stellgröe

		$u = \frac{\sum_i v(i)^2 \cdot s_i}{\sum_i v(i)^2}$
		Ausgabe der Temperatur, des Drucks und der errechneten Ventilstellung
Grafische Anzeige des Kennfeldes		

Für die Auswertung legen wir ein neues Tabellenblatt an und für den Aufruf der Auswertung ein Menü.

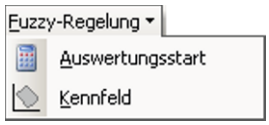


Abbildung 11-13: Menü zur Fuzzy-Regelung

Code 11-1: Die Auswertungsprozeduren zur Fuzzy-Regelung

```
Option Explicit
Option Explicit
Dim wT(4), wD(3) As Double

Sub Auswertung()
   Dim T, M(3, 5), v(4) As Double
   Dim i, i1, i2, z As Integer
   Dim u, u1, u2, sv As Double
   Dim MyDoc As Object
   Dim Shp As Shape
   Set MyDoc = ThisWorkbook.Worksheets("Fuzzy")
   MyDoc.Activate
   MyDoc.Cells.Clear
'
'alle Charts löschen
   For Each Shp In MyDoc.Shapes
      Shp.Delete
   Next
'
'Auswertungsstart
   z = 0
   '
   'Über alle Temperaturen
   For T = 600 To 1000 Step 50
      '
      'Fuzzyfizierung
      Call FuzzyTemperatur(T)
      For i = 0 To 2
         M(0, i + 1) = wT(i)
      Next i
      z = z + 1
      Cells(1, z) = T
      '
      'Über alle Drücke
      For i = 1 To 9
```

```
         Call FuzzyDrücke(i)
         For i1 = 0 To 2
            M(i1 + 1, 0) = wD(i1)
         Next i1
         '
         'Inferenzen
         For i1 = 1 To 3
            For i2 = 1 To 5
               M(i1, i2) = Minimum(M(i1, 0), M(0, i2))
            Next i2
         Next i1
         '
         'Defuzzyfizierung
         v(4) = M(1, 1) + M(1, 2) + M(2, 1)
         v(3) = M(1, 3) + M(2, 2) + M(3, 1)
         v(2) = M(1, 4) + M(2, 3) + M(2, 4) + M(3, 2)
         v(1) = M(1, 5) + M(3, 3)
         v(0) = M(2, 5) + M(3, 4) + M(3, 5)
         sv = v(0) + v(1) + v(2) + v(3) + v(4)
         For i1 = 0 To 4
            If Not sv = 0 Then
               v(i1) = v(i1) / sv
            End If
         Next i1
         u1 = v(1) ^ 2 * 0.25 + v(2) ^ 2 * 0.5 + _
              v(3) ^ 2 * 0.75 + v(4) ^ 2 * 1
         u2 = v(0) ^ 2 + v(1) ^ 2 + v(2) ^ 2 + _
              v(3) ^ 2 + v(4) ^ 2
         If Not u2 = 0 Then
            u = u1 / u2
         Else
            u = 0
         End If
         Cells(1 + i, z) = u
      Next i
   Next T
End Sub

Function Minimum(a, b) As Double
   If a <= b Then
      Minimum = a
   Else
      Minimum = b
   End If
End Function

Sub FuzzyTemperatur(T)
   Dim i As Integer
   If T < 600 Then
      wT(0) = 1
   Else
      If T <= 700 Then
         wT(0) = (700 - T) / 100
      Else
         wT(0) = 0
      End If
   End If
```

```
   For i = 1 To 4
      If T < 700 + (i - 1) * 100 Then
         wT(i) = 0
      Else
         If T <= 700 + i * 100 Then
            wT(i) = (T - (700 + (i - 1) * 100)) / 100
         Else
            If T <= 700 + (i + 1) * 100 Then
               wT(i) = (700 + (i + 1) * 100 - T) / 100
            Else
               wT(i) = 0
            End If
         End If
      End If
   Next i
   If T < 900 Then
      wT(4) = 0
   Else
      If T <= 1000 Then
         wT(4) = (T - 900) / 100
      Else
         wT(4) = 1
      End If
   End If
End Sub

Sub FuzzyDrücke(i)
   Select Case i
   Case 1
      wD(0) = 1: wD(1) = 0: wD(2) = 0
   Case 2
      wD(0) = 0.75: wD(1) = 0.25: wD(2) = 0
   Case 3
      wD(0) = 0.5: wD(1) = 0.5: wD(2) = 0
   Case 4
      wD(0) = 0.25: wD(1) = 0.75: wD(2) = 0
   Case 5
      wD(0) = 0: wD(1) = 1: wD(2) = 0
   Case 6
      wD(0) = 0: wD(1) = 0.75: wD(2) = 0.25
   Case 7
      wD(0) = 0: wD(1) = 0.5: wD(2) = 0.5
   Case 8
      wD(0) = 0: wD(1) = 0.25: wD(2) = 0.75
   Case 9
      wD(0) = 0: wD(1) = 0: wD(2) = 1
   End Select
End Sub
```

Das Ergebnis der Auswertung ist das nachfolgend dargestellte Kennfeld.

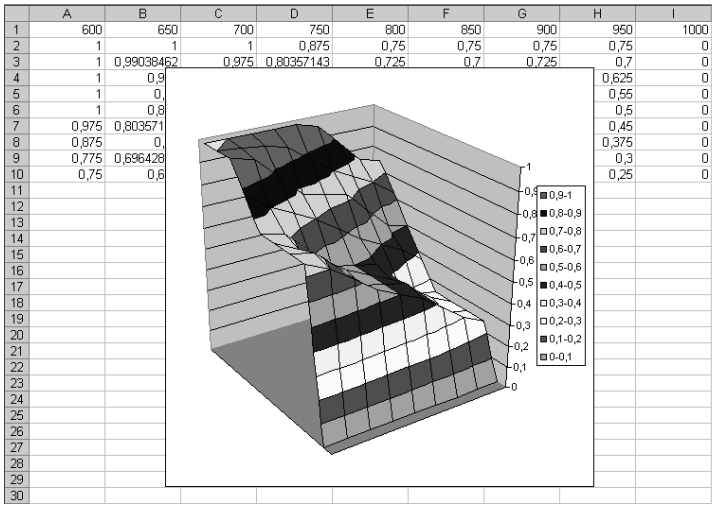


Abbildung 11-14: Berechnetes Kennfeld

Übung 11-1: Verfeinerung

Verfeinern Sie die Regelung durch eine feinere Aufteilung der Fuzzyfizierung und Defuzzyfizierung.

11.2 Expertensysteme

Im Bereich Künstliche Intelligenz (KI) nennt man ein Softwaresystem ein Expertensystem, wenn es in der Lage ist, Lösungen für Probleme auf einem Fachgebiet zu liefern, die von der Qualität her von denen eines menschlichen Experten nicht zu unterscheiden sind.

Besonders bewährt haben sich wissensbasierte Systeme, so dass Expertensysteme auch oft so benannt werden. Einen Ansatz kann dabei die Sammlung von Lösungen zu bestimmten Problemstellungen sein, die sich hierarchisch in Excel-Tabellen sammellassen. Zur Verwaltung dieser Datenmengen bietet sich die Methode der Hashfunktion an. Eine Ausarbeitung zu dem Thema Hashsuche in Listen finden Sie auf meiner Autorenwebsite. Leider habe ich in diesem Buch nicht den Raum, um über Expertensysteme mehr zu schreiben.

Literaturverzeichnis

[1] J.L. Adams, Ich habs! Wie man Denkblockaden mit Phantasie überwindet. Vieweg Verlag, 1984

[2] R. Groner, Hypothesen im Denkprozess, Grundlagen einer verallgemeinerten Theorie auf der Basis elementarer Informationsverarbeitung, Huber Verlag, 1978

[3] H. Nahrstedt, VBA für Maschinenbauer, Vieweg Verlag, 2005

[4] J. Ziegenbalg, Algorithmen, Von Hammurapi bis Gödel, Spektrum Akademischer Verlag, 1996

[5] I. Rechenberg, Evolutionsstrategie ´94, Friedrich Frommann Verlag, Günther Holzboog, 1994

[6] E. Horowitz, S. Sahni, Algorithmen, Entwurf und Analyse, Springer Verlag, 1981

[7] N. Wirth, Algorithmen und Datenstrukturen, Teubner Verlag, 2000

[8] W. Domschke, A. Drexl, Einführung in Operation Research, Springer Verlag, 2004

[9] V. Heun, Grundlegende Algorithmen, Vieweg Verlag, 2003

[10] Z. Michalewicz, Genetic Algorithms and Data Structures equal Evolution Programs, Springer Verlag, 1999

[11] N. Blum, Algorithmen und Datenstrukturen, Oldenbourg Verlag, 2004

[12] R. Sedgewick, Algorithmen, Addison-Wesley Verlag, 2002

[13] I. Wegener, Theoretische Informatik – eine algorithmenorientierte Einführung, Teubner Verlag, 1999

[14] A. Solymosi, U. Grude, Grundkurs Algorithmen und Datenstrukturen in Java, Vieweg Verlag, 2002

[15] E. Schöneburg, Genetische Algorithmen und Evolutionsstrategien, Addison-Wesley Verlag, 1993

[16] H. Pohlheim, Evolutionäre Algorithmen, Springer Verlag, 1999

[17] J. Orwant, Algorithmen in Perl, O'Reilly Verlag, 2000

[18] T. Peters, Rekursive Algorithmen, Dümmlers Verlag, 1989

[19] M. Lusti, Wissensbasierte Systeme, Algorithmen, Datenstrukturen und Werkzeuge, 1990

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2,

[20] G. F. Barth, Sensors and Sensing in Biology and Engeneering, Springer Verlag, 2003

[21] U. Küppers, H. Tributsch, Verpacktes Leben – verpackte Technik, Wiley-VCH, 2002

[22] W. Nachtigall, Bionik, Springer Verlag, 2002

[23] E. Börger, Berechenbarkeit, Komplexität, Logik, Vieweg Verlag, 1998

[24] M. Bretz, Algorithmen und Berechenbarkeit, Vieweg Verlag, 1992

[25] D. Drechsel, Regelbasierte Interpolation und Fuzzy Control, Vieweg Verlag, 1998

[26] E. Seiffart, K. Manteufel, Lineare Optimierung, Harri Deutsch Verlag, 1997

[27] H. Normann, Lineare Optimierung, ein Rezeptbuch, UTB Verlag, 1996

[28] C. Drösser, Fuzzy Logic, Methodische Einführung in krauses Denken, Rowohlt Taschenbuch Verlag, 1996

[29] L. Papula, Mathematik für Ingenieure und Naturwissenschaftler, Vieweg Verlag, 2001

[30] T. Ottmann, P. Widmayer, Algorithmen und Datenstrukturen, Spektrum Akademischer Verlag, 2002

[31] H. Nahrstedt, Programmieren von Maschinenelementen, Vieweg Verlag, 1985

[32] N. Boysen, Ameisenalgorithmus, Referat unter www.ameisenalgorithmus.de

[33] J. Dankert, Numerische Methoden der Mechanik, Springer Verlag, 1987

[34] M. Lawo, G. Thierauf, Stabtragwerke, Matrizenmethoden der Statik und Dynamik, Vieweg Verlag, 1986

[35] G. Engeln-Müllges, K. Niederdrenk, R. Wodicka, Springer Verlag, 2005

Index

Ada Lovelace 2
Adjungierte Matrix 117
Alan Turing 2
Algorithmeneigenschaften 3
Algorithmus 1
Ameisenalgorithmus 215
Arrays 176
Auftragsfolgenproblem 201

backtracking 205
Baugruppen-Stückliste 184
Baum 190
Bisektionsmethode 199
Black Box 6
Blechteilfläche 142

Chromosom 232
Crossover 231

Datenfeldern 176
Datenstrukturen 163
Defuzzyfizierung 240
Determinante 114
Deterministische Algorithmen 4
Differentialgleichungen 75
Differentialquotient 128
Differenzenquotient 128
Diskriminante 11
DNA 232

Drehschwingung 88
Eigenschwingung eines geraden Trägers 132
Eigenwertproblem 132
einschrittige Codes 207
Einseitig eingespannter Träger 129
Elektronenröhre 102
Engpaßproblem 165
Entscheidungsalgorithmen 5
Euklid 1
Euler-Cauchy-Verfahren 75

Evolutionsstrategie 224

Federkonstante 175
Fitnesstest 233
Fließbandproblem 165
Fritz Zwicky 8
Fuzzy Logic 237
Fuzzyfizierung 239
Fuzzy-Mengen 238
Fuzzy-Sets 238

Gaußsches Eliminationsverfahren 29
Genetische Algorithmen 5
Genetischer Algorithmus 230
gerichteten Graphen 189
Gewichtungsfaktoren 177
Gewöhnliche Differentialgleichung 75
Gleichung n-ten Grades 19
Gleichungen 11
Gozintographen 190
Graphentheorie 189
Gray-Code 206
Greedy-Methode 201

Härtebestimmung 11
Hash-Methode 201
Heuristik 7

Industrieofen 241
Inferenz 240
Instandhaltung 149
Inverse einer Matrix 116
Iterations-Verfahren nach von Mises 133
Iterativen Algorithmen 4
Jeep-Problem 212

Kanten 189
Knoten 189
Kognitive Algorithmen 5
Komplementäre Matrix 117
Korrelationsanalyse 169

H. Nahrstedt, *Algorithmen für Ingenieure,* DOI 10.1007/978-3-8348-1980-2,

Korrelationskoeffizienten 171
Kovarianz 171
Kubische Gleichungen 14
Künstliche Intelligenz 5
Kurbeltrieb 77
Kurt Gödel 2

Laplace-Gleichung 30
Laplace-Operator 96
Lebensdauer 149
Leibnitz 2
Lineare Gleichungssysteme 29
linguistische Werte 240
Listen 180

Maschinenbelegung 218
Maschinenwartung 144
Matrix 103
Matrizen 103
Matrizenaddition 106
Matrizenprodukt 112
Matrizensubtraktion 108
Maximales Volumen 25
Membrane 96
Minimaler Materialverbrauch 20
Monte Carlo Methode 139
Monte-Carlo-Methode 216
Morphologischen Kasten 8
Mutation 231, 234

Netzplantechnik 191
Numerische Algorithmen 4
Nutzwertanalyse 177

Operation Research 5
Optimierungsalgorithmen 5

Packproblem 231
Partielle Differentialgleichung 94
Permutation 163
PERT 191
Pheromonmatrix 216
Probabilistischer Simulation 144
Problemlösungen 6
Produktionsoptimierung 35
Pseudozufallszahlen 139
Pumpventile 148

Quadratische Gleichungen 11
Quicksort 181

Raimundus Lullus 2
Randomize-Funktion 142
Regressionsanalyse 169
Regressionsgeraden 171
Regula Falsi 19
Rekombinationsmethode 233
Rekursion 212
Rekursive Algorithmen 4
Rnd-Funktion 141
Rückverfolgung 205
Rückwärtsrechnen 212

Simplexmethode 35
Skalar 110
Skalarprodukt 110
Sortieralgorithmus auf Listen 180
Stabwerksoptimierung 225
Strukturstückliste 188
Stücklisten 183

Taylor-Reihenentwicklung 131
Teile und Herrsche 199
Temperaturverteilung in einem Kanal 30
Torsionspendel 87
Transponierte 103
Trichtervolumen 18

unscharfe Mengen 237
Varianzen 171
Vektoren 103
Verfahren nach Newton 25

Wahrscheinlichkeitsbetrachtungen 144
Weg eines Handelsreisenden 216

Zufallszahlen-Generator 141
Zuschnittoptimierung 43