Studienbücherei

I. O. Kerner

Numerische Mathematik mit Kleinstrechnern

VEB Deutscher Verlag der Wissenschaften

Mathematik für Lehrer
Band 18

Herausgegeben von:
W. Engel, S. Brehmer, M. Schneider, H. Wussing

Unter Mitarbeit von:
G. Asser, J. Böhm, J. Flachsmeyer, G. Geise, T. Glocke,
K. Härtig, G. Kasdorf, O. Krötenheerdt, H. Lugowski,
P. H. Müller, G. Porath

Studienbücherei

Numerische Mathematik mit Kleinstrechnern

I. O. Kerner

Mit 77 Abbildungen, 4 Bildtafeln
und 58 Tabellen

Zweite, überarbeitete Auflage

VEB Deutscher Verlag
der Wissenschaften
Berlin 1989

ISBN 3-326-00397-8

ISSN 0081-7384

Verlagslektor: Brigitte Mai
Verlagshersteller: Birgit Burkhardt
Umschlaggestaltung: Rudolf Wendt

Lizenz-Nr. 206· 435/75/89
Printed in the German Democratic Republic
Satz: VEB Druckhaus „Maxim Gorki", 7400 Altenburg
Offsetdruck und buchbinderische Verarbeitung: VEB Druckerei „Thomas Müntzer",
5820 Bad Langensalza
LSV 1084
Bestellnummer: 571 702 1
01980

Vorwort zur zweiten Auflage

Für den Autor ist es immer sehr erfreulich, wenn sein Buch schnellen Absatz findet. Dies zeigt den richtig eingeschätzten Bedarf und das Interesse der Nutzer. Er kann sicher sein, daß es auch gelesen und im Fall eines Lehrbuches auch benutzt wird. Vieles Lesen und Benutzen fördern dann naturgemäß auch zunächst verborgene Mängel ans Licht und erzeugen Hinweise zur Verbesserung. Solche erhielt ich mit großem Dank für den numerischen Teil von den Herren Prof. PIRL (Berlin) und ZIELKE (Halle) sowie für den Teil Mikroelektronik von den Herren Prof. GERDES (Rostock) und SCHMIDT (Dresden).

Der Abschnitt über Taschenrechner wurde nach den inzwischen selbst gewonnenen Erfahrungen in der Ausbildung ergänzt und geändert. Hier unterstützte mich Herr Dr. WINKLER (Dresden). Die Taschenrechner, vertreten durch den Typ SR 1, haben nun ihren Platz in der Schule ab Klasse 7. In der Lehrerausbildung muß ihr Einsatz von höherer Warte aus beleuchtet werden. Insbesondere gilt das für die Eigenarten des Rechnens mit begrenzter Stellenanzahl und für Taschenrechner-Resultatanzeigen, die oft von dem intern verwendeten Zahlsystem und der internen Stellenanzahl abweichen. Letzteres gilt sogar bei Kleincomputern und deren Programmiersprachen.

Seit dem Erscheinen der ersten Auflage ist die Entwicklung auch bezüglich des Einsatzes von Kleincomputern (KC 85/1 vom VEB Kombinat Robotron und KC 85/2 vom VEB Kombinat Mikroelektronik) in der Lehrerausbildung und sogar in Schulen in großen Schritten vorangekommen. Das Umsetzen der Programme in die BASIC-Version der genannten Computer ist sehr einfach. Lediglich bei den FOR-TO-NEXT-Zyklen ist zu beachten, daß die KC — ähnlich wie z. B. auch Commodore — den Zyklustest nachgestellt durchführen und daß Variablennamen nur aus zwei Zeichen bestehen dürfen.

Zum Buch sind von der Zentralstelle für Rationalisierte Lehrmittel (ZRL) Erfurt des Ministeriums für Volksbildung zwei Foliensätze und eine Programmkassette für den KC 85/1 beziehbar.

Dresden, Sommer 1987 I. O. KERNER

Aus dem Vorwort zur ersten Auflage

Um der Bedeutung der Numerischen Mathematik und des algorithmischen Aspekts in der Mathematik einerseits und der Entwicklung der Mikroelektronik sowie der damit zunehmenden Verbreitung der Taschenrechner und Videocomputer andererseits Rechnung zu tragen, erteilten die Herausgeber der Studienbücherei, Reihe „Mathematik für Lehrer", den Auftrag, einen Band des vorgelegten Titels abzufassen. Mathematiklehrer wissen aus der täglichen Praxis, daß der Taschenrechner schon nicht mehr im Mathematikunterricht ignoriert werden kann. Der 8. Pädagogische Kongreß 1978 hat zu der in diesem Zusammenhang entstehenden auch ökonomischen Problematik Stellung genommen und in den Grundzügen die künftigen Schritte fixiert. In einer Gruppe von Schulen des Bezirkes Dresden wurden methodisch-didaktische Schulversuche zum Einsatz von Taschenrechnern in 7. Klassen durchgeführt, und wissenschaftliche Untersuchungen im Auftrag der Akademie der Pädagogischen Wissenschaften zum „Für-und-Wider" und zum „Wann-Wo-Wie" liefen an bzw. wurden verstärkt fortgesetzt. Eine resultierende Maßnahme ist dieses Buch, denn bei einer sorgfältig abgewogenen Einführung und Nutzung der Taschenrechner in der Schule (ab 1984 in den EOS und ab 1985 in den POS) muß man wohl mit der Information der Lehrer und erst recht der zukünftigen Lehrer beginnen.

Unter Kleinstrechnern im Sinne des Titels verstehen wir Tisch- und Taschenrechner, wobei der Kreis bis auf die programmierbaren Heim- oder Videocomputer ausgedehnt wurde. Gerade die letztgenannten erlauben im Einsatzbereich der Numerischen Mathematik besondere Möglichkeiten der Demonstration. Darüber hinaus können sinnvoll die verschiedensten Probleme aus der ingenieurtechnischen oder ökonomischen Praxis oder auch aus anderen Gebieten bearbeitet werden. Da im wesentlichen Rechner des Typs „HC 900" bzw. „Z 9001" vom VEB Kombinat Mikroelektronik bzw. VEB Kombinat Robotron und ZX Spectrum von Sinclair Research Ltd. in den Darstellungen zugrunde gelegt wurden, hoffen wir auch auf eine Nutzung des Buches außerhalb der Volksbildung in Industrie, Forschung und Verwaltung — überall dort, wo diese nützlichen Geräte vorhanden sind. Derartige Kleinstrechentechnik erlaubt es, viele Aufgaben rascher und billiger zu bearbeiten, als es im Verkehr mit einem Rechenzentrum, welches größere Computer betreibt, möglich wäre. Das unvermeidliche Hin und Her bei der Programmentwicklung und Programmkorrektur mit der Einordnung in den Stapelbetrieb verursacht selbst bei einem Rechenzentrum im eigenen Betrieb oft tagelange und mitunter auch wochenlange Bearbeitungszyklen sogar für kleine Programme. Abhilfe wird erst zukünftig durch Datenendstellen und Dialogterminale geschaffen werden. Jetzt aber ist es durch Kleinstrechentechnik schon

möglich, sofort am Arbeitsplatz, in der Werkstatt oder sogar im freien Feld ein Programm zu erstellen und auch fehlerfrei zu machen, was je nach Problem eine Arbeit von Minuten oder wenigen Stunden ist. Die Kleinstrechentechnik hat sich auf diese Weise schon einen großen Freundes- und Nutzerkreis erworben und auch die Rechenzentren von vielen Miniproblemen befreit. Die Programmierung erfolgt in BASIC, ist also leider noch sehr maschinenorientiert. Das mindert die Universalität der im Buch enthaltenen Beispielprogramme, die bei einer mehr problemorientierten, wenn auch einfachen Sprache größer wäre. Kleinstrechentechnik mit problemorientierter Programmiersprache ist dringend erforderlich und würde eine effektivere Nutzung der Geräte ermöglichen.

Es kann nun gewiß nicht in kurzer Frist jede Schule mit einem Klassensatz Taschenrechner oder jedes mathematische Kabinett bzw. jeder Fachunterrichtsraum mit einem programmierbaren Tischrechner ausgestattet werden, ebensowenig wie es sehr bald Schulcomputer oder Terminalanschlüsse an Rechenzentren in den Schulen geben wird. Die jetzigen Lehrerstudenten und Absolventen werden aber rund 40 Jahre im Beruf stehen. Das heißt, ihr berufliches Wirken reicht beträchtlich in das nächste Jahrtausend hinein. Dies gilt um so mehr, wenn man bedenkt, daß sie dabei junge Menschen auf deren Tätigkeit im Arbeits- und Produktionsprozeß vorbereiten.

Im gegenwärtigen Zeitpunkt konzentrieren wir uns auf die Lehrerаus- und Weiterbildung. Die genannten Geräte finden sich deshalb an Universitäten und Hochschulen und werden selbstverständlich auch für die Ausbildung und Forschung anderer Disziplinen und Wissenschaften eingesetzt.

Im vorliegenden Buch wird besonders der dynamische algorithmische Aspekt der Numerischen Mathematik hervorgehoben. Es werden Lösungsverfahren vorgestellt und gegeneinander bezüglich der Effektivität abgewogen. Wegen der notwendigen Beschränkung des Umfangs mußte auf eine ausführliche Behandlung der analytischen oder besser funktionalanalytischen Seite verzichtet werden. Der künftige Lehrer soll durch dieses Hervorheben der numerischen Verfahrenstechnik angeregt werden, auch in der Schulmathematik nach Ansatzstellen zur Vermittlung einer algorithmischen Denk- und Arbeitsweise zu fahnden. In der Tat gibt es diese überaus zahlreich in allen Gebieten. Es sei hier nur auf die Möglichkeiten zur Klärung der Schülerfragen verwiesen: Wir berechnet man Quadratwurzeln, Kubikwurzeln oder höhere Wurzeln? Wie sind die Zahlenwerte in Logarithmen- und Funktionentafeln gefunden worden? Wie sind π und e auf so viele Stellen berechnet worden? Die Stoffauswahl aus der Numerischen Mathematik für den Inhalt dieses Buches wurde vom gültigen Lehrprogramm für die Mathematik-Fachlehrerausbildung bestimmt. Entsprechend dem Titel sind Kapitel zur Kleinstrechentechnik hinzugefügt. Diese reichen von einer Klassifizierung der Taschenrechnertypen über Hinweise, wie man mit einfacheren Geräten doch noch näherungsweise und mit relativ wenigen Tastungen zu den Werten einfacher transzendenter Funktionen kommt (Logarithmus, Exponentialfunktion, Winkelfunktionen, Arcusfunktionen) und wie in mikroelektronischen Geräten intern schnell, d. h. mit 2 bis 3 Divisions- oder Multiplikationszeiten, diese Werte berechnet werden, bis zu Informationen über die physikalisch-technische Seite der Mikroelektronik.

Der Taschenrechner im Mathematikunterricht könnte diesen revolutionierend verändern. Er ist in einfachen Formen sogar schon recht früh einsetzbar. Spielend erkennt der Schüler die Wirkung einfacher und elementarer arithmetischer Gesetze wie die der Kommutativität, der Assoziativität und der Distributivität. Er wird vertraut mit der Dezimalbruchdarstellung der Zahlenwerte. Die Übungsphasen können bereichert werden mit dem, was wirklich geübt werden soll, da belastende arithmetische Kleinarbeit vom Taschenrechner erledigt wird. Auch die Abschnitte über Tafelarbeit, Rechenstab und logarithmisches Rechnen könnten gekürzt, eventuell sogar gestrichen werden. Der Wegfall von Tafeln und Rechenschiebern kompensiert teilweise die Kosten eines Taschenrechners. Behandelte Aufgaben werden wirklichkeitsnäher. Es können auch „krumme" Eingangsdaten verwendet werden. Rechengang und Resultate brauchen nicht mehr „aufzugehen", was bisher das einzige Mittel der Aufgabenkonstrukteure war, die eigentlich unwesentliche arithmetische Arbeit zu reduzieren. Damit entfällt für den Schüler das unsachliche Kriterium „aufgegangen" für die Korrektheit seiner Lösung.

Ich möchte nur noch zu einem Argument der Skeptiker, Warner und Pessimisten Stellung nehmen, welches lautet: „Mit dem Einsatz der Taschenrechner wird die ohnehin in weiten Kreisen der Bevölkerung sehr gering entwickelte Fähigkeit zu arithmetischer Arbeit noch mehr verkümmern. Man wird abhängig von einem Hilfsmittel. Versagt dieses oder ist nicht zur Hand, so kann man überhaupt nicht mehr selbst rechnen." An diesem Argument ist im letzten Teil etwas Wahres. Ich bin aber mit vielen Fachkollegen der Meinung, daß mit den Taschenrechnern die Bereitschaft zum Rechnen vielmehr zunehmen wird, da die wegen der arithmetischen Arbeit abschreckende Wirkung vieler Probleme verblaßt.

Bei der Herstellung und Abfassung des Textes habe ich viel Unterstützung und Hilfe erhalten. Vor allem danke ich Herrn Prof. Dr. W. Engel, Rostock, für die grundlegende Anregung sowie Herrn Prof. Dr. G. Maess, Rostock, und Herrn Prof. Dr. M. Schneider, Karl-Marx-Stadt, für zahlreiche Ratschläge und verbessernde Hinweise. Herrn Prof. Dr. H. Gerdes, Rostock, danke ich für die Durchsicht des Abschnitts über Mikroelektronik. Meinen Kollegen von der Pädagogischen Hochschule Dresden, insbesondere Herrn Dr. H. Bauch, Frau Dr. E. Stahl (für die Durchsicht des Abschnittes Statistik) sowie bezüglich schulpraktischer und schulmethodischer Informationen Herrn Dr. S. Schneider, habe ich für Hinweise und Diskussionen zu mathematischen Teilfragen herzlich zu danken.

Die Herstellung des Typoskripts besorgten fleißig und mit persönlichem Engagement Frau W. Schmidt, Frau E. Furker und meine Frau. Auch ihnen sei für diese nicht immer leichte, öfteren Änderungen unterworfene Arbeit herzlich gedankt. Ebensolcher Dank gebührt für das Anfertigen vieler Zeichnungen den Mitarbeitern Frau M. Schmidt, Herrn G. Mokronowski und Frau H. Tanner. Schließlich ist den Mitarbeitern des Verlages, vertreten in Frau B. Mai, für die verständnisvolle Zusammenarbeit und den Mitarbeitern der Druckerei für die sorgfältige Arbeit im Prozeß der Herstellung aufrichtig zu danken.

Dresden, im Dezember 1986 Immo O. Kerner

Inhalt

1. Einführung

1.1. Algorithmisches Denken in der Schulmathematik

Bei der Durchsicht der Mathematiklehrbücher und Lehrpläne für Schulen bemerkt man bald eine Bevorzugung der funktionalen (und zwar in expliziter Formelgestalt) und formelmäßigen Zusammenhänge, und demzufolge wird auch diese Denkweise entwickelt und gefördert. Nach einer langen „euklidischen" Periode im Schulmathematikunterricht wurde diese nach einer Forderung von FELIX KLEIN um 1900 durch Einführung und Förderung des funktionalen Denkens abgelöst. Entsprechend der stürmischen Entwicklung der Mathematik im 18. und 19. Jahrhundert, insbesondere der Analysis, war das auch eine zwingende und folgerichtige Entscheidung. Jetzt aber stehen wir offenbar wiederum an der Schwelle einer Änderung — oder besser Ergänzung — des mathematischen Denkens und eben auch der Schulmathematik. Diese legt in jungen Menschen den Grundstein ihrer mathematischen Bildung, und für sehr große Teile der Bevölkerung vermittelt lediglich die Schule einen Einblick in die Mathematik als ein Ergebnis menschlicher Geistes- und Denkgeschichte überhaupt.

Es ist Anliegen und Aufgabe der Schule, den jungen Menschen in verschiedenen ausgewählten Wissensgebieten die Ergebnisse menschlichen Forschens und menschlicher Arbeit bleibend und nachhaltig zu vermitteln, um sie für das Leben vorzubereiten und ihnen ein Weltbild zu geben, welches der Realität nach dem gegenwärtigen Erkenntnisstand möglichst gut entspricht.

In Gesellschaftswissenschaften und Naturwissenschaften folgt man sehr genau dieser Richtlinie. Beispielsweise haben sich die Lehrinhalte der Biologie um die Mikrobiologie, der Physik und Chemie um Kernphysik, Raketentechnik und hochpolymere Kunststoffe erweitert.

Auch in der Mathematik hat man Reformen im Schulstoff durchgeführt. Beispielsweise wurde eine Umstellung des Grundwissens auf die Mengenlehre vorgenommen. Das entsprach völlig einem modernen Stand der mathematischen Wissenschaft und spiegelte sich in gleichartigen Prozessen in anderen Ländern wider.

1.1.1. Algebraische Gleichungen

Durch die Entwicklung von Hochleistungsgeräten für die praktische numerische Arbeit (elektronischen Rechengeräten oder Datenverarbeitungsanlagen) im Wechselspiel zwischen Anforderung und Angebot hat sich auch eine Rückkopplung auf die Mathematik selbst ergeben. Während man in der Schule jedenfalls oft zufrieden ist, wenn man für eine Aufgabenklasse eine „Lösungsformel" entwickelt hat, in welche

dann aktuelle Aufgabenparameter lediglich einzusetzen sind, werden in programmgesteuerten Automaten „Rechenverfahren" oder Algorithmen abgearbeitet. Ein typisches Beispiel für das Formeldenken ist die Vermittlung der Lösungsformel für quadratische Gleichungen:

$$x^2 + px + q = 0,$$

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \quad \text{mit} \quad p^2 - 4q \geqq 0.$$

Im Lehrplan werden natürlich Durchführen und Üben der Herleitungen und der Beweise zur Festigung exakter mathematisch logischer Denkprozesse gefordert. In der Praxis bleibt aber lediglich die Formel im Gedächtnis.

Viele mathematische und vor allem auch wirklichkeitsnahe Probleme entziehen sich jedoch einer formelmäßigen Behandlung oder Lösung. Das kann man sogar — und jedem Schüler verständlich — bereits am Problem der Gleichungsauflösung zeigen. Die lineare Gleichung

$$ax + b = 0, \qquad a \neq 0,$$

$$x = -\frac{b}{a},$$

hat eine beinahe triviale Lösung. Dies wird auch schon in der 6. Klasse gelehrt. Für die quadratische Gleichung ist die Lösungsformel oben angegeben. Sie ist im Stoffplan der 9. Klasse enthalten. Für die kubische Gleichung

$$x^3 + ax^2 + bx + c = 0$$

und die Gleichung vierten Grades

$$x^4 + ax^3 + bx^2 + cx + d = 0$$

gibt es noch Lösungsformeln, allerdings keine, in die man einfach die Koeffizienten einsetzt, sondern es sind Formelsätze, die in mehreren Schritten und mit Fallunterscheidungen abzuarbeiten sind. Sie bilden auch keinen Bestandteil der Schulmathematik mehr ([29], S. 260). Für die allgemeine Gleichung fünften oder höheren Grades weiß man seit Galois (1831) und fast gleichzeitig durch Abel, daß sie durch „Wurzelziehen formelmäßig" nicht lösbar sind. Sie besitzen aber im konkreten Fall Lösungen, wie es nach dem Fundamentalsatz von Gauss (1799) bekannt ist. Man kann sie auch berechnen, aber eben i. a. nicht durch eine Formel, sondern durch einen Algorithmus, der i. a. Näherungswerte ergibt. Man braucht also gar nicht zur höheren Analysis wie Integralen und Differentialgleichungen zu greifen, wenn man die Notwendigkeit von mathematischen Verfahren den Formeln gegenüberstellen will.

Mit der Berechnung von Näherungswerten, dem Abschätzen verbliebener Restfehler und dem Verhalten der Berechnungsalgorithmen befaßt sich die Numerische Mathematik. Durch die Existenz von Rechenautomaten kann man auch real die tat-

sächliche Rechnung ohne ermüdende langwierige, oft fehlerbehaftete menschliche Arbeit durchführen, und durch die Existenz kleiner leistungsstarker mikroelektronischer Taschen- und Tischrechner kann man diese Rechnungen selbst daheim, im Schulzimmer oder am Arbeitsplatz ohne oft langdauernde Konsultationen, ökonomische und verwaltungstechnische Vorbereitungen, viel Hin und Her und Wartezeiten bei einem Rechenzentrum erledigen bzw. vorführen.

1.1.2. QUICKSORT und Sortieren durch Mischen

Abgesehen von den numerischen Algorithmen, deren Herleitung und Vorführung natürlich die Entwicklung einer algorithmischen Denkweise bei jungen Menschen unterstützt, gibt es auch viele nichtnumerische Aufgaben, die im täglichen Leben eine mehr oder minder große Bedeutung besitzen, deren Behandlung aber in der Schulmathematik unterbleibt, weil es für sie keine Lösungsformeln, sondern Lösungsalgorithmen gibt, wobei auch die zur Verfügung stehende Zeit eine Rolle spielt. Ein solches Problem und Standardbeispiel ist das Sortieren der Elemente einer Menge nach einem vergleichbaren Merkmal (in einer Ordnungsrelation), im einfachsten Fall von Zahlen. Sortiert wird überall. Mit sortierten Mengen kommt man ständig in Berührung: Telefonbuch, Kontonummern, Materialkennzahlen, KFZ-Nummern, Patientennummern, Kundennummer beim Postzeitungsvertrieb, bei der Energieversorgung, Personalkennzahlen u. v. m. Die Aufgabenstellung des Sortierens ist jedem Schüler verständlich, wird sie doch für sehr kleine Mengen schon im Vorschulalter trainiert, und auch bei wenig Interesse für Mathematik werden Kartenspiele mit darin enthaltenen Sortiervorgängen fleißig und mit Ausdauer geübt. Einen einfachen Sortieralgorithmus, z. B. auf der Basis der Auswahl des Minimalelements, auszudenken, zu formulieren oder zumindest intuitiv auszuführen gelingt wohl allen Schülern. Hier kann man ansetzen und den Aufwand A untersuchen, der quadratisch von der Anzahl n der Elemente abhängt:

$$A \sim n^2.$$

Von Interesse ist es dann, Sortieralgorithmen zu zeigen, die wesentlich weniger Arbeit verursachen, nämlich z. B. QUICKSORT oder Sortieren durch Mischen mit dem Aufwand (im Mittel)

$$A \sim n \times \lg n.$$

Zwei Beispiele an dieser Stelle genügen zur Erläuterung dieser Verfahren.

QUICKSORT (quick (engl.) = schnell, Schnellsortieren)

- Von einer gegebenen zu sortierenden Elementefolge wird das erste als „Trennelement" genommen, und alle anderen Elemente werden davor (wenn sie kleiner oder gleich sind) oder dahinter (wenn sie größer sind) angeordnet. Das Trennelement steht dann an der richtigen Stelle.
- Mit den beiden nun entstandenen Teilfolgen, welche durch das Trennelement getrennt sind, wird ebenso verfahren.

— Eine Teilfolge aus nur einem Element ist natürlich fertig sortiert.
— Eine Teilfolge aus zwei Elementen benötigt zum Sortieren nur noch einen Vergleich.

8	12	5	14	3	10	4	7	15	6	11	2	13	1	6	9
5	3	4	7	6	2	1	6	8	12	14	10	15	11	13	9
3	4	2	1	5	7	6	6	8	10	11	9	12	14	15	13
2	1	3	4	5	6	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	6	7	8	9	10	11	12	13	14	15

Die Trennelemente sind in Quadrate eingefaßt, und die Trennung der Teilfolgen wurde durch senkrechte Striche gekennzeichnet.

Die Tragweite der Aufwandformel wird besonders bei großem n deutlich:

n	n^2	$n \lg n$	$n^2 : n \lg n$
100	10000	200	50fach
1000	10^6	3000	300fach
1000000	10^{12}	$6 \cdot 10^6$	160000fach

Sortieren durch Mischen

Der Name des Verfahrens erzeugt zunächst beim Schüler Schmunzeln und Verwunderung, da er vom Kartenspielen gewöhnt ist, daß durch Mischen gerade das Gegenteil des Sortierens eintritt, nämlich eine möglichst große Unordnung. Ähnlich ist es auch beim Betonmischen für die Bestandteile Zement, Sand und Wasser. Hier aber handelt es sich um einen gesteuerten Mischvorgang zweier bereits sortierter Teilmengen, den man besser durch das Wort „einordnen" kennzeichnen sollte.

— Von einer gegebenen zu sortierenden Elementefolge werden Zweierfolgen aus aufeinanderfolgenden Elementen gebildet und diese entsprechend jeweils einem Vergleich durch evtl. Tauschen sortiert.
— Zwei benachbarte Teilfolgen werden gemischt.
— Wenn eine Folge der beiden zu mischenden ausgeschöpft ist, kann der Rest der anderen einfach übernommen werden.

8	:	12	5	:	14	3	:	10	4	:	7	15	:	6	11	:	2	13	:	1	6	:	9
8	12	:	5	14	:	3	10	:	4	7	:	6	15	:	2	11	:	1	13	:	6	9	
5	8	12	14	:	3	4	7	10	:	2	6	11	15	:	1	6	9	13					
3	4	5	7	8	10	12	14	:	1	2	6	6	9	11	13	15							
1	2	3	4	5	6	6	7	8	9	10	11	12	13	14	15								

Der Mischvorgang (zweite Anweisung des Algorithmus) bedarf vielleicht noch einer Erläuterung. Im obigen Zahlenfeld sind die zu mischenden Partner eines Paares benachbarter Teilfolgen durch einen Doppelpunkt getrennt. Die Paare selbst sind durch senkrechte Striche voneinander getrennt.

Als Beispiel für das Mischen zweier Folgen betrachte man das Paar

5 8 12 14 : 3 4 7 10,

d. h. also den Übergang von der dritten zur vierten Zeile im Schema. Man stelle sich die beiden Teilfolgen als zwei Warteschlangen vor einem Tor vor, an dem jeweils das kleinere Element

von den unmittelbar vor dem Tor stehenden eingelassen wird. Dann entstehen die folgenden Arbeitsphasen:

0. Schritt (Anfang)		⌉5 8 12 14 ⌋3 4 7 10
1. Schritt	3	⌉5 8 12 14 ⌋4 7 10
2. Schritt	3 4	⌉5 8 12 14 ⌋7 10
3. Schritt	3 4 5	⌉8 12 14 ⌋7 10
. . . . .		
7. Schritt	3 4 5 7 8 10 12	⌉14 ⌋

Im 8. Schritt ist dann das Ende erreicht. Man erkennt auch die natürliche Wirkung der dritten Anweisung aus dem Algorithmus: Wenn alle Elemente einer Warteschlange durch das Tor getreten sind (hier nach dem 6. Schritt), kann der Rest der anderen Schlange einfach hindurchtreten (hier die Elemente 12 und 14).

1.1.3. Damenproblem

Eine weitere wichtige algorithmisch lösbare Aufgabenklasse ohne Einsatz der Numerik ist das sogenannte „Aufgabenlösen" selbst. Ein sehr schönes und einprägsames Beispiel dafür ist das *Damenproblem* oder *Königinnenproblem* aus der Theorie des Schachspiels: Auf einem Schachbrett sind acht Königinnen so zu stellen, daß sie sich gegenseitig nicht schlagen! Wieviel solche Stellungen gibt es? Man weiß heute, daß es 92 Stellungen gibt, von denen einige allerdings geometrisch äquivalent sind, da sie durch Drehung und Spiegelung an verschiedenen Achsen auseinander hervorgehen. Diese Aufgabe wurde 1848 in einer Schachzeitung gestellt. GAUSS befaßte sich mit ihr und gab 72 Lösungen an. Im Jahre 1850 wurde die Gesamtanzahl der Lösungen bekannt. Nach einem verhältnismäßig einfachen Algorithmus kann man alle Stellungen finden (man verwende die acht Bauern symbolisch als Damen):

0. In der Reihe a stelle eine Dame auf a1.
1. In der nächsten Reihe (anfangs also b) stelle eine Dame auf das erste mögliche Freifeld (nicht bedroht von den bereits stehenden Damen, anfangs also b3).
2. Wiederhole Schritt 1, soweit man mit den Reihen kommt.
3. Wird Reihe h erfolgreich erreicht, so ist eine Stellung gefunden; dann findet man die nächste Stellung durch Aufsuchen des nächsten Freifeldes in Reihe h.
4. Gibt es in einer Reihe kein Freifeld mehr, so geht man eine Reihe zurück und sucht dort nach dem nächsten Freifeld.
5. Das Ende ist erreicht, wenn es in der Reihe a für die erste Dame kein Freifeld mehr gibt.

Nach diesem Algorithmus findet man in fünf bis zehn Minuten die erste Stellung

a	b	c	d	e	f	g	h
1	5	8	6	3	7	2	4

durch manuelles Nacharbeiten der Schritte (Abb. 1.1 und 1.2). Der Algorithmus läßt sich programmieren, und ein Rechenautomat findet in kürzester Zeit alle 92 Stellungen, so daß heute mit moderner Rechentechnik und algorithmischer Denkweise leicht und schnell eine Aufgabe gelöst werden kann, an der GAUSS scheiterte. Dieser Vergleich ist zwar dreist, aber durch die inhaltliche Maßlosigkeit der Formulierung wird doch eine reale Tendenz deutlich. Moderne Technik steigert nicht nur in Physik, Chemie, Biologie die wissenschaftlichen Potenzen, sondern auch in der Mathematik.

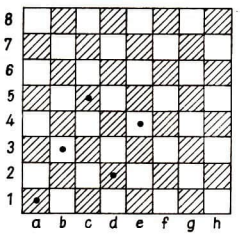


Abb. 1.1. Erste Stellung der Damen nach dem Algorithmus, bei der in Reihe f keine Dame mehr gestellt werden kann

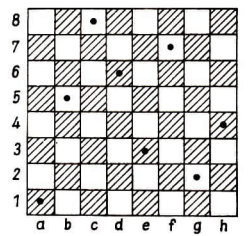


Abb. 1.2. Erste auffindbare Stellung von acht Damen

1.1.4. Größter gemeinsamer Teiler, Euklidischer Algorithmus

Wo findet man heute in der Schulmathematik Ansätze zum algorithmischen Denken oder direkt Algorithmen?

Neben der Tatsache, daß jedes Ausführen einer arithmetischen Operation bereits das Abarbeiten eines Algorithmus ist und daß auch das Einsetzen von Parametern und Ausführen von Formelrechnen einen Ablauf darstellt, sind Algorithmen vorwiegend in den Konstruktionsbeschreibungen der Geometrie zu finden. Die zeichnerische oder konstruktive Geometrie widersetzt sich naturgemäß einer formelmäßigen Behandlung. In anderen mathematischen Teilgebieten sind Algorithmen im Stoffplan der Schule nur spärlich vertreten. Es gibt aber ein historisch bedingtes Standardbeispiel, welches einen Einstieg ermöglicht. Das ist der Euklidische Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier ganzer Zahlen. Zur Lösung dieser Aufgabe gibt es keine Formel, sondern eben nur eine Arbeitsvorschrift, deren Befolgung am Ende das gewünschte Resultat liefert. Da dieser Algorithmus im Mathematiklehrbuch Klasse 6, in MfL Bd. 1, S. 144, und MfL Bd. 9, S. 18, ausführlich dargestellt ist, werden hier lediglich einige Ergänzungen angefügt, und zwar aus der Sicht praktischer rechentechnischer Algorithmenabarbeitung. Ohne Formelanwendung kann man den Algorithmus verbal so formulieren:

0. Der größte gemeinsame Teiler zweier natürlicher Zahlen a und b werde mit GGT(a, b) oder $a \sqcap b$ bezeichnet (die erste Schreibweise gibt ihn als Wert einer Funktion von zwei Parametern oder Argumenten an und die zweite als Wert einer Formel mit dem Operator $\sqcap$).

1. Die größere beider Zahlen (der Dividend) werde modulo der kleineren (dem Divisor) zerlegt und der Rest bestimmt (natürlich ist der Rest kleiner als der Divisor).
2. Schritt 1 wird wiederholt, wobei der Divisor zum Dividenden und der Rest zum Divisor gemacht wird.
3. Der Prozeß endet, sobald ein Rest verschwindet, d. h. sobald die für die Moduloperation nötige Division aufgeht. Der letzte von Null verschiedene Rest ist dann der gesuchte Wert von GGT(a, b) oder von $a \sqcap b$.

Zur rechentechnischen Realisierung genügen die vier arithmetischen Grundoperationen nur schlecht. Man benötigt entweder eine sogenannte *ganzzahlige Division* (Operanden ganzzahlig, Resultat ganzzahliger Anteil des gewöhnlichen Quotienten (in ALGOL 60 durch das Zeichen $\div$ oder ***div*** dargestellt)) oder die Operation *ganzer Anteil* einer Zahl (in ALGOL 60 durch die Standardfunktion ***entier*** (x), vgl. MfL Bd. 9, 3.1., in der Mathematik durch Gauß-Klammern $[x]$ und bei vielen Taschenrechnern durch die Taste „int x", integer (engl.) = ganzzahlig, dargestellt). Dann errechnet man den Rest

$$r = a - a \,\mathbf{div}\, b * b \quad \text{oder} \quad r = a - \mathbf{entier}\,(a/b) * b.$$

Im Zahlenbeispiel $a = 144$ und $b = 32$ ist

$$r = 144 - 144 \,\mathbf{div}\, 32 * 32 = 144 - 4 * 32 = 144 - 128$$
$$= 16$$

oder

$$r = 144 - \mathbf{entier}\,(144/32) * 32 = 144 - \mathbf{entier}\,(4.5) * 32$$
$$= 144 - 4 * 32,$$

dann weiter wie oben.

Man lasse sich bitte durch das Benutzen verschiedener Bezeichnungen mit gleicher Bedeutung nicht verwirren. In der Praxis treten diese Bezeichnungen alle auf. Noch einfacher wird das Verfahren, wenn man durch das verwendete Rechenhilfsmittel eine Modulo- oder Restoperation ***mod*** angeboten erhält:

$$r = a \,\mathbf{mod}\, b,$$
$$r = 144 \,\mathbf{mod}\, 32 = 16.$$

Der Algorithmus lautet dann sehr einfach:

1. Bilde einen Rest

   $r := a \,\mathbf{mod}\, b$;

   wenn $r = 0$, dann ist

   GGT $:= b$ und STOP,

   sonst Schritt 2.

2. Nimm als neues a das alte b ($a := b$), als neues b den in Schritt 1 errechneten Rest ($b := r$) und wiederhole Schritt 1.

Dies liefert bei Ausführung das Rechenschema in Form einer Tabelle:

a	b	$r = a \,\mathbf{mod}\, b$	
144	32	16	letzter Rest ungleich Null
32	16	0	ist GGT(144, 32) oder $144 \sqcap 32$

Dabei geben die Pfeile das Übernehmen der Zahlenwerte an. Hierbei ist es noch lästig, jede Zahl zweimal zu schreiben. Es genügt sogar eine Spalte r, wobei sich ein neuer Wert jeweils aus zwei unmittelbar vorhergehenden ergibt (Rekursion oder Iteration),

$$r_i := r_{i-2} \; \mathbf{mod} \; r_{i-1},$$

mit $r_1 := a$ und $r_2 := b$:

i	r_i	
1	144	
2	32	
3	16	letzter Rest ungleich Null ist GGT(144, 32) oder $144 \sqcap 32$
4	0	

Der Index i ist dabei kein Vektorkomponentenindex, sondern ein Rekursions- oder Iterationsindex. Man braucht jeweils nur die letzten drei Zeilen der letzten Tabelle oder die letzte Zeile der dreispaltigen Tabelle, die anderen können vergessen werden. Das ist von Bedeutung für den Speicherbedarf bei evtl. Programmieren.

Die Restbildung wird vielleicht als kompliziert empfunden, besonders von Schülern. Durch Rückführung oder Erinnerung an den Zusammenhang der Division mit der einfachen Subtraktion kann man den Rest sehr einfach gewinnen. Dieser Gedanke geht auf NICOMACHUS (um 100) zurück. Der Divisor wird, so oft es geht, vom Dividenden abgezogen, und dann verbleibt der Rest, sobald der Dividend kleiner als der Divisor geworden ist. Zur Fortsetzung wird dann einfach die Rolle beider Werte vertauscht.

Dies gibt für das Zahlenbeispiel die wirklich sehr einfache Tabelle

$a := a - b$	b	
144	32	
112		
80		
48		
16		
32	16	letzter Rest ungleich Null ist GGT(144, 32)
16		
0		

Hier kann der Einwand erwartet werden: Ist es nicht möglich, daß unangenehm viele Subtraktionen bis zur Feststellung eines neuen Restes nötig werden? Möglich ist das schon, aber man untersuche, wie häufig solch eine unangenehme Zahlenkombination auftritt. Derartige Abschätzungen sind auch typisch für die Untersuchung der Effektivität von Algorithmen. Es sind dies die Verhaltensuntersuchungen für ungünstig gelagerte Fälle (worst case behaviour).

Abb. 1.3 zeigt den Anteil des Gebietes aller Punktkoordinaten-Kombinationen $a > b$ (Winkelraum 45°), bei denen höchstens zwei Subtraktionen nötig sind. Natürlich sind nur die ganzzahligen Gitterpunkte zu betrachten. Aber als Maß für ihre relative Häufigkeit wird das Verhältnis der Öffnungswinkel genommen, was eine Abschätzung liefert. Es ist

$$\arctan 3 = 71.57°,$$

woraus sich der Öffnungswinkel des schraffierten Gebietes zu 26.57° ergibt und damit schließlich das Verhältnis 0.59. Also in 59% aller Fälle, d. h. mehr als der Hälfte, genügen zwei Subtraktionen. Daraus kann man schließen, daß der vereinfachte Algorithmus ohne Division auch effektiv genug ist.

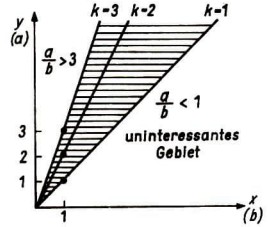


Abb. 1.3. Zur Abschätzung der benötigten Subtraktionen bei der Bestimmung des größten gemeinsamen Teilers von a und b. Für Wertekombinationen (a, b) im schraffierten Gebiet sind höchstens zwei Subtraktionen nötig. Eingezeichnet sind die Geraden $a = kb$ mit $k = 1, 2, 3$

Ein Programm (Tastenfolge) z. B. für den Taschenrechner MR 610 ist leicht zu überlegen:

1. b ($< a$)	eintasten
2. x → M	b in den Speicher führen
3. CE	Eingabe (b) löschen
4. a	eintasten
5. −	
6. MR	Speicherrückruf von b
7. =	$a - b$ in der Anzeige; wenn $a - b >$ b, dann Wiederholung ab Schritt 5, wenn $a - b = b$, dann ist GGT $= b$, STOP, wenn $0 < a - b < b$, dann Schritt 8
8. x ↔ M	Speichertausch mit Anzeige, d. h., nun übernimmt $a - b$ die Rolle von a, weiter bei Schritt 5.

Hier muß der ausführende Mensch durch Tastendrücken und Beachten der Fälle in Schritt 7 den Algorithmus selbst steuern und ausführen. Der Taschenrechner nimmt ihm lediglich die aufwendigen Arbeiten des Subtrahierens und Notierens von Zwischenwerten ab.

Ein echtes Programm mit automatisch ablaufenden Zyklen und Entscheidungen ist das folgende in der Programmiersprache BASIC (einige Informationen über BASIC werden in 1.3. gegeben), welches bei Kenntnis einiger englischer Wörter „von selbst" verständlich ist:

```
10 INPUT a, b
20 IF a = b THEN GOTO 100
30 IF a < b THEN GOTO 60
40 LET a = a - b
50 GOTO 20
60 LET h = a
70 LET a = b
80 LET b = h
90 GOTO 40
100 PRINT " GGT = "; a
110 STOP
```

Die Zeilen 60 bis 80 (im Programm werden die Nullen zur Unterscheidung vom Großbuchstaben O durchstrichen) besorgen das Vertauschen des mittlerweile veränderten a mit b. Dazu ist eine Hilfsgröße h notwendig. Die — bei manchen BASIC-Versionen weglaßbare — Befehlsangabe LET sorgt für die Deutung des Gleichheitszeichens als dynamisches Zuweisungs- bzw. Ergibtzeichen (in ALGOL und PASCAL als := geschrieben).

Natürlich ist in einem GGT-Programm nach NICOMACHUS keine Sicherheit dagegen gegeben, doch in einen Fall mit sehr vielen Subtraktionen zu geraten. So liefert

$$a = 2000000000 \quad \text{und} \quad b = 999999999$$

nach zweimaliger Subtraktion des b die neuen Werte

$$a = 999999999 \quad \text{und} \quad b = 2,$$

wonach dann 499999999-mal die zwei zu subtrahieren wäre, was auch in Heimcomputern sehr lange dauert. Besonders schnell arbeitet das nun folgende Programm. Man sollte unbedingt versuchen, es zu verstehen. Es benutzt die BASIC-Funktion INT für „ganzer Anteil von" und stellt mit ihrer Hilfe den Rest der Division her:

```
10 INPUT a, b
20 LET a = a - b * INT (a/b)
30 IF a = 0 THEN GOTO 60
40 LET b = b - a * INT (b/a)
50 IF b < > 0 THEN GOTO 20
60 PRINT a + b
70 STOP
```

1.2. Diskrete Arithmetik. Zahlendarstellung und numerische Effekte

Im Schulunterricht wird der Schüler im Fach Mathematik beim Kennenlernen der Zahlen und des Rechnens schrittweise von den natürlichen (ganzen positiven) Zahlen über negative und rationale Zahlen (Brüche) zu den reellen geführt. Der zuerst dünn besetzte Zahlenraum wird immer mehr gefüllt und schließlich zum Kontinuum. Dabei bleibt der letzte Schritt zu den reellen Zahlen problematisch. Einmal sind die Mittel der Einführung, der Dedekindsche Schnitt oder die Intervallschachtelung, den Schülern fremd und damit dem Lehrer unlieb, und zum anderen kommt der Schulunterricht nicht recht über das bloße Einführen hinaus. Jede praktische arithmetische Arbeit verwendet dann doch nur rationale Zahlen in der Form endlicher Dezimalbrüche. Es ist natürlich unbestritten, daß als Abrundung des Zahlbegriffs reelle Zahlen nötig sind. Sie sind aber ein abstrakter Begriff, auch wenn am Beispiel von $\sqrt{2}$ im Unterricht nachgewiesen wird, daß dies als Länge der Diagonale eines Quadrats der Seitenlänge 1 auftritt. Man kann mit ihnen symbolhaft $(\sqrt{2}, \pi, e)$ hantieren und, wie es in der Analysis geschieht, Weiteres fruchtbringend aufbauen. Die gesamte Lehre von den reellen Funktionen, der Begriff der Stetigkeit, der Differential- und Integralkalkül werden für das Kontinuum der reellen Zahlen entwickelt. (In der Topologie kommt man beim Aufbau des Begriffs der Stetigkeit auch ohne reelle

Zahlen aus.) Aber unsere arithmetische Darstellung von Zahlen in Form der Dezimalbruchschreibweise erlaubt es nicht, eine reelle nichtrationale Zahl $(\sqrt{2}, \pi, e)$ aufzuschreiben. Wir sind gezwungen, die Notation an irgendeiner Stelle abzubrechen. Uns sind nicht einmal „weit hinten" liegende Stellen bekannt. Wir könnten sie berechnen, aber wir tun es wegen der Nutzlosigkeit nicht. Im praktischen Leben rechnen wir sogar nur mit einer Teilmenge der rationalen Zahlen, mit zwei-, drei-, ..., zehn-, zwölfstelligen Dezimalbrüchen, je nach verlangter Genauigkeit. Die arithmetischen Hilfsmittel im mathematischen Schulunterricht begrenzten bisher die Rechengenauigkeit auf drei (Rechenstab) und vier (Tafelwerk) Stellen. Die Umständlichkeit ihrer Handhabung begrenzten darüber hinaus Umfang und Häufigkeit arithmetischen Trainings. Dieses ertrank schon bei halbwegs praktischen Beispielen (in den Aufgaben) in ermüdender und eigentlich stupider Ziffernrechnerei, in Stabschieben und Tafelblättern.

Erst der Taschenrechner bringt hier im Unterricht eine Abhilfe. Er erlaubt die Intensivierung der Übungsphase jedes Stoffabschnitts. Er bietet die Möglichkeit wirklichkeitsnaher und ungekünstelter Zahlenbeispiele und macht dem Schüler das unsachliche Kriterium des „Aufgehens" einer Rechnung für die Korrektheit seiner Arbeit zunichte. Er ist für arithmetisch Unbegabte, die es zweifelsfrei bei sonst guter mathematischer Veranlagung gibt, eine Arbeitsprothese, wie sonst eine Brille, ein Hörgerät, ein Gebiß oder ein künstliches Bein.

Soviel Lobendes man also über den Einsatz des Taschenrechners in der Schule sagen kann, man muß sich über gewisse arithmetische Konsequenzen völlig klar sein. Dies gilt besonders für den leitenden und den Bildungsprozeß führenden Lehrer.

1.2.1. Computerzahlen

Alle Computer und auch die Taschenrechner müssen sich in ihrer Zahlendarstellung auf eine kleine endliche Anzahl von Ziffernstellen beschränken. Üblich sind 8, 10 und 12 dezimale Ziffern. Wenn die Rechner auch im Innern, in der unserem Zugriff entzogenen internen Logik, dezimal arbeiten, entsteht zunächst kein weiteres Problem. Das der Schaltungstechnik am besten angepaßte Zahlensystem ist jedoch das Dualsystem, und viele Rechner arbeiten intern in diesem System. Nur im Verkehr mit dem bedienenden Menschen, also in der Ein- und Ausgabe, wird dann das Dezimalsystem verwendet. Durch die Umrechnung von einem System in das andere (Konvertierung) entstehen Fehler. Zum Beispiel liefern alle Brüche, in deren Nennern nur die Primfaktoren 2 und 5 auftreten, abbrechende Dezimalbrüche. Wenn sie nicht zu lang sind, haben sie eine gute Chance der exakten Darstellung in einem Dezimalcomputer. Aber nur die Brüche mit Zweierpotenzen im Nenner brechen im Dualsystem ab. Bei allen anderen periodischen muß im Rechner die Darstellung künstlich beendet werden, sei es durch Abbruch oder sei es durch eine Rundung. Auf alle Fälle entsteht ein Fehler. Bei Mikroprozessoren wird durchweg das Dualsystem bevorzugt, und zwar meist nur mit acht Stellen (1 Byte), gelegentlich mit 16. In größeren Rechnern findet man 16, 24, 32, 48 oder 64 Dualstellen, und das dezimale Äquivalent reicht von 4 oder 5 bis zu 16 bis 20 Ziffernstellen.

Festpunktdarstellung

Der Rechner verwende m Ziffern zur Darstellung einer Zahl, ein Vorzeichen und einen Trennungspunkt für das Trennen von ganzem und gebrochenem Anteil. Als Ziffernvorrat werde (0, 1, ..., 9), also das Dezimalsystem benutzt. Wenn die Lage des Trennungspunktes fixiert oder festgelegt ist, spricht man von *Festpunktdarstellung*.

Im englischen Sprachbereich — und von dort kam fast die gesamte Technologie der Computer — wird ein Dezimalpunkt anstelle eines Dezimalkommas verwendet. Im Schulunterricht gebräuchlich ist

1000000,36 oder 1 000 000,36,

dagegen in englischer Literatur mit anderer Benutzung des Kommas

1,000,000.36 oder 1000000.36,

und diese letzte Form gilt für Computer.

In der Computerarithmetik und sogar in der Numerischen Mathematik hat es sich eingebürgert, den Dezimalpunkt zu verwenden. In älterer deutscher Literatur findet man auch Festkommadarstellung. Der Punkt bietet unbestreitbare Vorteile. Bei einer Funktion $f(x, y)$ mit mehreren Variablen wird das Komma zur Trennung in der Liste der Variablen verwendet. Bei aktuellen konkreten Variablenwerten $f(3.4, 7.2)$ kann bei Benutzung des Dezimalpunktes kein Zuordnungsproblem entstehen bzw. man braucht nicht plötzlich doch zum Semikolon als Trennzeichen überzugehen.

Der Einfachheit halber (und weil es im Prinzip gleichgültig ist) wird der feste Dezimalpunkt ganz an den Anfang gelegt. Ist beispielsweise $m = 4$, d. h., vier Stellen nach dem Dezimalpunkt stehen zur Verfügung, so gibt es für den Betrag einer darstellbaren Zahl z die Möglichkeiten

0.0000
0.0001
⋮
0.9999

Die Null vor dem Punkt ist ohne Informationsgehalt und könnte wegbleiben. In der Anzeige vieler Taschenrechner wird sie auch weggelassen. Es ist also allgemein

$$0 \leqq |z| \leqq 1 - 10^{-m} < 1$$

mit einem endlich großen (wenn auch kleinen) Abstand

$$\Delta z = 10^{-m}$$

zwischen benachbarten darstellbaren Zahlen.

Auf der Zahlengeraden liegen die Computerzahlen (positiv, negativ und Null) *diskret*. Sie bilden eine endliche Menge, deren Mächtigkeit $2 \cdot 10^m - 1$ ist. Sie ist als Menge abbrechender m-stelliger Dezimalbrüche eine endliche Teilmenge der rationalen Zahlen, die ihrerseits ja unendlich groß, wenn auch abzählbar ist. Für die Absolutbeträge gibt es deshalb ein kleinstes Element und ein größtes Element:

$$|z_{\min}| = 0, \qquad |z_{\max}| = 1 - 10^{-m}.$$

Es wird noch einmal ausdrücklich darauf verwiesen, daß die darstellbaren Computerzahlen diskret auf der Zahlengeraden liegen (Abb. 1.4), während die Gesamtheit der rationalen Zahlen dort zwar *dicht*, aber noch nicht *kontinuierlich* liegt (vgl. MfL Bd. 4). In einem solchen Zahlenbereich sind nicht mehr alle arithmetischen

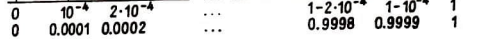


Abb. 1.4. Die diskret auf der Zahlengeraden liegenden Computerzahlen der Festpunktdarstellung (Dezimalsystem, vier Ziffern, die Eins gehört nicht mehr zum Bereich)

Operationen ausführbar. Addition, Subtraktion, Multiplikation und Division rationaler Zahlen ergeben wieder eine rationale Zahl. Die Menge der rationalen Zahlen bildet einen Körper. Für die Computerzahlen der Festpunktdarstellung gilt das nicht:

$0.5802 + 0.8217 \rightarrow$ Überlauf, Zahlbereich wird verlassen

$0.3249 + 0.2134 = 0.5383$ liegt im Zahlbereich

$0.5802 \times 0.8217 = 0.47675034$ ist nicht darstellbar, muß auf 0.4768 verkürzt werden.

(Weitere Beispiele und Folgerungen siehe 1.2.2.)

Ein Überlauf wird wohl immer zum Abbruch der Rechnung führen. Das Verkürzen verletzt im allgemeinen elementare arithmetische Gesetzmäßigkeiten, die den Schülern vermittelt wurden und die nun bei dem sonst so wunderbaren Hilfsmittel Taschenrechner nicht mehr voll gültig sind. Solche verletzbaren Gesetze sind bereits das Assoziativgesetz der Addition bzw. der Multiplikation

$$(a + b) + c = a + (b + c),$$
$$(a \times b) \times c = a \times (b \times c),$$

und das Distributivgesetz

$$a \times (b + c) = a \times b + a \times c.$$

Im nächsten Abschnitt werden dazu Beispiele angegeben. Als Folge der Verletzung solcher elementaren Gesetze, also letztlich als Folge der diskreten Arithmetik, stellen sich dann auch Abweichungen von tieferliegenden mathematischen Aussagen beim numerischen Rechnen ein. Ein den Schüler verblüffendes und ihm verständliches Beispiel wird durch die folgenden leicht überprüfbaren Identitäten geliefert:

$$99 - 70\sqrt{2} = \frac{1}{99 + 70\sqrt{2}} = \frac{1}{(1+\sqrt{2})^6}.$$

(Den ersten Übergang erreicht man durch das sogenannte *Rationalmachen*, das in der Schule oft als Anwendung und Übung der Formel $(a + b)(a - b) = a^2 - b^2$ auftritt, und den zweiten durch Ausmultiplizieren und Anwendung der Wurzelgesetze bzw. durch Anwendung des binomischen Satzes und Anwendung der Wurzelgesetze.) Rechnet man zehnstellig, so erhält man (je nach verwendeter Rundungsmethode auch andere) mit

$$0.005\,050\,660\,000, \qquad 0.005\,050\,633\,885, \qquad 0.005\,050\,633\,890$$

zwar unterschiedliche Ergebnisse, aber die Fehler liegen für praktische Anwendungen doch „genügend weit hinten". Rechnet man fünfstellig (beim Nachprüfen nicht zehnstellig rechnen und nur das Endresultat fünfstellig anzeigen lassen, sondern nach jeder Operation fünfstellig verkürzen), so ergibt sich

$$0.006\,0000, \qquad 0.005\,0507, \qquad 0.005\,0508.$$

Fünfstelliges Rechnen bedeutet „5 Stellen nach der ersten geltenden Ziffer"! Bei dreistelliger Rechnung, wie sie bei Verwendung eines Rechenstabes nötig wäre, erhält man sogar

$$0.300, \qquad 0.005\,06, \qquad 0.005\,06,$$

wobei der erste Wert um zwei Größenordnungen falsch wird und gegenüber dem zweiten und dritten den 30000fachen Fehler zeigt.

Dieses Beispiel macht sogar aufmerksam auf die Tatsache, daß das im Schulunterricht empfohlene und bei Resultatangaben oft geforderte Rationalmachen, wenn sich dabei Differenzen ergeben, numerisch gesehen geradezu schädlich ist. Man sollte auf das Vermeiden von Differenzen fast gleichgroßer ($99 \approx 70\sqrt{2} = 98.994\,949\,34$) Zahlen orientieren und die Schüler auf entsprechende Umformungen trainieren:

Für große x-Werte:

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

$$\frac{1}{x+1} + \frac{1}{x-1} - \frac{2}{x} = \frac{2}{x(x^2-1)},$$

$$\sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}},$$

$$\sqrt{\sqrt{x+1}} - \sqrt{\sqrt{x}} = \frac{1}{\left(\sqrt{\sqrt{x+1}} + \sqrt{\sqrt{x}}\right)\left(\sqrt{x+1} + \sqrt{x}\right)}.$$

Beim Subtrahieren fast gleichgroßer Zahlen in endlich genauer Zahlendarstellung heben sich die vorderen bedeutsamen Stellen weg, was man *Stellenauslöschung* nennt, und nur die letzten Stellen verbleiben. Da diese oft ungenau sind, kann das Ergebnis bis zur Unkenntlichkeit verfälscht werden. Bei nachfolgender Multiplikation mit einer genügend großen Zahl werden wieder alle Stellen „aufgefüllt", und eine nicht mehr vorhandene Genauigkeit wird vorgetäuscht. (Weiteres dazu siehe in 1.2.2.)

Gleitpunktdarstellung

Bei der Festpunktdarstellung ist man auf den Zahlenbereich $(-1, +1)$ beschränkt, was für viele Aufgaben sehr lästig ist, da man die Zahlenwerte (auch alle Zwischenresultate) so transformieren muß, daß sie in diesen Bereich fallen. Unter dem Namen „Kommaautomatik" bieten einfachere Taschenrechner die Fähigkeit an, das Komma — oder besser gesagt den Dezimalpunkt — innerhalb der zur Verfügung stehenden Ziffern zu verschieben und bei Resultaten an der richtigen Stelle zu liefern. Ist etwa wieder $m = 4$ die Stellenanzahl, so können nun Zahlen der Form

$$\left.\begin{matrix}.0000\\.0001\\\vdots\\.9999\end{matrix}\right\} \Delta z = 10^{-4}$$

$$\left.\begin{matrix}1.000\\1.001\\\vdots\\9.999\end{matrix}\right\} \Delta z = 10^{-3}$$

$$\left.\begin{matrix}10.00\\10.01\\\vdots\\99.99\end{matrix}\right\} \Delta z = 10^{-2}$$

$$\left.\begin{matrix}100.0\\100.1\\\vdots\\999.9\end{matrix}\right\} \Delta z = 10^{-1}$$

$$\left.\begin{matrix}1000.\\1001.\\\vdots\\9999.\end{matrix}\right\} \Delta z = 1$$

dargestellt werden. Der Zahlbereich ist erheblich vergrößert:

$$0 \leqq |z| \leqq 10^m - 1.$$

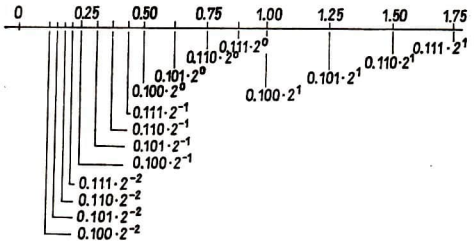
Die Zahlen liegen weiterhin diskret auf der Zahlengerade, aber nicht mehr äquidistant. Die Abstände schwanken zwischen

$$\Delta z_{\min} = 10^{-m} \quad \text{und} \quad \Delta z_{\max} = 1,$$

wobei sie sich von Stufe zu Stufe um den Faktor 10 (Basis des verwendeten Zahlsystems) vergrößern. Um Null herum liegen die Zahlen recht dicht, aber an den Grenzen des Zahlbereichs verhältnismäßig dünn, so daß dort ziemlich große Abschnitte der Zahlengeraden bei notwendigen Verkürzungen von Resultatziffernfolgen auf eine darstellbare Zahl zusammengezogen werden müssen (Abb. 1.5).

Der Gedanke der Kommaautomatik, also des verschiebbaren oder gleitenden Kommas, ist verallgemeinerungsfähig. Man kann das Komma, d. h. den Dezimal-

punkt, über den Rand der Ziffernfolge hinaus verlagern, wenn man einen Skalierungsfaktor in Form einer Zehnerpotenz (oder Potenz der Basis des verwendeten Zahlensystems) hinzufügt.



1.5. Die diskret auf der Zahlengeraden liegenden Computerzahlen der Gleitpunktdarstellung. Um eine maßstäbliche Zeichnung zu ermöglichen, wurde als Basis 2, also das Dualsystem, als Ziffernanzahl $m = 3$ und als Exponentenbereich $-2 \leqq E \leqq 1$ gewählt

Man erreicht damit eine Zahlendarstellung, die Schülern bereits aus dem Chemie- oder Physikunterricht bekannt sein dürfte:

Loschmidtsche Zahl $= 6.022 \cdot 10^{23}\ \mathrm{mol}^{-1}$,

Plancksches Wirkungsquantum $= 6.626 \cdot 10^{-34}\ \mathrm{J \cdot s}$

als Anzahl der Moleküle in einem Molvolumen (bei Gasen 22.4 l) bzw. Proportionalitätsfaktor h für die Energie der Lichtquanten $E = h\nu$, wobei ν die Frequenz des Lichtes ist.

Da die Basis des verwendeten Zahlsystems immer bekannt ist, seine ständige Angabe demnach redundant ist, wird sie in der Regel bei der internen Zahlendarstellung im Computer weggelassen. Auch bei einer Zahlenanzeige oder beim Druck fehlt meist die Basis. Es erscheint lediglich der Exponent, zur besseren Erkennung wohl etwas abgesetzt, z. B.

6.022 23, 6.626 −34

für die beiden oben angegebenen Zahlenwerte. Dieser Exponent ist auch die Kennzahl (ganzer Anteil) des Logarithmus der darzustellenden Zahl, und deshalb ist für diese Schreibweise auch der Name *halblogarithmische Darstellung* bekannt, halblogarithmisch deshalb, weil von der Ziffernfolge selbst kein Logarithmus gebildet wird. Dessen ungeachtet wird die Ziffernfolge trotzdem *Mantisse* wie beim Logarithmus genannt. Diese Gleitpunktdarstellung ist nicht eindeutig, z. B. bedeuten die Angaben

6.022 23, 60.22 22, 602.2 21

alle dieselbe Zahl. Diese Mehrdeutigkeit wird in der Regel durch eine zusätzliche *Normalisierungsvorschrift* beseitigt. Meist verlangt man, daß die Mantisse M dem Betrag nach im Bereich oder Intervall $0.1 \leqq |M| < 1$ oder $1 \leqq |M| < 10$ liegt. Das

bedeutet entweder, daß der ganze Anteil Null ist und die erste Ziffer nach dem Dezimalpunkt verschieden von 0 sein muß oder aber daß der ganze Anteil eine von 0 verschiedene Ziffer sein muß. Die beiden Beispielzahlen würden nach diesen Vorschriften die Formen

$$0.6022\ \ 24, \qquad 0.6626\ \ -33$$

oder, wie schon gehabt,

$$6.022\ \ 23, \qquad 6.626\ \ -34$$

annehmen. Im ersten Fall wird dann oft noch die überflüssige, weil stets vorhandene und damit redundante Null weggelassen:

$$.6022\ \ 24, \qquad .6626\ \ -27$$

In Taschenrechnern bevorzugt man die zweite Form mit einer echten Ziffernstelle vor dem Dezimalpunkt. Allerdings geht durch diese Normalisierung die Darstellungsmöglichkeit für die Null verloren. Diese bekommt eine Sonderstellung und wird mit der Mantisse Null und dem Exponenten Null angegeben. Es sind auch andere Exponentenwerte dabei üblich, wodurch „Nullen unterschiedlicher Genauigkeit" angezeigt werden können. Als Mantissenlängen sind 6, 8, 10 und 12 Stellen üblich und als Exponentenangaben zwei Stellen, so daß der Exponentenbereich $-99 \leqq E \leqq +99$ ist.

Werden bei Computern als Exponentenbereich E verwunderliche „krumme" Angaben gemacht, z. B. $-19 \leqq E \leqq +19$, so verbirgt sich dahinter meist eine interne Zahlendarstellung im Dual- oder Oktalsystem, welche sechs Dualstellen bzw. zwei Oktalstellen für den Exponenten reserviert,

$$\begin{aligned} -111111 \leqq E \leqq +111111 \quad & \text{dual,} \\ -77 \leqq E \leqq +77 \quad & \text{oktal} \\ (-63 \leqq E \leqq +63 \quad & \text{dezimal),} \end{aligned}$$

und es ist $2^{63} \approx 10^{19}$.

1.2.2. Numerische Effekte

Durch den Exponenten kann also der Zahlenbereich so umfangreich konstruiert werden, daß für die meisten praktischen Rechnungen keine Überschreitungen auftreten und man relativ ohne Sorge einen arithmetischen Prozeß ablaufen lassen kann. Diese Sorglosigkeit gilt aber nur relativ, denn manchmal äußert sich ein Computer trotz fehlerfreier Bedienung und technisch fehlerfreier Arbeit in den Schaltelementen mit gänzlich falschen und damit offensichtlich falschen Resultaten. Dieser Fall ist noch der angenehmste, denn offensichtlich falsche Ergebnisse kann man leicht aussortieren. Die Angelegenheit wird sehr problematisch bei verfälschten Resultaten, die noch im Plausibilitätsbereich liegen, aber hohe Fehler aufweisen. Leider ist das möglich, und diese Tatsache verlangt vom Computerbenutzer — auch vom Anwender der Taschen- und Tischrechner — eine kritische Einstellung, ein gutes numerisches Verständnis und ständige Resultatkontrolle. Gewarnt sei vor kritikloser Akzeptierung

der Computerergebnisse und blinder Computer- bzw. Elektronikgläubigkeit! (Einige Beispiele zum geschilderten Sachverhalt wurden bereits in 1.2.1. vorgeführt.)

Letzten Endes rührt dieses Verhalten der Rechnerarithmetiken vom diskreten und endlichen Zahlenraum her. Die Mathematik, genauer die reelle Analysis, ist auf der Basis des Kontinuums entwickelt worden. Elementare Rechengesetze und auch tieferliegende bzw. höhere Aussagen legen den Körper der reellen Zahlen zugrunde. Im Computer arbeitet man mit einer Teilmenge der rationalen Zahlen, die keinen Körper im strengen Sinne bildet, obwohl sie dafür weitgehend ein gutes Modell abgibt. Gerade die an sich so vorteilhafte Gleitpunktdarstellung der Zahlen sorgt bei den höheren Exponenten für eine gähnende Leere auf der Zahlengeraden. Nimmt man als Beispiel zehn Mantissenstellen an, so ist

$$\Delta z_{\max} = 0.0000000001 \times 10^{99} = 10^{89}$$

der größtmögliche Abstand zweier „benachbarter" Zahlen. Selbst in der Längeneinheit Ångström (10^{-7} mm) gezählt, wäre dies noch ein Abstand von 10^{63} Lichtjahren, läge also in Bereichen der Abmessungen unserer Galaxis oder gar des uns bekannten oder optisch erreichbaren Universums. Das ist natürlich weit vom Kontinuum entfernt. Ungeheuere Bereiche müssen durch Rundung zusammengezogen werden. Die Rundungsvorschrift gewinnt dadurch eine eminente Bedeutung, die

Mantisse	Exp
−4 3 5 0 8	+0 3
+6 0 2 1 4	+0 1

−4 3 5 0 8	+0 3
+0 0 6 0 2	+0 3

Abb. 1.6. Ausrichten der Operanden und Verlust von Ziffern bei Gleitpunktaddition

weit über ihre vertraute mindere Wirkung im täglichen Leben hinausgeht. Durch die nun existierenden Computerarithmetiken praktisch angeregt, beginnen Mathematiker die zugehörigen algebraischen Strukturen, nun keine Körper mehr, sondern deshalb *Raster* genannt, genauer zu untersuchen [30, 33].

Im Zahlenbeispiel der Abb. 1.6 sind die Ziffern 1 4 der Mantisse des absolut kleineren Operanden verloren und werden auch durch Rundung nicht gerettet. Es ist also numerisch stets besser, erst mehrere kleine Zahlen für sich zu summieren und dann diese Summe zu einer größeren Zahl zu addieren, als jede der kleinen Zahlen einzeln zur größeren Zahl zuzuschlagen.

Wie das Beispiel aus Abb. 1.6 deutlich macht, denke man bei jeder Addition und Subtraktion daran, daß ein sogenanntes *Ausrichten* der Ziffernfolgen zur Herstellung gleicher Exponenten nötig ist. Dadurch werden vom kleineren Operanden Ziffern abgestoßen, gehen also für die weitere Rechnung verloren. Lediglich durch eine Rundung finden sie noch gewisse Berücksichtigung.

Abb. 1.7 zeigt ein Operandenraster für zweistellige (dyadische) Operationen. Nur durch die Gitterpunkte werden Operandenkombinationen angezeigt. Man sieht dadurch deutlich den Unterschied zu dem das ganze Quadrat lückenlos ausfüllende und darüber bis ins Unendliche reichende zweidimensionale Operandenkontinuum der Analysis.

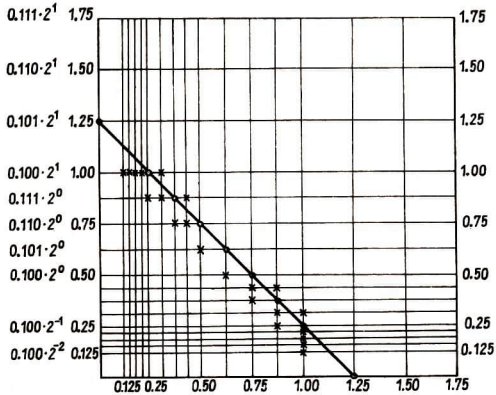


Abb. 1.7. Ein Operandenraster für zweistellige Operationen. Eingetragen ist die Gerade $x + y = 1.25$ im Kontinuum. Im Raster ergeben sich neun exakte Summen dieses Wertes (○), und 22 entstehen durch Rundung (×)

Damit die zeichnerische Darstellung technisch möglich ist, wurde eine Gleitpunktdarstellung mit drei Stellen im Dualsystem und dem Exponentenbereich $-2 \leqq E \leqq 1$ gewählt, wie sie bereits auch in Abb. 1.5 gezeigt wurde.

Eingetragen ist die Gerade

$$x + y = 1.25,$$

aber im Zahlenraster ist es keine Gerade, sondern nur eine endliche Punktmenge aus neun Elementen. Durch Rundung wird 22 weiteren Operandenkombinationen, die im Kontinuum nicht die Summe 1.25 bilden, ebenfalls dieser Wert als Summe zugeordnet. Dabei wird bei diesem Beispiel trotz symmetrischer Rundungsvorschrift nur viermal abgerundet.

Beispiele

1. *Verletzung des Assoziativgesetzes der Addition*

$$(a + b) + c = a + (b + c).$$

Bei dezimaler vierstelliger Rechnung mit

$$a = 0.4872 \cdot 10^2, \quad b = 0.4671 \cdot 10^{-2}, \quad c = 0.4023 \cdot 10^{-2}$$

und bei Rundung ergibt sich

$$\text{links } 0.4872 \cdot 10^2 \quad \text{und} \quad \text{rechts } 0.4873 \cdot 10^2.$$

Der Unterschied scheint unbedeutend, ist jedoch eine Übertretung des Gesetzes.

2. *Verletzung des Assoziativgesetzes oder Kommutativgesetzes der Addition im großen (global).* Die Summe

$$S = \sum_{k=1}^{n} \frac{1}{k}$$

werde für verschiedene n „von vorn", d. h. beginnend mit $k = 1$, und „von hinten", d. h. beginnend mit $k = n$, berechnet. Bei zehnstelliger Rechnung ergibt sich

n	S(vorwärts)	S(rückwärts)
10	2.928968254	2.928968254
100	5.187377520	5.187377519
1000	5.485470857	5.485470865

Dabei ist die Rückwärtssumme korrekter; bei ihr werden nicht so viele Ziffern abgestoßen, da auch die Zwischensummen vorerst noch klein sind.

3. *Numerisches Differenzieren, Grenzwertbildung.* Es wird in der Analysis gelehrt, daß der Differentialquotient der Grenzwert des Differenzenquotienten ist,

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x},$$

und daß man aus Berechnungen des Differenzenquotienten um so bessere Näherungswerte des Differentialquotienten erhält, je kleiner man Δx wählt. Beim Einsatz von Taschenrechnern ist man geneigt, dies experimentell nachzuprüfen oder vorzuführen. In [29] findet man für die Funktion

$$y = x,$$

deren Ableitung konstant gleich 1 ist, die numerischen Werte für die Ableitung an der Stelle $x_0 = 1$.

Die Computerrechnung versucht, den Grenzübergang

$$y'_0 = \lim_{h \to 0} \frac{(x_0 + h) - x_0}{h} = 1$$

numerisch zu simulieren. In Abb. 1.8 wird der qualitative Verlauf von

$$W = \frac{(1 + h) - 1}{h}$$

im Bereich von $h = 10^{-9}$ bei zehnstelliger Rechnung gezeigt. Im Zähler gehen Ziffern der Darstellungen für h gegenüber 1 verloren, wenn als Rundung einfach Abbruch genommen wird — anderenfalls kann bei Aufrundung auch W größer als 1 werden. Unterhalb von 10^{-9} wird sogar $W = 0$.

Es handelt sich hierbei auch wieder um die bereits als gefährlich genannte Subtraktion fast gleichgroßer Zahlen.

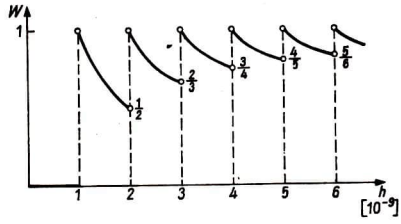


Abb. 1.8. Numerisch simuliertes Differenzieren von $y = x$ an der Stelle $x_0 = 1$

4. *Eine Aussage über alternierende Reihen.* Die Reihenentwicklung für e^x,

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots,$$

ist bekanntlich beständig konvergent, d. h., sie konvergiert für alle x. Für negative x wird daraus eine alternierende Reihe. Für diese gilt [35], daß der Abbrechfehler kleiner als der Betrag des ersten vernachlässigten Gliedes ist. Diese Aussage scheint für numerische Berechnungen besonders günstig zu sein, braucht man doch nur den Betrag des nächsten Reihengliedes zu beobachten. Wenn er unter ein vorgegebenes ε sinkt, welches die Rechengenauigkeit bestimmt (beispielsweise 10^{-8}), so bricht man ab und könnte sicher sein, daß dann auch das Resultat die gewünschte Genauigkeit besitzt.

Tab. 1.1 zeigt die tatsächlich auf diese Weise erhaltenen Resultate [29, 30]. Zwischen -4 und -5 wird erstmalig die analytisch-mathematische Aussage über das Fehlerverhalten nicht mehr bestätigt. Bei -12 ist keine Ziffer des Resultats mehr richtig. Bei -13 steigen die Werte der alternierenden Reihe sogar wieder an, und bei -50 wird ein Ergebnis geliefert, das um 33 Größenordnungen falsch ist. Alles das liegt natürlich an dem verwendeten Rechenraster und der diskreten, hier 10stelligen Gleitpunktarithmetik. Bei der mathematischen, vom Kontinuum

Tab. 1.1. Vergleich exakter, d. h. auf neun geltende Ziffern gerundeter Werte von e^x mit denen aus der alternierenden Reihe und dem analytisch gesicherten Abbruchkriterium. Es wurde 10stellig mit der Genauigkeitsschranke 10^{-8} gerechnet. Bei der angegebenen Anzahl von Gliedern konnte abgebrochen werden

x	alternierende Reihe	e^x exakt	absoluter Fehler	Anzahl der Glieder
-1	$3.6787944 \cdot 10^{-1}$	$3.6787944 \cdot 10^{-1}$	10^{-8}	12
-10	$4.4802480 \cdot 10^{-5}$	$4.5399929 \cdot 10^{-5}$	$6 \cdot 10^{-7}$	41
-11	$1.7772109 \cdot 10^{-5}$	$1.6701700 \cdot 10^{-5}$	10^{-6}	43
-12	$3.6000740 \cdot 10^{-6}$	$6.1442123 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	46
-13	$4.7107442 \cdot 10^{-6}$	$2.2603294 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	49
-20	$2.1825293 \cdot 10^{-3}$	$2.0611536 \cdot 10^{-9}$	$2 \cdot 10^{-3}$	69
-50	$1.1589190 \cdot 10^{11}$	$1.9287498 \cdot 10^{-22}$	$2 \cdot 10^{11}$	151

der reellen Zahlen ausgehenden Formulierung wurde stillschweigend vorausgesetzt, daß alle zur Berechnung benutzten Zahlen beliebig genau vorliegen. Um z. B. $e^{-50} = 1.929 \cdot 10^{-22}$ zu berechnen, müssen alle Werte und Zwischenresultate auf mindestens 22 Stellen nach dem Dezimalpunkt bekannt sein. Die Reihenglieder $(-50)^k/k!$ wachsen aber erst einmal bis zum fünfzigsten Glied ständig an. Es wird dabei eine Größe von $2.92 \cdot 10^{20}$ erreicht. Diese 21 Ziffern vor dem Dezimalpunkt und weitere 21 danach müssen sich während der Rechnung aufheben, damit das korrekte Ergebnis erzielt werden kann. Also müßte man für eine korrekte Resultatziffer etwa 43stellig arbeiten. Tatsächlich wurde 10stellig gerechnet, und somit wird der zunächst verwunderlich erscheinende Fehler von 33 Größenordnungen plausibel.

Für eine numerisch richtige Abbruchbedingung kann die numerisch richtige Angabe der Größenordnung G des Abbrechfehlers Δ

$$G(\Delta) = G\big(\max(|g_n|, |z|/10^a)\big)$$

verwendet werden, wobei g_n das erste vernachlässigte Glied ist, z der größte Zwischensummand der Rechnung und a die Stellenzahl in der verwendeten dezimalen Gleitpunktarithmetik. Erst wenn a sehr groß ist, reduziert sich diese Regel auf die mathematisch-analytische Formulierung, denn dann ist immer $|g_n|$ der größere Partner.

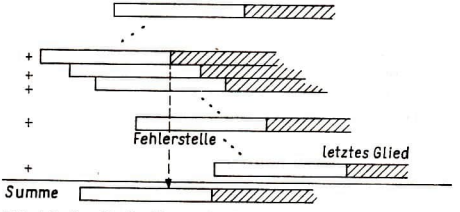


Abb. 1.9. Graphische Veranschaulichung der Entstehung des Fehlers beim Abbruch alternierender konvergenter Reihen unter Verwendung einer Gleitpunkt-Rasterarithmetik

Die Skizze aus Abb. 1.9 zeigt, daß das letzte Glied bereits kleiner als die Wertigkeit der letzten Ziffer der Summe ist. Trotzdem liegt der Fehler wesentlich weiter vorn, weil von einem Zwischenglied unbekannte Ziffern (schraffiert) auf diese Stelle wirken. Tatsächlich berechnet der numerische Praktiker e-Potenzen mit negativem Exponenten nicht nach der alternierenden Reihe, sondern er berechnet die e-Potenz mit dem entsprechenden positiven Exponenten und bildet davon den Kehrwert gemäß $e^{-x} = 1/e^x$.

Die Beispiele für die Auswirkung der diskreten oder Rasterarithmetik im Gegensatz zur Arithmetik und Analysis im Kontinuum der reellen Zahlen lassen sich beliebig vermehren. Mit der Vorführung einiger davon sollte eine Warnung vor kritikloser Anwendung moderner Rechengeräte, auch vor kleinen Taschenrechnern, ausgesprochen werden. Bei allen praktischen Anwendungen sind Fehleruntersuchungen notwendig, um die geschilderten Effekte zu erkennen und evtl. zu vermeiden.

1.3. Charakterisierung von Taschenrechnern

1.3.0. Formelschreibweisen

Im täglichen Leben und im Beruf sind wir heutzutage bei der Durchführung arithmetischer Rechnungen auf das sogenannte „Infixsystem" geprägt. Dies bedeutet, daß die arithmetischen Operatoren ($+$, $-$, $\times$ oder $\cdot$, / oder :) *zwischen* den Operanden stehen, z. B.

$$a + b, \qquad a \times b,$$

was historisch, durch jahrhundertelange Gewöhnung entstanden und schließlich durch unsere Schulbildung auf uns gekommen ist. Taschenrechner, welche diese Gewöhnung des rechnenden Menschen berücksichtigen, damit auch einen großen Interessentenkreis ansprechen und für den Einsatz in der Schule vielleicht als besonders geeignet erscheinen, verwenden auch die Infixmethode, die gewöhnlich als *algebraische Logik* bezeichnet wird. Die einfacheren von ihnen lassen dann aber aus Gründen eines billigeren inneren Systems (weniger aufwendige Steuerung) die uns ebenfalls sehr vertrauten Vorrangbeziehungen (Hierarchien) der Rechenarten unberücksichtigt, es sind dies die in 1.3.1. betrachteten Rechner.

Es gibt jedoch auch andere Formelschreibweisen, und weil diese Einfluß auf bestimmte Taschenrechnertypen haben, müssen sie und die zugehörige Umwandlung hier behandelt werden. Es handelt sich um die *Präfix-* und die *Postfixsysteme*, bei denen die Operatoren *vor* bzw. *nach* den Operanden stehen.

Normale Infixdarstellung:

$$a + b \times c \quad \text{bzw.} \quad (a + b) \times c$$

Funktionsschreibweise:

$$\mathrm{ADD}(a, \mathrm{MUL}(b, c)) \quad \text{bzw.} \quad \mathrm{MUL}(\mathrm{ADD}(a, b), c)$$

Daraus Präfixdarstellung durch Weglassen der „technischen" Zeichen und Operatorverwendung:

$$+\, a \times b\, c \quad \text{bzw.} \quad \times + a\, b\, c$$

Methodisches Hilfsmittel zur Umwandlung sind die Rechenbäume und die zugehörigen sogenannten *Durchläufe* (Abb. 1.10). An jeden Knoten stößt man dreimal: Das erste und das dritte Berühren liefert die Reihenfolgen

$$\begin{array}{l} +\ a \times b\, c \\ \times + a\ b\, c \end{array} \quad \text{bzw.} \quad \begin{array}{l} a\, b\ c \times + \\ a\, b + c\ \times \end{array}$$

also die Präfix- bzw. die Postfixdarstellung.

Für Arbeitsgruppen oder fakultative Kurse ist der folgende Hinweis gedacht: Präfix- und Postfixdarstellungen von Formeln sind klammerfrei und benötigen keine Vorrangregeln (Hierarchien).

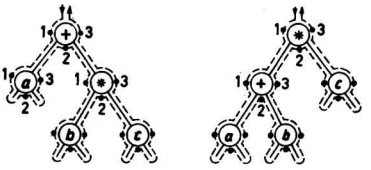


Abb. 1.10. Rechenbäume mit Durchläufen

Interessant ist das Umschreiben bekannter Gesetzmäßigkeiten in diese Form:
Kommutativgesetze

$$a\,b\,+\,=\,b\,a\,+ \qquad a\,b\,+\,b\,a\,+\,=$$
$$a\,b\,*\,=\,b\,a\,* \quad \text{oder sogar} \quad a\,b\,*\,b\,a\,*\,=$$

Assoziativgesetze

$$a\,b\,+\,c\,+\,=\,a\,b\,c\,+\,+ \qquad a\,b\,+\,c\,+\,a\,b\,c\,+\,+\,=$$
$$a\,b\,*\,c\,*\,=\,a\,b\,c\,*\,* \quad \text{oder sogar} \quad a\,b\,*\,c\,*\,a\,b\,c\,*\,*\,=$$

Distributivgesetz

$$a\,b\,+\,c\,*\,=\,a\,c\,*\,b\,c\,*\,+ \quad \text{oder sogar} \quad a\,b\,+\,c\,*\,a\,c\,*\,b\,c\,*\,=$$

Binomische Formel

$$a\,b\,+\,2\uparrow\,=\,a\,2\uparrow\,2\,a\,*\,b\,*\,+\,b\,2\uparrow\,+$$
$$\text{oder sogar} \quad a\,b\,+\,2\uparrow\,a\,2\uparrow\,2\,a\,*\,b\,*\,+\,b\,2\uparrow\,+\,=$$

Über Hinweise zur Methodik der Arbeit mit Taschenrechnern im Unterricht siehe [77].

1.3.1. Infixrechner (algebraische Logik) ohne Hierarchie

Man rechnet mit ihnen wie bei den Kopfrechnungen auf Zuruf mit Wiederholungen (Kettenrechnungen) in den unteren Schulklassen:

$$3 + 4 - 5 + 7 =$$

liefert das korrekte Resultat 9. Die Niederschrift der Aufgabe entspricht der Tastenbedienungsfolge des Rechners. Infixrechner erkennt man äußerlich an der Taste „=". Das Eintasten einer Zahl in das Anzeige-, Eingabe- und Resultatregister X wird durch einen Operator oder Drücken der Taste „=" beendet. Gewöhnlich, außer zu Beginn, löst ein Operator das Ausführen der zuvor eingetasteten Operation aus, deren Resultat dann in der Anzeige X erscheint und als neuer erster Operand zur Verfügung steht (*Kurzweg-Technik* im Gegensatz zu immer verlangtem Drücken der Taste „=").

Tasten	3 + 4 − 5 + 7 =
Anzeige X	3 3 4 7 5 2 7 9
Register Y	0 3 3 7 7 2 2 9
(Register Y	0 0 3 0 7 0 2 0)

Zu Beginn dagegen führt das Tasten einer Operation den Inhalt x von X in ein zweites Register Y über.

Beim Eintasten eines neuen (zweiten) Operanden muß der alte (erste) Operand oder das alte Resultat in diesem Register Y aufbewahrt werden. Dieses Register im Rechenwerk des Taschenrechners ist dem Benutzer nicht sichtbar. Entweder drückt das Betätigen einer Operatortaste den Inhalt der Anzeige (Resultat) als ersten Operanden für den Operator in das Register (diese Methode wird bevorzugt), oder aber dies geschieht erst beim Beginn des Eintastens des zweiten Operanden (oben in Klammern gesetzt).

Das Register Y hat nach Abschluß der Operation bei unterschiedlichen Taschenrechnern unterschiedliche Inhalte. Es kann
1. den ersten Operanden halten,
2. ebenfalls das Resultat beinhalten,
3. auf Null gelöscht sein.

Tasten	4	+	2	+	3	=
X	4	4	2	6	3	9
Y		4	4	6	6	6 oder 9 oder 0

Das Register Y wird zur Konstantenbildung herangezogen (siehe 1.3.6.). Die Aufgabe

$$3 \times 4 + 2 \times 5 =$$

liefert nicht, wie die Hierarchie es fordert, 22 als Resultat, sondern 70:

Tasten	$3 \times 4 + 2 \times 5 =$
Anzeige	3 3 4 12 2 15 5 70.

Das kann auch als Test zum Typ verwendet werden.

Will man Produkte von Summen oder Summen von Produkten,

$$(a_1 + a_2) \times (b_1 + b_2) \quad \text{oder} \quad a_1 \times a_2 + b_1 \times b_2,$$

korrekt berechnen, so muß man
- umformen,
- Zwischenergebnisse notieren,
- Zwischenergebnisse speichern.

Ein Taschenrechner mit einem Speicher, gewöhnlich durch eine Taste M (memory (engl.) = Erinnerung, Gedächtnis, Speicher) oder durch M+ bzw. M− erreichbar, ist anderen überlegen. Die Tastenfolgen würden lauten:

1.			2.		
	C	löschen (clean)		C	löschen (clean)
	MC	Löschen des Speichers		MC	Löschen des Speichers
	a_1	Eingabe		a_1	Eingabe
	+	Addition		×	Multiplikation
	a_2	Eingabe		a_2	Eingabe
	=	erste Summe		=	erstes Produkt
	M+	erste Summe nach M		M+	erstes Produkt nach M
	b_1	Eingabe		b_1	Eingabe
	+	Addition		*	Multiplikation
	b_2	Eingabe		b_2	Eingabe
	=	zweite Summe		=	zweites Produkt
	×	Multiplikation		+	Addition
	M	erste Summe aus M zurück		M	erstes Produkt aus M zurück
	=	Resultat		=	Resultat

Bei der Division von Summen muß man zuerst den Divisor (Nenner) errechnen und speichern,

$$(a_1 + a_2)/(b_1 + b_2),$$

da dieser dann als zweiter Operand aus M geholt werden kann.

Errechnet man zuerst den Dividenden (Zähler), so ist eine Taste „1/x" (Kehrwert) von großem Vorteil. Ein Rechner mit dieser Taste ist wiederum besser.

Interessant, wenn auch etwas mühsam, ist das Umformen von Ausdrücken so, daß sie mit Taschenrechnern mit algebraischer (Infix-) Logik, aber ohne Hierarchie in einem Zug lösbar werden.

Summe von Produkten

Die normal notierte arithmetische Umstellung wäre

$$(a_1 \times a_2) + (b_1 \times b_2) = (\frac{a_1 \times a_2}{b_2} + b_1) \times b_2,$$

woraus die Tastenfolge ohne Klammern

$$a_1 \times a_2/b_2 + b_1 \times b_2 =$$

entsteht. Dies läßt sich verhältnismäßig einfach fortsetzen:

$$(a_1 \times a_2) + (b_1 \times b_2) + (c_1 \times c_2) + (d_1 \times d_2) =$$

$$((\frac{a_1 \times a_2}{b_2} + b_1) \times \frac{b_2}{c_2} + c_1) \times \frac{c_2}{d_2} + d_1) \times d_2$$

Tastenfolge ohne Klammern:

$$a_1 \times a_2/b_2 + b_1 \times b_2/c_2 + c_1 \times c_2/d_2 + d_1 \times d_2 =$$

Ein wenig einfacher gestaltet sich das Berechnen einer Summe von Quotienten:

$$(a/b) + (c/d) + (e/f) = ((\frac{a \times d}{b} + c) \times \frac{f}{d} + e)/f$$

Tastenfolge ohne Klammern:

$$a \times d/b + c \times f/d + e/f =$$

Produkt von Summen

Es wird zunächst in eine Summe von Produkten umgeformt,

$$(a_1 + a_2) \times (b_1 + b_2) = a_1 \times b_1 + a_1 \times b_2 + a_2 \times b_1 + a_2 \times b_2,$$

und dann wie bei dieser weiter verfahren. Man erkennt den Vorteil auch nur eines einzigen Speicherregisters, und es wird deutlich, daß das technische Hilfsmittel Arbeit und Methode beeinflußt.

1.3.2. Infixrechner (algebraische Logik) mit Hierarchie

Schon bald wird in der Schule (Klasse 2) gelehrt, daß unter den arithmetischen Operatoren eine Vorrangbeziehung, eine Priorität oder Hierarchie, gilt. Die Multiplikation und Division werden gegenüber Addition und Subtraktion bevorzugt behandelt. Das ist von der Sache her völlig unbegründet. Es stellt lediglich eine Übereinkunft, eine praktische Verabredung, dar, welche Klammerungen einspart. In der Mathematischen Logik bzw. den Grundlagen der Mathematik werden algebraische Strukturen und ihr Aufbau behandelt, bei denen man entweder keine Vorrangbeziehungen festlegt oder diese erst später und eben aus technischen Gründen sparsamer als durch Verwendung von Klammern fixiert. Die Arithmetik ist eine algebraische Struktur von vielen. Taschenrechner, welche die arithmetischen Hierarchien automatisch befolgen, sind etwas aufwendiger konstruiert und scheinen dem Nutzer zunächst einfacher in der Handhabung zu sein. Da sie aber nicht ermöglichen, die Hierarchie durch Klammerung wieder aufzuheben, verliert sich der Vorteil recht bald. Wenn schon die Hierarchie befolgt wird, sollte der Rechner auch zugleich — möglichst mehrere — Klammerniveaus berücksichtigen können und damit genauer die algebraische Struktur der Arithmetik reflektieren.

Von diesem Typ ist der Schulrechner SR1. Er hat zwei zusätzliche Hierarchiestufen:

0. Stufe:	$+\ -$	plus minus
1. Stufe:	$\times\ /$	mal durch
2. Stufe:	y^x	hoch

Das hat zur Folge, daß er zu den X- und Y-Registern im Rechenwerk noch zwei weitere Register Z und T benötigt. Die Aufgabe $3 + 4 \times 2^5 = 131$ wird z. B. wie folgt abgearbeitet:

	3	+	4	×	2	y^x	5	=		Tastenfolge
X	3	3	4	4	2	2	5	(32	128) 131	Anzeige
Y		3_+	3_+	4_*	4_*	$2_\uparrow$	$2_\uparrow$	(4_*	3_+) 128	
Z				3_+	3_+	4_*	4_*	(3_+)		Registerinhalte
T						3_+	3_+			

Die noch nicht ausgeführten Operationen werden ebenfalls gespeichert, d. h. tiefer in den Stapel von Registern gedrückt, wenn eine neue Operation höherer Hierarchie folgt. Neue gleiche oder niedrigere Hierarchie löst aus dem Stapel alle gleichen oder höheren Operationen aus. Beim Gleichheitszeichen werden alle restlichen Operationen erledigt. Das geht so schnell, daß es dem Auge verborgen bleibt (Klammern in der Tabelle).

Die Tastenbezeichnung y^x für das Potenzieren ist nun verständlich. Aus methodischen Gründen wäre ein Pfeil oder POT oder HOCH o. ä. freilich besser.

Übrigens arbeitet y^x nach der Formel

$$\exp(x \cdot \ln y).$$

Daraus wird klar, daß negative y (Basis) zu Fehlern führen, daß aber auch rationale x (Exponenten, Wurzelziehen) möglich sind. Auch die deutliche Wartezeit auf das Resultat wird verständlich.

Für die Abiturstufe: Man hüte sich, den Grenzwert von

$$y^n = \left(1 + \frac{1}{n}\right)^n \quad \text{für} \quad n \to \infty,$$

d. h. die Zahl e, mit der Taste y^x und großem n vorzuführen! Es ist dann nämlich $\ln\left(1 + \frac{1}{n}\right) = \frac{1}{n}$, und aus $x \cdot \ln y$ wird $n \cdot \frac{1}{n} = 1$ und dann natürlich $\exp 1 = e$. Es kommt zwar richtig heraus, aber es wäre eine Vortäuschung des beabsichtigten Effekts. Dieser wird nur durch Verwenden der Multiplikations- oder noch besser der Quadriertaste, z. B. durch zehnmaliges Drücken der Taste x^2 für $n = 1024$, erreicht und korrekt demonstriert.

Man beachte: Zu groß darf n auch wieder nicht sein! Beim SR 1 ist z. B. für $n \geqq 10^9$

$$1 + \frac{1}{n} = 1 \quad \text{und} \quad \ln 1 = 0,$$

woraus $\exp 0 = 1$ und nicht e entsteht.

Konstantenbildung beim SR 1

Im SR1 wird als Konstante das letzte Zwischenergebnis (eventuell unsichtbar) vor dem Resultat und die letzte vor dem Resultat ausgeführte Operation genommen. Im obigen Beispiel ist das + 128. Diese Operation kann in der Tastenfolge schon lange zurückliegen.

Methodischer Hinweis zum Ersatz von Klammern

Von Lehrern, welche bereits Erfahrung im Einsatz der Taschenrechner im Schulunterricht haben, wird häufig die Ansicht vertreten, daß automatisches Befolgen der Hierarchie durch den Rechner für den Lern- und Übungsprozeß nachteilig ist. Die Schüler würden beispielsweise bei der Aufgabe

$$5 + 6 \times 3$$

dann ohne Taschenrechner viel öfter zu dem falschen Resultat $(5 + 6) \times 3 = 33$ durch einfaches Befolgen der Operationen von links nach rechts — wie beim Eintasten — kommen.

Die Hierarchie ist bei diesen Rechnern häufig durch die Resultattaste „=" zu durchbrechen, so daß man gewisse Klammerungen wirkungsmäßig ausführen kann, ohne eine Klammertaste zu besitzen. Wird die Aufgabe $5 + 6 \times 3 =$ in dieser Folge getastet, so ergibt sich korrekt 23. Tastet man $5 + 6 = \times 3 =$, so ergibt sich $33 = (5 + 6) \times 3$.

1.3.3. Infixrechner (algebraische Logik) mit Klammerung

Jede Klammerung legt einen momentanen Vorrang für die eingeklammerte Operation fest. Es gibt nun Taschenrechner mit einem und mit mehreren Klammerniveaus, die sich obendrein noch darin unterscheiden, daß sie die übliche Hierarchie der Operatoren ($\times$ und $\div$ vor $+$ und $-$) berücksichtigen oder nicht. Wie bei der einfachen

Hierarchie verlangt jedes Klammerniveau im Taschenrechner ein weiteres Register. Rechner mit Klammertasten verlangen zwar mehr Tastenbedienungen, sind aber im Ablauf der Operationen logisch einfacher überschaubar. Das Prinzip wird an einer Aufgabe für einen Rechner mit zwei Klammerniveaus ohne Hierarchie (d. h., Multiplikation und Division müssen zusätzlich geklammert werden) vorgeführt. Dieser Rechner ist einem mit Hierarchie und einem Klammerniveau gleichwertig. Die Klammertasten tragen die Bezeichnungen [(und)], und sie dürfen nur zweimal geschachtelt betätigt werden. Ein solcher Rechner ist der MR 610.

Die Aufgabe lautet

$$\frac{6+2\times 3}{3}\times(8-\frac{6}{2})\times 5+3=$$

und wird mit Klammern linear geschrieben zu

$$((6+(2\times 3))\div 3)\times((8-(6\div 2))\times 5)+3=$$

	6	+	(	2	×	3	)	/	3	×	(	8	−	(	6	/	2	)
X	6	6	0	2	2	3	6	12	3	4	0	8	8	0	6	6	2	3
Y		6_+	0	0	$2_\times$	$2_\times$	6_+	$12_/$	$12_/$	$4_\times$	0	0	8_-	0	0	$6_/$	$6_/$	8_-
Z			6_+	6_+	6_+	6_+						$4_\times$	$4_\times$	$4_\times$	8_-	8_-	8_-	8_-	$4_\times$
T																		

	×	5	)	+	3	=	Tastenfolge
X	5	5	25	100	3	103	Anzeige
Y	$5_\times$	$5_\times$	$4_\times$	100_+	100_+		Registrierinhalte
Z							

Das Resultat ist 103. In der arithmetisch geklammerten Niederschrift der Aufgabe treten sogar drei Klammerniveaus auf. Da diese Klammerungen aber am Anfang am tiefsten sind und der Rechner bei jeder Operatortaste eine noch anstehende Operation abarbeitet, kommt man bei der Bedienung mit zwei Niveaus aus. Bei der Anwendung erkennt man wiederum trotz Bemühen der Hersteller Differenzen zur üblichen gewohnten Notation. Am bequemsten für den Nutzer sind demnach Rechner mit Hierarchie und mehreren Klammerniveaus. Es werden gegenwärtig bereits bis zu 20 Klammerstufen angeboten, was praktisch völlig ausreicht, sogar wohl nur mit Mühe auszuschöpfen sein wird.

Konstantenautomatik beim MR 610

Die Regeln lauten:

1. Konstantenoperation ist die letzte ausgeführte Operation vor dem Resultat (im Beispiel Addition).
2. Konstante ist bei Addition, Subtraktion und Division der zweite Operand, d. h. Inhalt von X (im Beispiel 3). Konstante ist bei Multiplikation der erste Operand, d. h. Inhalt von Y.

Anzahl der Zusatzregister

Die verfügbare Technologie stellte ein Rechenwerk-Chip mit vier Registern her. Diese konnten in der Steuerungsschaltung zur Beachtung von zwei Hierarchiestufen (SR1) oder zwei Klammerniveaus (MR610) benutzt werden. Bei H Hierarchiestufen und K Klammerniveaus erhöht sich der Registerbedarf multiplikativ, im wesentlichen durch $H \times K$.

1.3.4. Gemischte Infix-Postfix-Rechner (arithmetische Logik)

Die technische Konstruktion einer Rechenwerklogik in Taschenrechnern ist dann besonders einfach, d. h., auf Zusatzschaltungen lediglich wegen menschlicher Konventionen wird weitgehend verzichtet, wenn die Additions- und Subtraktionstaste nachgestellt (postfix) werden. Da bei ihrer Betätigung das Resultat sichtbar wird, sind diese Tasten durch $\overset{+}{=}$" bzw. $\overset{-}{=}$" gekennzeichnet, und an diesen Tasten sowie der fehlenden Resultattaste „=" erkennt man Rechner mit dieser gemischten Technik. Eine gemischte Technik liegt vor, da die Multiplikations- und die Divisionstaste in der Infixform, also zwischen den Operanden, verwendet werden.

Aus der Aufgabe $7 - 4 =$ wird bei diesen Rechnern die Tastenfolge

$$7 \overset{+}{=} 4 \overset{-}{=},$$

worauf die Anzeige 3 sichtbar wird. Man spricht deshalb auch vom nachgestellten „Vorzeichen". Eingetastete Operanden werden je nach der Operation oder besser je nach dem Postfixzeichen zum Resultatregister im Rechenwerk addiert. Diese Rechner können auf eine Vorzeichenwechseltaste, d. h. einen monadischen Minusoperator (monadisch bedeutet: Wirkung nur auf einen Operanden (im Gegensatz zu dyadischen Operatoren für zwei Operanden)), verzichten. Aus $-4 + 7 =$ wird nämlich einfach

$$4 \overset{-}{=} 7 \overset{+}{=},$$

und wieder erscheint die Anzeige 3. Ein solcher Taschenrechner ist der MR201.

1.3.5. Postfixrechner

Ganz konsequente Bedienungsvorschriften nach logisch klarer Rechenwerkschaltung verlangen die Postfixrechner. Ein Operator wird immer seinen zwei Operanden nachgestellt. Aus $7 + 4 =$ wird

$$7\ 4\ +.$$

Da aber der Rechner das Ende der Eingabe des ersten Operanden erkennen muß (7 und nicht etwa 74), gibt es dafür eine spezielle Taste (ENTER oder ↑), woran (und an der fehlenden Resultattaste „=") diese Rechner schon am Tastenfeld erkennbar sind: Die Tastenfolge ist also

$$7 \uparrow 4 +$$

Die Postfixschreibweise wurde nicht etwa erst mit den Computern oder gar erst mit den Taschenrechnern entwickelt. Die Mathematik, insbesondere die Logik, kennt sie bereits seit J. ŁUKASIEWICZ (1878—1956, polnischer Mathematiker, Begründer der „Warschauer Schule" der Mathematischen Logik). Man nennt sie deshalb auch die *polnische* oder *Warschauer Schreibweise*. Zuerst wurde das Prinzip sogar als Präfixschreibweise, d. h. vorangestellter Operator, entwickelt, weshalb die Postfixschreibweise meist *umgekehrte polnische Notation* (*UPN*) genannt wird. Der Vorteil dieser Schreibweise besteht darin, daß man keine Prioritätsverabredungen und keine Klammern benötigt, weshalb man auch von *klammerfreier* Schreibweise spricht.

Die Aufgabe $7 + 4 \times (3 - 2) =$ wird zu

$$7 \uparrow 4 \uparrow 3 \uparrow 2 - \times +,$$

und $(7 + 4) \times 3 - 2 =$ wird zu

$$7 \uparrow 4 + 3 \times 2 -.$$

Auch Infixrechner machen Konzessionen an die Postfixschreibweise. Diese erfolgt beispielsweise beim Vorzeichenwechsel (Taste „+/−") und bei den Funktionen, die in der Folge

5 SIN

für sin(5) getastet werden. Selbst die Mathematik kennt einen Postfixoperator, das Zeichen ! für die Fakultät.

Postfixrechner benutzen für die Arithmetik einen sogenannten *Keller* oder *Stapel* (*stack*) von Registern. Meist sind es drei oder vier. Sie werden oft mit X, Y, Z, T bezeichnet, wobei X zugleich das Anzeigeregister am Stapelende ist. Man stelle sich diese Stapel- oder Kellerregister wie in Abb. 1.11 vor.

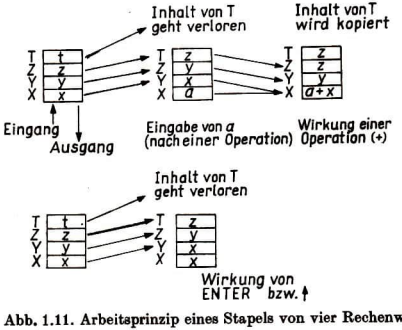


Abb. 1.11. Arbeitsprinzip eines Stapels von vier Rechenwerkregistern

T	t	z	z	y	y	7	7	7	7	7
Z	z	y	y	7	7	4	4	7	7	7
Y	y	7	7	4	4	3	3	4	7	7
X	7	7	4	4	3	3	2	1	4	11
Taste	7	↑	4	↑	3	↑	2	−	*	+

T	t	z	z	z	z	z	z	z
Z	z	y	y	z	y	z	y	z
Y	y	7	7	y	11	y	33	y
X	7	7	4	11	3	33	2	31
Taste	7	↑	4	+	3	*	2	−

Abb. 1.12. Inhalte der Stapelregister während der Abarbeitung der Aufgaben $7 + 4 * (3 - 2) =$ und $(7 + 4) * 3 - 2 =$. Die Ergebnisse sind 11 bzw. 31

Die Abarbeitung der oben notierten Beispielaufgaben zeigt die Abb. 1.12. Bei der ersten Aufgabe wird ein Keller der Tiefe 4 benötigt. Da die Tischrechner K 1001/2/3 nur drei Kellerregister besitzen, kann bei ihnen diese Aufgabe nicht einfach in UPN eingetastet werden. Man müßte z. B. umformen in $(3 - 2) \times 4 + 7$, woraus

$$3 \uparrow 2 - 4 \times 7 +$$

mit Kellertiefe 2 entsteht. Ein Rechner dieses Typs ist z. B. der sonst etwas veraltete KONKRET 600.

1.3.6. Taschenrechner mit Konstantenbildung

Industrie und Handel bieten Taschenrechner mit sogenannter Konstantenlogik oder sogar Konstantenautomatik an. Diese Geräte sind besonders für Aufgaben geeignet, bei denen eine größere Zahlenmenge mit derselben Größe, eben der Konstanten, und einer Operation — meist wird es die Multiplikation sein — verarbeitet werden soll. Diese Aufgabenart tritt besonders in ökonomischen Bereichen auf (verschiedene Warenmengen mit demselben Preis multiplizieren, von verschiedenen Beträgen denselben Prozentsatz bilden usw.), jedoch lassen sich solche Aufgaben natürlich auch in Technik und Wissenschaft finden.

Taschenrechner mit Konstantenbildung sind also nützliche Geräte, aber für den Gebrauch in der Schule kann man auf diesen Vorzug durchaus verzichten.

Die Konstantenlogik der einzelnen Taschenrechner ist sehr vielfältig angelegt. Liegt keine Automatik vor, so erfolgt die Konstantenbildung, d. h. das Einspeichern eines Operanden in ein besonderes Konstantenregister, erst nach Betätigen einer Konstantentaste K. Der Rechner ist dann auf Konstantenbildung eingestellt und bleibt es, bis die K-Taste zurückgestellt wird. Es gibt Rechner, welche den ersten Operanden, und andere, welche den zweiten Operanden als Konstante speichern. Andere verfahren nach anderen Vorschriften, so daß die folgende Übersicht einen Einblick gibt. Stets aber muß man bei dem vorliegenden Gerät die Bedienungsanleitung genau lesen und den oft nicht völlig klaren Text durch Beispiele und Testaufgaben bezüglich des logischen Aufbaus der Geräte regelrecht erforschen.

1. Konstantenrechnung ist möglich nur bei Multiplikation und Division.
2. Konstantenrechnung ist möglich bei allen vier Rechenarten.
3. Die Verwendung des ersten Operanden als Konstante ist günstig für Aufgaben der Art $k \cdot a_i$.

4. Die Verwendung des zweiten Operanden als Konstante ist günstig für Aufgaben der Art $a \cdot k^n$.

— Konstantenbildung wird automatisch möglich, wenn das Y-Register, d. h. das zweite Register im Rechenwerk, bei der Bedienung der Resultattaste „=" nicht gelöscht wird. Sein Inhalt wird dann in das Konstantenregister übergeführt.
— Rechner mit Konstantenrechnung sind Infixrechner (algebraischer oder arithmetischer Logik) ohne Hierarchie und Klammerungen.
— Bei Konstantenrechnung nach Wunsch (K-Taste) wird gewöhnlich der Inhalt des X- oder Anzeigeregisters in das Konstantenregister übernommen.
— Rechner mit Postfixlogik (UPN-Rechner) haben nie besondere Konstantenlogik. Bei ihnen können Konstanten aus dem Stapel gezogen werden, wenn dieser bis zum Anstoß gefüllt war (siehe Abb. 1.11). Die Größe z kann nach einer weiteren Operation als Konstante verwendet werden (siehe auch Abb. 1.12, erstes Beispiel; die Sieben kann nach dem Resultat als Konstante beliebig oft verwendet werden).

5. Die Verwendung des ersten oder zweiten Operanden als Konstante braucht bezüglich der Operationen nicht einheitlich in einem Rechner zu erfolgen (Bedienungsanleitung!).

Die Fortsetzung der Konstantenrechnung, d. h., das weitere Verwenden der Konstanten (was ja der Sinn der Konstruktion ist), aber auch der Übergang zu einer neuen Konstanten, wird wiederum von den einzelnen Rechnern unterschiedlich behandelt.

6. Die Konstante und die bei ihrer Bildung getastete Operation werden durch die Resultattaste „=" wieder verwendet bzw. ausgelöst. Dies ist eine sehr klare und vorzuziehende Arbeitsweise. Eine neue Konstante wird bei Betätigen einer Operationstaste gebildet.

7. Die Konstante wird mit jeder Operationstaste wieder verwendet. Diese viele Möglichkeiten bietende Variante hat oft den Nachteil, daß nicht die eben getastete Operation, sondern die zuvor getastete ausgelöst wird. Beim Wechsel der Operation muß sehr aufgepaßt werden, und es gibt zahlreiche Fehlermöglichkeiten. Eine neue Konstante wird nach der Resultattaste „=" und einer neuen Operationstaste gebildet.

8. Mischform, die Konstante wird bei der Resultattaste und jeder Operationstaste wieder verwendet. Die Operationstastungen hängen um einen Bedienungstakt der Ausführung nach. Beim Übergang von der Taste „=" zu einer Operation wird eine neue Konstante gebildet.

1.3.7. Leistungsklassen der Taschenrechner

Wenn auch die *Bedienungslogik* hier ausführlicher dargestellt wurde, ist sie jedoch (abgesehen von der Infixlogik mit Hierarchie und genügend hoher Klammerschachtelung oder abgesehen von Postfixlogik mit genügender Stapelhöhe — also abgesehen von den kostenintensiveren Geräten) von geringer Bedeutung für die Einstufung in eine Leistungsklasse.

Gegenüberstellungen der Tastschritte in den einzelnen Logiken zeigen bei unterschiedlichen Aufgaben auch unterschiedlich günstiges Verhalten (d. h. weniger Tastungen). Die Postfixlogik schneidet jedoch sehr oft günstig ab. Dafür verlangt sie aber das Umdenken in der Arithmetik und wird wegen des nötigen Überlegens langsamer und wegen der ungewohnten Denkweise zumindest anfangs mehr Fehler verursachen. Bis auf den Infixrechner mit Hierarchie und Klammerungen (genügend viele) verlangen aber alle anderen Logiken ebenfalls ein Umdenken oder doch Beachten von Besonderheiten, so daß dieser oft zitierte Nachteil der Postfixrechner Allgemeingut fast aller Typen ist. Wegen der im Mittel geringeren Tastzahl ist der Postfixrechner bei den obersten Leistungsklassen, nämlich den programmierbaren Taschenrechnern, leicht bevorzugt. Das gilt aber auch nur, solange der Programm-

speicher vom Hersteller noch klein gehalten wird. Mit zunehmender Steigerung der Speicherkapazität versiegt auch dieser Vorteil der Postfixschreibweise.

Entscheidender als die Logik des Rechners ist unbedingt die *Zahlendarstellung*. Für viele Bereiche wird die sogenannte *Festkommadarstellung* (*Festpunkt* zu sagen wäre besser; ein Taschenrechner verwendet meist den Dezimalpunkt, vgl. 1.2.1.) ausreichend sein, und auch sechs oder acht Dezimalstellen werden genügen.

Da der Dezimalpunkt im Taschenrechner nicht etwa so fixiert ist wie im mechanischen Tischrechner, sondern eher in der Weise einer Komma-Automatik von ganz links bis ganz rechts variieren kann, hat man einen Zahlenbereich von 10^k bis 10^{-k} ($k = 6$ oder 8) zur Verfügung, jeweils k-stellige Zahlen. Beim Einsatz für technische und wissenschaftliche Berechnungen sollte man beim Taschenrechner aber Wert auf die sogenannte *wissenschaftliche Zahlendarstellung* — gemeint ist die *Gleitpunktdarstellung* (vgl. 1.2.1.) legen.

Die *Ausstattung mit Bedienungsfunktionen* ist ein weiteres und sehr wichtiges Merkmal für die Leistungseinstufung eines Taschenrechners. Dies ist außerdem schon einfach äußerlich durch *Betrachten des Tastenfeldes* erkennbar. Von den besonders einfachen Geräten, die nur die vier Grundrechenarten anbieten, soll hier nicht gesprochen werden.

Erste nützliche Zusatztasten bzw. -funktionen sind

$x \leftrightarrow y$	Registerwechsel im Rechenwerk Austausch erster und zweiter Operand (wichtig bei Subtraktion und Division, wo es kein Kommutativgesetz gibt)
$+/-$	Vorzeichenwechseltaste, monadisches Minuszeichen, für negative Operanden
$1/x$, x^2, $\sqrt{x}$	nützlich, da sehr häufig
%	für viele Zwecke von Vorteil
CE	Löschtaste für irrtümlich zuletzt eingegebene Ziffer (clean entry), man braucht nicht die gesamte oft lange Zahl neu einzutasten
M, M+, M−, M → x	Verwendung eines Speichers (memory)
STO, RCL	Verwendung von mehreren Speichern (store = speichern, recall = zurückrufen), wobei dann natürlich noch eine Adresse (d. h. Nummer des Speichers, siehe MfL Bd. 9, 2.1.) anzugeben ist.

Die bis hier genannten Tasten gehören eigentlich zu einer Mindest- oder Grundausstattung.

Am Tastenfeld ist ferner die verwendete Bedienungslogik leicht erkennbar:

Infixrechner	(algebraische Logik) an der Resultattaste „="

Infix-Postfix-Rechner	(arithmetische Logik) an den Tasten	„+" und „−" „="
Postfixrechner	(UPN) am fehlenden „=" und an ENTER oder ↑	
Klammerungen		(und) [(und)].

Von der Grundausstattung ausgehend werden nun eine Vielzahl von problemorientierten Funktionen in der Mittelklasse der Geräte angeboten. Das fängt bei den mathematischen Funktionen

$$\text{SIN, COS, TAN, LN, LOG, EXP, } x^y\text{, SIN}^{-1}\text{, COS}^{-1}\text{, TAN}^{-1}$$

an (der Exponent -1 deutet Umkehrfunktion an, auch oft mit der Taste ARC erreichbar) und setzt sich bis in die Spitzenklasse mit Hyperbel- und Areafunktionen fort. Ferner werden Umwandlungen von kartesischen Koordinaten in Polarkoordinaten, Minuten und Sekunden in Dezimalbruchteile und umgekehrt angeboten. Auch verschiedene statistische Grundrechnungen findet man: Mittelwert, Streuung und sogar lineare Regression. Manche Hersteller rühmen sich für 70 und mehr Tastenfunktionen. Oft sind aber dabei eine größere Anzahl für europäische Verhältnisse unnötiger Umrechnungen wie gallon in Liter, foot und inch in Zentimer usw.

Die Spitzenklasse der Taschenrechner wird durch die *programmierbaren* gebildet und unter diesen wieder durch diejenigen, welche schon das Stadium der Maschinenprogrammierung hinter sich ließen und nutzerfreundliche Hilfsmittel wenigstens auf dem Assemblerniveau anbieten. Es gibt sogar Geräte, welche bereits eine einfache höhere Programmiersprache (BASIC) anzuwenden gestatten. Technisch wird die gegenwärtige Spitzenklasse durch austauschbare Funktionenmoduln, Magnetkartenein- und -ausgabe sowie Anschlußmöglichkeiten von Peripheriegeräten (Drukker, Magnetbandkassetten) gekennzeichnet.

Zahlendarstellungen in Taschenrechnern

Der Schulrechner errechnet Resultate mit maximal neun Stellen, von denen minimal acht, die letzte gerundet, angezeigt werden. Erfolgt die Resultatangabe in Gleitpunktdarstellung, so werden sogar nur fünf Ziffern gezeigt. Im Rechner sind aber die Werte neunstellig gespeichert! Wie läßt man sich die verborgenen Ziffern vorführen? Bekanntlich ist $\frac{1}{7} = 0.\overline{142857}$ ein periodischer Bruch. Das Resultat von „$1 + 7 =$" wird als 1.4285-01 in der Anzeige dargestellt. Multiplikation mit 10 (Beseitigung des Exponentenanteils) liefert schon 1.4285714, also acht Ziffern, und in diesem Fall schon die volle Periode. Subtraktion der ersten Ziffer (Resultat in der Anzeige 4.2857-01) und erneute Multiplikation mit 10 zeigt 4.2857143, also mit der bisher verborgenen neunten Ziffer „3", die aus „28" gerundet wurde. Eine Wiederholung des Verfahrens (jetzt Subtraktion von 4 und Multiplikation mit 10) befördert keine neue Ziffer mehr in die Anzeige. Dieses Verfahren kann man allgemein anwenden, um festzustellen, ob ein Rechner intern mehr — und wieviel mehr — Stellen verwendet, als er anzeigt.

Wenn Taschenrechner sowohl Fest- als auch Gleitpunktanzeige besitzen, sind sie gewöhnlich so konstruiert, daß vom intern berechneten Resultat möglichst viele geltende Ziffern in die Anzeige gebracht werden. Das ist beim SR1 und auch beim MR610 der Fall. Somit zeigt der MR610 bei der achtstelligen Anzeige

$\frac{1}{128} = 0.0078125$	denn die acht Stellen reichen,
$\frac{1}{256} = 3.90625_{10}\text{-}03$	denn bei Festpunktanzeige würden die Stellen nicht reichen,
$\sin 45° = 7.0710678_{10}\text{-}01$	um eine Stelle mehr als bei 0.7071067 anzuzeigen,
$\sin 30° = 0.5$	was niemanden wundert,
$\sin 33° = 5.4463904_{10}\text{-}01$	als irrationale Zahl mit einer Stelle mehr als bei 0.5446390,
$\sin 34° = 0.5591929$	doch wieder zum Staunen, denn auch das ist eigentlich eine irrationale Zahl; aber das interne SIN-Programm liefert in acht gültigen Ziffern 0.55919290, und es besteht kein Grund für das Umsteigen auf Gleitpunktdarstellung.

Nun sind die Anzeigen

$$\sqrt{0.129} = 0.3591657 \quad \text{und} \quad \sqrt{0.128} = 3.5777088_{10}\text{-}01$$

ebenso erklärbar.

Beim SR1 gilt dieselbe Überlegung, und dieselbe Reaktion tritt ein. Aber: Seine Anzeige nimmt drei Stellen für die Angabe der Zehnerpotenz fort! Also statt — wie beabsichtigt — mehr anzuzeigen, kommt es oft vor, daß er sogar weniger anzeigt. Die Beispiele vom MR610 werden beim SR1 zu

$\frac{1}{256} = 3.9062\text{-}03$	eine Stelle fehlt noch,
$\sin 45° = 7.0710\text{-}01$	drei Stellen fehlen,
$\sin 33° = 5.4463\text{-}01$	drei Stellen fehlen,
$\sqrt{0.128} = 3.5777\text{-}01$	drei Stellen fehlen.

„Fehlen" bedeutet „fehlen in der Anzeige". Intern hat der SR1 sie wie der MR610 berechnet, und man kann sie sich zeigen lassen, wie es oben beschrieben wurde.

1.3.8. Technische Charakterisierung der Taschenrechner

Ein recht entscheidendes Merkmal ist die *Stromversorgung*. Sie ist mit Batterien, Netzteil oder ladbaren Akkumulatoren möglich.

Alle heutigen Geräte arbeiten mit äußerst geringem Stromverbrauch (s. u. LCD-Anzeige) und Batterien langer Lebensdauer, die 2000 Betriebsstunden und mehr

Abb. 1.13. Die Ziffern 5, 8 und 9 in der Anzeige mit LED Balkensegmenten. Bei der Ziffer 8 sind alle sieben Segmente verwendet

garantieren, oder sogar Solarzellen. Bei diesen — und zu ihnen gehören die Typen SR1, MR411 und MR610 — entfallen die Forderungen nach einem Netzteil.

Ein weiteres in die Betrachtung einzubeziehendes Element ist die Anzeige. Äußerlich machen sich ältere Geräte durch die beiden zum Einsatz gebrachten Techniken roter oder grüner Leuchtziffern kenntlich.

Die rote LED- (light emitting diode) Anzeige wird durch Leuchtdioden (Halbleitertechnik) gebildet. Ein Ziffernfeld hat sieben Segmente (vgl. Abb. 1.13) oder seltener $5 \cdot 7 = 35$ Leuchtpunkte. Die Lebensdauer ist hoch. Es werden 100000 Stunden oder 10 Jahre angegeben. Außerdem ist die Leuchtdiode stoßsicher. Heute werden die von Digital-Quarz-Uhren bekannten Flüssigkristallanzeigen LCD (liquid crystal display) angewendet, da diese extrem wenig Energie verbrauchen. Daß sie nur bei Lichteinfall arbeiten und auch eine größere Trägheit aufweisen, dürfte kaum störend sein. Bei diesen Geräten reicht eine Batterie ein bis zwei Jahre, so daß ein Netzanschluß entbehrlich wird. Das Problem des Energieverbrauchs der Anzeige — er stellt den weitaus größten Anteil dar — hat zu oft gepriesenen automatischen Sparschaltungen geführt. Nach einigen Sekunden Anzeige schaltet sie sich ab. Das ist aber gefährlich, da man bei solchen Geräten das Ausschalten leicht vergißt.

1.3.9. Kurze Information zu BASIC

Die zur Verfügung stehenden Kleincomputer oder Videocomputer sind in der Programmiersprache BASIC (beginner all purpose symbolic instruction code) und einem zugehörigen Betriebssystem bedienbar. Man kann sie, natürlich auch unter Zurückweichen auf den Basisbaustein Mikroprozessor U 808 D oder 8080, in direktem Maschinencode im Hexadezimalsystem (16er System) bzw. in der zugeordneten Assemblersprache programmieren. Da dies aber eine Sache für Fachleute bzw. Hobbyenthusiasten ist, bleibt es gewiß auf diesen nicht zu großen Kreis begrenzt. Ein Lehrer und auch Schüler werden im allgemeinen doch BASIC bevorzugen.

Der Einstieg in BASIC ist sehr bequem, wie schon die zwei Beispiele für den GGT in 1.1.4. zeigten. Diese Programmiersprache ist „nutzerfreundlich". Die meisten Sprachelemente erklären sich sozusagen selbst. Formeln können in BASIC nun wirklich fast genau in der üblichen Notation geschrieben werden. Die Unterschiede liegen lediglich in der Linearisierung, in der unbedingt notwendigen Angabe jeder Operation — auch bei Multiplikation — und der konsequenten Klammerung aller Funktionsparameter:

nicht	$\frac{a+b}{c}$	$a(b+c)$	$\sin x$
sondern	(a + b)/c	a * (b + c)	sin(x)

Der Programmautor braucht sich nicht um Speicherplatzzuweisungen für Variable oder Variablenfelder (in BASIC-Dialekten sind meist nur Vektoren, oft auch Matrizen, verfügbar) und deren Adressen zu kümmern. Er schreibt symbolisch Namen für Variable auf. Für grundlegende Formulierungselemente von Algorithmen, nämlich Zyklen und Entscheidungen, bietet BASIC spezielle Formulierungshilfen an.

Während der Einstieg in BASIC fast immer ohne Schwierigkeiten gelingt, wird man bei der Lösung komplizierterer Probleme mit notwendigerweise komplizierteren Algorithmen zunehmend Mühe haben. Die Programme werden immer unübersichtlicher, ihre Korrektur oder Abänderung löst Fehler an anderen Stellen aus, die vielen Sprünge GOTO lassen ein strukturelles Gefitze, „Spaghetti-Programme", entstehen. BASIC ist eben auch sehr maschinenprogrammgebunden, was zu der Formulierung im Vorwort geführt hat. Echte höhere Programmiersprachen, wie z. B. PASCAL, oder in diese Richtung zielende Aufbauversionen zu BASIC, wie z. B. das in Dänemark für den Schulgebrauch entwickelte COMAL, sind dann zu bevorzugen. Man hat in höheren Programmiersprachen (ALGOL, PASCAL, COBOL u. a.) zwar mehr Mühe am Anfang, ehe man das erste Programm schreiben kann, aber der Lern- und Arbeitsaufwand für das tiefere Eindringen und das Anfertigen komplizierterer Programme ist geringer als in BASIC.

Man kann selbstverständlich auch in BASIC diszipliniert und gut strukturiert programmieren. Tausende von Programmen beweisen das. Das aber muß antrainiert und anerzogen werden, während andere gute Programmiersprachen einen solchen guten Programmierstil aus ihrer Anlage heraus fordern und fördern [80].

So konfus manche BASIC-Programme strukturiert sind, so konfus hat sein Programmierer eben auch gedacht, als er durch „Hackerei", d. h. durch dauerndes lokales Korrigieren und Ändern das Programm „zum Laufen brachte". Das „Hacker"-Unwesen muß ein Lehrer bekämpfen; am besten ist, wenn er es gar nicht erst aufkommen läßt.

Die meisten Heimcomputer bieten BASIC an, deshalb ist es in diesem Buch als verfügbar verwendet worden. Alle angegebenen Programme sind in dem BASIC-Dialekt eines speziellen Videocomputers getestet worden und haben erfolgreich gearbeitet. Das Umschreiben in andere Dialekte oder Versionen dürfte kaum Probleme bereiten.

Ein BASIC-Handbuch hat meistens einen Umfang von etwa 100 Seiten. Hier ist lediglich ein Überblick oder eine Zusammenfassung möglich.

Numerische Variable

Angabe durch einen Namen, der entweder aus einem Buchstaben allein oder auch weiteren Buchstaben und Ziffern besteht (oft aber maximal zwei Zeichen):

```
a     x     ai     x1
```

Variable werden einfach durch Ergibtanweisungen oder Eingabeanweisungen eingeführt.

Numerische Konstante

Zahlenwerte im Dezimalsystem dargestellt:

1	125	ganze Zahlen
.5	1.5	Dezimalbrüche, Festpunktzahlen
1.25E-11		Gleitpunktzahlen

Vektoren, indizierte Variable (numerisch)

Angabe durch einen Namen, der nur aus einem Buchstaben besteht. Vor dem Gebrauch des Namens muß eine Dimensionsdefinition im Programm stehen:

DIM h(15)	der Vektor h hat 15 Komponenten, welche durch Indizes aus dem Bereich 1 (auch 0) bis 15 unterschieden werden
h(7)	ist die Angabe der Komponente von h mit Index 7

Ausdrücke oder Formeln

Die Schreibweise ist weitgehend identisch mit der üblichen Notation, aufgebaut aus einfachen oder indizierten Variablen (Feldkomponenten), Konstanten, Operationssymbolen ($+ - \times / \uparrow$), Klammern (nur runde) und Funktionsnamen.

Jede Variable, einfach oder indiziert, muß vor dem Gebrauch oder ihrem „angewandten Auftreten" in einer Formel ein einführendes oder „definierendes Auftreten" auf der linken Seite einer Ergibtanweisung oder als Argument einer Eingabeanweisung erlebt haben. Dadurch hat sie einen Wert erhalten, der nun in der Formel verwendet werden kann, um den Wert der Formel zu bestimmen.

Funktionen

Mit x ist ein beliebiger Ausdruck bezeichnet, dessen Wert den Beschränkungen der Funktion entsprechen muß.

ABS(x)	absoluter Betrag von x
INT(x)	ganzer Anteil von x
SQR(x)	Quadratwurzel (square root von x)
SIN(x)	Sinus von x (x in Bogenmaß)
COS(x)	Cosinus von x
TAN(x)	Tangens von x
ASN(x)	Arcussinus von x (Wert in Bogenmaß)
ACS(x)	Arcuscosinus von x
ATN(x)	Arcustangens von x
RND	Erzeugung einer Zufallszahl (random)
SGN(x)	Vorzeichen (signum) von x
LN(x)	Logarithmus von x
EXP(x)	Exponentialfunktion von x

Anweisungen

Anweisungen werden nach dem Eintasten mit einer vorgeschriebenen Taste, z. B. ENTER, bestätigt. Sind sie korrekt formuliert, werden sie sofort ausgeführt, anderenfalls erfolgt eine Fehlermeldung. Werden Anweisungen mit einer Anweisungsnummer (am Anfang oder ganz links) eingegeben, so werden sie nach ENTER nicht abgearbeitet, sondern zu einem Programm entsprechend der Anweisungsnummer zusammengestellt.

Ergibtanweisung

Eine Ergibtanweisung beginnt mit LET, dem die Variable folgt, der etwas zugewiesen werden soll, dann ein Gleichheitszeichen und dann ein Ausdruck. Im einfachsten Fall ist der Ausdruck eine Konstante.

```
LET x = 1
LET h(7) = 1.2 E4 + x
```

Ergibtanweisungen können als „definierende Auftreten" der links stehenden Variablen betrachtet werden. (In manchen BASIC-Versionen kann LET weggelassen werden.)

Eingabe/Ausgabe-Anweisungen

Um Werte von außen einem Programm zur Verfügung zu stellen, gibt es die INPUT-Anweisung. Sie kann mehrere durch Komma oder Semikolon getrennte Parameter in Form einfacher oder indizierter Variabler haben. Bei Abarbeitung einer INPUT-Anweisung

```
INPUT a, b, c, d
```

werden am Bildschirm vier Zahlenangaben, z. B.

```
35        −18
−1.5E3    Ø.5
```

in zwei, auch drei Spalten erwartet. Jede Angabe ist mit ENTER zu beenden. Die Variablen *a*, *b*, *c* und *d* haben nach dem ENTER nach 0.5 die Werte 35, −18, −1500 und 0.5. Diese Werte kann man durch die Ausgabeanweisung

```
PRINT a, b, c, d
```

in obiger Anordnung erscheinen lassen. Bei Abarbeitung einer INPUT-Anweisung

```
INPUT a; b; c; d
```

werden die Zahlenwerte in zusammenhängender Gestalt, aber zeitlich beim Eintasten durch ENTER unterbrochen, am Bildschirm erwartet.

```
35 − 18 − 1.5 E3 Ø.5
```

Bei der Ausgabe

```
PRINT a; b; c; d
```

erscheint

35 — 18 — 15ØØØ.5 (auch Standardabstände möglich).

Es gibt noch eine andere Eingabeanweisung

READ

mit durch Komma getrennten Variablenangaben, die jedoch ihre einzulesenden Werte aus einer — meist am Ende eines Programmes befindlichen — Datenliste holt.

READ a, b, c, d

und

DATA 35, −18, −1.5 E3, Ø.5

gehören zusammen. Die Datenliste kann länger als die Parameterliste sein. Dann wird von der Stelle an weiter eingelesen, wenn eine neue READ-Anweisungsabarbeitung erfolgen soll, an der aufgehört wurde.

Variable in Eingabeanweisungen dienen als „definierende Auftreten" dieser Variablen, denn auf diese Weise erhalten sie einen Wert, mit dem in „angewandten Auftreten" weiter gearbeitet werden kann.

Zyklen

In BASIC gibt es nur eine Zyklenform:

FOR i = 5 TO 17 STEP 2
⋮
NEXT i

Anstelle des i kann irgendeine Variable (einfach) stehen (nach dem FOR ist es ein definierendes Auftreten), anstelle der 5 (Anfangswert), 17 (Endwert) oder 2 (Schrittgröße) können irgendwelche Ausdrücke stehen. Die Formulierung STEP 1 kann weggelassen werden. Die Anweisungen anstelle der drei Punkte werden x-mal wiederholt, wobei

$$x = \mathrm{INT}\left(\frac{\text{Endwert} - \text{Anfangswert}}{\text{Schrittgröße}}\right)$$

ist. Es kann also durchaus $x \leqq 0$ sein! Dann wird der Zyklus nicht durchlaufen (doch Vorsicht: bei gewissen Interpretern einmal).

Entscheidungen oder Verzweigungen

In BASIC gibt es nur eine Entscheidungsform

IF Bedingung THEN Anweisung

Die Anweisung wird ausgeführt, wenn die Bedingung erfüllt ist, anderenfalls wird sie übergangen. In den meisten Fällen ist die Anweisung eine Sprunganweisung, es wird also gesprungen oder nicht, d. h., es geht „geradeaus weiter". Wird „nach oben" (zu kleineren Anweisungsnummern) gesprungen, so entsteht ein Zyklus. Wird „nach unten" (zu größeren Anweisungsnummern) gesprungen, so wird ein Programmteil ausgelassen.

Bedingungen

Im einfachsten Fall ist eine Bedingung ein Vergleich:

```
a < b
```

Anstelle von *a* oder *b* können auch komplizierte Ausdrücke, aber auch einfach Konstante stehen, und anstelle des < können

```
<=    =    >=    >    <>
```

für

```
≦     =    ≧     >    ≠
```

auftreten. Solche einfachen Bedingungen können noch mit logischen Operatoren

```
NOR    OR    AND
```

zu komplizierteren zusammengesetzt werden. So entsteht aus $2 \leqq x \leqq 10$ in BASIC

```
2 < = x AND x < = 1Ø
```

Sprunganweisung

Die Sprunganweisung lautet

```
GOTO n n
```

wobei n n eine im Programm auftretende Anweisungsnummer ist. Dort wird die Arbeit fortgesetzt bzw. bei einem sofort abzuarbeitendem Sprung (vom Bediener am Bildschirm ausgelöst) aufgenommen.

Stoppanweisung und Pause

Eine STOP-Anweisung läßt das Programm an dieser Stelle halten (die Arbeit kann dort durch eine Maßnahme des Bedieners fortgesetzt werden):

```
STOP
```

Eine PAUSE-Anweisung unterbricht das Programm für *n* Zeiteinheiten (meist Sekunden), was zur Anzeige von Programmnachrichten oder Zwischenwerten günstig ist:

```
PAUSE n
```

Zeichenketten

Für INPUT und PRINT ist es zur Information des Bedieners günstig, wenn Zeichenketten auf dem Bildschirm erscheinen. Eine Zeichenkette wird in " (Anführungsstriche) eingeschlossen. Die Zeichenkette zwischen den Anführungsstrichen wird abgebildet.

```
INPUT "a ="; a, "b =";b
```

läßt zunächst

a =

erscheinen. Nach Eingabe des a-Wertes 18 (beendet mit ENTER) wird

a = 18 b =

erscheinen. Dieser Aufforderung könnte mit dem b-Wert 1.5

a = 18 b = 1.5

(noch ohne ENTER) genügt werden. Nach ENTER ist der Schirm wieder leer, aber a und b haben diese Werte. Die Anweisung

PRINT "a = "; a, "b = "; b

erzeugt, wenn die Werte zuvor nicht etwa verändert wurden, auf dem Bildschirm

a = 18 b = 1.5

Weitere BASIC-Befehle

In Tab. 1.2 sind die bis jetzt erwähnten BASIC-Standardwörter als Bestandteile von Anweisungen alphabetisch gelistet. Es sind keineswegs alle, z. B. fehlen noch Bezeichnungen, welche man zur Definition und zum Gebrauch von Unterprogrammen (Subroutinen) und Funktionen benötigt.

Ferner enthält jeder BASIC-Dialekt oder jede Version der Sprache noch Befehle, die speziell von dem verwendeten Grundgerät bestimmt werden. Das sind Befehle zum Anfertigen von Zeichnungen, zum Erzeugen von Tönen, zur Text- oder Zeichenkettenverarbeitung, zur farbigen Gestaltung des Bildschirmes u. a. m. Alle solchen speziellen Befehle und Möglichkeiten hat man der betreffenden Betriebsanleitung bzw. dem zum Gerät gehörenden BASIC-Handbuch zu entnehmen.

ABS	DATA	IF	NOT	(RUN)	TAN
ACS	DIM	INPUT	OR	SGN	THEN
AND	(ENTER)	INT	PAUSE	SIN	TO
ASN	EXP	LET	PRINT	SQR	
ATN	FOR	LN	READ	STEP	
COS	GOTO	NEXT	RND	STOP	

Betriebssystembefehle

Zur Zusammenstellung des Programms, seiner Kontrolle und Korrektur, zum Aufruf des Programms, zu seiner Unterbrechung oder seinem Abbruch gibt es Befehle, die hier nicht Gegenstand der Betrachtung sein können. Ferner gibt es Befehle zur Aufbewahrung (externe Speicherung) von Programmen auf Kassetten über Recorder und auch zur Einfüllung solcher Konserven zurück in den Computer.

In Tab. 1.2 sind in Klammern zwei Betriebssystembefehle aufgenommen, welche im Text erwähnt wurden.

2. Numerische Mathematik mit Kleinstrechnern

2.1. Iterationsalgorithmen

Eine ganz wesentliche Arbeitsmethode der numerischen Mathematik ist die *Iteration* bzw. sind die Algorithmen, welche sie benutzen (iterare (lat.) = wiederholen).

Zunächst wird eine Einstiegsmethode geschildert, die unter Verwendung auch einfacher Taschenrechner in der Schule oder mit dem Schüler eingesetzt werden könnte.

Man kann den Taschenrechner, wenn er nur die Quadratwurzeltaste hat, zum Berechnen „höherer" Wurzeln veranlassen! Das wird alle Besitzer einfacher Taschenrechner (ohne x^y-Taste) interessieren.

Beispiel

Berechnung der dritten Wurzel $x = \sqrt[3]{a}$ (a gegeben und x gesucht).

Eine Umformung ist nötig, wobei zugleich auch die Rechengesetze für Wurzeln wiederholt und angewendet werden:

$x^3 = a$ Beseitigung der Wurzel

$x^4 = ax$ Maßnahme (hier Multiplikation mit x), damit der Exponent von x eine Zweierpotenz wird.

$x^2 = \sqrt{ax}$

$x = \sqrt{\sqrt{ax}}$ Fortgesetztes Quadratwurzelziehen (Wurzeltaste) liefert x, eine Rechenformel ist gefunden.

Da aber auf der rechten Seite auch noch das gesuchte x auftritt, ergibt sich folgender Algorithmus für das Beispiel:

1. Einen *Startwert* (eine nullte Näherung) für x schätzen,
2. Formel zur Berechnung eines neuen (nächste bessere Näherung) x anwenden:

$$x := \sqrt{\sqrt{ax}},$$

3. Formel wiederholen (Schritt 2), indem das errechnete neue x nun rechts eingesetzt wird.

Der Schritt 3 in Verbindung mit Schritt 2 beinhaltet das *Iterieren*, es entstehen fortlaufend neue *iterierte* Werte für x. Oft wird diesen Werten auch ein Index gegeben: $x_0, x_1, x_2, \ldots$ Aber man mache sich frei von der Vorstellung, daß dadurch die Elemente oder Komponenten eines

Vektors bezeichnet werden. Diese Indizes hier sind *Iterationsindizes* und haben einen dynamischen oder zeitlichen Inhalt. Um dies besonders zu charakterisieren, gebraucht man auch die Schreibweisen

$$x_{(0)}, x_{(1)}, x_{(2)}, \dots$$

oder

$$x^{(0)}, x^{(1)}, x^{(2)}, \dots$$

und könnte dann in der Iterationsformel auf das Ergibtzeichen verzichten. Der Algorithmus nimmt dann folgende Form an:

1. $x^{(0)}$ schätzen,
2. $x^{(i+1)} = \sqrt{\sqrt{a}x^{(i)}}$ für $i = 0, 1, 2, \dots$

oder etwas deutlicher wieder mit Ergibtzeichen

1. $x^{(0)}$ schätzen
2. $x^{(1)} := \sqrt{\sqrt{a}x^{(0)}}$
3. $x^{(0)} := x^{(1)}$ und Schritt 2 wiederholen.

Hier bedeutet der Schritt 3 das Umbenennen des neuen iterierten Wertes $x^{(1)}$ in einen alten, der dann in der Formel auf der rechten Seite eingesetzt wird.

Eine Tastenfolge für einen Taschenrechner (MR 410) ist leicht zu finden. Man stelle durch

Eingabe a
MC
M+
Eingabe Schätzwert $x^{(0)}$

den Radikanden a in das Speicherregister.

Die Tastenzyklen aus den Operationen

$\times$
MR
=
F $\sqrt{\ }$
F $\sqrt{\ }$

zeigen dann beispielsweise für $a = 8$ und $x^{(0)} = 1$ bzw. $x^{(0)} = 3$ die Zahlenwerte der Tab. 2.1, denn nach jedem Zyklus steht im x-Register der neue iterierte Wert.

Nach dem elften Zyklus bzw. der elften Iteration ändert sich der angezeigte Wert nicht mehr. Er steht fest (fix) und heißt auch *Fixpunkt* der Iteration. Dieser Wert ist die Lösung der gestellten Aufgabe. (Bei einem Taschenrechner mit Konstantenrechnung ließe sich der Zyklus verkürzen.) Es liegt einmal ein zu kleiner und einmal ein zu großer Startwert vor. Die Annäherung an die Lösung oder an den Fixpunkt erfolgt einmal von „unten", d. h. von zu kleinen zu immer größer werdenden Näherungswerten, und einmal von „oben", d. h. von zu großen zu immer kleiner werdenden Näherungswerten.

Tab. 2.1. Näherungswerte für $\sqrt[3]{8}$ nach der Iterationsformel $x := \sqrt{\sqrt{8x}}$

i	$x^{(i)}$	$x^{(i)}$
0	1	3
1	1.681799	2.213364
2	1.915207	2.051331
3	1.978456	2.012711
4	1.994592	2.003170
5	1.998647	2.000792
6	1.999662	2.000198
7	1.999915	2.000049
8	1.999979	2.000012
9	1.999995	2.000003
10	1.999999	2.000001
11	2.000000	2.000000
12	2.000000	2.000000

Bei der Aufgabe $x = \sqrt[3]{a}$ wurde die Iterationsformel

$$x = \sqrt{\sqrt{xa}}$$

hergestellt. Das Wesentliche war dabei die Isolierung eines x auf der linken Seite, gleichgültig ob und in welcher Stellung noch weitere x rechts verbleiben.

Erste Regel zur Herstellung einer Iterationsformel

Allgemein muß man aus einer Aufgabe

$$g(x) = 0 \tag{1}$$

eine *iterierfähige* Formel

$$x = f(x) \tag{2}$$

durch Isolierung eines x herstellen.

Im speziellen Beispiel der Aufgabe

$$g(x) = x - \sqrt[3]{a} = 0$$

wurde diese Isolierung zunächst durch die Umformung

$$x^3 = a$$

vorbereitet. Das weitere Ziel, daß der Exponent des x eine Zweierpotenz sein soll, kann auch durch Division erreicht werden,

$$x^2 = a/x,$$

und man erhält die Iterationsformel

$$x = \sqrt{a/x}$$

mit nur einer Quadratwurzel. Die Rechnung mit einem Taschenrechner erfordert nun den Tastenzyklus

$1/x$
$\times$
MR
$=$
$\sqrt{x}$

nach Einstellung desselben Anfangszustandes.

Die Zwischenresultate sind in Tab. 2.2 für die gleichen Startwerte wie in Tab. 2.1 und für $a = 8$ gezeigt. Man erkennt, daß man „langsamer" zur Lösung kommt und daß die Näherungswerte *alternieren*, d. h. abwechselnd zu groß und zu klein sind.

Tab. 2.2. Näherungswerte für $\sqrt[3]{8}$ nach der Iterationsformel $x := \sqrt{8/x}$.
Die Iteration ist auch nach neun Schritten noch nicht zum Stillstand gekommen

i	$x^{(i)}$	$x^{(i)}$
0	1	3
1	2.828427	1.632993
2	1.681793	2.213364
3	2.181015	1.901160
4	1.915207	2.051331
5	2.043794	1.974818
6	1.978456	2.012711
7	2.010860	1.993675
8	1.994592	2.003170
9	2.002709	1.998417

Mit der gezeigten Methode lassen sich alle höheren Wurzeln einfach berechnen. Beispielsweise kann man für

$$x = \sqrt[5]{a} \quad \text{oder} \quad g(x) = x - \sqrt[5]{a} = 0$$

die iterierfähigen Formeln

$$x = \sqrt{\sqrt{a/x}} \quad \text{oder} \quad x = \sqrt{\sqrt{\sqrt{a\sqrt{ax}}}}$$

erhalten, je nach den Umformungen

$$x^5 = a \qquad \text{oder} \qquad x^5 = a$$
$$x^4 = a/x \qquad x^{15} = a^3$$
$$x^2 = \sqrt{a/x} \qquad x^{16} = a^3ax$$
$$x^8 = a\sqrt{ax}$$
$$x^4 = \sqrt{a\sqrt{ax}}$$
$$x^2 = \sqrt{\sqrt{a\sqrt{ax}}}$$

Bei solchen Rechnungen und Vorführbeispielen dazu stellen aufmerksame Schüler sofort die Fragen:

— Wie erkennt man das Erreichen des Iterationsendes, da man in realen Fällen den Wert des Fixpunktes ja nicht kennt und dieser auch irgendeine gewiß nicht ganze Zahl sein wird?

— Wie schnell kommt man zum Ziel?

oder sogar

— Kommt man mit dieser Methode immer zum Ziel?

Die Antwort auf die erste Frage ist einfach. Man beobachte die Zwischenwerte oder iterierten Werte und achte dabei auf die Ziffernstellen, die sich nicht mehr ändern. Wenn alle angezeigten Stellen (oder so viele, wie es für eine geforderte Genauigkeit ausreichend ist) „stehen", hat man den Fixpunkt (im Rahmen der Rechengenauigkeit, hier Stellenzahl des Taschenrechners) erreicht.

Oft wird in der Literatur der Fixpunkt durch einen Stern, also x^* bezeichnet. Der Abbruch oder das Ende der Iteration erfordert also den beobachtenden eingreifenden Menschen. Dies wäre für umfangreiche Iterationsprobleme ein nicht haltbarer Zustand. Er muß geändert werden in einen automatischen Abbruch, der vom Rechner selbst ausgelöst wird. Ein Weg dazu führt über die zweite Frage, deren Beantwortung etwas aufwendiger ist und auch neue Begriffsbildungen erfordert.

Dazu betrachte man in Tab. 2.1 und in Tab. 2.2 das Fehlerverhalten

$$|x^{(i)} - x^*|,$$

was ja durch die Kenntnis des Fixpunktes $x^* = 2$ möglich wird. Die Zahlenwerte sind in Tab. 2.3 angegeben. Man erkennt, daß für die Fehlerwerte nach Tab. 2.1 von Schritt zu Schritt annähernd eine Vierteilung und nach Tab. 2.2 eine Halbierung eintritt. Bezeichnet man den Fehler jeweils mit $\varepsilon^{(i)}$, also

$$\varepsilon^{(i)} = |x^{(i)} - x^*|,$$

Tab. 2.3. Fehlerverhalten $|x^{(i)} - x^*|$ der iterierten Werte aus Tab. 2.1 und 2.2. Der Fixpunkt ist $x^* = 2$

i	$x^{(i+1)} = \sqrt{7x^{(i)}}$		$x^{(i+1)} = \sqrt{8/x^{(i)}}$	
	$\lvert x^{(i)} - x^*\rvert$	$\lvert x^{(i)} - x^*\rvert$	$\lvert x^{(i)} - x^*\rvert$	$\lvert x^{(i)} - x^*\rvert$
0	1	1	1	1
1	0.32	0.21	0.83	0.37
2	0.075	0.051	0.32	0.21
3	0.022	0.013	0.18	0.10
4	0.0055	0.0031	0.085	0.051
5	0.0014	0.00079	0.044	0.025
6	0.00034	0.00020	0.022	0.013
7	0.000085	0.000049	0.011	0.0063
8	0.000021	0.000012	0.0054	0.0032
9	0.000005	0.000003	0.0027	0.0016
10	0.000001	0.000001		
11	0	0		
12	0	0		

und den Faktor der Fehlerverkleinerung als *Konvergenzfaktor* q, so ist

$$\varepsilon^{(i+1)} = q\varepsilon^{(i)}$$

oder auch

$$|x^{(i+1)} - x^*| = q|x^{(i)} - x^*| \tag{3}$$

(konvergent (lat.) = zusammenlaufend, sich zueinander neigend). Für die Iterationsformel der Tab. 2.1,

$$x = \sqrt{\sqrt{8x}},$$

ist also der Konvergenzfaktor $q = \frac{1}{4}$, und für eine Iterationsformel der Tab. 2.2,

$$x = \sqrt{8/x},$$

ist der Konvergenzfaktor $q = \frac{1}{2}$.

Zweite Regel für ein Iterationsverfahren

Damit also ein Iterationsverfahren *konvergiert*, d. h. zu einem Fixpunkt laufende Werte produziert, muß es einen positiven Konvergenzfaktor

$$q < 1$$

haben, und je kleiner dieses q ist, desto besser oder schneller konvergiert das Verfahren, desto höher ist seine *Konvergenzgeschwindigkeit.*

Der neue Restfehler nach einem Iterationsschritt hängt linear vom alten Fehler ab,

$$\varepsilon^{(i+1)} = q\varepsilon^{(i)},$$

und deshalb spricht man von *linearer Konvergenz.*

Es erhebt sich sofort die Frage nach der Konstruktion möglichst guter oder schneller Iterationsformeln, also nach Formeln mit möglichst kleinem (positiven) Konvergenzfaktor q und nach Methoden zur Berechnung oder Abschätzung von q.

Noch besser als jede noch so gute lineare Konvergenz ist aber, wenn der neue Fehler quadratisch bzw. noch stärker vom alten Fehler abhängt,

$$\varepsilon_{(i+1)} = q\varepsilon_{(i)}^2,$$

oder allgemeiner

$$\varepsilon_{(i+1)} + q\varepsilon_{(i)}^p \quad \text{mit} \quad p > 1. \tag{4}$$

Die Kenngröße p heißt *Konvergenzordnung*, und sobald $p \geqq 2$ ist, kann man von „Traumalgorithmen" sprechen, da sie traumhaft schnell zum Fixpunkt gelangen.

Beispielsweise kannten bereits die Sumerer oder Babylonier 2000 Jahre v. d. Z. folgende Iteration für $\sqrt{a}$, die wir heute mit dem Namen NEWTON verbinden (vgl. MfL Bd. 9, 1.1.), allerdings führten sie nur einen Iterationsschritt aus:

$$x_{(i+1)} = \frac{1}{2}\left(x_{(i)} + \frac{a}{x_{(i)}}\right). \tag{5}$$

Beim Fixpunkt muß natürlich

$$x^* = \frac{1}{2}\left(x^* + \frac{a}{x^*}\right)$$

gelten, woraus sich leicht

$$2x^* = x^* + \frac{a}{x^*} \qquad x^* = \frac{a}{x^*}, \quad x^{*2} = a, \qquad x^* = \sqrt{a}$$

wie gewünscht ergibt. Zur Untersuchung des Fehlerverhaltens setzt man

$$x_{(i)} = \sqrt{a}\,(1 + \varepsilon_{(i)}),$$

wobei also

$$\varepsilon_{(i)} = \frac{x_{(i)} - \sqrt{a}}{\sqrt{a}}$$

der *relative Fehler* ist, in die Iterationsformel ein und erhält nach kurzer Rechnung

$$x_{(i+1)} = \sqrt{a}\left(1 + \frac{\varepsilon_{(i)}^2}{2(1 + \varepsilon_{(i)})}\right).$$

Es ist also

$$\varepsilon_{(i+1)} = \frac{\varepsilon_{(i)}^2}{2(1 + \varepsilon_{(i)})}$$

oder bei positiven $\varepsilon_{(i)}$

$$\varepsilon_{(i+1)} < \frac{\varepsilon_{(i)}^2}{2}, \tag{6}$$

und das bedeutet nach dem oben Gesagten *quadratische Konvergenz.*

Das „traumhaft schnelle" Konvergieren verdeutlicht die folgende Überlegung. Ist etwa bereits

$$\varepsilon_{(n)} \approx 10^{-k},$$

d. h., sind schon k Dezimalstellen korrekt, so ist

$$\varepsilon_{(n+1)} \approx 10^{-2k},$$

und es sind dann $2k$ Dezimalstellen korrekt. In jedem Schritt verdoppelt sich die Anzahl korrekter Ziffern des gesuchten Ergebnisses. Ein kleines BASIC-Programm demonstriert dies sehr schön:

```
10 INPUT a, x0
20 LET x1 = (x0 + a/x0)/2
30 PRINT x1
40 LET x0 = x1
50 STOP
60 GOTO 20
```

Das Programm wird mit RUN (und ENTER) gestartet. Nach der Eingabe von $a = 2$ und $x_0 = 1$ (und ENTER) stoppt es nach Anzeige des ersten iterierten Wertes. Die Arbeit wird bei

```
60 GOTO 20
```

also beim Befehl 20 fortgesetzt, wenn der Betriebssystembefehl CONTINUE (setze fort) gegeben wird. Auf diese Weise erhält man die Werte der Spalte $x^{(i)}$ aus Tab. 2.4. (Es kann sein, daß in manchen BASIC-Versionen nicht zehn Ziffern angezeigt werden. Dann können mit

```
30 PRINT INT (x1 * 1000)/1000,  x1 - INT (x1 * 1000)/1000
```

sehr oft weitere, sonst in dem Standardformat der Zahlenanzeige unterdrückte Ziffern „hervorgelockt" werden. Man suche die neue Anweisung 30 zu verstehen!)

Tab. 2.4. Quadratische Konvergenz der Iterationsformel $x := (a/x + x)/2$ für $\sqrt{a}$ und $a = 2$. Die jeweils bereits (fast) korrekten Ziffern sind fett gedruckt

i	$x^{(i)}$	$\varepsilon^{(i)}$
0	1	
1	**1.50**	0.09
2	**1.41666667**	0.002
3	**1.41421568**7	0.000002
4	**1.41421356**8	0
5	1.41421356**3**	

2.1.1. Konstruktion von Iterationsverfahren

Die bereits formulierte Regel zur Konstruktion einer iterierfähigen Formel $x = f(x)$ aus einer gegebenen Aufgabe $g(x) = 0$ genügt leider nicht in allen Fällen, obwohl nun schon einiges zur Konvergenz gezeigt wurde. Es ist noch nicht deutlich geworden, wie die Konvergenz gesichert werden kann.

Man betrachte beispielsweise die Aufgabe

$$g(x) \equiv a \ln x + bx + c = 0,$$

woraus man die iterierfähige Form

$$x = -\frac{a \ln x + c}{b} \equiv f(x)$$

leicht herstellt. Durch aktuelle Wahl zweier Parameterkombinationen (a, b, c) erhält man zwei spezielle Aufgaben:

1. Aufgabe	2. Aufgabe
$a = 2$	$a = 2$
$b = -2$	$b = -0.2$
$c = 3.802775422$	$c = -1.597224578$

Diese Parameter wurden gewählt, da dann in beiden Fällen die Lösung $x^* = 3$ existiert, wovon man sich leicht überzeugt:

1. $\ln 3 - 3 + 1.901387711 = 1.098612289 - 3 + 1.901387711 = 0$,

2. $\ln 3 - 0.3 - 0.798612289 = 1.098612289 - 0.3 - 0.798612289 = 0$.

Die beiden aktuellen Iterationsvorschriften lauten

1. $x := \ln x + 1.901387711$,

2. $x := 10 \ln x - 7.986612289$,

und die Rechenergebnisse der Iterationen jeweils für den Startwert $x^{(0)} = 2$ zeigt Tab. 2.5.

Tab. 2.5. Iterationswerte für die Iteration $x := A \ln x + B$ in zwei Parameterkombinationen (siehe Text)

i	1. Fall $x^{(i)}$	2. Fall $x^{(i)}$
0	2	2
1	2.59	−1.05
2	2.85	
3	2.95	
4	2.98	
5	2.99	
6	3.00	
7	3.00	

Im Fall 1 wird der Fixpunkt $x^* = 3$ erreicht. Im Fall 2 versagt die Iteration im zweiten Schritt, da der Definitionsbereich von $f(x)$ (hier von $\ln x$) verlassen wird. Das Versagen der Iteration im zweiten Fall besagt dabei nicht etwa, daß kein Fixpunkt der Iteration existiert. Setzt man nämlich $x^* = 3$ in die Iterationsformel

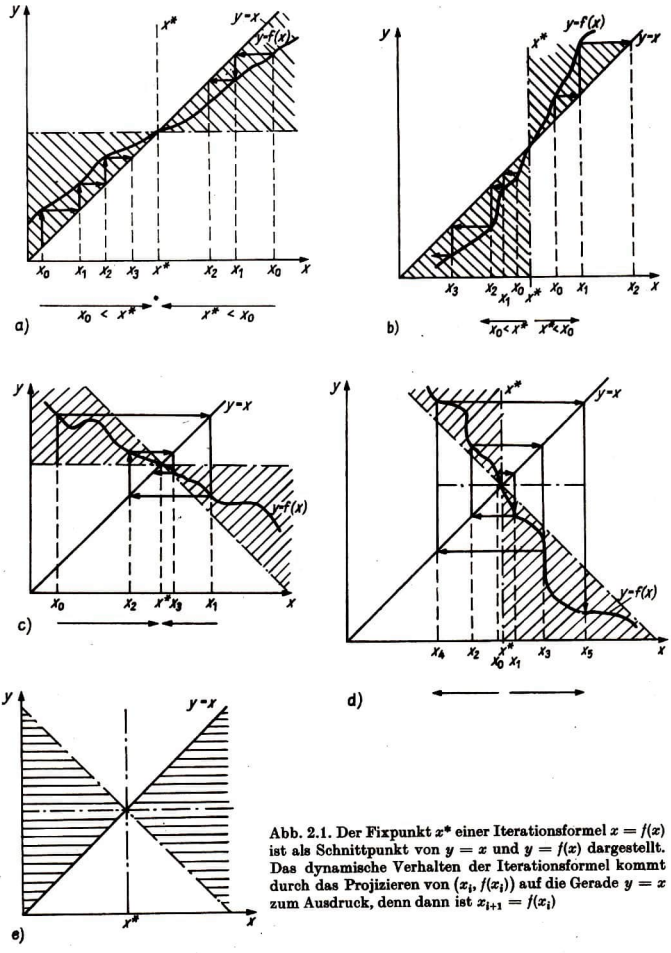


Abb. 2.1. Der Fixpunkt x^* einer Iterationsformel $x = f(x)$ ist als Schnittpunkt von $y = x$ und $y = f(x)$ dargestellt. Das dynamische Verhalten der Iterationsformel kommt durch das Projizieren von $(x_i, f(x_i))$ auf die Gerade $y = x$ zum Ausdruck, denn dann ist $x_{i+1} = f(x_i)$

ein, so erhält man auch wieder $x^* = 3$. Der Fixpunkt existiert. Der durch die Formel beschriebene Iterationsvorgang ist nicht konvergent.

Woran liegt das? Vier Skizzen werden den Sachverhalt verdeutlichen. Aus Abb. 2.1a—e erkennt man in Form einer Plausibilitätsbetrachtung:

a) Beim Fixpunkt x^* und in dessen Umgebung ist $0 < f'(x) < 1$, d. h., der Kurvenanstieg ist $< 45°$. Der Fixpunkt ist *anziehend*, gleichgültig, ob $x_0 < x^*$ oder $x_0 > x^*$ ist. Die Konvergenz erfolgt *monoton* von links bzw. rechts.

b) Beim Fixpunkt x^* und in dessen Umgebung ist $1 < f'(x)$, d. h., der Kurvenanstieg ist $> 45°$. Der Fixpunkt ist nach beiden Seiten *abstoßend*. Die iterierten Werte x_i divergieren (laufen auseinander).

c) Beim Fixpunkt x^* und in dessen Umgebung ist $-1 < f'(x) < 0$, d. h., der Kurvenabfall ist $< 45°$. Der Fixpunkt ist *anziehend*. Die Konvergenz ist *alternierend*.

d) Beim Fixpunkt x^* und in dessen Umgebung ist $f'(x) < -1$, d. h., der Kurvenabfall ist $> 45°$. Der Fixpunkt ist *alternierend abstoßend*. Die iterierten Werte x_i *divergieren*.

e) Verläuft $y = f(x)$ im schraffierten Bereich, so konvergiert das Iterationsverfahren $x = f(x)$. Hinreichend dafür ist, daß bei x^* und in seiner Umgebung $|f'(x)| < 1$ gilt.

Aus Abb. 2.1 läßt sich die Zusammenfassung ableiten, daß es bei der Konvergenz einer Iterationsformel auf den Anstieg, d. h. auf die Ableitung der Funktion $f(x)$ beim Fixpunkt und in dessen Umgebung ankommt. Ein *anziehender* Fixpunkt (eine *kontrahierende*, d. h. zusammenziehende *Abbildung* oder eine konvergente Iterationsformel $x = f(x)$) liegt vor, wenn

$$|f'(x)| \leqq q < 1 \tag{7}$$

beim Fixpunkt x^* und in dessen Umgebung gilt und der Startwert x_0 aus dieser Umgebung genommen wird. Die Aussage bedeutet (schärfer als oben formuliert), daß $|f'(x)|$ beschränkt sein muß und daß diese Schranke kleiner als 1 sein muß. Wird diese Schranke tatsächlich angenommen, ist es also die kleinstmögliche obere Schranke (obere Grenze von $f(x)$ im Intervall, supremum), so ist dieses q der Konvergenzfaktor des Verfahrens, und deshalb wurde auch derselbe Buchstabe zur Bezeichnung gewählt.

Beispiele

1. Die Iterationsformel $x = \sqrt{\sqrt{ax}} = (ax)^{1/4} = f(x)$ liefert

$$f'(x) = \frac{a}{4}\,(ax)^{-3/4},$$

für $a = 8$ und $x^* = 2$ ergibt sich

$$f'(x^*) = 2\sqrt[4]{\frac{1}{(8 \cdot 2)^3}} = \frac{1}{4} < 1,$$

und in Tab. 2.3, linker Teil, wurde auch $q = \frac{1}{4}$ erkannt.

2. Die Iterationsformel $x = \sqrt{a/x} = (a/x)^{1/2} = f(x)$ liefert

$$|f'(x)| = \frac{a}{2}\,(a/x)^{1/2} \cdot (1/x^2)\,;$$

für $a = 8$ und $x^* = 2$ ergibt sich

$$|f'(x)| = 4\sqrt{\frac{2}{8}\cdot\frac{1}{4}} = \frac{1}{2} < 1,$$

und in Tab. 2.3, rechter Teil, wurde auch $q = \frac{1}{2}$ erkannt.

3. Bei der Berechnung von fünften Wurzeln konnte man umformen auf die Iterationsformeln

$$x = \sqrt[5]{a/x} \quad \text{oder} \quad x = \sqrt{\sqrt{\sqrt{a\sqrt{ax}}}}.$$

Für $a = 32$ ist dann $x^* = 2$, und der Konvergenzfaktor errechnet sich folgendermaßen:

$$f(x) = (a/x)^{1/4} \qquad f(x) = a^{1/8}(ax)^{1/16}$$

$$|f'(x)| = \frac{a}{4}(a/x)^{-3/4}\cdot 1/x^2 \qquad f'(x) = a^{1/8}\frac{a}{16}(ax)^{-15/16}$$

$$= \frac{1}{4x}\sqrt[4]{(a/x)} \qquad = \frac{1}{16}\sqrt{\frac{1}{x}\sqrt{\frac{1}{x}\sqrt{\frac{a}{x}\sqrt{\frac{a}{x}}}}}$$

$$|f'(x^*)| = \frac{1}{4} < 1 \qquad f'(x^*) = \frac{1}{16} < 1$$

$$q = \frac{1}{4} \qquad q = \frac{1}{16}.$$

Die zweite Formel hat also ein vierfach stärkeres Konvergenzverhalten.

4. Die Iterationsformel $x = -\dfrac{a\ln x + c}{b}$ für die Aufgabe

$$a\ln x + bx + c = 0$$

liefert

$$|f'(x)| = \left|\frac{a}{bx}\right|,$$

und für die beiden oben betrachteten Fälle entstehen daraus mit

$a = 2$	$a = 2$
$b = -2$	$b = -0.2$
$x^* = 3$	$x^* = 3$

die Ungleichungen

$$|f'(x^*)| = \frac{1}{3} < 1 \qquad |f'(x^*)| = \frac{10}{3} > 1$$

damit ist der Grund für das Versagen der Iteration im Fall 2 aufgedeckt.

2.1.2. Fixpunktsatz von Banach

Der Fixpunktsatz von BANACH (vgl. MfL Bd. 4 und MfL Bd. 9, 4.1.1.) bildet eine der wichtigen theoretischen Grundlagen für Iterationsverfahren oder *kontrahierende* (zusammenziehende) *Abbildungen*. Er gehört zur Funktionalanalysis, die ihrerseits zum theoretischen Fundament der numerischen Mathematik beiträgt und diese vom Stand eines beinahe empirischen Wissens in den des exakten, mit mathematischer Strenge begründeten gehoben hat. Es gibt allerdings eine Einschränkung für den Einsatz von numerischen Rechenhilfsmitteln. Die Funktionalanalysis macht u. a. Aussagen über Elemente sogenannter *vollständiger Räume* (z. B. des Raumes der reellen Zahlen). Das sind Räume, in denen die Grenzelemente x der Folgen x_i mit

$$\lim_{n,m\to\infty} |x_n - x_m| = 0,$$

also der Cauchy-Folgen, ebenfalls zum Raum gehören, d. h., solche Räume haben keine „Löcher". Für vollständige Räume mit einer Norm, d. h. also für Banachräume, gilt auch der Fixpunktsatz. Aber der Zahlenraum, welcher den Rechenhilfsmitteln zur Verfügung steht (darstellbare Maschinenzahlen), ist kein Banachraum. Es werden ja nur endlichstellige Dezimalzahlen, also nur eine endliche Teilmenge des auch schon nicht vollständigen Raumes der rationalen Zahlen, verwendet. (Der Lehrer denke an den in der Schule geführten Beweis, daß $\sqrt{2}$ nicht zu den rationalen Zahlen gehört. Aber man kann eine Cauchy-Folge rationaler Zahlen konstruieren, deren Grenzwert $\sqrt{2}$ ist.)

Der Zahlenraum der Rechengeräte hat demnach nicht einfach nur Löcher, sondern er ist nicht einmal überall dicht besetzt. Die Elemente des Raumes sind durch (z. T. sogar beträchtliche) Abstände getrennt. Es ist ein diskreter Raum. Damit gilt die Grundvoraussetzung des Banachschen Fixpunktsatzes nicht mehr, und man kann auch nicht die Gültigkeit erwarten. Jedoch bildet der Raum der Computerzahlen bei 6-, 8- oder 10stelliger Genauigkeit für viele Anwendungsfälle ein genügend gutes „Modell eines Banachraumes", und man kann sehr oft (aber eben nicht immer!) die Aussagen des Fixpunktsatzes verwenden.

Bezüglich der Cauchy-Folgen in einem diskreten Zahlenraum läßt sich argumentieren, daß ein solcher Raum vollständig ist. Ist x_i eine Folge mit

$$\lim_{n,m\to\infty} |x_n - x_m| = 0,$$

dann bedeutet dies, daß von einer Stelle k an alle Elemente x_i mit $i > k$ den gleichen Wert x^* haben müssen. Das muß dann eintreten, sobald

$$|x_k - x_{k+1}| < d_{\min}$$

wird, wenn $d_{\min}$ der minimale Abstand zweier benachbarter Zahlen im diskreten Raum ist. Damit ist x^* als Grenzelement auch Element des Raumes. In der Praxis begnügt man sich jedoch mit Folgen, welche der Forderung

$$\lim_{n,m\to\infty} |x_n - x_m| < \varepsilon$$

genügen, d. h., die x_i können noch in den letzten Ziffernstellen schwanken. Es gibt für sie keinen Fixpunkt, kein Grenzelement, sondern ein „Grenzintervall möglichst eng benachbarter Werte". Man verlangt also nicht die Vollständigkeit im ursprünglichen Sinn.

Für einen Banachraum wird außer der Existenz einer Norm auch verlangt, daß er ein Vektorraum ist, und das heißt, es müssen insgesamt zehn Forderungen [32] bezüglich einer Addition und Vervielfachung seiner Elemente erfüllt sein. Für Zahlenräume in Rechenanlagen gelten diese Forderungen, z. B. das Assoziativgesetz, nicht in allen Fällen, so daß sich ein weiterer Grund für vorsichtiges Anwenden der aus der Funktionalanalysis gewonnenen Aussagen ergibt. Formulierung und Beweis des Banachschen Fixpunktsatzes dürften für den Schulunterricht und sicher sogar für Arbeitsgemeinschaften zu kompliziert und speziell sein. Der Mathematiklehrer aber müßte sich gut mit seinen Aussagen und Konsequenzen vertraut machen.

Bemerkungen zu den Voraussetzungen und Aussagen des Satzes. Die oben aus der geometrischen Anschauung gewonnene Konvergenzbedingung (7) für die Iterationsformel (2) wird im Fixpunktsatz durch die schwächere Lipschitz-Bedingung

$$\|f(x_1) - f(x_2)\| \leqq q \, \|x_1 - x_2\| \quad \text{mit } q < 1 \tag{8}$$

bei x_1, x_2 aus der Umgebung von x^* ersetzt.

Diese allgemeine Formel bedeutet speziell für den $\mathbf{R}_1$, daß nicht die Ableitung, sondern der Differenzenquotient oder, anschaulich geometrisch, nicht der Anstieg der Tangenten, sondern der der Sehnen beschränkt sein muß. (Im Raum $\mathbf{R}_1$ der reellen Zahlen kann als Norm (Doppelstriche) die Betragsfunktion (einfache Striche) verwendet werden.)

Die beiden geltenden Fehlerabschätzungen

$$\text{a priori:} \quad |x^* - x^{(i)}| \leqq \frac{q^i}{1-q} \, |x^{(1)} - x^{(0)}| \tag{9}$$

$$\text{a posteriori:} \quad |x^* - x^{(i)}| \leqq \frac{q}{1-q} \, |x^{(i)} - x^{(i-1)}| \tag{10}$$

erlauben aus der Kenntnis des Unterschiedes zwischen Startwert und erstem iterierten Wert, also nach einem einzigen Iterationsschritt (d. h. sehr früh, a priori (lat.) = von vornherein), bei bekanntem q (Lipschitz-Konstante) eine Aussage über den Fehler nach dem i-ten Schritt zu machen bzw. bei Überwachung der Unterschiede, d. h. der Änderung, der iterierten Werte von Schritt zu Schritt (also später, a posteriori (lat.) = im nachhinein) ebenso. Man kann also die erste Formel bereits nach dem ersten Iterationsschritt benutzen, um abzuschätzen, wieviel Schritte nötig sind, bis eine gewünschte Rechengenauigkeit erzielt ist. Das bedeutet Auflösung der Formel nach i.

Mit den Bezeichnungen

$$|x^* - x^{(i)}| = \varepsilon^{(i)}, \quad |x^{(i)} - x^{(i-1)}| = d^{(i)} \quad \text{(Differenz } d\text{)}$$

wird $q^i \geqq \dfrac{\varepsilon^{(i)}}{d^{(1)}} (1-q)$, also

$$i \geqq \frac{\lg\left(\dfrac{\varepsilon^{(i)}}{d^{(1)}} (1-q)\right)}{\lg q}. \tag{11}$$

Beispiel

Bei der Berechnung von $x = \sqrt[3]{a}$ mit $a = 8$ und der Iterationsformel

$$x := \sqrt{\sqrt{a}x}$$

war $q = \frac{1}{4}$. Mit dem Startwert $x^{(0)} = 1$ ergab sich $x^{(1)} = 1.68$, also $d^{(1)} = 0.68$ (vgl. Tab. 2.1). Wieviel Iterationsschritte sind mindestens nötig, um den Fixpunkt x^* auf sechs Stellen genau zu berechnen? Es ist also $\varepsilon^{(i)} = 10^{-6}$,

$$i \geqq \frac{\lg (10^{-6} \cdot 0.75/0.68)}{\lg 0.25} = \frac{-5.96}{-0.60} = 9.90.$$

Es sind $i = 10$ Schritte nötig, um diese Genauigkeit zu garantieren, und Tab. 2.1 bestätigt dies auch.

Die logarithmische Rechnung wurde hier mit einem Taschenrechner durchgeführt. Dieser liefert negative Logarithmen direkt und nicht in der Form

$$\lg (1.07 \cdot 10^{-6}) = 0.04 - 6, \quad \lg 0.25 = 0.40 - 1,$$

wie man es bei Tafelverwendung gewöhnt ist. Da die Formel einen Quotienten von Logarithmen enthält, ein Umrechnungsfaktor sich also herauskürzt, kann man auch sofort und eben ohne Umrechnung natürliche Logarithmen benutzen, wenn etwa der Rechner keine dekadischen anbietet:

$$i \geqq \frac{\ln \left(\frac{\varepsilon^{(i)}}{d^{(1)}} (1 - q) \right)}{\ln q}.$$

Im Beispiel ergibt sich dann

$$i \geqq \frac{-13.72}{-1.39} = 9.90,$$

natürlich der gleiche Zahlenwert.

In der praktischen Rechnung schätzt man jedoch kaum a priori die Anzahl der Iterationsschritte ab, sondern überwacht während der Rechnung die Differenzen $d^{(i)}$ zwischen aufeinanderfolgenden Iterationswerten und bricht die Rechnung gemäß der a-posteriori-Formel ab. Ist nämlich ein bestimmtes ε als zulässiger Restfehler gefordert, so läßt man iterieren, bis die Bedingung

$$\frac{q}{1 - q} d^{(i)} \leqq \varepsilon \tag{12}$$

erfüllt ist, denn dann ist auch $\varepsilon^{(i)} \leqq \varepsilon$.

Dieses allgemein angewendete Verfahren führt auf die Vorschrift für Iterationsverfahren mit einer Abbruchbedingung:

— Aus der Aufgabe

$$g(x) = 0 \tag{1}$$

bilde man eine iterierfähige Form

$$x = f(x) \tag{2}$$

und bestimme (im Rechenintervall) für $f(x)$ die Lipschitz-Konstante $q < 1$

— Man schätze einen Startwert $x^{(0)}$

— Man wende die Iterationsformel

$$x^{(1)} = f(x^{(0)})$$

an und bilde

$$d = |x^{(1)} - x^{(0)}|$$

— Test:

$$\frac{qd}{1-q} \leqq \varepsilon? \tag{12}$$

ja: STOP, $x^{(1)}$ genügt der geforderten Genauigkeit $|x^* - x^{(1)}| \leqq \varepsilon$;

nein: Man wiederhole die Iterationsformel mit $x^{(1)}$ als $x^{(0)}$.

Es hat sich in der Rechenpraxis eingebürgert, an Stelle von

$$\frac{qd}{1-q} \leqq \varepsilon$$

als Abbruchtest einfach

$$d \leqq \varepsilon$$

zu nehmen, weil die Bestimmung einer möglichst scharfen Lipschitz-Konstanten oft schwierig und umständlich ist. Das widerspricht der Aussage des Fixpunktsatzes und ist unter Umständen auch gefährlich. Man muß in jedem Fall gesondert überlegen, ob diese vereinfachte Testvariante für den Abbruch berechtigt ist. Sie ist erlaubt bei $q \leqq 1/2$ und bei mindestens quadratisch konvergenten Verfahren (z. B. Newton).

Gegenbeispiel

Die Reihe $\sum_{k=1}^{\infty} \frac{1}{k}$ bzw. die Folge der Partialsummen

$$S_n = \sum_{k=1}^{n} \frac{1}{k}$$

ist bekanntlich divergent. Sieht man davon ab und bildet eine Iterationsformel für ihre Berechnung,

$$S_{(n)} = S_{(n-1)} + \frac{1}{n} \quad \text{mit} \quad d_{(n)} = \frac{1}{n},$$

und ist $\varepsilon = 10^{-k}$ gefordert, so liefert der zulässige Abbruchtest

$$\frac{1}{n} \leqq 10^{-k}, \quad \text{d. h. } n \geqq 10^k$$

und damit einen endlichen und somit falschen Resultatwert.

Es ist aber durchaus richtig, daß das Bestimmen einer möglichst scharfen Lipschitz-Konstanten eine lästige Arbeit sein kann. Hier hilft eine Überlegung weiter. Es ist nicht nur

$$\lim_{i \to \infty} \varepsilon^{(i)} = \lim_{i \to \infty} d^{(i)} = 0,$$

sondern auch

$$\lim_{i \to \infty} \frac{\varepsilon^{(i+1)}}{\varepsilon^{(i)}} = \lim_{i \to \infty} \frac{d^{(i+1)}}{d^{(i)}} = q,$$

so daß der Quotient aus zwei aufeinanderfolgenden Differenzen iterierter Werte das q immer besser annähert. Die Abbruchbedingung wird damit

$$\frac{d^{(i+1)}}{1 - \frac{d^{(i+1)}}{d^{(i)}}} \leqq \varepsilon. \tag{12}$$

Im obigen Gegenbeispiel der divergenten Reihe ergäbe sich bei diesem (besseren) Test für die linke Seite

$$\frac{\frac{1}{n+1}}{1 - \frac{n}{n+1}} = 1,$$

und das kann nie kleiner als ε werden, so daß kein Abbruch erfolgt und ein Stopp bei Maschinenrechnung erst bei einem Überlauf des zur Verfügung stehenden Zahlbereiches eintritt. Das reflektiert auch besser die Divergenz der Reihe. (Infolge der endlichen Maschinenarithmetik würde allerdings trotz beliebig vieler Iterationen kein Überlauf entstehen, da die Addition eines Gliedes $\frac{1}{n} < \frac{1}{10^m}$ (m-stellige Maschinenzahlen) keinen Beitrag mehr zur Partialsumme liefert.)

2.1.3. Regula falsi (1. Form)

Unter der Benutzung der Sekanten des Graphen der Funktion $g(x)$ aus der Aufgabe $g(x) = 0$ kann ein Iterationsverfahren konstruiert werden, welches mit „größerer Wahrscheinlichkeit" kontrahierend ist als die durch irgendeine Isolierung von x aus $g(x)$ algebraisch gewonnenen. Da ein anschaulicher Sachverhalt vorliegt, ist die entstehende Formel auch schon lange bekannt.

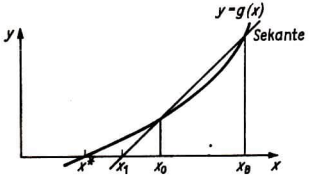


Abb. 2.2. Zur Herleitung der regula falsi. Der Punkt mit der Abszisse x_B ist der feste Bezugspunkt. Die Werte $x_0, x_1, \ldots$ bilden die Iterationsfolge, die gegen die Lösung x^* konvergiert

Anstelle des Schnittpunktes des Bildes von $y = g(x)$ mit der x-Achse wird der der Sekante bestimmt (Abb. 2.2). Die Punkte (x, y) der Sekante genügen der Beziehung

$$\frac{y - y_0}{x - x_0} = \frac{y_0 - y_B}{x_0 - x_B}.$$

Dabei ist (x_B, y_B) ein festgehaltener Bezugspunkt. Für den gesuchten Schnittpunkt x_1 ist $y_1 = 0$, und daraus erhält man

$$x_1 = x_0 - \frac{x_0 - x_B}{y_0 - y_B} y_0$$

oder

$$x_1 = x_0 - \frac{x_0 - x_B}{g(x_0) - g(x_B)} g(x_0) \tag{13}$$

als Iterationsformel. Die Werte x_0 und x_B sind die „falschen" Werte, welche zur Berechnung eines besseren benutzt werden, daher auch der Name der Regel. Man überprüfe nun das Konvergenzverhalten.

Es ist für die Iterationsvorschrift

$$x = f(x) = x - \frac{x - x_B}{g(x) - g(x_B)} g(x) = \frac{x_B g(x) - x g(x_B)}{g(x) - g(x_B)},$$

$$f'(x) = \frac{\big(x_B g'(x) - g(x_B)\big)\big(g(x) - g(x_B)\big) - g'(x)\big(x_B g(x) - x g(x_B)\big)}{\big(g(x) - g(x_B)\big)^2},$$

$$f'(x^*) = \frac{g(x_B) + g'(x^*)(x^* - x_B)}{g(x_B)},$$

$$|f'(x^*)| = \left|1 + \frac{g'(x^*)}{g(x_B)}(x^* - x_B)\right|, \tag{14}$$

wobei die Bedingung $|f'(x^*)| < 1$ z. B. für $g(x_B) > 0$, $g'(x^*) > 0$ und $x^* < x_B$ bei passenden Zahlenverhältnissen wie etwa in Abb. 2.2 erfüllt werden kann.

Beispiel

1. Die Aufgabe $g(x) = a \ln x + bx + c = 0$ führt bei

$$a = 2,\ b = -2 \text{ und } c = 3.802775422$$

für den Bezugswert $x_B = 4$ zu der Vorschrift

$$x_1 = x_0 - \frac{x_0 - 4}{\ln x_0 - x_0 - \ln 4 + 4}\,(\ln x_0 - x_0 + 1.901387711)$$

$$= x_0 - \frac{(x_0 - 4)\,(\ln x_0 - x_0 + 1.901387711)}{\ln x_0 - x_0 + 2.613705639}.$$

Die Rechenergebnisse sind für den Startwert $x_0 = 2$ in Tab. 2.6 dargestellt. Es zeigt sich ein Konvergenzfaktor $q < \frac{1}{10}$. Für die gleiche Aufgabe erhielt man in Tab. 2.5 nur $q \approx \frac{1}{3}$. Dort war ein x lediglich algebraisch isoliert worden.

Tab. 2.6. Rechenergebnisse nach der regula falsi für die beiden im Text genannten Beispiele. Im Vergleich zu Tab. 2.5 liegt nun beim ersten Beispiel ein besserer Konvergenzfaktor vor; im zweiten Beispiel wurde Konvergenz erreicht. Der Fixpunkt ist $x^* = 3$

i	Beispiel 1 x_i	Beispiel 2 x_i
0	2	2
1	2.9099	3.24
2	2.99941	2.942
3	2.999962	3.014
4	2.9999976	2.9965
5	2.99999984	3.0008
6	2.999999988	2.99979
7		3.000005
8		2.9999879
9		3.0000029

2. Die gleiche Aufgabe mit

$$a = 2,\ b = -0.2 \text{ und } c = -1.597224578$$

und ebenfalls mit $x_B = 4$ führt zu der Vorschrift

$$x_1 = x_0 - \frac{(x_0 - 4)\,(\ln x_0 - 0.1x_0 - 0.798612289)}{\ln x_0 - 0.1x_0 - 0.986294361}.$$

Die Rechenergebnisse sind für den Startwert $x_0 = 2$ ebenfalls in Tab. 2.6 dargestellt. Jetzt zeigt sich ein Konvergenzfaktor $q \approx \frac{1}{4}$. In Tab. 2.5 bei nur algebraisch isoliertem x wurde keine Konvergenz registriert.

3. Abschätzung der Konvergenzfaktoren in

a) Beispiel 1:

$$x^* - x_B = 3 - 4 = -1 \quad \text{und} \quad g'(x^*) = \frac{1}{x^*} - 1 = -\frac{2}{3},$$

$$g(x_B) = -0.712317928,$$

$$q = 1 - \frac{-0.667}{-0.712} = 0.063 \text{ positiv},$$

daher monotone Annäherung an x^*. Dieses q stimmt recht gut mit dem beobachteten Wert überein.

b) Beispiel 2:

$$x^* - x_B = -1 \quad \text{und} \quad g'(x^*) = \frac{1}{x^*} - 0.1 = 0.23333333$$

$$g(x_B) = 0.187682072,$$

$$q = 1 - \frac{0.2333}{0.1877} = -0.24 \text{ negativ},$$

daher alternierende Annäherung an x^*. Das $|q|$ stimmt gut mit dem beobachteten Wert überein.

2.1.4. Iterationsverfahren nach Newton

Anstelle der Sekante, welche ja außer dem iterierten Punkt noch einen Bezugspunkt oder doch stets zwei Punkte des Graphen der Funktion $g(x)$ aus der Aufgabe $g(x) = 0$ benötigt, kann man die Tangente verwenden und braucht dann nur noch einen Punkt (Abb. 2.3). Allerdings wird bei diesem auch noch die Ableitung der Funktion berechnet. Der Gedanke geht auf Newton (1700) zurück, und deshalb wird das Verfahren auch nach ihm benannt.

Die Punkte (x, y) der Tangente durch den Punkt (x_0, y_0) der Funktion $y = g(x)$ genügen der Beziehung

$$\frac{y - y_0}{x - x_0} = g'(x_0),$$

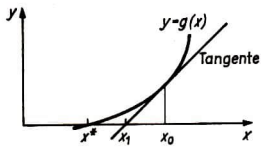


Abb. 2.3. Zur Herleitung des Iterationsverfahrens nach Newton

wobei man sich zu erinnern hat (Stoff der 11. Klasse), daß der Steigungswinkel α_0 der Tangente durch

$$\tan \alpha_0 = g'(x_0)$$

festgelegt ist. Der Schnittpunkt der Tangente mit der x-Achse ist durch (x_1, y_1) mit $y_1 = 0$ bestimmt, d. h.

$$\frac{x_1 - x_0}{-y_0} = \frac{1}{g'(x_0)},$$

$$x_1 = x_0 - \frac{y_0}{g'(x_0)} = x_0 - \frac{g(x_0)}{g'(x_0)}$$

oder

$$x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}, \tag{15}$$

und das ist die gesuchte Iterationsvorschrift. Hier liefert das Überprüfen der Konvergenzeigenschaften

$$x = f(x) = x - \frac{g(x)}{g'(x)},$$

$$f'(x) = 1 - \frac{(g'(x))^2 - g''(x)\, g(x)}{(g'(x))^2} = \frac{g''(x)\, g(x)}{(g'(x))^2} \tag{16}$$

und nun für $x = x^*$ wegen $g(x^*) = 0$

$$f'(x^*) = 0 < 1, \tag{17}$$

aber es muß $g'(x^*) \neq 0$ sein.

Das ist besonders gut, es ist kein q ungleich Null angebbar. Die Konvergenz ist überlinear. NEWTON hat einen „Traumalgorithmus" gefunden. Für eine nun nötige genauere Untersuchung des Verhaltens dieses wichtigen und praktisch sehr oft angewandten Verfahrens rechnet man mit Verwendung der Taylorentwicklung von $f(x)$:

$$f(x_0) = f(x^*) + f'(x^*)\,(x_0 - x^*) + \frac{f''(x^*)}{2}\,(x_0 - x^*)^2 + O\big((x_0 - x^*)^3\big).$$

Dabei wurde die O-Symbolik nach LANDAU benutzt. Sie besagt, daß hier der gesamte Rest der Taylorentwicklung für $x_0 - x^* \to 0$ sich nicht „schlimmer" als $c(x_0 - x^*)^3$ verhält oder, genauer, daß es ein positives c gibt mit der Eigenschaft

$$|\text{Rest}| \leqq c(x_0 - x^*)^3.$$

Da $x_1 = f(x_0)$, $x^* = f(x^*)$ und $f'(x^*) = 0$ gilt, wird

$$x_1 - x^* = \frac{f''(x^*)}{2}\,(x_0 - x^*)^2 + O\big((x_0 - x^*)^3\big)$$

bzw.

$$\varepsilon_1 = \frac{f''(x^*)}{2}\,\varepsilon_0^2 + O(\varepsilon_0^3). \tag{18}$$

Aus (16) erhält man durch nochmaliges Differenzieren an der Stelle $x = x^*$

$$f''(x^*) = \frac{g''(x^*)}{g'(x^*)}, \tag{19}$$

und damit ist auch

$$\varepsilon_1 = \frac{g''(x^*)}{2g'(x^*)}\,\varepsilon_0^2 + O(\varepsilon_0^3). \tag{20}$$

Es steht demnach der neue Fehler ε_1 in quadratischer Relation zum alten ε_0. Wie schon früher überlegt, liefert also das Newtonverfahren in der Nähe der Lösung pro Iterationsschritt eine Verdopplung der Anzahl korrekter Ziffern. Aber es muß $g'(x^*) \neq 0$ sein. Ist $g'(x^*) = 0$, so liegt beim Fixpunkt eine Tangente parallel zur x-Achse vor (Schulstoff 11. Klasse). Das bedeutet für $g(x)$ mindestens eine Doppelnullstelle, und eine ganz ähnliche Rechnung mit Hilfe der Taylorentwicklung zeigt, daß dann nur lineare Konvergenz vorliegt.

Beispiel

1. Es werde wieder die Aufgabe

$$g(x) = a \ln x + bx + c$$

mit $a = 2$, $b = -2$, $c = 3.802775422$ betrachtet. Da $g'(x) = \dfrac{a}{x} + b$ sehr einfach ist, entsteht die Iterationsvorschrift

$$x_1 = x_0 - \frac{\ln x_0 - x_0 + 1.901387711}{\dfrac{1}{x_0} - 1}.$$

Die Rechenergebnisse werden in Tab. 2.7 gezeigt. Die quadratische Konvergenz reduziert drastisch die nötigen Rechenschritte.

Tab. 2.7. Rechenergebnisse nach dem Newton-Verfahren für die beiden im Text genannten Beispiele. Die Verdopplung der Anzahl korrekter Zifferstellen pro Schritt infolge quadratischer Konvergenz geht in der Nähe des Fixpunktes wegen der endlichen Maschinenarithmetik meist wieder verloren

i	Beispiel 1 x_i	Beispiel 2 x_i
0	2	2
1	3.189	2.76
2	3.0027	2.987
3	3.000000059	2.999958
4	3.000000000	2.999999999

2. Die gleiche Aufgabe mit $a = 2$, $b = -0.2$, $c = -1.597224578$ liefert

$$x_1 = x_0 - \frac{\ln x_0 - 0.1x_0 - 0.798\,612\,289}{\frac{1}{x_0} - 0.1}.$$

Die Rechenergebnisse sind ebenfalls in Tab. 2.7 enthalten.

3. Das Newton-Verfahren eignet sich zur iterativen Wurzelberechnung. Die Aufgabe

$$\sqrt[n]{a} - x = 0$$

ist äquivalent zu der Aufgabe

$$g(x) = x^n - a = 0.$$

Die Ableitung ist leicht zu berechnen, $g'(x) = nx^{n-1}$, und das liefert die Iterationsvorschrift

$$x_1 = x_0 - \frac{x_0^n - a}{nx_0^{n-1}} = \frac{nx_0^n - x_0^n + a}{nx_0^{n-1}},$$

$$x_1 = \frac{1}{n}\left((n-1)\,x_0 + \frac{a}{x_0^{n-1}}\right) \quad (\textit{Heronsche Formel}) \tag{21}$$

als eine einfache Rechenvorschrift mit quadratischer Konvergenz zur Berechnung der n-ten Wurzel, die übrigens schon dem in Alexandria lebenden Griechen Heron (um 130) bekannt gewesen sein soll. Für $n = 2$, d. h. für Quadratwurzeln, erhält man

$$x_1 = \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right), \tag{5}$$

also die bereits den Sumerern in Babylon vor 4000 Jahren bekannte Formel. Für $n = 2$ ist ein Beispiel in Tab. 2.4 enthalten. Für $n = 3$ und $a = 8$ sowie für $n = 5$ und $a = 32$ enthält Tab. 2.8 die Resultate jeweils mit dem Startwert $x_0 = 1$. Die

Tab. 2.8. Rechenergebnisse für $\sqrt[3]{8}$ und $\sqrt[5]{32}$ nach der Heronschen Formel

i	$\sqrt[3]{8}$ x_i	$\sqrt[5]{32}$ x_i
0	1	1
1	3.33	7.2
2	2.46	5.76
3	2.081	4.62
4	2.0031	3.71
5	2.0000049	3.00
6	2.000000000	2.48
7		2.15
8		2.020
9		2.00040
10		2.000000156
11		2.000000000

Werte zeigen deutlich, daß die für quadratisch konvergente Verfahren typische Verdopplung der Anzahl korrekter Ziffern pro Schritt erst einsetzt, wenn der Fehler kleiner als 1 ist. Zuvor ist der Faktor $|g''(x^*)/g'(x^*)|$ von Bedeutung.

Ein *Programm zur Nullstellenberechnung* nach dem Newton-Verfahren ist in Tab. 2.9 enthalten. Es werden dabei zwei Unterprogramme zur Berechnung von $g(x)$ und von $g'(x)$ verwendet. Diese befinden sich zwischen 200 und 290 bzw. zwischen 300 und 390. Der Nutzer

Tab. 2.9. BASIC-Programm für das Newton-Verfahren

```
10 REM Newton
20 INPUT "Startwert x0 = " ;x0, "Epsilon ="; e
30 GOSUB 200
40 GOSUB 300
50 LET x1 = x0 - y0 / ys
60 IF ABS (x1 - x0) < e THEN GOTO 90
70 LET x0 = x1
80 GOTO 30
90 PRINT "Nullstelle ="; x1
100 STOP
200 REM Funktion y0 = g(x0)
290 RETURN
300 REM Ableitung von g(x0) = ys
390 RETURN
```

hat die Möglichkeit, dort seine speziellen Anweisungen für die jeweils vorliegende Aufgabe einzufügen. Zu Beginn verlangt das Programm Eingabe eines Startwertes x_0 und einer Genauigkeitsschranke ε oder e, wie diese im Programm genannt wird. Für die Nullstellenberechnung der kubischen Parabel

$$y = x^3 - 2x^2 - x/2 + 5$$

wurden beispielsweise eingefügt bzw. eingegeben

```
210 LET y0 = x0 * x0 * x0 - 2 * x0 * x0 - x0 / 2 + 5
310 LET ys = 3 * x0 * x0 - 4 * x0 - .5
```

und

```
x0 = 5      e = .0001
```

worauf auf dem Bildschirm

```
Nullstelle = -1.307438
```

erschien. (Im Programm wurden die Potenzen in Produkte aufgelöst, da der Potenzoperator ↑, ähnlich wie die Taschenrechnertaste x^y, über Logarithmus- und Exponentialfunktion arbeitet. Bei negativer Basis stößt man also auf einen Argumentfehler!)

Ein zusätzlicher Prüfdruck des Funktionswertes y_0 für das letzte x_0

```
PRINT "y0 = "; y0
```

lieferte

```
y0 = -0.000000009
```

Möchte man alle iterierten Werte wahlweise sehen, so kann man im Programm noch

```
25 INPUT "Zwischendrucke?"; a$
55 IF a$ = "j" THEN PRINT "x = "; x1
```

einfügen. Es wird dann zu Beginn noch nach der Entscheidung gefragt (j für „ja" und jede andere Taste für „nein"; das Dollarzeichen $ ist im verwendeten BASIC-Dialekt eine Kennung für Zeichenketten- bzw. Text-Variable). Bejahendenfalls erhält man die Zwischenwerte

$x = 3.58$	$x = -10.85$	$x = -1.47$
$x = 2.59$	$x = -7.05$	$x = -1.322$
$x = 1.76$	$x = -4.56$	$x = -1.3076$
$x = -0.18$	$x = -2.95$	$x = -1.3074384$
$x = -16.56$	$x = -1.97$	$x = -1.3074384$

deren Aufeinanderfolge recht lehrreich ist. Beim gewählten Startwert springen die iterierten Werte zunächst suchend hin und her, bis einer in den Anziehungsbereich fällt. Die quadratische Konvergenz wird wirksam ab

$$x = -1.322.$$

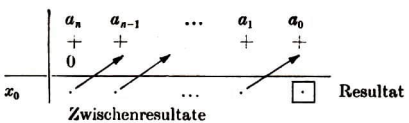
Das Newton-Verfahren wird gern zur *Berechnung von Polynomnullstellen* herangezogen. Die Programme zur Berechnung von $g(x)$ und $g'(x)$ folgen in diesem Fall dem sogenannten Horner-Schema, das aber auch schon NEWTON bekannt gewesen sein soll. Das Polynom n-ten Grades $g(x) = P_n(x)$, gewöhnlich in der Form

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{k=0}^{n} a_k x^k \tag{22}$$

geschrieben, läßt sich schneller für gegebenes x wertmäßig berechnen, wenn die Form

$$P_n(x) = \left(\cdots((0 + a_n)\, x + a_{n-1})\, x + \cdots + a_1\right) x + a_0 \tag{23}$$

verwendet wird. Dies führt auf das einfach zu handhabende Schema



Dabei wird in den Senkrechten addiert, und diese Summen, mit x_0 multipliziert, werden in Pfeilrichtung übertragen.

Hält man bei einem kleinen Rechner mit Postfixarithmetik der Kellertiefe 3 die Koeffizeiten a_k in Speicherregistern 0 bis n, so ist für einen x_0-Wert die Befehlsfolge zur Abarbeitung des Schemas

```
R → x n
  +
  *
R → x n − 1
  +
  *
  ⋮
R → x 1
  +
  *
R → x 0
  +
```

Dabei ist die Anfangsbesetzung des Kellers mit x_0 im Register y und z und 0 im Register x anzusetzen. Es entsteht also ein Zyklus aus drei immer gleichen Befehlen, bei dem sich nur die Registeradresse jeweils um 1 verringert.

Die Nutzung des Horner-Schemas geht jedoch noch weiter. Die Zwischenresultate unter dem Strich sind nämlich die Koeffizienten des Polynoms $P_{n-1}(x)$, welches der Beziehung

$$P_n(x) = P_{n-1}(x)\,(x - x_0) + P_n(x_0) \tag{24}$$

genügt, die Ableitung $P'_n(x) = P'_{n-1}(x)\,(x - x_0) + P_{n-1}(x)$ nimmt also an der Stelle $x = x_0$ den Wert

$$P'_n(x_0) = P_{n-1}(x_0) \tag{25}$$

an. Demzufolge kann einfach unter Benutzung der Zwischenresultate wiederum nach dem Horner-Schema auch der Ableitungswert berechnet werden.

Beispiel

$P_5(x) = 3x^5 - 4x^3 + 5x^2 - 17x - 51$ und $x_0 = 2$

	3	0	−4	5	−17	−51	
	0	6	12	16	42	50	
2	3	6	8	21	25	−1	$= P_5(2)$
	0	6	24	64	170		
2	3	12	32	85	195		$= P'_5(2)$

Ohne diese einfache Beziehung hätte man wie folgt zu rechnen:

$P'_5(x) = 15x^4 - 12x^2 + 10x - 17$

	15	0	−12	10	−17	
	0	30	60	96	212	
2	15	30	48	106	195	$= P'_5(2)$

Das ergibt natürlich dasselbe Resultat. Zur Berechnung einer Nullstelle des Polynoms $P_5(x)$ erhält man also mit dem Startwert $x_0 = 2$ eine erste Näherung

$$x_1 = 2 + \frac{1}{195} = 2.005\ldots$$

Die Folge der iterierten Werte ist hier

2,
2.005,
2.005 098 675,
2.005 098 663,
2.005 098 663,

und für den letzten ist $P_5(x) = -0.8 \cdot 10^{-7}$, und unter diesen Polynomwert kommt man hier bei 10stelliger Rechnung nicht, denn bei 2.005 098 664 ist der Wert sogar $1.2 \cdot 10^{-7}$.

Weitere Nullstellen müßte man aus dem Polynom $P_4(x)$ mit den Koeffizienten

3,
6.015 295 989,
8.061 261 950,
21.163 625 56,
25.435 157 31

gewinnen, und diese Koeffizienten sind natürlich ungenau, da die abgespaltene Nullstelle ungenau ist, wie oben gezeigt wurde. Auf diese Weise entstehen immer größere Fehler, und jede Nullstelle muß am Ursprungspolynom nachiteriert werden.

Ein Programm für einen Postfixrechner für das auf die beschriebene Weise erweiterte Horner-Schema wird die Zwischenresultate nun nicht mehr im Keller aufsammeln können, sondern benutzt für sie zwei Register P und P'. Dabei sind dann die Zwischenresultate aus P die Koeffizienten für P'.

Anfang:	R → x	n	
	x → R	P	
	x → R	P'	Höchster Koeffizient ist in den Registern
Zyklus:	R → x	x	Multiplikation mit x in den Registern
	x → R ∗ P		
	x → R ∗ P'		
	x → R	k	k-ter Koeffizient von P
	x → R + P		Zwischenresultat von P ist Koeffizient von P'
	R → x	P	
	x → R + P'		Zwischenresultat von P'

Der Zyklus muß für P' einen Durchgang eher als für P verlassen werden, und zwar nach der Additionsstelle.

Es wurde hier von der oft bei Postfixrechnern angebotenen Registerarithmetik Gebrauch gemacht: Mit den Inhalten des obersten Kellerregisters und des adressierten Registers wird die Operation ausgeführt, das Resultat im adressierten Register abgelegt, während der Inhalt des obersten Kellerregisters erhalten bleibt.

Mit zwei Infixrechnern (z. B. Schulrechner SR 1) und vorhandener Hierarchiesteuerung, jedoch nur mit je einem Speicher M, kann das erweiterte Horner-Schema „kollektiv" bearbeitet werden (x befindet sich im Speicher M):

a_n		
$*$		a_n
MR		$*$
$+$		MR
a_{n-1}		$+$
$=$	$\rightarrow$	a'_{n-1}
$*$		$=$
MR		$*$
$+$		MR
...		...
$+$		MR
a_1		$+$
$=$	$\rightarrow$	a'_1
$*$		$=$
MR		
$+$		
a_0		
$=$		

Jeweils bei einem „="-Zeichen werden eine Multiplikation mit x und eine Addition mit dem neu eingegebenen Koeffizienten ausgeführt. Die dabei entstandenen Resultate des „linken" Rechners wurden als Koeffizienten in den „rechten" Rechner übernommen. Bei einem Rechnertyp ohne Hierarchie (z. B. MR 610) drückt man nicht die „="-Taste, sondern verharrt zum Übertragen des Zwischenresultats nach der „*"-Taste; denn nach dem Drücken dieser Taste wurde die Addition des zuvor eingetasteten Koeffizienten durchgeführt und das Ergebnis angezeigt.

Tab. 2.10. BASIC-Programm für das erweiterte Horner-Schema zum Anschluß an das Newton-Programm in Tab. 2.9

```
210 LET y0 = 0: LET ys = 0
220 FOR i = 6 TO 1 STEP -1
230 LET ys = ys * x0 + y0
240 LET y0 = y0 * x0 + a(i)
250 NEXT i
```

Das BASIC-Programm für das erweiterte Horner-Schema in Tab. 2.10 ist für sechs Koeffizienten eines Polynoms ausgelegt und berechnet den Funktionswert und den Ableitungswert, so daß das Programm für die Ableitung (300 bis 390 in Tab. 2.9) leer bleiben kann. Es wird zur Berechnung das Vorhandensein eines Koeffizientenvektors a vorausgesetzt. Diese Bedingung erfüllt man mit

```
21 DIM a(6)
22 FOR i = 6 TO 1 STEP -1
23 READ a(i)
24 NEXT i
```

und für das genannte Beispiel mit den Daten

400 3, 0, −4, 5, −17, −51

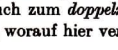
als Zusätze zum Newton-Programm in Tab. 2.9.

Bei Polynomen höheren Grades können im Horner-Schema erhebliche Stellenauslöschungen durch Subtraktion auftreten. Es ist also Vorsicht bei der Benutzung der Ergebnisse geboten, und strenge Kontrolle ist angebracht. Das Horner-Schema kann auch zum *doppelzeiligen* für die Behandlung komplexer Argumente erweitert werden, worauf hier verzichtet wird (vgl. [29], S. 259ff.; [32], S. 107ff.).

2.1.5. Regula falsi (2. Form)

Bei einer Kurve geht eine Sekante in eine Tangente über, wenn die beiden Kurvenpunkte, welche die Sekante bestimmen, gegeneinander streben. Das kann man bei der regula falsi ausnutzen, indem man den Bezugspunkt nicht mehr starr hält, sondern durch den iterierten Punkt $(x_i, g(x_i))$ ersetzt: Die Sekante wird durch jeweils zwei aufeinanderfolgende iterierte Werte (und zugehörige Funktionswerte) bestimmt:

$x_0, x_1, x_2, x_3, \ldots$



Da diese Werte konvergieren, streben die Sekanten gegen eine Tangente, und diese Form der regula falsi wird dem Verfahren von Newton im Verlauf der Rechnung immer ähnlicher. Man wird also bessere als nur lineare Konvergenz erwarten dürfen. Tatsächlich zeigt eine ähnliche Rechnung (Verwendung von Taylor-Entwicklungen, vgl. MfL Bd. 9, 4.1.4., und [32]) wie beim Newton-Verfahren, daß der Konvergenzgrad (Konvergenzordnung (4))

$$p = \frac{1+\sqrt{5}}{2} = 1.62 \tag{26}$$

ist, also zwischen 1 (linear) und 2 (quadratisch) liegt.

Diese Zahl ist übrigens das Verhältnis des „Goldenen Schnittes" (vgl. MfL Bd. 8) und charakterisiert auch das Wachstum der Fibonaccizahlen. Diese Bemerkung dient nur als Hinweis darauf, daß interessante Zahlen in verschiedenen Gebieten der Mathematik auftreten.

Die Iterationsformel lautet jetzt (vgl. regula falsi 1. Form) bei Ersatz des Bezugspunktes durch einen iterierten Wert

$$x_{(2)} = x_{(1)} - \frac{x_{(1)} - x_{(0)}}{g(x_{(1)}) - g(x_{(0)})}\, g(x_{(1)}). \tag{27}$$

Es muß bei der Anwendung dieser Formel pro Iterationsschritt ein neuer Funktionswert $g(x_{(1)})$ berechnet werden. Zweimaliges Anwenden liefert wegen des Konvergenz-

grades 1.62 dann oft bessere Werte als ein Newton-Schritt, denn $1.62^2 = 2.62$ ist größer als 2. In einem Newton-Schritt müssen auch zwei Funktionswerte, $g(x_{(0)})$ und $g'(x_{(0)})$, berechnet werden.

Abb. 2.4 zeigt das Verhalten der Vorgehensweise gemäß (27). Es kann im Verlauf der Rechnung durchaus eintreten, daß zwei aufeinanderfolgende iterierte Werte auf derselben Seite des Fixpunktes liegen (im Bild x_1 und x_3) und Funktionswerte gleichen Vorzeichens (vgl. auch Tab. 2.11) liefern (im Bild positiv). Wenn man glaubt, dies sei ein Nachteil, und eine „Verbesserung" einfügt dahingehend, daß von den aufeinanderfolgenden iterierten Werten immer derjenige durch den neuen Wert

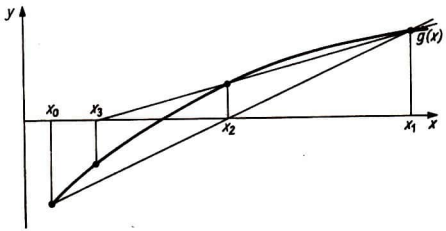


Abb. 2.4. Zur regula falsi (2. Form)

ersetzt wird, der einen Funktionswert mit dem gleichen Vorzeichen liefert, so gibt man in überraschender Weise den Konvergenzgrad 1.62 auf und fällt zurück auf lineare Konvergenz. Man erkennt dies schon aus Abb. 2.4, denn dann wird x_0 zum festen Bezugspunkt x_B. Obwohl man also durch diese „verbesserte" Vorgehensweise den Fixpunkt immer zwischen den iterierten Werten einschließt, was optisch oder anschaulich besser zu sein scheint, ist diese in der Praxis verbreitete Variante der regula falsi schlechter.

Tab. 2.11 zeigt für das schon öfter demonstrierte Beispiel

$$g(x) \equiv a \ln x + bx + c = 0$$

mit $a = 2$, $b = -2$ und $c = -2 \ln 3 + 6 = 3.802\,775\,422$, also leicht verändert auf

$$\ln x - x + 1.901\,387\,711 = 0$$

mit dem Fixpunkt $x^* = 3$ die bessere Konvergenz gegenüber der regula falsi 1. Form in Tab. 2.6, Beispiel 1.

Die regula falsi 2. Form gibt durch die Tatsache, daß zwei vorhergehende iterierte Werte zur Berechnung eines neuen verwendet werden, den Anlaß zu einer neuen Begriffsbildung.

Iterationsformeln der Gestalt

$$x_n = f(x_0, x_1, \ldots, x_{n-1})$$

nennt man n-stufig (vgl. MfL Bd. 9, 4.1.4., und [57] unter diesem Stichwort). Die oben angegebene Fassung des Banachschen Fixpunktsatzes müßte für solche Verfahren noch verallgemeinert werden, was natürlich möglich ist.

Die regula falsi 2. Form ist also ein zweistufiges Verfahren.

Tab. 2.11. Iterierte Werte und zugehörige Funktionswerte für das im Text angegebene Beispiel zur regula falsi 2. Form. Die Funktionswerte zeigen, daß der Fixpunkt $x^* = 3$ nicht immer zwischen zwei aufeinanderfolgenden iterierten Werten liegt. Es müßten sonst die Funktionswerte im Vorzeichen alternieren

i	x_i	$g(x_i)$
0	1	0.90
1	2	0.59
2	3.94	−0.67
3	2.91	0.057
4	2.995	0.0035
5	3.0000215	−0.0000143
6	3.0000000	$6.4 \cdot 10^{-9}$
7	3.0000000	0

2.1.6. Bisektion

Beim Einsatz von Taschenrechnern, besonders bei nichtprogrammierbaren, wird man die längeren Tastenfolgen für die beschriebenen Iterationsverfahren (diejenigen für $g(x)$ und evtl. $g'(x)$ kommen noch dazu) als lästig empfinden. Auch kann man sich allzu leicht dabei vertippen. Man wird sich dann auf die Berechnung von $g(x)$ für mehrere x beschränken und nur den Verlauf der Funktionswerte beobachten. Hat man x_0 und x_1 mit der Eigenschaft $g(x_0) < 0$ und $g(x_1) > 0$ gefunden, so muß nach dem Satz vom Zwischenwert (MfL Bd. 4) für stetiges $g(x)$ jeder Wert zwischen $g(x_0)$ und $g(x_1)$ wenigstens einmal angenommen werden und also auch eine Nullstelle zwischen ihnen liegen. Man kann sie nach „Augenmaß" linear interpoliert schätzen mit x_2. Man kann aber auch einfach die Mitte des Intervalls (x_0, x_1) nehmen:

$$x_2 = \frac{1}{2}(x_0 + x_1).$$

Jedenfalls wird x_2 dieses Intervall in zwei Teile „zerschneiden", und daher kommt der Name dieser Methode (bi = Vorsilbe für zwei, sezieren = schneiden, Sektion = Teil, Schnitt). Man berechnet dann $g(x_2)$ und prüft das Vorzeichen. Ist

$g(x_2) > 0$, so wird das neue Intervall $(x_0, x_1) := (x_0, x_2)$,

$g(x_2) < 0$, so wird das neue Intervall $(x_0, x_1) := (x_2, x_1)$.

Auf diese Weise wird das die Nullstelle einschließende Intervall immer kleiner.

Abb. 2.5 veranschaulicht das Verfahren. Der Konvergenzfaktor ist offensichtlich

0.5, und nach drei bis vier Iterationsschritten gewinnt man eine neue korrekte Resultatstelle. Das Verfahren genügt demnach für die oft nur praktisch geforderte Genauigkeit von zwei bis drei Stellen.

Tab. 2.12 zeigt die Rechenwerte für die Aufgabe

$$g(x) \equiv \ln x - x + 1.901387711 = 0,$$

wenn als Startintervall (2.8, 3.1) genommen wird.

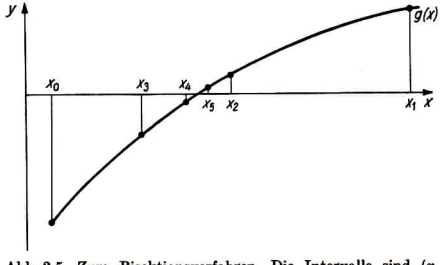


Abb. 2.5. Zum Bisektionsverfahren. Die Intervalle sind (x_0, x_1), (x_0, x_2), (x_3, x_2), (x_4, x_2), (x_4, x_5), die Länge von (x_4, x_5) ist 1/16 der Länge von (x_0, x_1)

Tab. 2.12. Zum Bisektionsverfahren. Das Startintervall ist (2.8, 3.1), und den nächsten x_i-Wert erhält man immer als Mitte des betrachteten Intervalls

i	x_i	$g(x_i)$	Intervall
0	2.8	0.13	—
1	3.1	−0.07	(x_0, x_1)
2	2.95	0.03	(x_2, x_1)
3	3.025	−0.02	(x_3, x_2)
4	2.987	0.01	(x_4, x_2)
5	3.006	−0.004	(x_4, x_5)
6	2.997	0.001	(x_6, x_5)

2.1.7. Konvergenzverbesserungen

Für programmierbare Rechner, aber auch für nichtprogrammierbare, lohnt es sich, über Beschleunigungen der Konvergenz nachzudenken. Man verkürzt dadurch die Rechenzeit bzw. verringert die Anzahl der Iterationsschritte. Besonders bei Handrechnungen mit nichtprogrammierbaren Taschenrechnern wird man für raschere Konvergenz wegen der verringerten Tastenbedienungen dankbar sein.

Die folgende Methode hat schon Huygens (1654) bei der iterativen Berechnung von π angewendet. Sie wurde erst 1936 von Kommerell wieder aufgegriffen und 1955 von dem Norweger Römberg vervollkommnet.

Verbesserung linearer Konvergenz — Schema von Romberg

Bei linearer Konvergenz mit dem (sogar evtl. schlechten) Konvergenzfaktor $0 < q$ (d. h. also, q liegt zu nahe bei 1) gilt die a-posteriori-Fehlerformel (10) aus dem Fixpunktsatz von BANACH

$$x^* - x^{(n+1)} \approx \frac{q}{1-q}\,(x^{(n+1)} - x^{(n)}),$$

die hier ohne Absolutstriche und mit dem „ungefähr-gleich"-Zeichen geschrieben wurde. Daraus erhält man

$$x^* \approx x^{(n+1)} + \frac{q}{1-q}\,(x^{(n+1)} - x^{(n)}) \approx \frac{x^{(n+1)} - qx^{(n)}}{1-q},$$

und da dies nur ungefähr den Fixpunkt liefert, hat man statt dessen einen besseren iterierten Wert

$$\bar{x}^{(n+1)} = \frac{x^{(n+1)} - qx^{(n)}}{1-q}. \tag{28}$$

Die Folge der $\bar{x}^{(k)}$ konvergiert zwar ebenfalls linear gegen x^*, jedoch ist nun

$$\bar{q} = q^2$$

der Konvergenzfaktor, und das ist wegen $q < 1$ besser als zuvor.

Tab. 2.13 zeigt für das Beispiel der Berechnung von $\sqrt[3]{8}$ nach der Formel $x := \sqrt{\sqrt{8x}}$ mit dem Startwert $x_0 = 3$ (vgl. Tab. 2.1) die Verbesserung. Tatsächlich erkennt man auch den besseren Konvergenzfaktor $\frac{1}{16} = 0.0625$. Die Werte $\bar{x}_i$ werden wegen $q = \frac{1}{4}$ gemäß

$$\bar{x}_i = \frac{x_i - \frac{1}{4}\,x_{i-1}}{1 - \frac{1}{4}} = \frac{4x_i - x_{i-1}}{3}$$

gewonnen.

Bei alternierender Iteration ist $-1 < q < 0$, also q negativ, und dies ist in der Formel der Konvergenzverbesserung zu beachten. Beispielsweise erhält man für $q = -\frac{1}{4}$

$$x_i = \frac{4x_i + x_{i-1}}{5}.$$

Man kann nun auf die $\bar{x}_i$, welche mit q^2 linear konvergieren, eine ähnliche (s. u.) Überlegung anwenden — dies war der Gedanke von RÖMBERG — und erhält eine neue, wiederum besser konvergente Folge (mit q^3) usw. Es soll nun der Iterationsindex

Tab. 2.13. Verbesserung linearer Konvergenz. Durch Rundungsfehler und Stellenauslöschung bei der Subtraktion fast gleicher Größen ergeben sich die falschen Werte in den letzten beiden Zeilen der vierten Spalte

i	x_i	Fehlerquotient	$\bar{x}_i$	Fehlerquotient
0	3	—	—	—
1	2.213363839	0.21	1.951151785	—
2	2.051330793	0.24	1.997319778	0.0549
3	2.012711007	0.248	1.999837745	0.0605
4	2.003170206	0.249	1.999989939	0.0620
5	2.000792081	0.2499	1.999999373	0.0623
6	2.000197991	0.24996	1.999999961	0.0622
7	2.000049496	0.24999	1.999999998	0.0513
8	2.000012374	0.25	2.000000000	0

mit i (tiefgestellt) und der Romberg-Index mit r (hochgestellt) angegeben werden. Mit $r = 0$ werden die Werte nach der ursprünglichen Iterationsformel bezeichnet, und die in Tab. 2.13 dazu berechneten verbesserten Werte würden $r = 1$ haben. Für die Romberg-2-Werte gilt

$$x_i^{(2)} = \frac{x_i^{(1)} - q^2 x_{i-1}^{(1)}}{1 - q^2},$$

nnd da dies mit q^3 konvergiert, erhält man für die Romberg-3-Werte

$$x_i^{(3)} = \frac{x_i^{(2)} - q^3 x_{i-1}^{(2)}}{1 - q^3}.$$

Diese werden dann mit q^4 konvergieren. Somit ist allgemein

$$x_i^{(r)} = \frac{x_i^{(r-1)} - q^r x_{i-1}^{(r-1)}}{1 - q^r}. \tag{29}$$

Tab. 2.14 zeigt den Effekt dieser äußerst wirkungsvollen Methode, die auch unter dem Namen *Extrapolation zur Grenze* oder *Richardson-Extrapolation* (1927) bekannt ist (vgl. [18], [29], [32]).

Tab. 2.14. Romberg-Schema für die Iteration $x = \sqrt{8x}$, $x^* = 2$. In der nullten Spalte gilt $q = \frac{1}{4}$, in der ersten Spalte $q = \frac{1}{16}$, in der zweiten Spalte $q = \frac{1}{64}$ und in der dritten Spalte $q = \frac{1}{256}$

i	$x_i^{(0)}$	$x_i^{(1)}$	$x_i^{(2)}$	$x_i^{(3)}$	$x_i^{(4)}$
0	3	—	—	—	—
1	2.213363839	1.951151785	—	—	—
2	2.051330793	1.997319778	2.000397645	—	—
3	2.012711007	1.999837745	2.000005609	1.999999387	—
4	2.003170206	1.999989939	2.000000085	1.999999997	1.999999999

Zur Herleitung der Romberg-Formel muß man wissen, daß die iterierten Werte $x_i^{(0)}$ bei linearer Konvergenz mit dem Kovergenzfaktor q sehr oft der Beziehung

$$x_i^{(0)} = x^* + a_1 q^i + a_2 q^{2i} + a_3 q^{3i} + \cdots$$

genügen. Es kommt dabei gar nicht auf die Kenntnis der Koeffizienten a_k an. Für den unmittelbar vorhergehenden iterierten Wert gilt also

$$x_{i-1}^{(0)} = x^* + a_1 q^{i-1} + a_2 q^{2(i-1)} + a_3 q^{3(i-1)} + \cdots,$$

und es ergibt sich

$$x_i^{(0)} - q x_{i-1}^{(0)} = (1-q)\, x^* + a_2 \frac{q-1}{q} q^{2i} + a_3 \frac{q^2-1}{q^2} q^{3i} + \cdots,$$

$$x_i^{(1)} = x^* - a_2 q^{2i-1} - a_3 q^{3i-2}(q+1) - \cdots.$$

Daraus erkennt man den Konvergenzfaktor q^2. Bildet man

$$x_i^{(1)} - q^2 x_{i-1}^{(1)} = (1-q^2)\, x^* + a_3 \frac{1-q^2}{q^3} q^{3i} + \cdots$$

oder

$$x_i^{(2)} = x^* + a_3 (q^{3i-3}) + \cdots,$$

so erkennt man nun den Konvergenzfaktor q^3. In dieser Weise verfährt man weiter. Die Bauart der erhaltenen Entwicklungen macht auch deutlich, daß die Werte $x_i^{(r)}$ mit wachsendem r abwechselnd größer und kleiner als x^* sein werden. Dies bestätigt ein Blick auf Tab. 2.14. Natürlich kommt es dabei auch auf die Vorzeichen der unbekannten a_k an. Tab. 2.14 läßt demnach vermuten, daß für diese spezielle Iteration a_1 positiv, a_2 positiv, a_3 positiv, a_4 positiv, aber a_5 negativ sein wird. Die in Tab. 2.14 genannten Werte von q erhält man nach (3) näherungsweise als Fehlerquotient.

Konvergenzbeschleunigung durch Übergang auf quadratische Konvergenz

Es werden zwei Verfahren vorgestellt. Das erste Verfahren geht auf AITKEN (MfL Bd. 9, 4.1.3.) und auf STEFFENSEN ([32] S. 102) zurück. Der Übergang von beliebiger (schlechter) linearer Konvergenz auf quadratische ist äußerst verlockend, da dann in jedem Iterationsschritt die Anzahl korrekter Ziffern verdoppelt wird. Jedenfalls geschieht dies, sobald der Fehler wesentlich kleiner als 1 geworden ist.

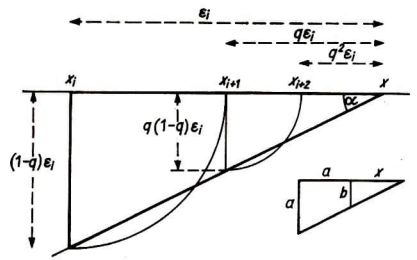


Abb. 2.6. Zur Herleitung der Steffensen-Konvergenzbeschleunigung

Abb. 2.6 zeigt die Annäherung der Iterationswerte bei linearer Konvergenz an den unbekannten Fixpunkt x^*. Durch Herausdrehen der beiden Änderungen

$$|x_i - x_{i+1}| = (1-q)\,\varepsilon_i, \qquad |x_{i+1} - x_{i+2}| = q(1-q)\,\varepsilon_i,$$

die ja bekannt sind, entsteht eine Figur, die das Anwenden des Strahlensatzes gestattet. Es ist

$$\tan\alpha = \frac{(1-q)\,\varepsilon_i}{\varepsilon_i} = \frac{q(1-q)\,\varepsilon_i}{q\varepsilon_i} = 1-q,$$

und aus der Übersichtsfigur mit $a = |x_i - x_{i+1}|$, $b = |x_{i+1} - x_{i+2}|$ ist x zu bestimmen. Ein besserer iterierter Wert $\bar{x}_i$ ergibt sich also aus

$$\bar{x}_i = x_{i+1} + x.$$

Der Strahlensatz besagt, daß $\dfrac{a}{b} = \dfrac{a+x}{x}$ ist. Daraus erhält man $x = \dfrac{ab}{a-b}$ und schließlich

$$\bar{x}_i = \frac{x_{i+1}^2 - x_i x_{i+2}}{2x_{i+1} - x_i - x_{i+2}}. \tag{30}$$

Dies läßt sich mit

$$\Delta x_i = x_{i+1} - x_i, \quad \Delta^2 x_i = \Delta x_{i+1} - \Delta x_i = x_{i+2} - 2x_{i+1} + x_i$$

wegen der leicht nachzurechnenden Formel

$$\bar{x}_i = x_i - \frac{(x_{i+1} - x_i)^2}{x_{i+2} - 2x_{i+1} + x_i}$$

auch als

$$\bar{x}_i = x_i - \frac{(\Delta x_i)^2}{\Delta^2 x_i} \tag{31}$$

schreiben, und deshalb nennt man diesen Prozeß zum Gewinnen schneller konvergierender Iterationswerte auch den *Aitken-Δ^2-Prozeß* (1926). Diese Werte konvergieren allerdings noch linear, wie es auch Tab. 2.15 ausweist, mit dem Faktor q^2. Hier setzt nun aber der Gedanke von Steffensen (1933) ein. Unter Beachtung der ursprünglichen Iterationsformeln

$$x_{i+1} = f(x_i) \quad \text{und} \quad x_{i+2} = f(x_{i+1}) = f\big(f(x_i)\big)$$

wird daraus eine neue Iterationsformel gewonnen:

$$x_{i+1} = \frac{\big(f(x_i)\big)^2 - x_i f\big(f(x_i)\big)}{2f(x_i) - x_i - f\big(f(x_i)\big)} = x_i - \frac{\big(f(x_i) - x_i\big)^2}{f\big(f(x_i)\big) - f(x_i) + \big(x_i - f(x_i)\big)}. \tag{32}$$

Auf der rechten Seite (zweite Form ist numerisch günstiger) kommt nur noch x_i vor, also kann

$$x_{i+1} = F(x_i)$$

geschrieben werden, und eine ausführliche Betrachtung (für das zweite Verfahren wird sie unten vorgeführt) zeigt, daß diese Iteration quadratisch konvergent ist. Der

Mehraufwand an Arbeit ist allerdings beträchtlich. Diese Formel oder dieses Verfahren ist ein typisch mehrstufiges Verfahren:

Aus einem iterierten Wert

$$x_i$$

werden durch Anwenden der Formel f zwei weitere hergestellt:

$$x_i' = f(x_i),$$
$$x_i'' = f(x_i').$$

Dann wird die *Rechenrunde* abgeschlossen durch Benutzen dieser drei Werte in einer anderen Formel:

$$x_{i+1} = \frac{x_i'^2 - x_i x_i''}{2x_i' - x_i - x_i''}.$$

In Tab. 2.15 wird die Leistung dieses Verfahrens für die Iteration $x = \sqrt{\sqrt{8x}}$ demonstriert.

Da in der Steffensen-Formel Differenzen fast gleicher Werte auftreten, ist Vorsicht geboten. Rundungsfehler können das Eintreten der quadratischen Konvergenz sabotieren. In der Literatur wird für dieses Verfahren dann auch oft das Anwenden doppelt genauer Rechnung empfohlen. Das verbietet sich natürlich bei Kleinstrechnern.

Zur Herleitung der Formel von STEFFENSEN wird zur Überlegung auch eine andere Figur herangezogen (vgl. [32] S. 102), die besonders überzeugend bei alternierenden Iterationen ist (Abb. 2.7). Die Sekante durch die Punkte

$$(x_0, f(x_0)) \quad \text{und} \quad (x_1, f(x_1)) \quad \text{bzw.} \quad \big(f(x_0), f(f(x_0))\big)$$

ist

$$\frac{y - f(x_0)}{x - x_0} = \frac{f(x_0) - f(f(x_0))}{x_0 - f(x_0)},$$

Tab. 2.15. Anwendung der Aitken-Δ^2-Methode und der Steffensen-Konvergenzbeschleunigung für die Berechnung von $\sqrt[3]{8}$ nach der ursprünglichen Iteration $x = \sqrt{\sqrt{8x}}$ mit $q = \frac{1}{4}$. Die Δ^2-Methode von AITKEN liefert $q = \frac{1}{16}$, die von STEFFENSEN quadratische Konvergenz

i	$x_i = \sqrt{\sqrt{8x_{i-1}}}$	AITKEN x_i	STEFFENSEN x_i
0	3	—	3
1	2.213364	—	2.009296567
2	2.051331	2.009297	2.000001337
3	2.012711	2.000626	2.000000000
4	2.003170	2.000039	
5	2.000792	2.000003	
6	2.000198	2.000000	
7	2.000049		

und ihr Schnittpunkt $\bar{x}_1$ mit der Geraden $y = x$ ergibt sich aus

$$\frac{\bar{x}_1 - f(x_0)}{\bar{x}_1 - x_0} = \frac{f(x_0) - f(f(x_0))}{x_0 - f(x_0)}$$

durch Auflösung wieder in Gestalt der bereits angegebenen Formel (32).

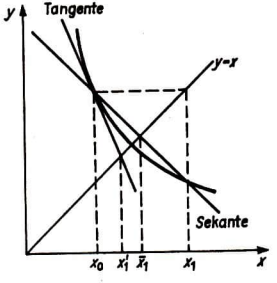


Abb. 2.7. Zur Herleitung der Steffensen-Konvergenzbeschleunigung (Sekante) und der nach Steffensen-Newton-Kombination (Tangente)

Das zweite angekündigte Verfahren benutzt die Skizze aus Abb. 2.7 und dabei aber, dem Gedanken von NEWTON folgend, die Tangente anstelle der Sekante. Man kann dieses Verfahren das *SNK-Verfahren* (*Steffensen-Newton-Kombination*) nennen. Die Tangente an die Kurve $y = f(x)$ im Punkt $\big(x_0, f(x_0)\big)$ ist

$$\frac{y - f(x_0)}{x - x_0} = f'(x_0),$$

und der Schnittpunkt mit $y = x$ liefert x_1' aus Abb. 2.7, das aber nun mit x_1 bezeichnet werden soll. Die Auflösung nach x_1 liefert eine bessere, sogar quadratisch konvergente Formel $F(x)$:

$$\frac{x_1 - f(x_0)}{x_1 - x_0} = f'(x_0),$$

$$x_1 = \frac{f(x_0) - f'(x_0)\, x_0}{1 - f'(x_0)} = F(x_0). \tag{33}$$

(Natürlich muß $f'(x_0) \neq 1$ sein.) Diese Formel bietet den Vorteil, daß keine Differenzen fast gleicher Zahlen auftreten. Allerdings muß eine Ableitung gebildet werden.

Die Formel (33) hat einen ähnlichen Aufbau wie (28) beim Romberg-Verfahren, da $q \approx f'(x_0)$ ist. In (28) liegt aber nur lineare Konvergenz mit q^2 vor, während (33) quadratische Konvergenz anbietet. Zunächst sieht man leicht ein, daß der Fixpunkt x^* von $x = f(x)$ auch Fixpunkt von $x = F(x)$ ist:

$$x^* = \frac{f(x^*) - f'(x^*)\, x^*}{1 - f'(x^*)} = \frac{x^* - f'(x^*)\, x^*}{1 - f'(x^*)} = x^*.$$

Die Taylorentwicklung von $F(x)$ an der Stelle $x = x^*$ mit $\varepsilon = x_0 - x^*$ liefert

$$F(x_0) = F(x^* + \varepsilon) = F(x^*) + F'(x^*)\,\varepsilon + O(\varepsilon^2),$$

und da (mit Verzicht auf Argumente und Indizes)

$$F'(x) = \frac{(f' - f' - xf'')\,(1 - f') + f''(f - xf')}{(1 - f')^2} = \frac{(f - x)f''}{(1 - f')^2}$$

ist, ergibt sich wegen $f(x^*) - x^* = 0$ auch $F'(x^*) = 0$ und damit

$$F(x_0) = x_1 = x^* + O(\varepsilon^2) \quad \text{oder} \quad |x_1 - x^*| = O(|x_0 - x^*|^2).$$

Beispiel

Als Beispiel wird wieder die Iterationsformel $x = \sqrt{\sqrt{8x}}$ zur Berechnung von $\sqrt[3]{8}$ mit dem Startwert $x_0 = 3$ genommen. Sie ist linear konvergent mit dem Konvergenzfaktor $q = \frac{1}{4}$.

$$x = f(x) = (8x)^{1/4} = \sqrt{\sqrt{8x}}, \quad f'(x) = 2(8x)^{-3/4} = \frac{1}{4}\sqrt{\sqrt{\frac{8}{x^3}}},$$

$$x_1 = \frac{\sqrt{\sqrt{8x_0}} - \frac{x_0}{4}\sqrt{\sqrt{\frac{8}{x_0^3}}}}{1 - \frac{1}{4}\sqrt{\sqrt{\frac{8}{x_0^3}}}} = \frac{3x_0\sqrt{\sqrt{8x_0}}}{4x_0 - \sqrt{\sqrt{8x_0}}}.$$

Die hier entstandene Formel läßt sich leicht auf Kleinstrechnern programmieren bzw. auf Taschenrechnern „durchtasten". Tab. 2.16 zeigt die guten Resultate.

Tab. 2.16. Anwendung der Konvergenzbeschleunigung nach der Steffensen-Newton-Kombination für die Berechnung von $\sqrt[3]{8}$ nach der ursprünglichen Iteration $x = \sqrt{\sqrt{8x}}$. Die Abweichung beim Schritt 3 ist auf Rundungsfehler zurückzuführen

i	x_i
0	3
1	2.035456741
2	2.000076643
3	1.999999999
4	2.000000000

2.2. Interpolation

2.2.1. Einführung

Unter Interpolation versteht man das „Zwischenschalten" von Werten einer Funktion, die nur an diskreten Stellen bekannt ist (siehe 2.2.2.). Entweder sind diese Stellen durch tabulierte Werte gegeben und man braucht zwischen diesen liegende Werte, oder die bekannten Funktionswerte wurden durch Messungen gewonnen und man sucht nach einem stetigen funktionellen und möglichst ökonomisch auswertbaren Zusammenhang, um Zwischenwerte durch Berechnung zu gewinnen. Es kann auch sein, daß der funktionelle Zusammenhang zwar bekannt, seine Realisierung aber so aufwendig ist, daß man eine einfachere Berechnung für Zwischenstellen sucht.

Die mit der Interpolation und ihren Verfahren verbundenen Namen von Wissenschaftlern, LAGRANGE und NEWTON, aber auch GAUSS, LAPLACE, STIRLING, BESSEL und GREGORY, deuten darauf hin, daß die oben geschilderte Problemstellung bereits im 17. und 18. Jahrhundert eine Rolle spielte, aber bis in die Gegenwart ihre Bedeutung behielt. Das Problem entstand mit der damaligen Entwicklung der geodätischen (Erdmessung), astronomischen, nautischen (sphärische Geometrie) und technischen Wissenschaften. Für den praktischen Rechengebrauch wurden umfangreiche Tafelwerke der verschiedensten Zusammenhänge hergestellt. Deren Produktion und auch ihre Nutzung benötigten die Interpolation. Die Entwicklung elektronischer Rechengeräte macht zwar die Interpolation elementarer transzendenter Funktionen (sin, tan, lg, exp) unnötig, selbst Taschenrechner bieten diese als Tastenfunktionen an, jedoch bleibt sie weiter bedeutungsvoll für den bereits genannten Sachverhalt diskreter Meßwerte oder komplexerer Zusammenhänge. Solche liegen beispielsweise bei numerischer Werkzeugmaschinensteuerung speziell für die Fertigung von Zahnrädern (Evolventen-Abwälzkurven räumlich) vor.

Bereits in der Schule werden im mathematischen und naturwissenschaftlichen Unterricht Tafelwerke für einfache Funktionen verwendet (Klasse 7 Wurzelfunktion, Klasse 9 Logarithmen, Klasse 10 Winkelfunktionen). Auch für die ökonomischen Berechnungen der Zinseszinsrechnung gibt es Tafeln.

Für das Aufsuchen von Zwischenwerten sind Interpolationen — und seien es meist auch nur grobe lineare Schätzungen — unerläßlich. Im Schulunterricht wird zwar meist auf echtes (lineares) Interpolieren verzichtet, und der Stoffplan sieht es auch nicht mehr vor. Der Lehrer sollte aber auf Schülerfragen nach den Zwischenwerten vorbereitet sein.

Ein Ausschnitt aus einer vierstelligen Logarithmentafel ist

	...	4	5	...
.		.	.	
.		.	.	
.		.	.	
21	...	3304	3324	...
.		.	.	
.		.		
.		.	.	

d. h.

$$\lg 2.140 = 0.3304, \qquad \lg 2.150 = 0.3324,$$

und die vierte Stelle des Arguments ist durch lineare Interpolation zu berücksichtigen („in die Tafel hinein", Aufschlagen von Funktions-(Logarithmus-) Werten) bzw. zu gewinnen („aus der Tafel heraus", Aufsuchen von Argumenten zu den gegebenen Funktionswerten). Es sei

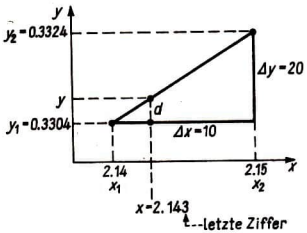


Abb. 2.8. Bei der Herleitung der linearen Interpolationsformel kann der Strahlensatz verwendet werden:
$\Delta y : \Delta x$ = gesuchte Differenz : letzte Ziffer = $d : z$

beispielsweise lg 2.143 gesucht. Abb. 2.8 zeigt den Sachverhalt. Entweder man benutzt nun die Aussage des Strahlensatzes oder verwendet Gedanken aus der analytischen Geometrie, hier die Gleichung der Geraden durch zwei Punkte:

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{\text{Tafeldifferenz}}{10}.$$

Dabei ist die Tafeldifferenz in Einheiten der letzten Tafelstelle zu nehmen. Im Beispiel ist also

$$\Delta y = 20.$$

Die Δx-Größe wird in Einheiten der gegebenen Stelle genannt. Man erhält zur Interpolation

$$y - y_1 = \frac{\Delta y}{\Delta x}(x - x_1) \tag{1}$$

oder zum besseren Merken

$$\text{unsere Differenz} = \frac{\text{Tafeldifferenz}}{10} \cdot \text{neue Ziffer}, \tag{2}$$

im Beispiel

$$d = \frac{20}{10} \cdot 3 = 6.$$

Diese Differenz ist nun laut Abb. 2.8 zum Wert y_1 zu addieren, und zwar in Einheiten der letzten Stelle. Damit erhält man

$$\lg 2.143 = 0.3304 + 0.0006 = 0.3310.$$

Auf vier Stellen gerundet liefert ein Taschenrechner denselben Wert. In diesem Bereich der Logarithmusfunktion und mit vierstelligen Werten genügt lineare Interpolation für eine

Argumentdichte mit drei Stellen, und so ist die Schülertafel auch aufgebaut. In vielen Tafelwerken sind am Rand Interpolationshilfstäfelchen enthalten, die das bei häufiger Anwendung lästig werdende Rechnen mit der oben angegebenen Interpolationsformel (1) oder (2) vermeiden helfen. Sie sind unter den Überschriften der Tafeldifferenzen angegeben und enthalten die *Ziffer* und *unsere Differenz*. Für das Beispiel:

20

Ziffer	1	2	3	4	5	6	7	8	9
Δy	2	4	6	8	10	12	14	16	18

Will man umgekehrt das Argument x zu einem gegebenen Logarithmus aufschlagen,

$$\lg x = 0.3315,$$

so muß man „unsere Differenz" = 11 Einheiten bilden, findet dazu die Ziffer 5 (oder 6) und erhält

$$x = 2.145 \quad \text{(oder } 2.146\text{)}.$$

Der gleiche Wert ergibt sich natürlich auch aus der Interpolationsformel. Ein Taschenrechner liefert nach Rundung ebenfalls den Wert 2.145. Ein genauerer Wert wäre 2.1454, so daß der nach Interpolation ebenfalls akzeptierbare Wert 2.146 doch schon ein wenig ungenau ist. Das ist auch verständlich, da die Logarithmusfunktion eben keine lineare Funktion ist. Man kann nun den Gedanken verfolgen — und mitunter werden auch Schüler nach höherer Genauigkeit fragen — und anstelle der Ersatzgeraden bei linearer Interpolation eine Parabel, kubische Parabel oder Kurven noch höhergradiger Polynome verwenden, um auch das Krümmungsverhalten der Graphen der Funktionen besser zu berücksichtigen. Natürlich müssen dann als Stütze anstelle von nur zwei Punkten deren drei, vier oder mehr einbezogen werden. Man kann als Resultat für die eingesetzte Mehrarbeit bei quadratischer und kubischer Interpolation auf höhere Genauigkeit hoffen, und bei den in der Praxis meist auch nur auftretenden „gutartigen" Funktionen trifft das auch zu.

2.2.2. Allgemeine Interpolationsaufgabe

Bereits in anderen Bänden dieser Reihe (MfL Bd. 4, 2.7., MfL Bd. 9, 4.2.1.) findet man eine Einführung bzw. detailliertere Angaben zur Interpolation. Die Interpolation ist eine spezielle Form des Funktionsersatzes bei möglichst geringem Fehler, d. h. letzten Endes der Approximation oder Funktionsannäherung, in einem vorgegebenen *Intervall* (a, b) für das Argument. Es wird oft übersehen, daß die Angabe des Intervalls wichtig ist. Bei der Interpolation wird für die Forderung eines kleinen Fehlers (und zunächst unter Verschweigen einer Angabe wie dieser gemessen werden soll) vereinfachend gefordert, daß die Differenz zwischen interpolierter (gegebener) und interpolierender (konstruierter) Funktion an mehreren Stellen x_i Null wird:

$$y(x_i) - y_p(x_i) = 0 \quad \text{für} \quad i = 0(1)n. \tag{3}$$

Die Gesamtheit dieser Stellen wird *Referenz* genannt. Die Gleichungen (3) bilden die *Interpolationsbedingung*. Die Konstruktion der Interpolationsfunktion erfolgt dabei als Linearkombination von Funktionen eines ebenfalls vorgegebenen Systems von *Basisfunktionen*

$$g_0(x), g_1(x), \ldots, g_n(x), \ldots$$

Beispiele für solche Systeme sind die Potenzfunktionen

$$1, x, x^2, \ldots,$$

Winkelfunktionen (trigonometrische Interpolation)

$$1, \sin x, \cos x, \sin 2x, \cos 3x, \ldots$$

oder Exponentialfunktionen

$$1, e^x, e^{-x}, e^{2x}, e^{-2x}, \ldots$$

Ganz beliebig dürfen aber diese Funktionen nicht gewählt werden (s. u.). Die zu konstruierende Linearkombination ist

$$y_p(x) = \sum_{k=0}^{n} a_k g_k(x), \tag{4}$$

und die Koeffizienten a_k sind so zu bestimmen, daß die Interpolationsbedingung (3) erfüllt wird:

$$y(x_i) = \sum_{k=0}^{n} a_k g_k(x_i), \qquad i = 0(1)n. \tag{5}$$

Das ist nun ein lineares Gleichungssystem für die a_k. Seine Lösbarkeitsbedingung ist

$$H = \begin{vmatrix} g_0(x_0) & g_1(x_0) & \ldots & g_n(x_0) \\ g_0(x_1) & g_1(x_1) & \ldots & g_n(x_1) \\ \cdots & \cdots & \cdots & \cdots \\ g_0(x_n) & g_1(x_n) & \ldots & g_n(x_n) \end{vmatrix} \neq 0, \tag{6}$$

und das muß für *jede* Referenz aus dem Intervall (a, b) gelten. In der Referenz müssen also zumindest lauter verschiedene Werte stehen, $x_i \neq x_j$ für $i \neq j$. Diese ***Haarsche Bedingung*** (6) ist eine Bedingung für das gewählte System von Basisfunktionen und das Intervall. Es ist also, wie schon gesagt, beides nicht beliebig wählbar, sondern muß dieser Bedingung genügen. Auch hier ist die Angabe des Intervalls (a, b) wichtig.

Beispielsweise wäre das System

$$x, x^2, x^3$$

zur Interpolation im Intervall $(-1, 1)$ nicht geeignet. Dieses Intervall enthält die Null, und wenn die Null als zugehörig zu einer Referenz angesehen wird, ist $H = 0$, wodurch die Haarsche Bedingung verletzt wird. Es läßt sich dann mit diesem System von Basisfunktionen und einer Referenz mit Null, z. B. $(-1, 0, +1)$, nicht immer eine Interpolationsfunktion konstruieren. Ist etwa $y = 5$ (konstant), so nimmt das zu lösende Gleichungssystem (5) die Form

$$\begin{aligned} 5 &= -a_1 + a_2 - a_3, \\ 5 &= 0, \\ 5 &= a_1 + a_2 + a_3 \end{aligned}$$

an, und das ist widersprüchlich wegen $5 = 0$.

Für die Potenzfunktionen als Interpolationssystem liefert die Haarsche Bedingung eine spezielle Determinante, die *Vandermondesche Determinante* (vgl. MfL Bd. 9, 3.3., Aufg. 5). Mit $g_0(x) = 1, g_1(x) = x, \ldots, g_n(x) = x^n$ ($n + 1$ Funktionen) ist

$$V = \begin{vmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} & x_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} & x_n^n \end{vmatrix} \tag{7}$$

für die Referenz aus $n + 1$ Punkten $(x_0, x_1, \ldots, x_n)$. Auch hier sieht man deutlich, was schon allgemein galt, daß nämlich die Referenz lauter unterschiedliche Werte haben muß. Anderenfalls wären zwei Zeilen identisch und damit $V = 0$. Die Vandermondesche Determinante läßt sich durch geschickte Spaltenkombinationen verhältnismäßig leicht in eine Produktdarstellung umformen:

Man multipliziert der Reihe nach die vorletzte, drittletzte, ..., erste Spalte mit x_0 und subtrahiert die Produkte von der letzten, vorletzten, ..., zweiten Spalte. Dadurch entsteht

$$V = \begin{vmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & x_1 - x_0 & x_1^2 - x_0 x_1 & \ldots & x_1^n - x_0 x_1^{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n - x_0 & x_n^2 - x_0 x_n & \ldots & x_n^n - x_0 x_n^{n-1} \end{vmatrix}.$$

Nach dem Streichen der ersten Spalte und ersten Zeile (Entwicklungssatz für Determinanten) und Herausheben der gemeinsamen Faktoren $(x_1 - x_0), \ldots, (x_n - x_0)$ aus der neuen ersten bis zur letzten Zeile entsteht

$$V = \begin{vmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & \ldots & x_n^{n-1} \end{vmatrix} \cdot \prod_{k=1}^{n} (x_k - x_0),$$

also außer dem Produkt der herausgehobenen Faktoren wieder eine Vandermondesche Determinante mit einer um 1 verringerten Ordnung. Es fehlen die Funktion x^n und der Referenzpunkt x_0. Führt man dies nun weiter, so erhält man

$$V = \prod_{k=1}^{n} (x_k - x_0) \cdot \prod_{k=2}^{n} (x_k - x_1) \cdots \prod_{k=n-1}^{n} (x_k - x_{n-2}) \cdot \begin{vmatrix} 1 & x_{n-1} \\ 1 & x_n \end{vmatrix}.$$

Da nun

$$\begin{vmatrix} 1 & x_{n-1} \\ 1 & x_n \end{vmatrix} = x_n - x_{n-1}$$

ist und man die Produkte zusammenfassen kann, entsteht die bekannte Formel

$$V = \prod_{j=0}^{n-1} \prod_{k=j+1}^{n} (x_k - x_j) = \prod_{\substack{k=1 \\ j<k}}^{n} (x_k - x_j). \tag{8}$$

Es ist also $V \neq 0$, wenn kein Faktor Null wird, d. h., wenn in der Referenz $x_k \neq x_j$ für $k \neq j$ gilt. Es folgt aber auch, daß viele dichtliegende Werte in der Referenz zu einem kleinen V-Wert führen und damit das Gleichungssystem (5), dessen Koeffizientendeterminante V für den

Spezialfall der Potenzfunktionen ist, schlecht konditioniert ist. Das ist hier bereits ein erster Hinweis darauf, daß die Interpolation durchaus nicht immer besser werden muß, je mehr und je dichter die Referenz bestückt ist.

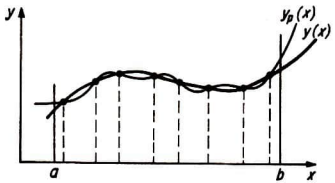


Abb. 2.9. Die im Intervall (a, b) zu interpolierenden Funktion $y(x)$ und die Interpolationsfunktion $y_p(x)$ erfüllen für die gegebene Referenz (hier acht Punkte) die Bedingung (5)

Abb. 2.9 zeigt noch einmal den Sachverhalt, daß die interpolierende Funktion an den Referenzstellen mit der interpolierten Funktion übereinstimmt. Sie wird durch diese Funktionswerte *gestützt*. Die Punkte $(x_i, y(x_i))$ oder einfacher notiert (x_i, y_i) heißen *Stützpunkte* und die Werte x_i aus der Referenz auch *Stützstellen*.

2.2.3. Interpolation nach Lagrange

Das Gleichungssystem (5) kann natürlich zur Bestimmung der Koeffizienten a_k in der Linearkombination des Interpolationspolynoms benutzt werden. Es gibt aber auch Konstruktionsansätze für dieses Polynom, welche das bei mehreren (vielen) Stützstellen doch aufwendige Lösen des Gleichungssystems vermeiden und auf der Grundforderung *interpolierende Funktion geht durch die Stützpunkte* beruhen. So hat der französische Mathematiker J. L. Lagrange (1736—1813) allgemeine Basispolynome konstruiert, die jeweils an allen Stützstellen bis auf eine den Wert 0 haben und an der Ausnahmestützstelle den Wert 1. Basisfunktionen nach Lagrange:

$$L_k(x) = \frac{\prod\limits_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i)}{\prod\limits_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)} = \frac{Z_k}{N_k} \quad \text{für} \quad k = 0(1)n. \tag{9}$$

Es sind dies also $n + 1$ Polynome vom Grad n, da der Faktor $x - x_k$ im Zähler Z_k fehlt (im Nenner N_k fehlt natürlich der Faktor $x_k - x_k = 0$). An allen Stützstellen $x = x_i$ mit $i \neq k$ tritt ein Faktor $x_i - x_i = 0$ auf, und es ist

$$L_k(x_i) = 0 \quad \text{für } i \neq k.$$

Wenn aber $x = x_k$ ist, haben Zähler und Nenner die gleiche Form, und es ist

$$L_k(x_k) = 1.$$

Abb. 2.10 zeigt zwei solche Basispolynome nach LAGRANGE. Mit diesen Eigenschaften der Basisfunktionen und für die zu interpolierende Funktion $y = y(x)$ mit

$$y_k = y(x_k) \quad \text{für} \quad k = 0(1)n$$

nimmt das Gleichungssystem (5) der Interpolationsbedingungen folgende Gestalt an:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Es ist also ein Diagonalsystem und liefert sofort die Lösung

$$a_k = y_k \quad \text{für} \quad k = 0(1)n.$$

Die gesuchten Koeffizienten der Systemfunktionen sind also einfach die y-Werte der Stützstellen.

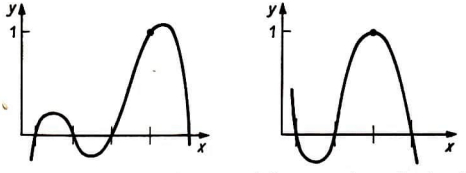


Abb. 2.10. Zwei Basispolynome nach LAGRANGE in qualitativer Darstellung: $L_3(x)$ für fünf Stützstellen und $L_2(x)$ für vier Stützstellen. Die „Ausnahmestelle" ist durch • markiert

Damit kann nun das Interpolationspolynom in der Form

$$y_p(x) = \sum_{k=0}^{n} y_k L_k(x) \tag{10}$$

angesetzt werden.

Das Interpolationspolynom $y_p(x)$ tritt hier nicht in der üblichen nach x-Potenzen geordneten Form auf. Es ist aber identisch mit dem in der allgemeinen Aufgabenstellung der Interpolation geforderten Polynom.

Beispiel

Im Beispiel wird eine Referenz mit äquidistanten Stützstellen verwendet. Das ist aber bei der Lagrangeinterpolation nicht erforderlich. Man kann die Stützstellen auch in beliebiger Reihenfolge numerieren. Natürlich wird man meist — so auch hier — ein Ordnungsprinzip dabei beachten. Hier wurden sie der Größe nach numeriert.

Intervall $(-1, 2)$; Referenz $(-1, 0, 1, 2) = (x_0, x_1, x_2, x_3)$

Das erste der zugehörigen Basispolynome nach LAGRANGE ist

$$L_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{x(x-1)(x-2)}{(-1)(-2)(-3)}$$

$$= -\frac{1}{6}x(x-1)(x-2), \tag{11}$$

und die drei anderen

$$L_1(x) = \frac{1}{2}(x+1)(x-1)(x-2),$$

$$L_2(x) = \frac{1}{2}(x+1)x(x-2),$$

$$L_3(x) = \frac{1}{6}(x+1)x(x-1)$$

werden ebenso gewonnen.

Es ist für dieses Beispiel übrigens

$$L_0(x) = L_3(1-x),$$

$$L_1(x) = L_2(1-x),$$

wovon man sich leicht überzeugt und was bei numerischer Auswertung ein Vorteil ist, weil nur noch L_0 und L_1 berechnet zu werden brauchen. In der Anwendung der Tafelverfeinerung (s. u.) wird dies ausgenutzt werden.

Rechentechnische Aufbereitung und Aufwand bei der Interpolation nach Lagrange

Bei der rechentechnischen Aufbereitung sind die Arbeitsanteile, welche bei jeder Interpolation wieder auftreten, vorbereitend ein für allemal zu erledigen, und der Arbeitsanteil, der speziell für jede Interpolation anfällt, ist möglichst gering zu halten. Als Vorbereitung kann man so alle Nenner N_i der Lagrange-Hilfspolynome berechnen und die Stützwerte y_i durch diese dividieren, $y_i' = y_i/N_i$. Man arbeitet nach folgendem Schema:

y_i	x_i	d_1x_i	d_2x_i	... d_nx_i	N_i	y_i'
y_0	x_0	x_0-x_1	x_0-x_2	x_0-x_n	N_0	y_0/N_0
y_1	x_1	x_1-x_2	x_1-x_3	x_1-x_0	N_1	y_1/N_1
y_2	x_2	x_2-x_3	x_2-x_4	x_2-x_1	N_2	y_2/N_2
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
y_n	x_n	x_n-x_0	x_n-x_1	x_n-x_{n-1}	N_n	y_n/N_n

(12)

Bei den Differenzen d_kx_i in den Spalten läuft der Minuend stets von x_0 bis x_n und der Subtrahend von x_k an aufwärts, wobei die Zählung wieder bei 0 beginnt, sobald

n erreicht ist, also modulo $n+1$ durchgeführt wird. Der Index k ist die Punktnummerndifferenz in den Spalten. Für die Nenner gilt

$$N_i = \prod_{k=1}^{n} d_k x_i .$$

Sie sind also als die Zeilenprodukte aller Differenzen herzustellen. Für y_i' ist die Rechenvorschrift bereits genannt. Damit ist die Vorbereitung abgeschlossen.

Bei der Auswertung der Lagrange-Interpolationsformel (9) für $x \neq x_i$ bedient man sich des folgenden Schemas:

y_i'	x_i	$x - x_i$	$Z_i = P/(x - x_i)$	$Z_i \cdot y_i'$
y_0'	x_0	$x - x_0$	Z_0	$Z_0 y_0'$
y_1'	x_1	$x - x_1$	Z_1	$Z_1 y_1'$
y_2'	x_2	$x - x_2$	Z_2	$Z_2 y_2'$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
y_n'	x_n	$x - x_n$	Z_n	$Z_n y_n'$
		$P = \prod_{i=0}^{n} (x - x_i)$		$S = \sum_{i=0}^{n} Z_i y_i'$

(13)

Es werden also die Zähler Z_i der Lagrange-Basispolynome für x zur Vermeidung allzu vieler Multiplikationen mit Hilfe des Spaltenproduktes

$$P = \prod_{i=0}^{n} (x - x_i)$$

gebildet, welches für jeden Zähler einen anderen Faktor $x - x_i$ zuviel enthält, und dieser wird wegdividiert. Die auszuwertende Summe

$$S = y_p(x)$$

entsteht dann als Skalarprodukt der Spalten Z_i und y_i'. Ein Rechenbeispiel wird weiter unten nachgeholt.

Für die Auszählung der benötigten Operationen gilt also ebenfalls, daß man zwischen einmal zu leistendem Vorbereitungsaufwand und dem Aufwand pro Interpolationswert zu unterscheiden hat. Als Vorbereitungsaufwand kann man bei der Lagrange-Interpolation das Berechnen der Nenner N_k in den Basispolynomen ansehen. Es erfordert

$N(N-2)$ Multiplikationen

und

$N(N-1)$ Subtraktionen,

da es N Stück der $L_k(x)$ gibt, denn es gibt $N = n+1$ Stützstellen $(x_0, \ldots, x_n)$. Für

die Berechnung eines Interpolationswertes benötigt man dann für die N Zähler Z_k wieder

$N(N-2)$ Multiplikationen

und

$N(N-1)$ Subtraktionen

oder bei geschickterer Auswertung (erst N Differenzen bilden, dann diese *alle* multiplizieren und schließlich jeweils eine Differenz wieder herausdividieren)

N Subtraktionen,

$N-1$ Multiplikationen,

N Divisionen.

Für die Auswertung der Summe können die Divisionen mit den vorbereiteten Nennern auch schon vorbereitend auf die y_k angewendet werden. Es entstehen die Werte $y'_k = y_k/N_k$, und es verbleiben somit nur noch

N Multiplikationen mit den y'_k

und

$N-1$ Additionen.

Für die Auswertung in Kleinstrechnern kann man sicher auf Beachtung der Unterschiede in der Operationszeit bei arithmetischen Operationen verzichten und nur deren Anzahl als Maß für den Aufwand nehmen.

Eigentlich sind auch Hilfsoperationen wie Speichern oder gar Notieren von Werten bedeutungsvoll. Für die Lagrange-Interpolation benötigt man bei N Stützstellen als Vorbereitung

$2N^2 - 2N$ arithmetische Operationen

und pro Interpolationspunkt (bei geschickter Arbeitsweise)

$5N - 2$ arithmetische Operationen.

Der Aufwand wächst also linear mit der Stützstellenanzahl. Den geringsten Arbeitsaufwand pro Iterationspunkt erhält man freilich, wenn das Interpolationspolynom nicht in der Lagrangeschen Gestalt, sondern geordnet nach x-Potenzen vorliegt. Dann kann das Horner-Schema benutzt werden, und pro Punkt hat man nur noch $2N - 1$ Operationen auszuführen. Dazu ist allerdings ein größerer Vorbereitungsaufwand nötig. Außer den y'_i (man gewinnt sie nach dem oben gegebenen Schema) benötigt man alle Koeffizienten a_i aus den Umordnungen

$$L_k(x) = \sum_{i=0}^{n} a_i x^i$$

und muß diese noch für alle k zusammenfassen. Es gelingt dies noch verhältnismäßig einfach nach den Formeln (14) und einem Schema, das man sich aus der Bauart

und der rekursiven Entstehung der Koeffizienten herleitet, wenn man zwei Lagrangepolynome ausführlich notiert:

für zwei Stützstellen

$$(x - x_1)\, y_0' + (x - x_0)\, y_1' = (y_0' + y_1')\, x - (x_0 y_1' + x_1 y_0'),$$

für drei Stützstellen

$$\begin{aligned}&(x - x_1)(x - x_2)\, y_0' + (x - x_0)(x - x_2)\, y_1' + (x - x_0)(x - x_1)\, y_2' \\ &= (y_0' + y_1' + y_2')\, x^2 - (x_1 y_0' + x_2 y_0' + x_0 y_1' + x_2 y_1' + x_0 y_2' - x_1 y_2')\, x \\ &\quad + (x_1 x_2 y_0' + x_0 x_2 y_1' + x_0 x_1 y_2').\end{aligned}$$

Das Ausfüllen des Schemas beginnt mit den beiden Schritten

i	a_i	b_i
$n+1$	0	1
n	0	0
$n-1$	0	0
.	.	.
.	.	.
.	.	.
0	0	0
	$-x_0$	y_0'
		$-x_0$

und

i	a_i	b_i
$n+1$	0	1
n	y_0'	$-x_0$
$n-1$	0	0
.	.	.
.	.	.
.	.	.
0	0	0
	$-x_1$	y_1'
		$-x_1$

Diese beiden Schritte haben nur rechenorganisatorische Bedeutung. Dabei sind die unten stehenden $-x_k$ und y_k' Multiplikatoren für die in den Spalten a und b aufzubauenden Polynomkoeffizienten a_i und b_i.

Man bildet ein neues Schema nach den Formeln:

$$\begin{aligned}&a_{n+1} := 0, \qquad b_{n+1} := 1, \\ &a_i := -a_{i+1} x_k + b_{i+1} y_k' + a_i \qquad \text{für} \quad i = n(-1)0 \\ &\text{(Multiplikator } y_k' \text{ für die Spalte } b\text{) und dann} \\ &b_i := -b_{i+1} x_k + b_i \qquad \text{für} \quad i = n(-1)0 \\ &\text{(Multiplikator } -x_k \text{ für die Spalte } b\text{),}\end{aligned} \tag{14}$$

also geht man von oben nach unten und kann natürlich beim Auftreten der Nullen abbrechen. Diese Formeln sind eng verwandt mit den Ausdrücken (in Spalte b_i sind sie es genau), nach denen man bei Kenntnis der Nullstellen die Polynomkoeffizienten herstellt, und zwar unter Ausnutzung der Vietaschen Beziehungen zwischen beiden. Das beschriebene Verfahren kann *Vieta-Lagrange-Kombination* oder kurz *VLK-Interpolation* genannt werden. Es entsteht für $k = 1$

i	a_i	b_i
$n+1$	0	1
n	$y_0' + y_1'$	$-x_1 - x_0$
$n-1$	$-x_0y_1' - x_1y_0'$	x_1x_0
$n-2$	0	0
.	.	.
.	.	.
.	.	.
0	0	0
	$-x_2$	y_2'
		$-x_2$

und in der Spalte a erkennt man die Koeffizienten für zwei Stützstellen. Unter dem Schema sind bereits die neuen Multiplikatoren vermerkt. Geht man nun wiederum wie eben erläutert vor, so entsteht für $k = 2$

i	a_i	b_i
$n+1$	0	1
n	$y_0' + y_1' + y_2'$	$-x_2 - x_1 - x_0$
$n-1$	$-y_0'x_2 - y_1'x_2 - x_1y_2' - x_0y_2' - x_0y_1' - x_1y_0'$	$x_1x_2 + x_0x_2 + x_1x_0$
$n-2$	$x_0x_2y_1' + x_1x_2y_2' + x_1x_0y_2'$	$-x_2x_1x_0$
$n-3$	0	0
.	.	.
.	.	.
.	.	.
0	0	0

und hier sind nun die Koeffizienten des Polynoms mit drei Stützstellen erkennbar. So fährt man weiter fort bis x_n und y_n', wobei das letzte Schema keine neuen b_i mehr zu enthalten braucht. Ein Beispiel wird unten angegeben. Beim Anwenden mit einem programmierbaren Rechner wird man nur zwei Vektoren a und b speichern und bezüglich i „vorwärts" rechnen, um durch neue Werte noch benötigte Werte nicht zu überspeichern: im k-ten Schritt also $i = (n - k)(1)n$.

Tafelverfeinerung

Eine Anwendung der Lagrange-Interpolation. Diese Anwendung ist möglicherweise das ursprüngliche Anliegen von LAGRANGE gewesen. Für kompliziert und aufwendig berechnete Funktionswerte sollen neun Zwischenwerte auf einfachere Weise und genau genug gewonnen werden. Dazu wird eine kubische Interpolation, also mit vier Stützstellen genommen. Das betrachtete Tafelintervall wird auf $(-1, 2)$ mit der Referenz $(-1, 0, 1, 2)$ abgebildet, so daß der mit Zwischenwerten aufzufüllende Teilbereich auf $(0, 1)$ fällt. Die Funktionswerte erleiden dabei keinerlei Veränderung. Die Intervalltransformation bringt aber den Vorteil, daß die Hilfsfunktionen $L_0(x)$ und $L_1(x)$ für die neun Werte $x = 0.1(0.1)0.9$ im voraus berechnet werden können.

Tab. 2.17. Funktionswerte zweier spezieller Hilfspolynome nach LAGRANGE, die zur Tafelverfeinerung verwendet werden

x	$L_0(x)$	$L_1(x)$	
0	0	1	1
0.1	−0.0285	0.9405	0.9
0.2	−0.048	0.864	0.8
0.3	−0.0595	0.7735	0.7
0.4	−0.064	0.672	0.6
0.5	−0.0625	0.5625	0.5
0.6	−0.056	0.448	0.4
0.7	−0.0455	0.3315	0.3
0.8	−0.032	0.216	0.2
0.9	−0.0165	0.1045	0.1
1	0	0	0
	$L_3(x)$	$L_2(x)$	x

Es werden zwar vier Hilfspolynome benötigt, aber es sind gerade die vier gemäß (11) im obigen Beispiel genannten, für welche die bereits genannten bequemen Beziehungen

$$L_0(x) = L_3(1 - x) \qquad \text{und} \qquad L_1(x) = L_2(1 - x)$$

gelten. Tab. 2.17 zeigt die errechneten Werte. Man schreibt leicht Programme oder Tastenfolgen für die Funktionen $L_0(x)$ und $L_1(x)$. Es ist

$$L_0(x) = x(x - 1)\,(x - 2)/(-6)$$

und

$$L_1(x) = (x + 1)\,(x - 1)\,(x - 2)/2 = (x^2 - 1)\,(x - 2)/2.$$

Ein BASIC-Programm zur Berechnung von $L_0(x)$ und $L_1(x)$ ist in Tab. 2.18 enthalten. Seine Arbeit ist an zwei Stellen beachtenswert. Die Laufanweisung in Zeile 10 enthält als obere Grenze „1 + halbe Schrittweite". Da der interne Mikroprozessor

Tab. 2.18. BASIC-Programm zur Berechnung der Lagrange-Hilfspolynome in Tab. 2.17

```
10 FOR x = 0 TO 1.05 STEP 0.1
20 LET L0 = -x * (x - 1) * (x - 2)/6
30 LET L1 = (x * x - 1) * (x - 2)/2
40 PRINT L0, L1
50 NEXT x
60 STOP
```

im Dualsystem arbeitet (8-Bit-Prozessor), werden Dezimalzahlen nicht immer exakt dargestellt. Die obere Grenze 1 wird deshalb schon nach dem zehnten Zyklus geringfügig überschritten, wodurch die Funktionswerte für $x = 1$ nicht mehr berechnet

werden würden. Bei der oberen Grenze 1.05 werden sie wie gewünscht geliefert, lauten aber 7.8E-11 bzw. −4.7E-10

statt glatt 0. Es haben also auch die anderen im Computer errechneten Funktionswerte auf der zehnten bzw. elften Stelle nach dem Dezimalpunkt geringe Abweichungen. Die in einem Computer realisierte interne Arithmetik beeinflußt die Ergebnisse. Das muß gegebenenfalls durch Zusatzüberlegungen beachtet werden. Hier erhält man: Die in Tab. 2.17 angegebenen Resultate sind exakt und nicht durch Rundungen verfälscht.

Nun soll konkret die lg-Funktion zwischen $x_1 = 2.14$ und $x_2 = 2.15$ in einer Tafel verfeinert werden. Dieser Bereich wurde bereits in der Einleitung zur Interpolation für lineare Interpolation verwendet. Zur Demonstration der erreichbaren Genauigkeit wird zehnstellig gerechnet. Der Tafelausschnitt ist

		x_0	x_1	x_2	x_3	
x	...	3	4	5	6	...
2.1	...	328379604	330413773	332438460	334453751	...

und zwischen x_1 und x_2 werden neun Werte eingeschoben. Diese erhält man durch ein Skalarprodukt aus den vier angegebenen Funktionswerten und vier Werten aus Tab. 2.17. Das Resultat verdeutlicht Tab. 2.19. Die verwendete kubische Interpolation liefert hier also drei bis vier Dezimalstellen mehr als die lineare, d. h., der Interpolationsfehler ist 1‰ bis 0.1‰ des Fehlers aus der linearen Interpolation.

Tab. 2.19. Tafelverfeinerung nach LAGRANGE für $y = \lg x$ im Intervall (2.14, 2.15). Zum Vergleich sind zehnstellig gerundete exakte Werte angegeben; sie zeigen, daß hier die kubische Interpolation zehnstellige Genauigkeit liefert, während die lineare Interpolation nur fünf bis sechs Stellen bringt

x	$y_p(x)$	$\lg x$	linear interpoliert
2.140	0.330413773		
2.141	0.330616667	0.330616667	0.330616242
2.142	0.330819466	0.330819467	0.330818711
2.143	0.331022171	0.331022171	0.331021179
2.144	0.331224781	0.331224781	0.331223648
2.145	0.331427297	0.331427297	0.331426117
2.146	0.331629718	0.331629718	0.331628585
2.147	0.331832044	0.331832044	0.331931054
2.148	0.332034277	0.332034277	0.332033523
2.149	0.332236416	0.332236416	0.332235991
2.150	0.332438460		

Beispiel

Die bei der Tafelverfeinerung demonstrierte Genauigkeit der (kubischen) Interpolation nach LAGRANGE darf nicht zum Glauben führen, daß dies immer so günstig abläuft. Bei der Approximation von Funktionen mit Hilfe der Interpolation kann

man böse Überraschungen erleben. Gegeben sei z. B. das Polynom fünften Grades

$$y = p_5(x) = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60.$$

Es soll im Bereich (0, 1) kubisch interpoliert werden unter Verwendung der Referenz (−1, 0, 1, 2). Mit dem Horner-Schema erhält man rasch die zugehörigen y_i-Werte. Das oben empfohlene Rechenschema nach (12) und (13) nimmt die Gestalt aus Tab. 2.20 an und zeigt Rechnung und Rechnungsgang. Der erhaltene Wert − 30.24 ist jedoch sehr ungenau, da der exakte Wert − 12.5428 ist. Über das Fehlerverhalten von Interpolationspolynomen ist später noch einiges zu sagen.

Tab. 2.20. Interpolation des Polynoms

$p_5(x) = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60$

an der Stelle $x = 0.8$ mit einem kubischen Lagrange-Polynom über der Referenz (−1, 0, 1, 2). Der exakte Funktionswert ist $p_5(0.8) = -12.5428$

i	y_i	x_i	d_1x_i	d_2x_i	d_3x_i	N_i	$y_i' = y_i/N_i$
0	27	−1	−1	−2	−3	−6	−4.5
1	−60	0	−1	−2	1	2	−30
2	33	1	−1	2	1	−2	−16.5
3	936	2	3	2	1	6	156

i	y_i'	x_i	$x - x_i$	$Z_i = P/(x - x_i)$	$Z_i \cdot y_i'$
0	−4.5	−1	1.8	0.192	−0.864
1	−30	0	0.8	0.432	−12.96
2	−16.5	1	−0.2	−1.728	28.512
3	156	2	−1.2	−0.288	−44.928
			$P = 0.3456$		$S = -30.24$

Da in diesem Beispiel die Referenz mit der aus dem Anwendungsfall „Tafelverfeinerung" übereinstimmt und der gewünschte Interpolationswert für $x = 0.8$ auch auf dem Verfeinerungsraster liegt, könnte die dort vorbereitete Tab. 2.17 verwendet werden. Es ergibt sich natürlich derselbe Wert,

$$\begin{aligned} -0.032 \cdot 27 \quad &= -0.864 \\ +0.216 \cdot (-60) \quad &-12.96 \\ +0.864 \cdot 33 \quad &+28.512 \\ -0.048 \cdot 936 \quad &-44.928 = -30.24, \end{aligned}$$

und es treten dieselben Zahlenwerte auf wie in der letzten Spalte des zweiten Teils der Tab. 2.20.

Tab. 2.21 zeigt das Umordnen nach x-Potenzen bei der VLK-Interpolation (14) und danach die Auswertung für $x = 0.8$ nach dem Horner-Schema.

Tab. 2.21. Umordnung der Lagrange-Interpolationspolynome nach x-Potenzen, wie im Text zur VLK-Interpolation beschrieben. Es entsteht

$y = 105x^3 + 90x^2 - 102x - 60.$

Die anschließende Auswertung nach dem Horner-Schema liefert natürlich wieder -30.24

i	a_i	b_i
4	0	1
3	0	0
2	0	0
1	0	0
0	0	0

$-x_0 = 1 \qquad y_0' = -4.5$
$\qquad\quad -x_0 = 1$

i	a_i	b_i
4	0	1
3	−4.5	1
2	0	0
1	0	0
0	0	0

$-x_1 = 0 \qquad y_1' = -30$
$\qquad\quad -x_1 = 0$

i	a_i	b_i
4	0	1
3	−34.5	1
2	−30	0
1	0	0
0	0	0

$x_2 = -1 \qquad y_2' = -16.5$
$\qquad\quad -x_2 = -1$

i	a_i	b_i
4	0	1
3	−51	0
2	−12	−1
1	30	0
0	0	0

$x_3 = -2 \qquad y_3' = 156$

i	a_i
4	0
3	105
2	90
1	−102
0	−60

	105	90	−102	−60
	0	84	139.2	29.76
0.8	105	174	37.2	−30.24

2.2.4. Interpolation nach Newton

Auf Newton (1643—1727), also noch vor Lagrange, geht ein anderer spezieller Ansatz des Interpolationspolynoms zurück, der ebenfalls das Auflösen des Gleichungssystems (5) vermeidet und außerdem den großen Vorteil bietet, daß eine Erweiterung der Referenz durch Hinzunahme einer (oder mehrerer) neuen Stütz-

stellen einfach durchführbar ist und keine neue Rechnung von Anfang an auslöst. Die vorgelegte Referenz sei wieder $(x_0, x_1, \ldots, x_n)$ in einem Intervall (a, b). Der Newton-Ansatz für das Interpolationspolynom ist mit den Basisfunktionen

$$P_i(x) = \prod_{k=0}^{i} (x - x_k), \tag{15}$$

$$y_p(x) = \sum_{i=0}^{n} s_{i0} \prod_{k=0}^{i-1} (x - x_k) = \sum_{i=0}^{n} s_{i0} P_{i-1}(x) \tag{16}$$

oder optisch einprägsamer ($P_{-1}(x) = 1$ gesetzt)

$$y_p(x) = s_{00} + s_{10}(x - x_0) + s_{20}(x - x_0)(x - x_1) + \cdots + s_{n0}(x - x_0) \cdots (x - x_{n-1}).$$

Es ist also ebenfalls ein Polynom n-ten Grades. Es sei ausdrücklich aufmerksam gemacht, daß die Stützstelle x_n hier nicht explizit auftritt. Sie ist lediglich (wie die anderen Stützstellen auch) bei der Berechnung der Koeffizientenwerte s_{i0} beteiligt. An dieser Konstruktion des Interpolationspolynoms erkennt man das leichte Erweitern um eine weitere Stützstelle x_{n+1}. Es gibt lediglich ein neues Glied

$$s_{n+1,0}(x - x_0) \cdots (x - x_{n-1})(x - x_n),$$

und es ist dabei gleichgültig, wo x_{n+1} in der geordneten Folge der Stützstellen liegt, denn auch bei der Newton-Interpolation brauchen die Stützstellen nicht äquidistant zu liegen und nicht geordnet numeriert zu werden.

Die Koeffizienten s_{i0} werden nun aus der Interpolationsbedingung (3) für die zu interpolierende Funktion $y = y(x)$

$$y_k = y(x_k) = y_p(x_k) \qquad \text{für} \qquad k = 0(1)n$$

gewonnen. Dabei kommt vorteilhaft zum Tragen, daß das Newton-Interpolationspolynom für $x = x_k$ jeweils nach dem Glied mit s_{k0} abbricht, da danach stets der Faktor $(x_k - x_k) = 0$ auftritt.

Das Gleichungssystem (5) der Interpolationsbedingungen nimmt für die Newton-Basisfunktionen die folgende Gestalt an:

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & x_1 - x_0 & 0 & \ldots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \ldots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \ldots & (x_n - x_0) \cdots (x_n - x_{n-1}) \end{pmatrix} \begin{pmatrix} s_{00} \\ s_{10} \\ s_{20} \\ \vdots \\ s_{n0} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{17}$$

Es hat also Dreiecksgestalt und ist deshalb leichter auflösbar. Man erhält für $k = 0$ (erste Interpolationsbedingung)

$$y_0 = y_p(x_0) = s_{00},$$

$$s_{00} = y_0,$$

für $k = 1$ (zweite Interpolationsbedingung)

$$y_1 = y_p(x_1) = s_{00} + s_{10}(x_1 - x_0) = y_0 + s_{10}(x_1 - x_0),$$

$$s_{10} = \frac{y_1 - y_0}{x_1 - x_0} \quad \text{eine Steigung (Sekantenanstieg),}$$

für $k = 2$ (dritte Interpolationsbedingung)

$$y_2 = y_p(x_2) = s_{00} + s_{10}(x_2 - x_0) + s_{20}(x_2 - x_0)(x_2 - x_1)$$

$$= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + s_{20}(x_2 - x_0)(x_2 - x_1),$$

und es ist die Auflösbarkeit nach s_{20} erkennbar. Da aber hier bereits die Übersichtlichkeit verloren zu gehen droht, verwendet man eine einprägsame Schreibweise für Steigungen, die man GAUSS verdankt:

$$\frac{y_1 - y_0}{x_1 - x_0} = [x_1 x_0] = s_{10}$$

oder allgemein

$$\frac{y_k - y_{k-1}}{x_k - x_{k-1}} = [x_k x_{k-1}] = s_{1,k-1},$$

und nennt dies *Steigungen erster Ordnung*. Formal kann man noch

$$s_{0k} = y_k = [x_k] \tag{18}$$

als *Steigung nullter Ordnung* bilden. Es ist dann

$$\frac{[x_k] - [x_{k-1}]}{x_k - x_{k-1}} = [x_k x_{k-1}]$$

die Steigung erster Ordnung als Differenzenquotient aus den Steigungen nullter Ordnung erhältlich.

Man prüft nun, allerdings schon mit etwas Mühe, nach, daß

$$s_{20} = \frac{[x_2 x_1] - [x_1 x_0]}{x_2 - x_0} = [x_2 x_1 x_0]$$

und damit eine *Steigung zweiter Ordnung* ist. Allgemein findet man

$$s_{i0} = [x_i x_{i-1} \dots x_0]$$

als *Steigung i-ter Ordnung* mit der rekursiven Berechnung dieser Steigungen

$$s_{ik} = \frac{s_{i-1,k+1} - s_{i-1,k}}{x_{i+k} - x_k} \tag{19}$$

sowie den Anfangswerten (18)

$$s_{0k} = y_k.$$

Praktisch berechnet man sie nach einem *Steigungsschema* und der Formel (19):

k	x_k	$i = 0$ y_k/s_{0k}	1 s_{1k}	2 s_{2k}	...	n s_{nk}
0	x_0	y_0	$[x_1x_0]$	$[x_2x_1x_0]$		$[x_nx_{n-1}\dots x_0]$
1	x_1	y_1	$[x_2x_1]$	$[x_3x_2x_1]$		—
2	x_2	y_2	$[x_3x_2]$	$[x_4x_3x_2]$		—
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.		.
$n-1$	x_{n-1}	y_{n-1}	$[x_nx_{n-1}]$	—		—
n	x_n	y_n	—	—		—

(20)

Es ist also

$$s_{ik} = [x_kx_{k-1}\dots x_{k-i}],$$

und die gewünschten Koeffizienten s_{i0} sind in der obersten Zeile des Schemas zu finden. Die Formel der Berechnungsvorschrift (19) kann verbal wie folgt beschrieben werden: Die neue Steigung s_{ik} in der Spalte i und Zeile k des Schemas erhält man aus der Differenz zweier Steigungen der vorhergehenden Spalte $i-1$, und zwar aus der nachfolgenden Zeile $k+1$ und derselben Zeile k, welche durch die x-Differenz aus dem x-Wert i Zeilen weiter voran und demjenigen aus derselben Zeile k dividiert wird.

Nimmt man einen weiteren Punkt x_{n+1} zur Referenz hinzu, d. h. einen weiteren Stützpunkt (x_{n+1}, y_{n+1}), so werden die Spalten x_k und y_k um einen Wert verlängert und dann im Schema eine „Diagonale" nach rechts oben dazugerechnet, bis man schließlich den neuen Koeffizienten $s_{n+1,0}$ am oberen Rand erhält. Alle anderen Koeffizienten bleiben unberührt, was gerade den Vorteil des Newton-Interpolationspolynoms ausmacht. Es ist dabei auch ohne Belang, ob die Punkte der Referenz äquidistant und geordnet vorliegen. Er erhält lediglich die letzte Nummer, d. h. den letzten Index.

Bei der *rechentechnischen Aufbereitung* braucht man nur die Vektoren der x_k, s_{i0} und y_k zu speichern, mehr nicht. Am Platz der letzteren wird man nach und nach die Diagonalelemente entstehen lassen und jeweils das letzte (s_{i0}) im Schritt i an den Vektor s_i abführen. In PASCAL ergäbe das den folgenden Programmausschnitt:

```
for i := 0 to n
do begin
   for k := i - 1 down to 0
   do y[k] := (y[k + 1] - y[k])/(x[k + i] - x[k]);
   s[i] := y[0]
   end
```

Diese Rechenvorschrift erzeugt in den Vektoren y und s_i der Reihe nach folgende Werte (für $n = 3$):

Anfang

k	$i=0$ y	$i=1$	$i=2$	$i=3$
0	y_0	$[x_1x_0]$	$[x_2x_1x_0]$	$[x_3x_2x_1x_0]$
1	y_1	y_1	$[x_2x_1]$	$[x_3x_2x_1]$
2	y_2	y_2	y_2	$[x_3x_2]$
3	y_3	y_3	y_3	y_3

(21a)

Schritte

i	$i=0$ s_i	$i=1$	$i=2$	$i=3$
0	y_0	y_0	y_0	y_0
1		$[x_1x_0]$	$[x_1x_0]$	$[x_1x_0]$
2			$[x_2x_1x_0]$	$[x_2x_1x_0]$
3				$[x_3x_2x_1x_0]$

(21b)

Der *Aufwand an Operationen* für diese Vorbereitung zur Interpolation nach NEWTON wächst quadratisch mit der Anzahl der Stützstellen $N = n + 1$. Für jede Einzelabarbeitung der Formel (19) sind zwei Subtraktionen und eine Division, also drei Operationen, nötig. Bei der Anwendung von Kleinstrechnern, bei denen die Operationszeit im wesentlichen durch die Tastenbedienung bestimmt wird, kann man auf Beachtung der unterschiedlichen Operationszeiten verzichten. Im Schritt i finden, wie man oben sieht, i Formelanwendungen statt, und es gibt $N - 1 = n$ Schritte. Somit werden

$$\frac{3N(N-1)}{2} \text{ Operationen}$$

benötigt.

Für die Auswertung des gewonnenen Interpolationspolynoms kann man ein dem Horner-Schema ähnliches Schema verwenden:

Newton-Horner-Kombination (NHK)

	x_{n-1}	x_{n-2}	$\cdots$	x_1	x_0
	s_{n0}	$s_{n-1,0}$		s_{10}	s_{00}
	$+$	$+$		$+$	$+$
x	0	$a_n(x - x_{n-1})$		$a_2(x - x_1)$	$a_1(x - x_0)$
	a_n	a_{n-1}		a_1	$a_0 = y_p(x)$

(22)

Die Polynomkoeffizienten s_{i0} werden wie im Horner-Schema mit dem höchsten voran aufgeschrieben, versetzt darüber die n Stützstellen x_{n-1} bis x_0. Wie beim Horner-Schema wird vertikal addiert. Aber vor der jeweiligen Multiplikation mit x ist dieser

Wert durch Subtraktion um x_i zu verringern. Für die Auswertung sind also bei jedem interpolierten Wert

$$3(N-1) \text{ Operationen}$$

nötig. Der Aufwand ist somit wieder linear bezüglich der Referenzlänge, aber es gibt den Faktor 3 gegenüber 2 bei der VLK-Interpolation.

Die Newton-Interpolationspolynome können mit wachsender Referenzlänge (Hinzunahme neuer Stützpunkte) auch *rekursiv nacheinander berechnet* werden. Man startet mit einem Polynom nullten Grades (konstant) bei nur einer Stützstelle x_0 und $y_0 = y(x_0)$

$$y_p^{(0)}(x) = y_0 = s_{00}. \tag{23}$$

(Es muß jetzt zur Unterscheidung der Polynome ein oberer Index mitgeführt werden.) Hat man bereits ein Polynom n-ten Grades mit $n+1$ Stützstellen $(x_0, \ldots, x_n)$

$$y_p^{(n)}(x) = \sum_{i=0}^{n} s_{i0} \prod_{k=0}^{i-1} (x - x_k) = \sum_{i=0}^{n} s_{i0} p_{i-1}(x)$$

bestimmt, welches die Bedingungen

$$y_p^{(n)}(x_k) = y_k \quad \text{für} \quad k = 0(1)n$$

erfüllt, so erhält man mit dem neuen Referenzwert x_{n+1} und der zusätzlichen Systemfunktion

$$p_n(x) = \prod_{k=0}^{n} (x - x_k) = (x - x_n)\, p_{n-1}(x)$$

das neue Interpolationspolynom vom Grad $n+1$ rekursiv,

$$y_p^{(n+1)}(x) = y_p^{(n)}(x) + \frac{y_{n+1} - y_p^{(n)}(x_{n+1})}{p_n(x_{n+1})}\, p_n(x), \tag{24}$$

und man erkennt deutlich, wie der neue Stützpunkt (x_{n+1}, y_{n+1}) hierbei wirksam wird. Jedoch ist diese Formel im Nachteil gegenüber der Berechnungsvorschrift im obigen Steigungsschema (20) bezüglich des Aufwandes oder nach dem kleinen PASCAL-Text für (21a, b). Das neue Interpolationspolynom $y_p^{(n+1)}(x)$ erfüllt alle Bedingungen

$$y_p^{(n+1)}(x_k) = y_k \quad \text{für} \quad k = 0(1)n+1,$$

denn es ist $p_n(x_k) = 0$ für $k = 0(1)n$, und das verbleibende $y_p^{(n)}(x)$ erfüllte die ersten $n+1$ Bedingungen. Für die letzte Bedingung erhält man leicht

$$y_p^{(n+1)}(x_{n+1}) = y_p^{(n)}(x_{n+1}) + \frac{y_{n+1} - y_p^{(n)}(x_{n+1})}{p_n(x_{n+1})}\, p_n(x_{n+1}) = y_{n+1}.$$

Beispiele

1. Für das Intervall $(-1, 2)$ seien für $n = 3$ die Referenz $(-1, 0, 1, 2)$ und die Funktionswerte $(0, 0, 1, 0)$ gegeben. Das Schema für die Berechnung der Koeffizienten des Newton-Interpolationspolynoms

$$y_p(x) = s_{00} + s_{10}(x+1) + s_{20}(x+1)\, x + s_{30}(x+1)\, x(x-1)$$

zeigt Tab. 2.22. Es ist also

$$y_p(x) = \frac{1}{2}\, x(x+1) - \frac{1}{2}\, x(x+1)\,(x-1),$$

Tab. 2.22. Berechnung der Koeffizienten s_{i0} (oberste Zeile) eines Newton-Interpolationspolynoms nach der Formel (19)

k	x_k	$i=0$ $y_k = s_{0k}$	1 s_{1k}	2 s_{2k}	3 s_{3k}
0	−1	0	0	0.5	−0.5
1	0	0	1	−1	–
2	1	1	−1	–	–
3	2	0	–	–	–

und das stimmt überein mit der Lagrange-Basisfunktion $L_2(x)$ bei vier Stützstellen. Durch Ausklammern erhält man

$$y_p(x) = \frac{1}{2}\, x(x+1)\,(1-(x-1))$$

$$= -\frac{1}{2}\, x(x+1)\,(x-2) = L_2(x)$$

(vgl. (11)).

Tab. 2.23. Kubische Interpolation des Polynoms

$y = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60$

bei $x = 0.8$ und mit der Referenz $(-1, 0, 1, 2)$ nach NEWTON

k	x_k	$i=0$ $y_k = s_{0k}$	1 s_{1k}	2 s_{2k}	3 s_{3k}
0	−1	27	−87	90	105
1	0	−60	93	405	
2	1	33	903	–	
3	2	936	–	–	

x_i		1	0	−1	
s_{i0}	105		90	−87	27
	0		−21	55.2	−57.24
$x = 0.8$	105		69	−31.8	**−30.24**

2. Das Polynom

$$y = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60$$

soll wie schon bei der Lagrange-Interpolation an der Stelle $x = 0.8$ für die Referenz $(-1, 0, 1, 2)$ kubisch interpoliert werden. Die y-Werte sind $(27, -60, 33, 936)$. Tab. 2.23 zeigt die Berechnung der Koeffizienten und die Auswertung nach der angegebenen Newton-Horner-Kombination (22). Natürlich ergibt sich derselbe interpolierte Wert −30.24, den man auch bei der Lagrange-Interpolation erhielt (vgl.

Tab. 2.20). Das entstehende Interpolationspolynom nach NEWTON lautet

$$y_p(x) = 27 - 87(x+1) + 90x(x+1) + 105x(x+1)(x-1).$$

Die Interpolationsrechnung nach NEWTON wird besonders einfach, wenn *äquidistante Stützstellen* mit $\Delta x = h$ der Größe nach geordnet in der Referenz vorliegen. Das Schema der Steigungen oder dividierten Differenzen wird dann zum einfachen *Differenzenschema*, und die Polynomkoeffizienten werden gemäß

$$s_{k0} = \frac{1}{k!}\frac{\Delta^k y_0}{h^k}$$

berechnet. Sie ähneln in der Bauart sehr den Taylor-Koeffizienten

$$\frac{1}{k!}\frac{d^k y}{dx^k}.$$

Das Differenzenschema ist

k	y_k	Δy_k	$\Delta^2 y_k$	$\cdots$	$\Delta^n y_k$
0	y_0	Δy_0	$\Delta^2 y_0$		$\Delta^n y_0$
1	y_1	Δy_1	$\Delta^2 y_1$		—
2	y_2	Δy_2	$\Delta^2 y_2$		—
.	.	.	.		.
.	.	.	.		.
.	.	.	.		.
$n-1$	y_{n-1}	Δy_{n-1}	—		—
n	y_n	—	—		—

(25)

und es ist

$$\Delta^i y_k = \Delta^{i-1} y_{k+1} - \Delta^{i-1} y_k \quad \text{für} \quad k = 0(1)n-i \quad \text{und} \quad i = 1(1)n; \tag{26}$$

außerdem gilt

$$\Delta^0 y_k = y_k. \tag{27}$$

In Tab. 2.24 wird das Beispiel 2 der Interpolation

Tab. 2.24. Differenzenschema für das Polynom
$y = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60$
auf der Referenz $(-1, 0, 1, 2)$ mit äquidistanten Werten

k	y_k	Δy_k	$\Delta^2 y_k$	$\Delta^3 y_k$
0	27	−87	180	630
1	−60	93	810	—
2	33	903	—	—
3	936	—	—	—

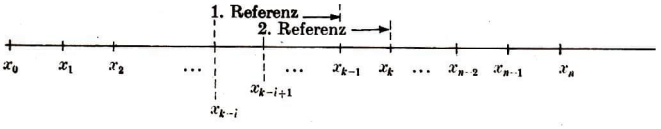
nach NEWTON unter Beachtung der vorliegenden äquidistanten Stützstellen mit $h = 1$ noch einmal vorgerechnet. Mit den Werten (27, −87, 180, 630) ergeben sich die Polynomkoeffizienten $(s_{00}, s_{10}, s_{20}, s_{30}) = (27, -87, 90, 105)$, wie sie bereits in Tab. 2.23 direkt erhalten werden konnten.

2.2.5. Aitken-Neville-Interpolation

In ähnlicher und verwandter Weise zur rekursiven Konstruktion (23) und (24) immer höher-gradiger Newton-Interpolationspolynome hat AITKEN (um 1925) ein Verfahren entwickelt, das auf folgender Idee beruht. Es sind zwei sich überschneidende Referenzen mit je i Stützstellen gegeben, und dazu hat man bereits für die vorgelegte zu interpolierende Funktion $y = y(x)$ auch zwei Interpolationspolynome vom Grad $i - 1$:

$$y_{k-1}^{(i-1)}(x) \quad \text{und} \quad y_k^{(i-1)}(x).$$

Zur Unterscheidung beider dient der untere Index. Dieser gibt die Nummer der Stützstellen an, an der die jeweilige Referenz für das Interpolationspolynom endet. Die zweite Referenz geht also aus der ersten durch Weiterrücken um eine Stelle hervor. Die Überschneidung betrifft $i - 1$ Stellen:



Natürlich sind die Interpolationsbedingungen

$$y_{k-1}^{(i-1)}(x_j) = y_j \quad \text{für} \quad j = k - i(1)k - 1,$$
$$y_k^{(i-1)}(x_j) = y_j \quad \text{für} \quad j = k - i + 1(1)k$$

erfüllt. Dann gewinnt man nach AITKEN ein neues Interpolationspolynom i-ten Grades durch $i + 1$ Punkte:

$$y_k^{(i)}(x) = \frac{(x - x_{k-1})\, y_{k-1}^{(i-1)}(x) - (x - x_k)\, y_k^{(i-1)}(x)}{x_k - x_{k-1}}, \tag{28}$$

und die Interpolationsbedingungen

$$y_k^{(i)}(x_j) = y_j \quad \text{für} \quad j = k - i(1)k$$

sind dann erfüllt, wie man unter Verwendung der erfüllten Bedingungen für die Vorgängerpolynome überprüft. Diese rekursive Konstruktionsvorschrift liefert Lagrange-Interpolationspolynome, wenn man nach den y_j ordnet. Man erkennt unmittelbar die Zählerbestandteile $x - x_k$ und die Nennerbestandteile $x_k - x_{k-i}$ der Lagrange-Basisfunktionen $L_k(x)$. In Abb. 2.11 ist das Überschneiden der Teilreferenzen besonders deutlich im Fall der quadratischen Parabeln zu sehen. Die bei der Inter-

polation an der Stelle $x = \bar{x}$ auftretenden Rechenwerte sind durch $\times$ markiert; vgl. dazu Tab. 2.25.

Für den Anfang der Rekursion (28) setzt man

$$y_k^{(0)}(x) = y_k, \tag{29}$$

d. h. konstant als y-Werte in der k-ten Stützstelle, an. Die Ausnutzung der Aitken-Interpolations-Rekursion durch NEVILLE besteht nun darin, daß man für einen vorgegebenen x-Wert $x = \bar{x}$ den Wert $y_n^{(n)}(\bar{x})$ des Interpolationspolynoms berechnet, ohne die einzelnen Rekursionspolynome (und auch das letzte nicht) zu kennen. Es werden also keinerlei Koeffizienten als Lösung des Systems (5) berechnet. In der Formel (28) werden vier Subtraktionen, zwei Multiplikationen und eine Division, insgesamt also sieben arithmetische Operationen, benötigt. Das Rechenschema hat die folgende Gestalt:

k	x_k	$i=0$	1	2	...	n
		y_k	$y_k^{(1)}$	$y_k^{(2)}$	...	$y_k^{(n)}$
0	x_0	y_0	—	—		—
1	x_1	y_1	$y_1^{(1)}$	—		—
2	x_2	y_2	$y_2^{(1)}$	$y_2^{(2)}$		—
.	.	.	.	.		.
.	.	.	.	.		.
.	.	.	.	.		.
n	x_n	y_n	$y_n^{(1)}$	$y_n^{(2)}$	...	$y_n^{(n)}$

(30)

Die Formel (28) ist demnach $n(n+1)/2$-mal anzuwenden, so daß sich bei $N = n + 1$ Stützstellen für jeden interpolierten Wert ein Aufwand von

$$\frac{7}{2}\, n(n+1) = \frac{7}{2}\, N(N-1) \text{ Operationen}$$

ergibt. Man kann den Aufwand drücken, wenn man allgemein die Nenner $x_k - x_{k-i}$ vorbereitend berechnet und für jede Einzelinterpolation die Zählerbestandteile $\bar{x} - x_k$. Aber er bleibt quadratisch mit N. Das zeigt die Einsatzgrenzen der AN-Interpolation. Hat man nur einen einzigen Wert zu interpolieren, so kann man sie getrost anwenden. Aber bei jedem weiteren Wert empfiehlt es sich, eine Vorbereitungsrechnung durchzuführen und dann das erhaltene Interpolationspolynom mit nur linearem Aufwand auszuwerten. Dabei ist die VLK-Interpolation die Variante mit dem geringsten Aufwand.

Tab. 2.25 zeigt den Rechengang zur Interpolation der Funktion $y = \lg x$ an der Stelle $\bar{x} = 2.147$ mit $n = 3$, also mit vier Stützstellen. Es ergibt sich der Wert

$$0.331\,832\,045,$$

der mit dem interpolierten Wert nach der Lagrange-Interpolation zur Tafelverfeinerung aus Tab. 2.19 gut übereinstimmt. Es handelt sich ja auch um das gleiche Interpolationspolynom, nur daß es bei der AN-Interpolation nicht explizit erscheint.

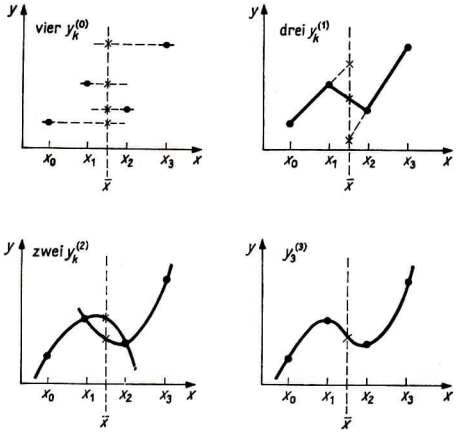


Abb. 2.11. Zur Aitken-Neville-Interpolation sind die abschnittsweisen Interpolationspolynome für eine Referenz mit vier Punkten schematisch dargestellt. Es endet mit einer kubischen Parabel

Tab. 2.25. Aitken-Neville-Interpolation für die Funktion $y = \lg x$ bei $\bar{x} = 2.147$ auf der Referenz (2.13, 2.14, 2.15, 2.16). Bei äquidistanten Stützstellen verringert sich der Aufwand etwas, da $x_k - x_{k-i}$ für jeweils eine Spalte im Rechenschema konstant ist

k	x_k	y_k, $i = 0$	$y_k^{(1)}$, $i = 1$	$y_k^{(2)}$, $i = 2$	$y_k^{(3)}$, $i = 3$
0	2.13	0.328379604			
1	2.14	0.330413773	0.331837692		
2	2.15	0.332438460	0.331831054	0.331832050	
3	2.16	0.334453751	0.331833863	0.331832041	0.331832045
		$x_k - x_{k-i}$	0.01	0.02	0.03

Bei der Berechnung mit dem programmierbaren Tischrechner oder Taschenrechner braucht man nur einen Vektor y mit $N = n + 1$ Komponenten und nicht etwa eine Matrix, die dem gesamten Rechenschema entspricht. Der Vektor wird anfangs mit den $y_k^{(0)} = y_k$-Werten gefüllt. Dann rechnet man nach folgendem PASCAL-Programm-Ausschnitt ($z = \bar{x}$):

```
for k: = 1 to n
do for i := 1 to n − k + 1
   do y[i − 1] := ((z − x[i − 1] − (z − x[k + i − 1]) ✳ y[i])/(x[k + i − 1]
                  − x[i − 1])
```

Dadurch entstehen im Vektor y der Reihe nach die Spaltenwerte des Rechenschemas, und zwar jeweils von $y[0]$ beginnend, so daß das Resultat am Ende aus $y[0]$ abgefragt werden kann. Nicht weiter benutzte Werte werden am Anfang des Vektors y überspeichert und bleiben am Ende gewissermaßen als Rechenmüll zurück. Die ursprünglichen y-Werte gehen also verloren. Sollten entgegen der Empfehlung nach Formel (28) und Schema (30) mehrere Interpolationen durchgeführt werden, so muß man dies beachten.

2.2.6. Interpolationsfehler

Das Beispiel der Interpolation eines Polynoms fünften Grades durch ein Polynom dritten Grades (vgl. Tab. 2.20, 21, 23) machte schon deutlich, daß es beim Interpolieren auch größere Fehler geben kann. Der echte Funktionswert war -12.5428, und der interpolierte Wert ergab -30.24. Wie kann man den Interpolationsfehler abschätzen, damit man eine Aussage über die Güte der Interpolation erhält?

An den Stützstellen x_k ist der Fehler oder das Restglied

$$R(x) = y(x) - y_p(x)$$

nach Konstruktion Null. Man kann zum Polynom $y_p(x)$ ohne Verletzung der Interpolationsbedingungen ein Polynom $(n+1)$-ten Grades addieren, welches selbst an allen Stützstellen den Wert Null annimmt. Das Restglied kann man also in der Form

$$R(x) = S \prod_{k=0}^{n} (x - x_k)$$

ansetzen. In der Gleichung

$$y(x) = y_p(x) + R(x)$$

zwischen interpolierter und interpolierender Funktion sowie dem Restglied interessiert nun der Faktor S, der natürlich auch variabel, d. h. $S = S(x)$, sein kann. Der Bestandteil

$$\prod_{k=0}^{n} (x - x_k)$$

im Restglied legt wegen seiner Bauweise den Gedanken nahe, das Newton-Interpolations-Rechenschema, Steigungsschema oder Differenzenschema (Beispiel Tab. 2.22) zu verwenden, und zwar mit beliebigem $\bar{x}$ als (letztem) Zusatzwert in der x_k-Spalte. Man erhält dann rechts oben eine zusätzliche Steigung oder einen Koeffizienten $[x_0 x_1 \dots x_n \bar{x}] = S_{n+1,0}$ bzw. im Differenzenschema einen Zusatzwert $\Delta^{n+1}\bar{y}$. Da dieses $\bar{x}$ oder $y(\bar{x})$ beliebig ist, kann man diese Rechnung natürlich nicht real vollziehen. Man weiß aber nun, daß $R(x)$ von der Form

$$R(x) = [x_0 x_1 \dots x_n \bar{x}] \prod_{k=0}^{n} (x - x_k) \tag{31}$$

oder

$$R(x) = \frac{1}{(n+1)!} \frac{\Delta^{n+1}\bar{y}}{h^{n+1}} \prod_{k=0}^{n} (x - x_k) \tag{32}$$

ist. Nach einer Folgerung aus dem Mittelwertsatz der Differentialrechnung (Folgerung aus dem Satz von ROLLE) gibt es immer einen solchen Wert $\bar{x}$ im Innern des Intervalls, der im entsprechenden Taylor-Entwicklungsglied mit der $(n+1)$-ten Ableitung

$$\frac{d^{n+1}y(\bar{x})}{dx^{n+1}}$$

für die Kompensation des Fehlers sorgt. Eine Abschätzung für den Interpolationsfehler erhält man demnach durch Abschätzung der $(n+1)$-ten Ableitung, die natürlich dann auch existieren muß, der zu interpolierenden Funktion und des Produktbestandteiles des Restgliedes.

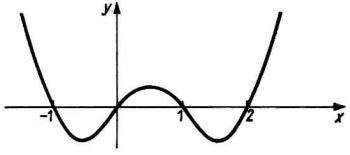


Abb. 2.12. Der Produktanteil $\prod_{k=0}^{n}(x - x_k)$ des Restgliedes $R(x)$ für die Interpolation durch Polynome hat in der „Mitte" der Referenz den kleinsten Extremwert. Die Skizze zeigt qualitativ den Produktanteil für die Referenz $(-1, 0, +1, +2)$

Eine nähere Untersuchung zeigt (vgl. Abb. 2.12), daß der Produktbestandteil „in der Mitte" das kleinste Maximum hat. Das war auch der Grund für die gewählte Lage des Verfeinerungsintervalls im Anwendungsbeispiel der Lagrange-Interpolation (vgl. Tab. 2.19).

Schätzt man nun z. B. das zu interpolierende Polynom (vgl. Tab. 2.20) bezüglich seiner $(n+1)$-ten Ableitung (dort also der vierten Ableitung) im Intervall $(-1, 2)$ ab, so erhält man

$$p_5(x) = 4x^5 + 40x^4 + 5x^3 + 50x^2 - 6x - 60,$$
$$p_5^{(4)}(x) = 480x + 960 = 480(x + 2).$$

Also hat das Restglied die Gestalt (es muß durch 4! dividiert werden)

$$R(x) = 20(\bar{x} + 2)(x + 1)x(x - 1)(x - 2),$$

und für die Interpolationsstelle $x = 0.8$ ist dies

$$R(0.8) = 6.912(\bar{x} + 2).$$

Der hier bekannte Interpolationsfehler

$$-12.5428 + 30.24 = R(0.8)$$
$$= 6.912(\bar{x} + 2)$$

erlaubt die Berechnung der Stelle $\bar{x}$ (auf vier Stellen gerundet):

$$\bar{x} = 0.5604.$$

Anderenfalls muß man für $\bar{x}$ den ungünstigsten Fall im Intervall wählen, hier wäre er $\bar{x} = 2$, und man erhält dann den schlimmstenfalls zu erwartenden Fehler 27.648.

Der größtmögliche Interpolationsfehler für $y = \lg x$ auf der Referenz (2.13, 2.14, 2.15, 2.16) ist demnach

$$|R(x)| \leqq \frac{|(x - 2.13)\,(x - 2.14)\,(x - 2.15)\,(x - 2.16)|}{4 \cdot \bar{x}^4 \cdot \ln 10},$$

und die rechte Seite wird im Zähler im Interpolationsintervall (2.14, 2.15) für $x = 2.145$ am größten, der Nenner wird für $\bar{x} = 2.13$ am kleinsten. Das liefert

$$|R(x)| \leqq 2.967 \cdot 10^{-11},$$

also einen Wert, der in der Rechnung von Tab. 2.19 noch unter den Rundungsfehlern bei zehnstelligen Werten liegt. Es ist also zu erwarten, daß dieses Beispiel der Tafelverfeinerung auch bei 11- bis 12stelliger Arbeit noch exakte Werte liefern würde.

2.2.7. Bemerkungen zu weiteren Interpolationsverfahren

Das verwendete Differenzenschema (25) bei der Newton-Interpolation nennt man genauer Schema der *absteigenden Differenzen*. Es zeichnet sich dadurch aus, daß die obersten Spaltenwerte alle den Index $k = 0$ tragen. Man kann die Differenzenformel auch so mit Indizes versehen, d. h. eine andere Numerierung der Werte einführen, daß die untersten Spaltenwerte alle denselben Index $k = n$ tragen. Es ist dann ein Schema der *aufsteigenden Differenzen* (MfL Bd. 9, *hintere und vordere Differenzen*, und [29]). Die im Schema bei konkreter Rechnung einzutragenden Zahlenwerte sind dabei identisch! Schließlich kann man das Schema so ordnen, daß die Differenzen „auf Lücke" stehen, und halbzahlige Indexwerte dafür einführen. Dann spricht man von *zentralen Differenzen* (MfL Bd. 9, 4.2.5., und [29]). Auch dann sind die Zahlenwerte natürlich wieder dieselben wie in den anderen Schemata. Weitere spezielle Interpolationsformeln ordnen die Stützstellen um, transformieren sie auf ganzzahlige Werte u. ä. Mit diesen speziellen Formeln sind die eingangs erwähnten Namen Gregory, Gauss, Laplace, Everett, Stirling und Bessel verbunden (MfL Bd. 9, 4.2.5., und [29]).

Liegt eine größere Anzahl N von Stützstellen vor, so werden die Berechnungen (der Aufwand der Vorbereitung wächst mit N^2) doch recht umfangreich. Man hat deshalb einer stückweisen oder intervallweisen Interpolation mit Polynomen niedrigeren Grades den Vorzug gegeben. Dabei geht man ebenfalls aus Aufwandsgründen nur selten über den Grad 3, also kubisch, hinaus. Man kann Interpolationsstücke dabei mit Knicken (stetig) aneinander stoßen lassen oder glatt (stetige erste Ableitung) mit gemeinsamer Tangente in den Stoßstellen.

Bezeichnet man das Interpolationsstück mit Intervall (x_k, x_{k+1}) mit $S_k(x)$, so gibt man für glatten Anschluß nach Hermite

$$S_k'(x_{k+1}) = S_{k+1}'(x_{k+1}) = y'(x_{k+1}) \tag{33}$$

mit den Ableitungswerten der zu interpolierenden Funktion $y(x)$ in den Stützstellen an den Rändern der Teilintervalle vor. Bei den sogenannten *Splines* verlangt man lediglich

$$S'_k(x_{k+1}) = S'_{k+1}(x_{k+1}), \tag{34}$$

aber mit unbekannten Werten an diesen Stellen. Das Erfüllen der Bedingung (34) gelingt schon bei Interpolationspolynomen vom Grad 2. Praktisch haben Splines vom Grad 3 größere Bedeutung erhalten. (Zur Hermite-Interpolation vgl. [29] Bd. 2, 5.3., und [32], 5.4.2.; zu Splines vgl. MfL Bd. 9, 4.2.4., und [32], 5.4.3., sowie [3].)

Die Forderung nach stetiger erster Ableitung kann sowohl bei der Hermite-Interpolation als auch bei den Splines auf höhere Ableitungen erweitert werden. Praktisch beschränkt man sich oft auf stetige erste Ableitung.

Man könnte vermuten, daß eine Interpolation für jede einigermaßen gutartige Funktion immer besser wird, je mehr Stützstellen vorliegen und je dichter sie das Interpolationsintervall belegen. Das gilt in der Tat schon für stetige Funktionen, d. h., sie sind bereits gutartig genug. Aber man kann die Referenzen dabei nicht beliebig auffüllen. Es gibt lediglich immer eine Verfeinerungsmöglichkeit der Referenzen, so daß der Interpolationsfehler im ganzen Intervall gegen Null strebt ([49] S. 374; [32], 5.3.). Andererseits kann man für jede Verfeinerungsvorschrift der Referenzen immer eine sogar stetige Funktion finden, für die der Interpolationsfehler nicht überall verschwindet ([49] S. 372; [29] Bd. 2, Abschn. 4, Satz von Faber). Beispielsweise hat Bernstein ([49] S. 375; [32], 5.3.) für die einfache Verfeinerungsvorschrift immer enger liegender äquidistanter Stützstellen, d. h. $\Delta x = \dfrac{2}{N}$, im Intervall $(-1, 1)$ und den Potenzfunktionen als Basisfunktionen der Interpolation als eine solche Funktion

$$y = |x|$$

nachgewiesen. Für sie gilt sogar, daß der Fehler (außer bei $x = -1$, $x = 0$ und $x = 1$) an keiner Stelle x im Intervall gegen Null geht.

2.3. Numerische Integration

In der Schule (Abiturstufe Kl. 11 und 12) werden Anfangsgründe und gewisse Grundkenntnisse der Integralrechnung vermittelt. Es wird dabei einerseits von der Umkehroperation zum Differenzieren und andererseits (ohne es direkt zu nennen) vom Riemannschen Integralbegriff gesprochen (MfL Bd. 4).

Das erstere führt auf Operationen mit Funktionen als Argumenten und gewissen anderen Funktionen als Resultaten:

$$\frac{d}{dx}\, g(x) = f(x), \qquad \int f(x)\, dx = g(x). \tag{1}$$

Dabei sind die Operatoren des Differenzierens und Integrierens *einstellig*, d. h., sie haben nur ein Argument und werden recht umständlich geschrieben, was in der Geschichte oder Ent-

wicklung der Mathematik seine historischen Ursachen hat. (Die Integralrechnung geht auf G. W. Leibniz (1680) und etwas später auf Newton zurück. Leibniz hat auch die noch heute üblichen Rechensymbole entwickelt.) Von größerer Bedeutung ist aber, daß diese Operatoren mit Funktionen arbeiten. Das ist für die gesamte mathematische Schülererfahrung bis zum Kennenlernen dieser Operationen etwas vollständig Neues.

Es wird in der Schule vom *bestimmten* und vom *unbestimmten* Integral gesprochen. Dabei ist das bestimmte durch eine Flächenvorstellung (Abb. 2.13) festgelegt, und es läßt sich aus dem Riemannschen Integralbegriff herleiten. Das unbestimmte

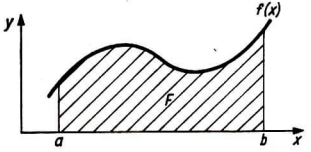


Abb. 2.13. Das bestimmte Integral einer Funktion $f(x)$ in den Grenzen von a bis b ist die schraffierte Fläche. Flächenteile unterhalb der x-Achse würden negativ gewertet werden

Integral dagegen ist die bereits erwähnte gewisse Funktion. Das bestimmte Integral wird durch Angabe der Integrationsgrenzen kenntlich gemacht, und es haben sich dazu ganz bestimmte Schreibweisen fest eingebürgert:

$$F = \int_a^b f(x)\,dx = g(x)\,|_a^b = g(b) - g(a). \tag{2}$$

Mit solchen Grenzen kann das unbestimmte Integral als *Funktion der oberen Grenze* geschrieben werden, wobei man der *Integrationsvariablen* zur besseren Unterscheidung eine andere Bezeichnung geben sollte:

$$\int_a^x f(\xi)\,d\xi = g(\xi)\,|_a^x = g(x) - g(a). \tag{3}$$

Die willkürlich eingesetzte untere Grenze a führt zu einem willkürlichen Wert $c = -g(a)$, so daß

$$\int f(x)\,dx = g(x) + c \tag{4}$$

geschrieben wird und die *Integrationskonstante* eingeführt ist (beim Differenzieren liefert sie den Beitrag Null). Das c kann also sogar außerhalb des Wertebereichs der Funktion $g(x)$ liegen.

In der Schule werden dann sowohl eine Reihe von Integrationsformeln, z. B.

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + c \quad (n \neq -1),$$

$$\int \frac{1}{x}\,dx = \ln x + c,$$

$$\int \cos x\,dx = \sin x + c,$$

$$\int e^x\,dx = e^x + c,$$

als auch Rechenregeln des Integrierens, z. B.

$$\int cf(x)\,dx = c\int f(x)\,dx \qquad \text{(konstanter Faktor)},$$

$$\int (u(x)+v(x))\,dx = \int u(x)\,dx + \int v(x)\,dx \qquad \text{(Integral einer Summe)},$$

$$\int u(x)\,v'(x)\,dx = u(x)\,v(x) - \int u'(x)\,v(x)\,dx \qquad \text{(partielle Integration als Ersatz für eine fehlende Regel zum Integral eines Produktes)}$$

vermittelt und damit zahlreiche zusammengesetzte Funktionen erfolgreich integriert. Im Grundkursabschnitt über das Integrieren (MfL Bd. 4) wurde für die große, aber nicht umfassende Klasse der rationalen Funktionen die Methode der *Partialbruchzerlegung* vermittelt. Damit ist eine größere Klasse von Funktionen geschlossen oder formelmäßig integrierbar. Jedoch ist dies nicht der Fall für alle nachweislich *integrierbaren Funktionen*, und die Partialbruchzerlegung (nicht zu verwechseln mit der partiellen Integration; s. o.) ist i. a. auch schon äußerst aufwendig und umständlich. Dies gilt beim Auftreten mehrfacher Nullstellen bei der Nennerfunktion der rationalen zu integrierenden Funktion, denn dann kommen Rekursionsformeln zum Einsatz, und es gilt besonders beim Auftreten mehrfacher konjugiert komplexer Wurzeln.

Der Lehrer muß also eine Antwort wissen auf die zu erwartende Schülerfrage: „Wie berechnet man die Integrale von integrierbaren Funktionen, bei denen aber alle bekannten Integrationsverfahren versagen?"

Die gleiche Frage entsteht natürlich viel zwingender bei Problemen aus der Praxis, denn dann wird das Integral zur Beantwortung oder Lösung einer konkreten Aufgabe benötigt. Solche Integrale gewinnt man unter Ausnutzung der Riemannschen Idee auf numerischem Weg, genähert, aber stets mit der erforderlichen Genauigkeit. Außerdem können Funktionen, die lediglich tabellarisch gegeben sind, nur durch numerische Verfahren integriert werden.

2.3.1. Der Riemannsche Gedanke und die Newton-Cotes-Formeln

Stufenformel

Die in Abb. 2.13 gezeigte schraffierte Fläche läßt sich näherungsweise durch Anlage von „Stufen" oder Rechtecken berechnen (Abb. 2.14). Es wird zur Zählung der Stufen und Streifen, die als gleichbreit (äquidistante Einteilung) eingeführt werden, folgender Ansatz gemacht:

$$x_0 = a, \quad x_n = b, \quad \Delta x = h = \frac{b-a}{n},$$

$$x_k = x_0 + k\cdot h \quad \text{mit} \quad k = 0(1)n.$$

Dann ist

$$F = \lim_{n\to\infty} \sum_{k=0}^{n-1} f(x_k)\cdot h,$$

und oft wird darauf verwiesen, daß diese Formel für die heute übliche Schreibweise des Integrals verantwortlich ist. Das Summenzeichen wird zum stilisierten S (Integralzeichen), und das $\Delta x = h$ wird zum dx.

Näherungswerte erhält man für endliche n, und deshalb gehört zur Wahrung des Gleichheitszeichens stets ein Rest R_n oder Fehlerglied R_n zur Formel. Da diese Fehlerglieder sich bezüglich n in den einzelnen im folgenden zu vermittelnden Formeln ändern, dient ein oberer Index zur Unterscheidung:

$$F = \sum_{k=0}^{n-1} f\left(a + k\,\frac{b-a}{n}\right) \cdot \frac{b-a}{n} + R_n^{\mathrm{R}},$$

$$F = h \sum_{k=0}^{n-1} f(a + kh) + R_n^{\mathrm{R}} \quad \text{mit} \quad h = \frac{b-a}{n} \quad (\textit{Rechteckregel}). \tag{5}$$

Trapezformel

Die Anschauung (Abb. 2.14) lehrt, daß die Rechteckregel bei monoton steigendem $f(x)$ stets zu kleine Integralwerte liefert. Anschaulich besser als die Rechteckregel wird es sein, wenn man die zu integrierende Kurve (als Bild der zu integrierenden Funktion) durch einen Polygonzug annähert und somit anstelle von Rechtecken nun von Trapezen ausgeht (Abb. 2.15).

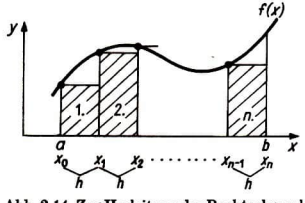


Abb. 2.14. Zur Herleitung der Rechteckregel. Es gibt n Streifen oder Rechtecke mit der Breite $h = \dfrac{b-a}{n}$

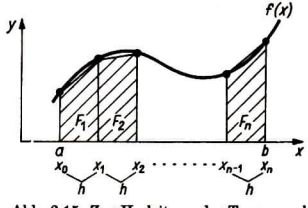


Abb. 2.15. Zur Herleitung der Trapezregel. Es gibt n Trapeze mit der Breite oder Höhe $h = \dfrac{b-a}{n}$

Das erste Trapez hat die Fläche

$$F_1 = \frac{h}{2}\,(f(x_0) + f(x_1)), \tag{6}$$

das zweite Trapez die Fläche

$$F_2 = \frac{h}{2}\,(f(x_1) + f(x_2)),$$

das k-te Trapez die Fläche

$$F_k = \frac{h}{2}\,(f(x_{k-1}) + f(x_k)),$$

das vorletzte Trapez die Fläche

$$F_{n-1} = \frac{h}{2}\,(f(x_{n-2}) + f(x_{n-1})),$$

und das letzte Trapez hat die Fläche

$$F_n = \frac{h}{2}\,(f(x_{n-1}) + f(x_n))\,;$$

die Höhe oder Breite der n Trapeze ist

$$h = \frac{b-a}{n}.$$

Nun ergibt sich die Summe aller Trapeze:

$$F = \sum_{k=1}^{n} F_k + R_n^{\mathrm{T}},$$

$$F = \frac{h}{2} \sum_{k=1}^{n} (f(x_{k-1}) + f(x_k)) + R_n^{\mathrm{T}}.$$

Näheres Betrachten dieser Formel oder auch der oben angegebenen Aufzählung der Trapezflächen zeigt, daß $f(x_0) = f(a)$ und $f(x_n) = f(b)$ nur einmal vorkommen, alle anderen $f(x_k)$ für $k = 1(1)n-1$ aber zweimal, denn diese Seiten sind rechter und linker Trapezrand zugleich für zwei benachbarte Trapeze.

Dies ergibt also:

$$F = \frac{h}{2}\,\big(f(a) + f(b)\big) + h \sum_{k=1}^{n-1} f(x_k) + R_n^{\mathrm{T}} \quad (\textit{Trapezregel}) \tag{7}$$

mit $h = \dfrac{b-a}{n}$, $x_k = a + kh$ (zum Restglied vgl. (24)). Der Unterschied der Trapezregel zur Rechteckregel besteht also lediglich darin, daß in ihr

$$\frac{h}{2}\, f(b) \text{ zuviel und } \frac{h}{2}\, f(a) \text{ zuwenig}$$

enthalten ist. Die numerische Auswirkung dieser Tatsache aber ist beträchtlich.

Simpson-Regel

Nun können wir diesen Gedanken weiter verfolgen — Newton und Cotes haben es getan — und statt der

Geradenstücke durch jeweils 2 Punkte

nun

Parabelstücke durch jeweils 3 Punkte,

⋮

durch Polynome n-ten Grades definierte Kurven durch jeweils $n+1$ Punkte

nehmen und diese exakt integrieren. Es wird also bei der Entwicklung der Formelgruppe nach Newton-Cotes zunächst für $f(x)$ eine Ersatzfunktion aus Polynomen zusammengesetzt und diese dann exakt integriert. Dies wird vorgeführt für den einfachen Fall, daß Parabeln benutzt werden, die durch äquidistante Punkte (d. h., die Abszissen sind äquidistant) gehen.

Die Parabel wird durch ein Lagrange-Interpolationspolynom dargestellt. Die Basisfunktionen sind

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{1}{2h^2}(x^2 - (x_1+x_2)x + x_1x_2),$$
$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = -\frac{1}{h^2}(x^2 - (x_0+x_2)x + x_0x_2), \tag{8}$$
$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{1}{2h^2}(x^2 - (x_0+x_1)x + x_0x_1),$$

und damit wird die Interpolationsparabel (Abb. 2.16)

$$y = y_0L_0(x) + y_1L_1(x) + y_2L_2(x)$$

gewonnen.

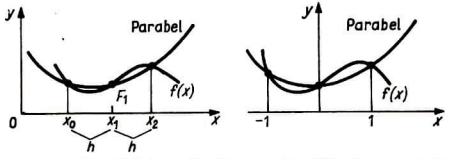


Abb. 2.16. Zur Herleitung der Simpson-Regel für die numerische Integration. Die zu integrierende Funktion $f(x)$ wird durch einen Parabelbogen ersetzt. Die Referenz dazu ist (x_0, x_1, x_2) und äquidistant mit $x_1 - x_0 = h$

Zum Integrieren von x_0 bis x_2 wird dieses Intervall auf die Länge 2 gedehnt und dann nach $(-1, +1)$ verschoben. Die Ordinaten oder Funktionswerte bleiben erhalten. Es wird also die Fläche um den Faktor $1/h$ zu groß, und dies wird durch Multiplikation mit h kompensiert. Für den betrachteten Flächenanteil F_1 gilt demnach

$$F_1 = y_0\int_{x_0}^{x_2} L_0(x)\,dx + y_1\int_{x_0}^{x_2} L_1(x)\,dx + y_2\int_{x_0}^{x_2} L_2(x)\,dx$$
$$= h\left(y_0\int_{-1}^{+1} L_0^+(x)\,dx + y_1\int_{-1}^{+1} L_1^+(x)\,dx + y_2\int_{-1}^{+1} L_2^+(x)\,dx\right).$$

Dabei haben die transformierten $L_k^+(x)$ die einfache Form

$$L_0^+(x) = \frac{1}{2}(x^2 - x),$$
$$L_1^+(x) = -(x^2 - 1), \tag{9}$$
$$L_2^+(x) = \frac{1}{2}(x^2 + x),$$

und deren Integrale werden

$$I_0 = \frac{1}{2}\int_{-1}^{+1}(x^2 - x)\,dx = \frac{1}{6}(1+1) = \frac{1}{3},$$
$$I_1 = -\frac{2}{3} + 2 = \frac{4}{3}, \tag{10}$$
$$I_2 = \frac{1}{3}.$$

Damit ist $F_1 = \frac{h}{3}(y_0 + 4y_1 + y_2)$, was auch als

$$F_1 = \frac{2h}{6}(y_0 + 4y_1 + y_2) \tag{11}$$

geschrieben wird. Diese letzte Form läßt den Nenner 6 für die sechs Funktionswerte in der Klammer erscheinen. Es entsteht ein Mittelwert der y-Werte, und deshalb werden die Formeln der Newton-Cotes-Gruppe auch *Mittelwertformeln* genannt, denn diese Eigenschaft tritt bei allen auf. Der Faktor $2h$ gibt die Breite des Parabelbereiches an. Die Formel (11) nennt man auch *Keplersche Faßregel.*

Bevor zu dieser Namenswahl eine Bemerkung folgt, zunächst noch die Zusammenfassung aller Teilflächen, wodurch die Simpson-Regel (um 1750) entsteht:

$$F = \sum_{k=1}^{n/2} F_k + R_n^S.$$

Die einzelnen F_k sind die den einzelnen Parabelbögen zugeordneten Flächen, und bei einer Streifenteilung in n Teile werden immer zwei Streifen zu einer Parabel gehören. Damit das bis zur letzten Parabel funktioniert, muß n eine gerade Zahl sein, und es gibt dann $n/2$ Parabeln.

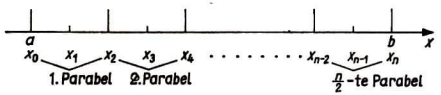


Abb. 2.17. Zur Herleitung der Simpson-Regel. Zu jedem Parabelbogen gehören drei Stützstellen. Die inneren Randwerte gehören dabei jeweils zu zwei Parabeln

Abb. 2.17 zeigt, was bei dieser Zusammenfügung zu beachten ist. Die äußeren Randwerte des Intervalls $x_0 = a$ und $x_n = b$ kommen nur einmal vor. Die zugehörigen Funktionswerte sind y_0 und y_n, die also von der ersten und der letzten Parabel stammen. Alle anderen (inneren) Randstellen der einzelnen Parabeln müssen doppelt gezählt werden und liefern den Beitrag

$$2 \sum_{k=2,4,6,\ldots}^{n-2} y_k.$$

Schließlich geben die Mittelpunkte aller Parabelintervalle den Beitrag

$$4 \sum_{k=1,3,5,\ldots}^{n-1} y_k,$$

und somit entsteht insgesamt

$$F = \frac{2h}{6}\left((y_0 + y_n) + 2 \sum_{k=2,4,6,\ldots}^{n-2} y_k + 4 \sum_{k=1,3,5,\ldots}^{n-1} y_k\right) + R_n^S \tag{12}$$

oder

$$F = \frac{2h}{6}\left(f(a) + f(b) + 2 \sum_{k=2,4,6,\ldots}^{n-2} f(a+kh) + 4 \sum_{k=1,3,5,\ldots}^{n-1} f(a+kh)\right) + R_n^S \tag{13}$$

mit $h = \dfrac{b-a}{n}$ und n gerade als *Simpson-Regel* (zum Restglied vgl. (25)). Es ist dabei eigentlich merkwürdig und unbefriedigend, daß die inneren Teilungspunkte unterschiedlich bewichtet werden müssen. Es deutet sich hier bereits eine negative Eigenschaft der Newton-Cotes-Formeln an, auf die später noch deutlicher hingewiesen werden wird.

Keplers Faßregel

Wie bereits gesagt, läuft die Simpson-Regel für nur einen Parabelbogen auch unter diesem Namen. JOHANNES KEPLER lebte von 1571 bis 1630. Wie damals üblich, sorgte auch er für sich und seine Familie durch das Einlagern von jährlich einigen Fässern Wein. Es wunderte ihn aber bald die Volumenvermessungstechnik der Faßmacher

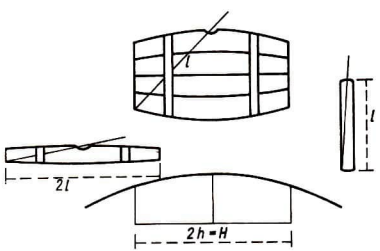


Abb. 2.18. Zur Zeit KEPLERS wurde das Faßvolumen durch eine Rute gemessen, die man durch das Spundloch steckte. Es gibt aber zwei extrem gestaltete Fässer mit derselben Rutenlänge l und dem Inhalt 0. KEPLER idealisierte das Faßprofil durch eine Parabel

(Küfer) bzw. der Weinlieferanten. Es wurde mit einer Rute durch das an der dicksten Faßstelle gelegene Spundloch zum Rand hin gemessen (Abb. 2.18). Man kann sich aber leicht zwei Extremfässer mit dem Inhalt Null und derselben Rutenlänge l denken. Das eine hat die Länge $2l$ und die Dicke 0, und das andere hat die Dicke l und die Länge 0. Also überlegte sich Kepler mit Hilfe einer Parabel, durch die er das Faßprofil annäherte, auf ähnliche Weise, wie oben vorgeführt, eine bessere Regel zum Berechnen des Volumens von Fässern. Wegen der Symmetrie gilt

$$F = \frac{2h}{6}\,(2r_{\text{Boden}} + 4r_{\text{Spund}}). \tag{14}$$

Es wird damit zunächst die halbe Querschnittsfläche und noch nicht das Volumen berechnet. Das Volumen ergibt sich vielmehr mit $H = 2h$ zu

$$V = \frac{H\pi}{60}\,(3r_{\text{Boden}}^2 + 8r_{\text{Spund}}^2 + 4r_{\text{Boden}}r_{\text{Spund}}), \tag{15}$$

und KEPLER kannte zu dessen Berechnung Verfahren ähnlich den *Guldinschen Regeln* (GULDIN 1577—1643), wobei sich das Volumen von Rotationskörpern als Produkt der halben Querschnittsfläche mit dem Weg des Flächenschwerpunktes ergibt.

Die 3/8-Regel

Ohne es vorzurechnen, wird noch eine weitere Formel der Newton-Cotes-Gruppe genannt.

Durch Interpolation mit kubischen Parabeln in den Teilintervallen entsteht die *3/8-Regel*:

$$F = \frac{3h}{8}\left(f(a) + f(b) + 3\sum_{k=1,4,7,\ldots}^{n-2} f(a+kh) + 3\sum_{k=2,5,8,\ldots}^{n-1} f(a+kh) + 2\sum_{k=3,6,9,\ldots}^{n-3} f(a+kh)\right) + R_n^{(3/8)} \tag{16}$$

mit $h = \dfrac{b-a}{n}$ und n durch 3 teilbar (zum Restglied vgl. (26)), die wegen ihres Faktors so heißt. Es ist auch wieder eine Mittelwertsformel, denn für den Einzelbogen gilt (z. B. für F_1)

$$F_1 = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3). \tag{17}$$

Es werden also acht Funktionswerte gemittelt.

Wiederum ist es störend, daß unterschiedliche Gewichte auftreten. Auf diese Weise lassen sich beliebig weitere Formeln aufstellen. Jedoch bei der Interpolation des Grades 8, wenn jeweils acht Streifen zusammengefaßt werden, tauchen erstmalig nicht nur unterschiedliche, sondern sogar negative Gewichte auf. Das ist nun sehr unschön, da ja ein bestimmtes Integral (Riemannscher Integralbegriff) Grenzwert einer Summe ist. Tatsächlich sagt ein Satz von KUSMIN [49] aus, daß die nach dem Verfeinerungsprozeß von NEWTON-COTES für gewisse Integranden erhältlichen Werte nicht gegen das Integral konvergieren. Unter dem Verfeinerungsprozeß wird dabei verstanden, daß im gesamten Intervall (a, b), über dem integriert werden soll, der Graph der Funktion erst durch eine Gerade, dann durch eine Parabel, dann durch eine kubische Parabel usw. ersetzt wird. Die Werte, die man nach den in den obigen Formeln beschriebenen Verfeinerungsprozessen (ein, zwei, drei, ... Trapeze, ein zwei, drei, ... Parabeln und ein, zwei, drei, ... kubische Parabeln) erhält, konvergieren jedoch gegen das Integral.

2.3.2. Restglieder oder Integrationsfehler

Die Herleitung des Restgliedes wird für den Spezialfall der Trapezregel und den Flächenanteil F_1, also für den ersten Streifen, vorgeführt. Bei anderen Formeln gelingt es dann auf ähnliche Weise durch Mitnahme von immer mehr Gliedern der Taylorentwicklungen und Ansatz von immer mehr y_i-Werten (Funktionsstützwerten). Zur weiteren Vereinfachung wird dieser Streifen vom Intervall $(a, a+h)$ auf $(0, h)$ verlegt. Es ist also

$$F_1 = \int_0^h f(x)\,dx = h(a_0 y_0 + a_1 y_1) + R_n^{T_1} \tag{18}$$

mit $y_0 = f(a)$ und $y_1 = f(a + h)$. Die Koeffizienten a_0 und a_1 sind uns schon aus (6) mit den Werten $\frac{1}{2}$ bekannt. Hier werden sie noch als unbekannt angenommen, dann kann gleich noch eine weitere Methode zum Gewinn der Mittelwertformeln vorgeführt werden. (Der obere Index T_1 am Restglied verweist auf die Tatsache, daß es sich um den Fehler beim ersten Streifen handelt.)

Nach TAYLOR erhält man für den linken Teil von (18) zunächst durch Entwicklung bei $x_0 = 0$ und Integration

$$f(x) = f(a) + f'(a)\,x + f''(a)\,\frac{x^2}{2} + O(x^3),$$

$$\int_0^h f(x)\,dx = \left[f(a)\,x + f'(a)\,\frac{x^2}{2} + f''(a)\,\frac{x^3}{6} + O(x^4)\right]_0^h$$

$$= h\left(f(a) + f'(a)\,\frac{h}{2} + f''(a)\,\frac{h^2}{6}\right) + O(h^4). \tag{19}$$

Die rechte Seite von (18) findet man ebenfalls mit Hilfe der Taylorentwicklung bei $x_0 = 0$ für $x = h$:

$$h(a_0 y_0 + a_1 y_1) = h\left(a_0 f(a) + a_1 f(a) + a_1 f'(a)\,h + a_1 f''(a)\,\frac{h^2}{2}\right) + O(h^4). \tag{20}$$

Durch Vergleich der Koeffizienten bei $f(a)$ und $f'(a)$ erhält man

$$a_0 + a_1 = 1 \quad \text{und} \quad a_1 = \frac{1}{2}, \tag{21}$$

so daß auch $a_0 = \frac{1}{2}$ ist.

Damit ist hier aus (18) die Trapezregel (7) gewonnen worden. Das Restglied ergibt sich nun aus

$$R_n^{T_1} = \int_0^h f(x)\,dx - \frac{h}{2}\,(y_0 + y_1) = f''(a)\,h^3\left(\frac{1}{6} - \frac{1}{4}\right) + O(h^4)$$

$$= -\frac{1}{12}\,f''(a)\,h^3 + O(h^4)$$

oder, da bei Verwendung des Restgliedes der Taylorentwicklung im Intervall $(0, h)$ bzw. $(a, a + h)$ eine Stelle ξ existiert,

$$R_n^{T_1} = -\frac{1}{12}\,f''(\xi)\,h^3. \tag{22}$$

Der Gesamtfehler ist somit

$$R_n^T = -\frac{h^3}{12}\sum_{k=1}^{n} f''(\xi_k), \tag{23}$$

denn diese Überlegung gilt für jeden Streifen. Eine Abschätzung gelingt mit dem Maximalwert $\max |f''|$ der zweiten Ableitung von $f(x)$ im Intervall (a, b):

$$|R_n^T| \leqq \frac{h^3 n}{12} \max |f''|$$

oder, da $h = \dfrac{b-a}{n}$ ist,

$$|R_n^T| \leqq \frac{(b-a)^3}{12n^2} \max |f''| \quad \text{für die Trapezregel (7).} \tag{24}$$

Das heißt nun, daß eine Verdopplung von n den Fehler auf den vierten Teil drückt. Der Konvergenzfaktor bei Verdopplung von n ist also

$$q = \frac{1}{4}.$$

Ganz mit derselben Gedankenfolge erhält man

$$|R_n^S| \leqq \frac{(b-a)^5}{180n^4} \max |f^{(4)}| \text{ für die Simpson-Regel (13)} \tag{25}$$

und

$$|R_n^{(3/8)}| \leqq \frac{(b-a)^5}{80n^4} \max |f^{(4)}| \text{ für die 3/8-Regel (16).} \tag{26}$$

Es ist dabei bemerkenswert, daß beim Übergang von der Trapez- zur Simpson-Regel die dritte Ableitung übersprungen wird. Daraus ist noch eine Schlußfolgerung zu ziehen. Es ist nach der Konstruktion der Simpson-Regel, welche ja Interpolationsparabeln verwendet, klar, daß Parabeln mit dieser Formel auch bei beliebig breiten Streifen und auch schon bei $n = 2$, also nur einer Parabel zur Interpolation, exakt

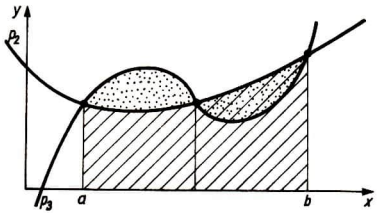


Abb. 2.19. Bei der Integration einer kubischen Parabel p_3 erhält man bei Verwendung der Simpson-Regel exakte Werte. Die punktierten Differenzflächen zur Parabel p_3 heben sich stets auf

integriert werden. Liegt aber $f(x)$ als zu integrierende Funktion in Form einer kubischen Parabel vor, so ist die vierte Ableitung konstant Null, demnach auch das Restglied Null, und kubische Parabeln werden mit der Simpson-Regel auch bei Einsatz nur einer Interpolationsparabel und beliebig breitem Integrationsintervall exakt integriert (Abb. 2.19). Dies kann der Lehrer bei Schülerübungen zur Integration gut verwenden. Die Schüler haben gegebene kubische Parabeln zu integrieren, der Lehrer kann unter Kenntnis der Simpson-Regel die Rechenarbeit für sich auf dreimaligen Einsatz des Horner-Schemas reduzieren. Beim gliedweisen Integrie-

ren kommt man zwar mit zweimaligem Einsatz des Horner-Schemas aus, man hat jedoch die Integrationsarbeit und beim Horner-Schema oft mit Bruchrechnung zu tun.

Beispiel

$$f(x) = 5x^3 - 4x^2 + 6x - 7$$

ist von -1 bis $+7$ zu integrieren. Man nimmt $h = \frac{7-(-1)}{2} = 4$, also ist $\frac{2h}{6} = 4/3 = 1.333$. Das Horner-Schema ist für $x = -1$, $x = 3$ und $x = 7$ anzuwenden:

	5	−4	6	−7
	0	−5	9	−15
−1	5	−9	15	−22
	0	15	33	117
3	5	11	39	110
	0	35	217	1561
7	5	31	223	1554

Also ist

$$F = \int_{-1}^{7} f(x)\,dx = \frac{4}{3}(-22 + 4 \cdot 110 + 1554) = \frac{4}{3} \cdot 1972 = 2629\frac{1}{3}$$

$$= 2629.333.$$

Der bereits erwähnte Nachteil der Formeln der Newton-Cotes-Gruppe wird wiederum deutlich beim Übergang zur 3/8-Regel. Sie zeigt trotz erhöhten Aufwandes nur dasselbe Konvergenzverhalten wie die Simpson-Regel, und in der Abschätzung für den Fehler ist der Faktor sogar schlechter.

2.3.3. Romberg-Verfahren

Aus den Fehler- oder Restgliedern kann man das Konvergenzverhalten der betreffenden Formel bezüglich n ablesen. Eine entsprechende Bemerkung wurde bei der Trapezregel eingefügt. Der Konvergenzfaktor ist dort $q = 1/4$. Diese Erkenntnis kann zu einer Konvergenzverbesserung oder -beschleunigung ausgenutzt werden, wie sie bereits im Abschnitt über Iterationsverfahren als *Extrapolation zur Grenze* beschrieben wurde. Gerade bei der numerischen Integration wurde dieser Gedanke von dem Norweger Romberg (1955) verwendet. Wird also die Anzahl n der Teilstreifen verdoppelt, so erhält man aus den Integralnäherungswerten nach der Trapezformel, T_n und T_{2n}, einen besseren Wert S_{2n}:

$$S_{2n} = \frac{4T_{2n} - T_n}{3}. \tag{27}$$

Die Folge der S-Werte konvergiert dann mit $q = \frac{1}{16}$. Ebenso erhält man wiederum bessere Werte R_{2n}:

$$R_{2n} = \frac{16S_{2n} - S_n}{15}. \tag{28}$$

Diese Werte konvergieren mit $q = \frac{1}{64}$.

Beispiel

Zur Demonstration der Leistungsfähigkeit der einzelnen Formeln wird das Integral mit bekanntem Ergebnis

$$\int_0^{\pi} \sin x\,dx = -\cos x\,|_0^{\pi} = -(\cos\pi - \cos 0) = -(-1-1) = 2$$

numerisch ausgewertet. Tab. 2.26 zeigt die Resultate, und es ist überraschend, daß der erste Romberg-Schritt bis auf Rundungsfehler die Werte der Simpson-Regel liefert (deshalb wurden diese Werte auch schon mit S bezeichnet). Eine Nachrech-

Tab. 2.26. Numerische Auswertung des Integrals $2 = \int_0^{\pi} \sin x\,dx$. Es ist $n = 2^k$. Die T_n werden nach der Trapezformel berechnet, die S_n daraus durch einen Romberg-Schritt und die R_n daraus wiederum durch einen Romberg-Schritt

k	n	Trapezformel	Romberg	Simpson	
		T_n	S_n	S_n	R_n
0	1	0	—	—	—
1	2	1.570796327	2.094395103	2.094395103	—
2	4	1.896118898	2.004559755	2.004559755	1.998570732
3	8	1.974231603	2.000269171	2.000269170	1.999983131
4	16	1.993570345	2.000016592	2.000016591	1.999999753
5	32	1.998393369	2.000001044	2.000001041	2.000000005

nung anhand der oben angegebenen Konvergenzverbesserungsformel bestätigt dieses Ergebnis. Es ist also besser, die Trapezformel zu verwenden und diese einmal nach Romberg zu verbessern, als die kompliziertere Simpson-Regel zu nehmen. Der nächste Romberg-Schritt liefert dann aber nicht etwa wiederum die nächsten Newton-Cotes-Formelwerte, nämlich die der 3/8-Regel, sondern dann ergeben sich neue bessere Werte.

Es läßt sich nachweisen (Stiefel [29], Bd. 2, S. 225, 230): Werden die Romberg-Werte auf eine lineare Kombination der ursprünglichen Stützstellenwerte y_k zurückgeführt,

$$R_n = \sum_{k=0}^{n} a_k y_k,$$

so sind alle Koeffizienten a_k positiv (solch eine Rückführung auf die y_k war z. B. notwendig, um nachzuprüfen, daß die ersten Romberg-Werte die Simpson-Werte liefern). Da weiterhin gilt (STEKLOV 1916 [49]): „Eine Interpolationsquadratur mit nicht negativen Koeffizienten konvergiert (mit wachsenden Teilungszahlen n) für jede stetige Funktion", kann man folgern, daß dies für das Romberg-Verfahren im besonderen gilt. Das Romberg-Verfahren ist deshalb in der Praxis von großer Bedeutung, es kann für numerische Integrationen stark empfohlen werden. Allgemein läßt es sich durch die folgenden Formeln beschreiben:

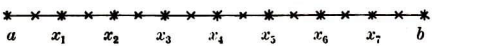
T_{0k} sind die Trapezwerte für $n = 2^k$ Streifen,

$$T_{mk} = \frac{4^m T_{m-1,k+1} - T_{m-1,k}}{4^m - 1}. \tag{29}$$

Die T_{1k}-Werte sind also die Simpson-Werte S_n von (27), und die T_{2k}-Werte stimmen mit den R_n von (28) überein.

Rechentechnische Aufbereitung

Bei der Verdopplung der Stützstellenanzahl (Tab. 2.26) fällt auf, daß man für die Errechnung der Funktionswertsumme jeden zweiten Wert wiederum zu berechnen hat, obwohl man dies schon einen Schritt zuvor tat. Die Skizze verdeutlicht das für den Übergang von 8 zu 16 Teilabschnitten:



Jedes zweite Kreuz (×) deckt sich mit einem der senkrechten Striche. Ist also für $n = 2^k$ bei der Trapezregel

$$T_{0k} = \frac{b-a}{2n}\big(f(a) + f(b)\big) + \frac{b-a}{n}\sum_{j=1}^{n-1} f(x_j) \quad \text{mit} \quad x_j = a + j\frac{b-a}{n} \tag{30}$$

und bei $2n = 2^{k+1}$

$$T_{0,k+1} = \frac{b-a}{4n}\big(f(a) + f(b)\big) + \frac{b-a}{2n}\sum_{j=1}^{2n-1} f(x'_j) \quad \text{mit} \quad x'_j = a + j\frac{b-a}{2n}, \tag{31}$$

so ist die alte Funktionswertsumme in der neuen enthalten, denn es ist

$$x_j = x'_{2j} \qquad \text{für} \qquad j = 1(1)\,n-1. \tag{32}$$

Somit ist

$$T_{0,k+1} = \frac{1}{2}\,T_{0k} + \frac{b-a}{2n}\sum_{j=1,3,5,\ldots}^{2n-1} f\left(a + j\frac{b-a}{2n}\right), \tag{33}$$

so daß man den vorherigen T_{0k}-Wert verwendet und lediglich die Summe der neu hinzutretenden Funktionswerte berechnet.

Man beginnt mit

$$k := 0, \quad n := 1, \quad h := b - a, \quad T_{00} := \frac{h}{2}\big(f(a) + f(b)\big)$$

und rechnet dann pro Verfeinerungsschritt

$$n := 2 \times n, \qquad h := h/2,$$

$$T_{0,k+1} := \frac{1}{2} T_{0k} + h \sum_{j=1,3,5,\ldots}^{n-1} f(a + jh), \tag{34}$$

$$k := k + 1.$$

Man kann eine ganz ähnliche Überlegung für die Verfeinerung der Simpson-Regel anstellen ([29] Bd. 2, S. 222), erhält dann aber eine kompliziertere Vorschrift.

Das BASIC-Programm in Tab. 2.27 benutzt als Speicherfeld nicht etwa eine Matrix (Dreiecksmatrix) zur Speicherung der Romberg-Werte (vgl. Tab. 2.26), sondern einen Vektor, der stets nur die letzte Romberg-Zeile enthält. Dabei wird nach einer Idee von BAUER (Algorithm 60 CACM) der neue Trapezwert ständig als letzte Komponente angefügt, und die Romberg-Schritte werden dann rückwärts zählend durchgeführt. Nach jedem Zyklus k steht der beste Integralwert demnach in $T[1]$. Es werden $N = 2^r$ Streifen als Maximum zugelassen, a und b sind die Integrationsgrenzen und $f(x)$ die zu integrierende Funktion. Das Resultat ist F:

```
begin array T [1 : r + 2];
      real h, s;
      integer j, k, n, p;
      h := b − a; n := 1;
      T [1] := h × (f(a) + f(b))/2;
      kzyklus: for k := 1 step 1 until r + 1
               do
               begin
               S := 0; h := h/2; n := 2 × n;
      comment Summe neuer Funktionswerte;
      for j := 1 step 2 until n − 1
      do s := s + f(a + j × h);
      comment neuer Trapezwert;
      T[k + 1] := T[k]/2 + h × s; p := 1;
      romberg: for j := k step −1 until 1
               do
               begin p := 4 × p;
               T [j] := (T [j + 1] × p − T [j])/(p−1)
               end romberg
      end kzyklus;
    F := T [1]
end
```

Tab. 2.27. BASIC-Programm zum Romberg-Verfahren der numerischen Integration

```
  5 INPUT "a =";a, "b =";b, "r =";r
 10 LET h = b - a: LET n = 1:
 20 DIM T(r + 2): REM f(a)
 30 LET x = a
 40 GOSUB 500
 50 LET s = y: REM f(b)
 60 LET x = b
 70 GOSUB 500
 80 LET s = s + y
 90 LET T(1) = h * s/2
 95 REM k Zyklus
100 FOR k = 1 TO r + 1
110 LET s = 0: LET h = h/2: LET n = 2 * n
115 REM Summe neuer Funktionswerte:
120 FOR j = 1 TO n - 1 STEP 2
130 LET x = a + j * h
140 GOSUB 500
150 LET s = s + y
160 NEXT j
165 REM neuer Trapezwert:
170 LET T(k + 1) = T(k)/2 + h * s
175 REM Romberg
180 LET p = 1
190 FOR j = k TO 1 STEP -1
200 LET p = 4 * p
210 LET T(j) = (T(j + 1) * p - T(j))/(p - 1)
220 NEXT j
230 NEXT k
240 LET F = T(1)
250 STOP
```

Aus einer solchen Algorithmen-Dokumentation als Programm-Modell in einer höheren Programmiersprache (hier ALGOL 60 — entnommen aus der Algorithmensammlung der Zeitschrift „Communications of the Association of Computing Machinery" — aber es geht auch jede andere, z. B. PASCAL oder auch FORTRAN) adaptiert man relativ leicht ein BASIC-Programm für einen Heimcomputer. Die recht bequeme Umsetzung in BASIC demonstriert, daß BASIC trotz vieler und auch schwerwiegender Einschränkungen doch zu den höheren Programmiersprachen

gezählt werden kann. Ältere Typen programmierbarer Tischrechner oder auch viele programmierbare Taschenrechner müssen dagegen in einer ASSEMBLER-Sprache programmiert werden. Die Transparenz solcher Tastenfolge-Programme ist sehr gering. Beispielsweise ist das Programm für das hier zu betrachtende Romberg-Verfahren $2^1/_2$ Druckseiten lang (Robotron K 1001/2/3 bzw. Hewlett-Packard-Tischrechner) und besteht aus ca. 240 Tastensymbolen.

Das in Tab. 2.27 niedergelegte BASIC-Programm verwendet dieselben Variablenbezeichnungen wie die ALGOL-Vorlage. Formeln ohne Aufrufe von Nichtstandard-Funktionen haben dieselbe Gestalt. Beim Aufruf des Integranden $f(x)$ muß in BASIC oft der Parameter oder das Funktionsargument als Wert einer Variablen x bereitgestellt werden. Nach dem Subroutinen-Sprung „GOSUB 5ØØ" arbeitet das frei programmierbare Unterprogramm mit diesem x und errechnet

$$y = f(x).$$

Das Resultat als Wert einer Variablen y steht dem Hauptprogramm nach der Rückkehr „RETURN" unmittelbar nach dem Subroutinen-Sprung zur Verfügung. Dieses Verfahren der Parametervermittlung über Variable als „Tote Briefkästen" ist eine der Schwächen von BASIC. Aber in diesem Beispiel ist alles noch gut überschaubar.

Zum Test wurde als Unterprogramm der Integrandenfunktion einfach

```
5ØØ  y = SIN (x)
51Ø  RETURN
```

verwendet. Einige BASIC-Versionen bieten die Möglichkeit der Funktionsdefinition mit bequemer Parametervermittlung. Derartige Funktionen können dann wie Standardfunktionen in Formeln benutzt werden.

Statt der Zeilen 500 und 510 kann man im vorliegenden Programm

```
5ØØ  DEF FNf(x) = SIN x
```

und dann statt 30 bis 90 viel einfacher

```
9Ø  LET T(I) = h * (FNf(a) + FNf(b))/2
```

und ebenso statt 130 bis 150

```
15Ø  LET s = s + FNf(a + j * h)
```

schreiben.

Gibt man nach der Aufforderung aus Programmzeile 5

```
a = 0      b = PI
r = 4
```

ein, d. h. läßt man 16 Trapezstreifen zu, so äußert sich das Programm mit dem

Druck

Integral = 2,

wenn

245 PRINT "Integral ="; F

eingefügt wird.

Zur Kontrolle der Zwischenwerte gemäß Tab. 2.26 füge man

91 PRINT 10, $T(1)$

171 PRINT $(k+1) \times 10 + k$, $T(k+1)$

215 PRINT $j \times 10 + k$, $T(j)$

ein. Man erhält dann die Werte aus Tab. 2.28 mit einem Vermerk der zugehörigen (j, k)-Kombination als zweistellige Dezimalzahl.

Man kann ein Programm wie das aus Tab. 2.27 leicht als Unterprogramm in umfangreicheren Rechnungen einsetzen. Statt der Zeile 5 wird man a, b und r als Parameter zur Verfügung stellen, d. h. vor einem Aufruf mit Werten versehen, und den STOP-Befehl wird man durch

250 RETURN

ersetzen, worauf man den Integralwert als Wert von F zur Verfügung hat. Um die Arbeitsgenauigkeit noch steuerbar zu machen, wird man — bei genügend groß vorgegebenem r — nach Eingabe oder Parametervermittlung ein e (für ε) noch einfügen bzw. überschreiben:

212 IF ABS $(T(j+1) - T(j)) < e$ THEN GOTO 240

240 LET $F = T(j)$

Es wird dann der erste Zwischenwert, der die Genauigkeitsbedingung erfüllt, als Resultat zurückgegeben.

Tab. 2.28. Zwischenwerte aus dem BASIC-Programm für die Romberg-Integration im Beispiel $\int_0^{\pi} \sin x\,dx$. Die Werte aus Tab. 2.26 tauchen in anderer Anordnung wieder auf

	k				
	0	1	2	3	4	$5 = r + 1$
$j = 6$						1.9983934
5					1.9935703	2.0000010
4				1.9742316	2.0000165	2.0000001
3			1.8961189	2.0002692	1.9999997	2.0000000
2		1.5707963	2.0045597	1.9999831	2.0000000	2.0000000
1	0	2.0943952	1.9985707	2.0000056	2.0000000	2.0000000

2.3.4. Andere Integrationsformeln

Hermite-Formeln (1878)

In diesen Formeln werden nicht nur die Funktionswerte der zu integrierenden Funktion, sondern auch die Ableitungen benutzt. Grundgedanke ist dabei wie schon bei der Interpolation, daß für die Ersatzpolynome in den Stützstellen nicht nur die Funktionswerte, sondern auch die Ableitungswerte mit denen der zu ersetzenden Funktion übereinstimmen, so daß die Teilpolynome auch stetig in den Ableitungen, d. h. ohne Knicke *glatt* aneinanderstoßen. Die der Trapezformel von NEWTON-COTES entsprechende Formel ist z. B.

$$F = \frac{h}{2}\left(f(a) + f(b)\right) + h\sum_{j=1}^{n-1} f(a + jh) + \frac{h^2}{12}\left(f'(a) - f'(b)\right) + R_n^{\mathrm{H}} \tag{35}$$

mit

$$h = \frac{b-a}{n}$$

und dem Restglied

$$|R_n^{\mathrm{H}}| = \frac{(b-a)^5}{720n^4} \max |f^{(4)}|.$$

Im Beispiel

$$F = \int_0^{\pi} \sin x \, dx$$

bedeutet das lediglich eine Korrektur von $\frac{h^2}{6}$ gegenüber der Trapezregel. Tab. 2.29 veranschaulicht die Auswirkung.

Aus dem Konvergenzfaktor (siehe Restglied) erkennt man schon das gleiche Verhalten wie bei der Simpson-Regel. (In [29] Bd. 1, 4.9.1., und Bd. 2, S. 218, ist bezüglich dieser Hermite-Regel eine noch verblüffendere Gegenüberstellung angegeben.) Solche Formeln zeigen die Wirksamkeit und Effektivität des Einsatzes tiefergehender mathematischer Kenntnisse auch oder gerade im Zeitalter leistungsstarker Computer.

Man muß dem Argument gegenübertreten, welches das Verwenden schwacher, aber leicht einsichtiger Formeln empfiehlt, da der schnelle Computer die Mängel der Formel kompensiere. Es ist vielmehr so, daß es für jedes Hilfsmittel auch die Kostenfrage gibt, und außerdem kann man bei jedem Computer an die Leistungsgrenze stoßen.

In jedem Fall lohnt sich die Suche nach besseren Verfahren, und diese wird sogar durch die sich seit Existenz der Computer anbietende Chance der Realisierung kräftig stimuliert.

Tab. 2.29. Vergleich der Trapezregel nach NEWTON-COTES mit der Trapezregel nach HERMITE für das Beispiel $\int_0^{\pi} \sin x \, dx$

k	n	NEWTON-COTES	HERMITE
0	1	0	1.644934067
1	2	1.570796327	1.982027144
2	4	1.896118898	1.998927277
3	8	1.974231603	1.999933698
4	16	1.993570345	1.999995869
5	32	1.998393369	1.999999750

Gauß-Formeln

Die Gauß-Formeln zur numerischen Integration sind ebenfalls Mittelwertformeln. Es treten allerdings keine Ableitungen auf. Die Grundidee besteht darin, daß in der Formel

$$\int_{-1}^{+1} f(x)\,dx = \sum_{i=1}^{n} C_i \times f(x_i) + R_n^G$$

(das Intervall ist auf $(-1, +1)$ normiert) nicht nur die Gewichte C_i, sondern auch die Lagen der x_i variabel sind. Es gelingt mit dieser Variante, bei n Stützstellen noch Polynome bis zum Grad $2n - 1$ exakt zu integrieren, d. h., für diese Polynome ist

$$R_n^G = 0.$$

Man erinnere sich, daß die Simpson-Regel mit drei Stützstellen noch kubische Polynome exakt integrierte. Eine Gauß-Formel mit drei Stützstellen integriert Polynome bis zum Grad 5 exakt. Aber die Lage der Stützstellen ist kompliziert. Es sind die Nullstellen der sogenannten Legendreschen Polynome (vgl. [29], Bd. 2, S. 213ff.). Stützstellen und Gewichte bis zu $n = 40$ findet man bei KRONROD (1964) [37] und STROUD (1966) [59]. Man argumentiert, daß beim Einsatz von Computern die Funktionswertberechnung für „einfache" äquidistante Stützstellen ebenso aufwendig ist wie für die „komplizierten" Legendre-Polynom-Nullstellen. Aber die äquidistanten Stützstellen können einfach berechnet werden, die Nullstellen der Polynome müssen gespeichert werden. Das verdoppelt zwar nur den Speicheraufwand gegenüber den Newton-Cotes-Formeln, bei denen nur die Gewichte gespeichert zu werden brauchen, aber bei Kleinrechnern wie Tisch- und Taschenrechnern verbietet sich das vorläufig noch wegen der kleinen Speicherkapazität. Bei Videocomputern ist das Einlesen der Nullstellen von einer Kassette lästig.

Es gibt außer den genannten Typen von Formeln zur numerischen Integration natürlich noch weitere, deren Einsatz besonders bei speziellen Differentialgleichungen zu empfehlen ist (z. B. Differenzenformeln, die außer den Funktionswerten noch Werte aus der Differenzentabelle (siehe 2.2.) verwenden).

2.4. Lineare Gleichungssysteme

In MfL Bd. 10, 6.1. (auch in [29]) sind direkte und indirekte Verfahren zur Lösung linearer Gleichungen beschrieben und erläutert worden. Insbesondere wird dort auf das Verfahren der *Pivotisierung* zur Vermeidung numerischer Schwierigkeiten verwiesen.

Bei drehbaren Windmühlen, sogenannten Bockwindmühlen, nennt man den Hauptzapfen oder die sehr kräftige Drehachse der Mühle *Pivot*. Solche Mühlen kann man beispielsweise im Dorfmuseum Riga, aber auch bei uns sehen.

Auf Beachtung solcher und anderer numerischer Effekte und die zugehörigen Gegenmaßnahmen kann und muß zunächst in der vorliegenden Darstellung verzichtet werden.

Zwar sind die Speichervolumen der Kleinstrechner noch recht beschränkt, so daß einmal die behandelbaren Aufgaben noch klein bleiben, und zum anderen dürfen die Programme nicht zu umfangreich werden. (Beispielsweise kann gezeigt werden, daß beim Robotron K 1003 oder Hewlett-Packard-Tischrechner unter Verwendung des reinen Gauß-Algorithmus der Umfang eines Gleichungssystems kaum über 10×10 möglich ist.) Jedoch lohnt sich die automatische „Bekämpfung" schlechter Kondition und Pivotisierung offenbar doch, denn schon wenige und sogar „kleine" Beispiele zeigen, wie oft Nullen in der Diagonale entstehen. Dabei ist eine Pivotsuche aus dem gesamten Restfeld fast immer unnötig, denn eine Multiplikation einer Zeile mit einem genügend großen Faktor könnte diese stets zur Pivotzeile machen, wenn überhaupt von Null verschiedene Elemente in der Zeile auftreten. Multiplikationen mit Faktoren oder auch Linearkombinationen von Gleichungen ändern dabei nichts an der Lösung des Systems. Eine Pivotisierung ist im Prinzip nur sinnvoll, wenn man sie auf eine zuvor *skalierte* (*equilibrierte*) Matrix anwendet, d. h., wenn man die Koeffizienten in allen Gleichungen auf die gleiche Größenordnung gebracht hat. Die Rechenstrategie (hier die Reihenfolge der Gleichungen oder Zeilen, in denen man Koeffizienten vom Wert Null zur Herstellung der Dreiecksgestalt erzeugt) wird von der Pivotauswahl beeinflußt. Große Verbreitung in der Praxis hat die sogenannte *Zeilenpivotisierung* gefunden, bei der jeweils aus der k-ten Restspalte (Elemente a_{ik} für $i = (k + 1)\,(1)n$) im Schritt k das Pivotelement gesucht wird. Jede Rechenstrategie führt zu anderen Rundungsfehlern. Problematisch bleiben immer diejenigen Rechengänge, bei denen sich in den Matrixelementen Stellen durch Subtraktion auslöschen. Eine optimale Strategie wäre eine solche mit den wenigsten Auslöschungen.

Was an dieser Stelle aber als erstes gezeigt werden soll, ist die Methode, wie man von einer problemorientierten Algorithmusformulierung — etwa in ALGOL 60 — wieder zu einer BASIC-Form oder gar maschinenorientierten Form kommt. Dieses deduktive Vorgehen ist hier einmal absichtlich gewählt worden, da Algorithmenformulierungen in höheren Programmiersprachen oft bereits vorliegen und man vor die Aufgabe gestellt wird, diese in eine solche Form zu bringen, daß die Abarbeitung mit einem Kleinstrechner und dessen Programmiersprache ermöglicht wird. Ferner wird hier der Gebrauch von Indexregistern und des Befehls IND in der Programmierung ausführlich erläutert, der häufig bei programmierbaren Taschenrechnern PTR mit großem Speicher vorhanden ist.

2.4.1. Lösung eines linearen Gleichungssystems nach Gauss mit Hilfe eines Kleinstrechners

Bekanntlich besteht der Gauß-Algorithmus aus zwei Hauptschritten, der Herstellung der *Dreiecksgestalt* und der Berechnung der Unbekannten, der sogenannten *Rückrechnung*.

Aus dem ALGOL-Programm

```
Gauß:
begin integer n; read (n);
      begin array a [1 : n, 1 : n + 1], x[1 : n];
            integer i, j, k; real f;
            read (a);

Dreiecksgestalt for i := 1 step 1 until n − 1
                do for k := i + 1 step 1 until n
                   do begin f := a[k, i]/a[i, i]; a[k, i] := 0;
                      for j := i + 1 step 1 until n + 1
                      do a[k, j] := a[k, j] − a[i, j] × f
                      end Dreiecksgestalt;

Rückrechnung: x[n] := a[n, n + 1]/a[n, n];
              for i := n − 1 step−1 until 1
              do begin
                    x[i] := a[i, n + 1];
                    for k := i + 1 step 1 until n
                    do x[i] := x[i] −a[i, k] × x[k];
                    x[i] := x[i]/a[i, i]
                 end Rückrechnung;
           print (x)
      end
end Gauß
```

erhält man zunächst durch Auflösung der Laufanweisungen die Flußbilder in Abb. 2.20 und Abb. 2.22. Bereits beim ALGOL-Programm wurde angenommen, daß die rechte Seite des Systems als $(n + 1)$-te Spalte im Koeffizientenschema enthalten ist.

Im Kleinrechner liegt das Datenfeld linear gespeichert vor. Gerade die Umsetzung mehrdimensionaler Felder (Matrizen) in eine linearisierte (vektorielle) Gestalt ist ein gewöhnlich als schwierig empfundener Prozeß.

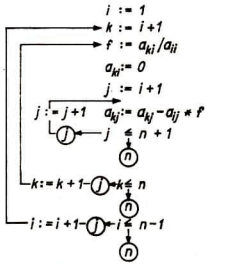


Abb. 2.20

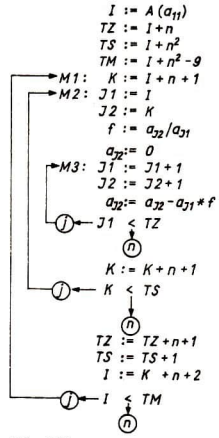


Abb. 2.21

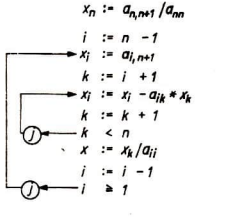


Abb. 2.22

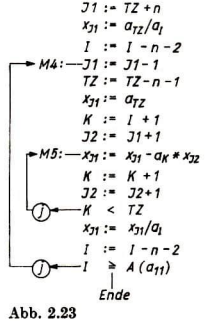


Abb. 2.23

Abb. 2.20. Flußdiagramm zur Herstellung der Dreiecksgestalt der Koeffizientenmatrix eines linearen Gleichungssystems gemäß Gauß-Algorithmus nach Auflösung der Laufanweisungen im entsprechenden ALGOL-Programm (siehe Text)

Abb. 2.21. Maschinenorientiertes Flußdiagramm entsprechend Abb. 2.20

Abb. 2.22. Flußdiagramm zu Berechnung der Unbekannten eines linearen Gleichungssystems gemäß Gauß-Algorithmus nach Auflösung der Laufanweisungen im entsprechenden ALGOL-Programm (siehe Text)

Abb 2.23. Maschinenorientiertes Flußdiagramm entsprechend Abb. 2.22

Nimmt man zeilenweise Speicherung an und zugleich in jeder Zeile als letztes Element das Element der rechten Seite, dann lautet die Adressierungsfunktion

$$A(a_{ij}) = A(a_{11}) + (n+1)(i-1) + j - 1. \tag{1}$$

Die Elemente der rechten Seite sind als $a_{i,n+1}$ enthalten, so daß j bis $n+1$ läuft.

An Stelle der Indizes i, j, k soll im Programm mit nur einem Index gearbeitet werden. Für die Rechenanweisung

$$a_{kj} := a_{kj} - a_{ij} \times f,$$

welche mit j in der innersten Schleife läuft, werden zwei Indexregister $I1$ und $I2$ für

$$A(a_{11}) + (n+1)(k-1) + j - 1 \qquad \text{(Zeile } k\text{)}$$

und

$$A(a_{11}) + (n+1)(i-1) + j - 1 \qquad \text{(Zeile } i\text{)}$$

eingeführt. An Stelle von $j := j + 1$ werden

$$I1 := I1 + 1 \quad \text{und} \quad I2 := I2 + 1$$

realisiert. An Stelle von $i := i + 1$, welches zum Aufruf der Diagonalelemente a_{ii} dient, wird $I := I + n + 2$ genommen. Der Anfangswert für k ist nicht mehr $i + 1$, sondern es gilt $K := I + n + 1$, und es ist auch nicht mehr $j := i + 1$, sondern

$$I1 := I, \qquad I2 := K.$$

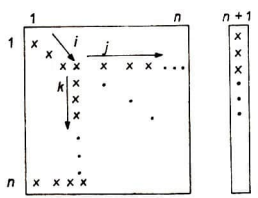


Abb. 2.24. Zur Veranschaulichung der Wertefolgen für die Testgrößen TZ Zeilenenden und TS Spaltenenden sowie der Adressenfolge für die Diagonalelemente

Für die Abschlußtests der Zeilen-, Spalten- und Matrixzyklen werden drei Testgrößen TZ, TS und TM eingeführt, die mit den Anfangswerten

$$TZ := A(a_{11}) + n, \quad TS := TZ + n^2, \quad TM := TZ + n^2 - 3 \tag{2}$$

versehen werden, wobei sich TZ und TS gemäß

$$TZ := TZ + n + 1, \quad TS := TS + 1 \tag{3}$$

ändern müssen. Somit erhält man die maschinenorientierten Flußdiagramme in Abb. 2.21 und 2.23, und daraus stellt man Programme her.

Abb. 2.24 dient zur Veranschaulichung der Adressenfolgen der Diagonalelemente

$$A(a_{ii}) = A(a_{11}) + (i-1)(n+2),$$
$$A(a_{i+1,i+1}) = A(a_{ii}) + (n+2),$$

der Zeilenenden nach (3) für TZ

$$A(a_{k,n+1}) = A(a_{11}) + k(n+1) - 1,$$
$$A(a_{k+1,n+1}) = A(a_{k,n+1}) + (n+1),$$

der Spaltenenden nach (3) für TS

$$A(a_{n,j}) = A(a_{11}) + n^2 + (j-2),$$
$$A(a_{n,j+1}) = A(a_{n,j}) + 1$$

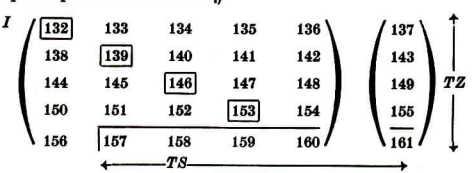
Man überzeugt sich an einem Beispiel mit $n = 5$ und $A(a_{11}) = 132$ nach Tab. 2.30 von der Korrektheit der Tests. (Wegen der Schrittweiten $\neq 1$ sind als Testgrößen Werte aus Intervallen von Schrittweitenlänge möglich.)

Ganz ähnliche Überlegungen führen zum Programmteil der Rückrechnung. Vom problemorientierten Flußdiagramm in Abb. 2.22 gelangt man zu einem maschinenorientierten in Abb. 2.23. Nach der Umsetzung in ein Programm schließt sich noch der Druck der Resultate an. Dieser kleine Abschnitt bedarf kaum einer Erläuterung.

Tab. 2.30. Werteprotokoll der Testgrößen TM (Matrixende), TZ (Zeilenende) und TS (Spaltenende) sowie der Indizes I und K für ein Beispiel mit $n = 5$ und $A(a_{11}) = 132$. Die Speicherbelegung ist mit Markierung der Werte für I, TS und TZ ebenfalls angegeben

TM	TZ	TS	I	K	Bemerkungen
154	137	157	132	138	1. Spalte
				144	
				150	
				156	
				162	Test $K < TS$ negativ
	143	158	139	145	2. Spalte
				151	
				157	
				163	Test $K < TS$ negativ
	149	159	146	152	3. Spalte
				158	
				164	Test $K < TS$ negativ
	155	160	153	159	4. Spalte
				165	Test $K < TS$ negativ
	161	161	160		Test $I < TM$ negativ, Ende

Speicherplätze der Elemente a_{ij}:



Tab. 2.31. BASIC-Programme zum Lösen linearer Gleichungssysteme nach GAUSS

```
  4 REM Gauss-Programm 1
  5 INPUT "n ="; n
 10 LET N1 = n * n + n: DIM a(N1): DIM x(n)
 20 FOR i = 1 TO N1
 30 READ a(i)
 40 NEXT i
 45 REM Dreiecksgestalt:
 50 LET TZ = 1 + n: LET TS = TZ + n * n: LET TM = TS
 60 FOR I = 1 TO TM STEP n + 2
 70 FOR K = I + n + 1 TO TS - 1 STEP n + 1
 80 LET J2 = K
 90 LET f = a(K)/a(I)
100 LET a(K) = 0
110 FOR J = I + 1 TO TZ
120 LET J2 = J2 + 1
130 LET a(J2) = a(J2) - a(J) * f
140 NEXT J
150 NEXT K
160 LET TZ = TZ + n + 1
170 LET TS = TS + 1
180 NEXT I
185 REM Rueckrechnung:
190 LET TZ = TZ - n - 1
200 LET I = I - n - 2
210 LET x(n) = a(TZ)/a(I)
220 LET J1 = n - 1
230 FOR I = I - n - 2 TO 1 STEP - n - 2
240 LET TZ = TZ - n - 1: LET x(J1) = a(TZ): LET J2 = J1 + 1
250 FOR K = I + 1 TO TZ - 1
260 LET x(J1) = x(J1) - a(K) * x(J2)
270 LET J2 = J2 + 1
280 NEXT K
290 LET x(J1) = x(J1)/a(I)
300 LET J1 = J1 - 1
305 NEXT I
310 STOP
400 DATA 3, 4, -5, 1, -35,
         5, 6, -13, -5, -67,
         3, 6, -1, 10, -31,
         9, 8, -19, -2, -75
```

Tabelle 2.31 (Fortsetzung)

```
  4 REM Gauss-Programm 2
  5 INPUT "n ="; n
 10 DIM x(n): DIM a(n, n + 1)
 20 FOR i = 1 TO n
 30 FOR j = 1 TO n + 1
 40 READ a(i, j)
 50 NEXT j
 60 NEXT i
 65 REM Dreiecksgestalt:
 70 FOR i = 1 TO n - 1
 80 FOR k = i + 1 TO n
 90 LET f = a(k, i)/a(i, i): LET a(k, i) = 0
100 FOR j = i + 1 TO n + 1
110 LET a(k, j) = a(k, j) - a(i, j) * f
120 NEXT j
130 NEXT k
140 NEXT i
145 REM Rueckrechnung:
150 LET x(n) = a(n, n + 1)/a(n, n)
160 FOR i = n - 1 TO 1 STEP - 1
170 LET x(i) = a(i, n + 1)
180 FOR k = i + 1 TO n
190 LET x(i) = x(i) - a(i, k) * x(k)
200 NEXT k
210 LET x(i) = x(i)/a(i, i)
220 NEXT i
230 STOP
Daten wie bei Gauss-1
```

In vielen BASIC-Versionen gibt es nur die Möglichkeit zur Dimensionierung (Befehl DIM) eindimensionaler Zahlenfelder, d. h. von Vektoren. Aber statt mit Adressen kann dann doch mit einem Index gearbeitet werden. Das bedeutet, daß in den soeben angestellten Überlegungen

$$A(a_{11}) = 1$$

als erster Indexwert einzusetzen ist. In den Laufanweisungen im BASIC-Programm können vorteilhaft die oben festgelegten Testgrößen und Schrittweiten verwendet werden. So entsteht das Programm 1 in Tab. 2.31.

Es gibt aber auch BASIC-Versionen mit der Möglichkeit zur Dimensionierung zweidimensionaler Zahlenfelder, d. h. von Matrizen. Diese Dialekte erlauben schon

einen weiteren Schritt auf dem Weg zu besser problemorientierten Sprachen. Das Programm 2 in Tab. 2.31 ist dem ALGOL-Vorbild viel ähnlicher.

Bei der Programmierung mit einem PTR oder älteren Tischrechner in maschinenorientierter oder ASSEMBLER-ähnlicher Form durch Angabe der Tastenfolgen muß man die soeben veranschaulichten Adressenwerte in Indexregistern bereitstellen. Ein Speicherabrufbefehl für ein Matrixelement lautet dann z. B.

MR IND I1 oder R → x IND I1

und bedeutet, daß der Inhalt des Indexregisters I1 (I1 ist eine Registernummer oder -adresse) dem Befehl MR bzw. R → x als Adresse zu geben ist, ehe er ausgeführt wird. Hat etwa I1 den Wert 141, so wird

MR 141 bzw. R → x 141

ausgeführt. Im Beispiel aus Tab. 2.30 wird demnach a_{24} aus dem Speicher geholt. Die Indexfortschreibung findet durch das Verändern des Inhalts vom Register I1 statt. Der Befehl zum Speicherabruf bleibt dabei unverändert.

Mit einem Tischrechner-Programm nach Abb. 2.21 und 2.23 wurden Testbeispiele mit ganzzahligen Koeffizienten, rechten Seiten und Lösungen gerechnet. Es ergaben sich dabei für

$n =$	2	3	4	5	6	7	8	9

Rechenzeiten in Sekunden (Aufwand A)

$A =$	6	12	23	38	59	88	123	166

so daß der Quotient $q = \dfrac{A}{n^3 + 3n^2 + 8n}$ die folgenden Werte annimmt:

$q =$	0.17	0.15	0.16	0.16	0.16	0.16	0.16	0.16

Man kann demnach für einen PTR bzw. einen HC in Abhängigkeit von der Aufgabengröße n (= Anzahl der Gleichungen) mit einem Aufwand A (= Rechenzeit) nach der Beziehung

$$A = 0.16(n^3 + 3n^2 + 8n)\ \text{s} \tag{4}$$

rechnen. Diese Formel spiegelt den Aufbau des Gauß-Algorithmus wider (vgl. Abb. 2.20 und 2.21), der drei geschachtelte Zyklen enthält, so daß der innerste Zyklus n^3-mal durchlaufen wird. Die Koeffizienten des quadratischen und des linearen Gliedes, 3 und 8, wurden empirisch durch Probieren gefunden. Sie berücksichtigen außer den arithmetischen Operationen auch die vielen Speicherungen, Transporte und organisatorischen Befehle des Programms.

Mit den Programmen aus Tab. 2.31 wurden ebenfalls Zeitstudien angestellt. Zunächst läuft das Zahlenbeispiel aus Tab. 2.32, wenn nach Zeile 5 $n = 4$ eingegeben wird, mit den Daten aus Zeile 400 in 1.5 s. Man wird sich natürlich die Lösung durch eingeschobene PRINT-Befehle zeigen lassen. Wegen der für die Tests sehr umfangreichen Zahleneingaben oder Datenlisten wurden mit einem BASIC-Pseudozufalls-

zahlengenerator ganzzahlige Werte zwischen −50 und +50 erzeugt. Es ergaben sich mit der im Heimcomputer mitlaufenden internen Zeitmessung:

	Gauss-1	Gauss-2
$n = 2$	$t =$ 0.48 s	$t =$ 0.34 s
3	0.86	0.68
4	1.50	1.24
5	2.42	2.08
6	3.72	3.24
7	5.42	4.82
8	7.60	6.82
9	10.32	9.32
10	13.62	12.36
20	91.78 = 1 : 31.78	85.20 = 1 : 25.20
50	1312.88 = 21 : 52.88	1205.12 = 20 : 0.512

Damit wurde festgestellt, daß der Heimcomputer mit BASIC etwa 16mal schneller ist als der programmierbare Tischrechner mit dem Tastenfolgeprogramm, welches ja nahe der Maschinensprache ist. Ferner erweist sich Gauss-2 als ein wenig (etwa 10%) schneller als Gauss-1, was auf schnell in Maschinensprache ablaufende, für BASIC intern bereitstehende Adressenberechnungsfunktionen aus den Indexwerten zurückzuführen ist. In Gauss-1 mußten diese Funktionen teilweise in BASIC-Befehlen ausgedrückt werden. Offenbar ist Gauss-2 auch übersichtlicher als Gauss-1.

Der Heimcomputer bot 48 kByte an und nahm eine Aufgabe mit $n = 88$ noch an, während er $n = 89$ als zu groß ablehnte. Es geht daraus hervor, daß er 5 Byte/Zahl verwendet. Die damit mögliche hohe Genauigkeit in der Zahlendarstellung äußert sich in den kleinen Defekten, die sich beim Einsetzen der gefundenen Lösungen in die Ausgangsgleichungen zeigten:

$n = 10$	größter Defekt $= 10^{-8}$
20	10^{-6}
50	10^{-5}

Dabei lagen diese Defekte in denjenigen Gleichungen, deren rechte Seite Null sein sollte. Nur die notwendige Stellenauslöschung hob den Defekt in vordere Positionen. Heimcomputer, welche nur 4 Byte/Zahl einsetzen, schaffen bezüglich der Rechengenauigkeit oft schon den Fall $n = 10$ nicht mehr.

2.4.2. Mechanisierter Gauß-Algorithmus nach Banachiewicz

Führt man den Gauß-Algorithmus mit einem Taschenrechner ohne genügend großen Speicher durch, so hat man ständig die jeweiligen neuen Belegungen der Matrix und der rechten Seite zu notieren. Solch häufiges Kopieren ist besonders beim Rechnen mit vielen (sechs, acht oder zehn) Stellen lästig und fehleranfällig. Es wird

Tab. 2.32. Ausfüllen des Rechenfeldes (Arbeitsformular) beim Lösen eines linearen Gleichungssystems nach dem Algorithmus von GAUSS-BANACHIEWICZ

a)

A				b	Zeilensumme
3	4	−5	1	−35	−32
6	6	−13	−5	−67	−73
3	6	−1	10	−31	−13
9	8	−19	−2	−75	−79

b)

F				r	Zeilensumme
3	4	−5	1	−35	−32
2					
1					
3					

c)

F				r	Zeilensumme
3	4	−5	1	−35	−32
2	−2	−3	−7	3	−9
1	−1				
3	2				

d)

F				r	Zeilensumme
3	4	−5	1	−35	−32
2	−2	−3	−7	3	−9
1	−1	1	2	7	10
3	2	2			

e)

F				r	Zeilensumme
3	4	−5	1	−35	−32
2	−2	−3	−7	3	−9
1	−1	1	2	7	10
3	2	2	5	10	15

vermieden durch den *mechanisierten Gauß-Algorithmus* oder *Algorithmus von Gauß-Banachiewicz* (1938; T. BANACHIEWICZ, 1982—1954, polnischer Mathematiker; zur Benennung vgl. [72]).

Bei der Erzeugung der Dreiecksgestalt werden die Werte in den Zeilen erst gebildet, wenn man sie endgültig festlegen kann. Dazu müssen die entsprechenden Multiplikatoren der vorher erzeugten Zeilen gemerkt werden. Dies gelingt auf den Stellen der Matrix, die zu Null gemacht werden sollen. In Tab. 2.32 ist dies in einem einfachen Beispiel vorgeführt. Das Rechenfeld F zerfällt in einen linken (unteren) Teil L, der die Multiplikatoren der Zeilen enthält, und einen rechten (oberen) Teil R, in dem die Dreiecksform der Koeffizientenmatrix entsteht. Die Rechenformeln sind:

Erster Rechenschritt (Start)

Erste Zeile von (F, r) bzw. (R, r) wird übernommen:

$$F_{1j} = R_{1j} := A_{1j} \quad \text{für } j = 1(1)n, \tag{5}$$

$$r_1 := b_1 \quad \text{rechte Seite};$$

erste Spalte von F bzw. L enthält die Multiplikatoren

$$F_{i1} = L_{i1} := A_{i1}/R_{11} \quad \text{für } i = 2(1)n. \tag{6}$$

(Es ist zwar $R_{11} = A_{11}$, aber aus Gründen der Symmetrie zu den Formeln für die weiteren Schritte wurde R_{11} gewählt.)

Zweiter Rechenschritt

Zweite Zeile von (F, r) bzw. (R, r):

$$F_{2j} = R_{2j} := A_{2j} - L_{21}R_{1j} \quad \text{für } j = 2(1)n,$$

$$r_2 := b_2 - L_{21}r_1;$$

zweite Spalte von F bzw. L:

$$F_{i2} = L_{i2} := (A_{i2} - L_{i1}R_{12})/R_{22} \quad \text{für } i = 3(1)n.$$

Hier wurde beachtet, daß die Faktoren nicht einfach aus den A_{i2} gewonnen werden können, da diese Werte inzwischen durch das Bilden der hier nicht sichtbaren Nullen in der ersten Spalte modifiziert wurden. Die Abwandlung der Werte wird im Zähler des Quotienten nachgeholt.

k-ter (allgemeiner) Rechenschritt

k-te Zeile von (F, r) bzw. (R, r):

$$F_{kj} = R_{kj} := A_{kj} - \sum_{p=1}^{k-1} L_{kp}R_{pj},$$

$$r_k := b_k - \sum_{p=1}^{k-1} L_{kp}r_k \quad \text{für } j = k(1)n; \tag{7}$$

k-te Spalte von F bzw. L:

$$F_{ik} = L_{ik} := (A_{ik} - \sum_{p=1}^{k-1} L_{ip}R_{pk})/R_{kk} \quad \text{für } i = k + 1(1)n. \tag{8}$$

Dieser allgemeine Rechenschritt ist für die Zeilen (7) für $k = 2(1)n$ und für die Spalten (8) für $k = 2(1)n - 1$ durchzuführen.

Die Formeln gelten sogar für $k = 1$ und erfassen ebenfalls das Bilden der ersten Zeile und Spalte, da eine Summe mit kleinerer oberer als unterer Grenze einen leeren Wert (Null) ergibt. In Tab. 2.32 erkennt man die Reihenfolge, in der die Elemente von R und L gebildet werden. Wie sie entstehen müssen, verdeutlicht Abb. 2.25:

a) Die Skalarprodukte für die Elemente der k-ten Zeile von R entstehen aus den $k - 1$ Elementen der k-ten Zeile von L mit den jeweils $k - 1$ Elementen in den Spalten von R.

b) Die Skalarprodukte für die Elemente der k-ten Spalte von L entstehen aus den $k - 1$ Elementen der k-ten Spalte von R mit den jeweils $k - 1$ Elementen in den Zeilen von L. Dividiert wird durch das neue Diagonalelement, welches durch einen Kreis markiert ist.

(Die Angaben $A, B, C, D, AS, AZ, TS, TZ, TM, I, I1, I2$ sind konstante bzw. variable Adressen- oder Indexwerte und beziehen sich auf ein evtl. zu erzeugendes PTR- oder BASIC-Programm mit Vektorform der Matrix.)

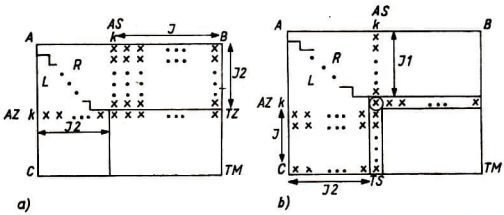


Abb. 2.25. Bilden der Elemente im Rechenfeld für das Verfahren nach GAUSS-BANACHIEWICZ

Die nötigen Skalarprodukte lassen sich mit einem Taschenrechner selbst mit nur einem Speicher recht bequem bilden. Nach der Subtraktion von A_{ij} (und eventueller Division durch R_{kk}) trägt man den Resultatwert in das Rechenfeld F ein.

In dem Beispiel nach Tab. 2.32a—e ist das Verfahren erläutert:

a) Ursprüngliche Aufgabe $Ax = b$.

b) Erster Rechenschritt im Rechenfeld: Die erste Zeile bleibt unverändert. Die erste Spalte enthält die Koeffizienten für die erste Zeile, mit denen Nullen in der ersten Spalte erzeugt werden können.

c) Zweiter Rechenschritt im Rechenfeld: Die zweite Zeile wird gebildet. Der Koeffizient für die erste Zeile ist bekannt (2). Die zweite Spalte (unter der Diagonalen) erhält danach die Koeffizienten für die zweite Zeile, mit denen Nullen in der zweiten Spalte erzeugt werden können. Dazu mußte aber der Multiplikator für die erste Zeile beachtet werden.

d) und e) Dritter und vierter Rechenschritt im Rechenfeld: entsprechen dem zweiten Schritt.

Tab. 2.33. Zwei Rechenabläufe für dasselbe Beispiel wie in Tab. 2.32. aber mit zeilenweiser Pivotisierung

a)

	A					Kontrolle
v:	1	2	3	4	b	K
	3	4	−5	1	−35	−32
	6	6	−13	−5	−67	−73
	3	6	−1	10	−31	−13
	9	8	−19	−2	−75	−79

b)

	F					
v:	3	2	1	4	r	K
	−5	4	3	1	−35	−32
	2.6	−4.4	−1.8	−7.6	24	
	0.2					
	3.8					

F					
		1.		r	K
3	4	−5	1	−35	−32
−1.8	−4.4	2.6	−7.6	24	
		0.2			
		3.8			

c)

v:	F 3	4	1	2	r	K
	−5	1	3	4	−35	−35
	2.6	−7.6	−1.8	−4.4	24	10.2
	0.2	−1.29	0.079	−0.047	6.95	6.55
	3.8	0.76				

F		1.	2.	r	K
3	4	−5	1	−35	−32
−1.8	−4.4	2.6	−7.6	24	10.2
0.079	−0.47	0.2	−1.29	6.95	6.55
		3.8	0.76		

d)

v:	F 3	4	2	1	r	K
	−5	1	4	3	−35	−32
	2.6	−7.6	−4.4	−1.8	24	10.2
	0.2	−1.29	−0.47	0.079	6.95	6.55
	3.8	−0.76	8.11	−1.67	16.76	−18.34

F				r	K
4.	3.	1.	2.		
−3	4	−5	1	−35	−32
−1.8	−4.4	2.6	−7.6	24	10.2
0.079	−0.47	0.2	−1.29	6.95	6.55
−1.67	8.11	3.8	−0.76	16.67	−18.34

Die Zeilensummen sind als Kontrolle (s. u.) mitgeführt worden. Die Determinante der Koeffizientenmatrix hat als Produkt der Diagonalelemente der Dreiecksmatrix im Beispiel den Wert -30.

Pivotisierung

Führt man eine Pivotisierung durch (aus der nächsten gerade errechneten Zeile), so kann man die Niederschrift eines neuen Rechenfeldes kaum vermeiden, es sei denn um den Preis eines immer unübersichtlicher werdenden Rechenablaufs, da die neuen Spalten nicht mehr der Reihe nach von links nach rechts zu bilden sind.

Die beiden in Tab. 2.33 dargestellten Rechenabläufe unterscheiden sich in der Behandlung der notwendigen Spaltenvertauschungen.

Links wurden die Spalten wirklich vertauscht, wodurch neue Niederschriften des Rechenfeldes F nötig wurden. Außerdem muß im Kopf von F die Lage der ursprünglichen Indizes vermerkt werden, damit die Nummern der Unbekannten korrekt festgestellt werden können.

Rechts werden die Kopien des Rechenfeldes vermieden. Dafür wird der Ablauf unübersichtlicher. Im Kopf ist vermerkt, in welcher Reihenfolge die Faktoren aus R zu entnehmen sind. In umgekehrter Reihenfolge ergeben sich bei der Rückrechnung die Unbekannten, hier also zuerst x_1.

Die Zeilensummen sind als Kontrolle mitgeführt worden. Die in der Tabelle angegebenen zwei- bis vierstelligen Werte stammen aus mehrstelliger Rechnung und sind gerundet.

Die Determinante der Koeffizientenmatrix hat als Produkt der Diagonalelemente der Dreiecksmatrix den Wert $30(-1)^3 = -30$, da drei Spaltenvertauschungen vorgenommen wurden.

Rechenkontrollen

Sobald man kein vollständiges Programm für die doch recht umfangreiche Rechen- und Schreibarbeit zur Verfügung hat, ist die Möglichkeit von Fehlern sehr groß. Das Durchführen von Kontrollen ist dringend zu empfehlen. Hier beim Verfahren von GAUSS-BANACHIEWICZ bietet sich eine *Zeilensummenkontrolle* an: Eine zusätzliche Spalte mit den Zeilensummen wird mitgeführt, und in jedem Schritt kann kontrolliert werden, ob die Werte die Eigenschaft der „Zeilensumme" noch bewahrt haben. Die Elemente dieser Spalte werden wie Elemente von R, d. h. nach den Formeln (7), behandelt. In Tab. 2.32 und 2.33 sind diese Kontrollspalten enthalten.

Rechenaufwand

Das manuelle Rechnen, auch mit Unterstützung eines Taschen- oder Tischrechners, zum Lösen eines linearen Gleichungssystems schon der Ordnung 5 ist eine beträchtliche Arbeit. Selbst geübte Arbeiter mit guter Konzentrationsfähigkeit werden etwa eine Stunde benötigen.

Bei Systemen der Ordnung 10 bis 15 wird oft ein ganzer Arbeitstag verbraucht, und bei der Ordnung 20 dürfte im allgemeinen die Grenze der Leistung eines aus-

führenden Menschen erreicht sein, denn dann reicht die sonst vertretbare Fehlerquote von 1‰ Fehltastungen aus, um den Rechenablauf einfach nicht zum Ende gelangen zu lassen. Diese Bemerkungen stützen sich auf Erfahrungen, die beispielsweise in den früher mit elektromechanischen Tischrechenmaschinen ausgerüsteten Rechenbüros des VEB Carl Zeiss Jena gesammelt wurden. Auch aus dieser Sicht ist also das Einführen von programmierbaren Tischgeräten notwendig.

Mehrere rechte Seiten

In der Praxis müssen öfter Gleichungssysteme mit zwar denselben Koeffizienten, aber mit unterschiedlichen rechten Seiten gelöst werden. Sind gleich zu Beginn die rechten Seiten — man wird sie als zusätzliche Spalten mitführen — alle bekannt, so können sie ohne Schwierigkeit in die Rechnung einbezogen werden, d. h., beim Prozeß der Herstellung der Dreiecksgestalt werden ebenfalls alle rechten Seiten bearbeitet. Lediglich die Rückrechnung ist dann noch mehrfach durchzuführen.

Werden zusätzliche rechte Seiten erst nachträglich bekannt, so kann man mit den Koeffizienten aus L in einfacher Weise die Vorbereitung der neuen rechten Seiten nachholen. Ist eine solche neue rechte Seite durch den Vektor b gegeben, so erhält man

$$r_k := b_k - \sum_{p=1}^{k-1} L_{kp} r_p \quad \text{für } k = (1)n, \tag{9}$$

und dann kann mit r und R die neue Rückrechnung durchgeführt werden.

2.4.3. Herstellung eines BASIC-Programms

Als Vorlage zur Herstellung des gewünschten BASIC-Programms wird ein ALGOL-Programm für das Herstellen der Matrizen L und R nach dem Verfahren von Gauss-Banachiewicz mit Gewinnung der Pivotelemente aus den Zeilen und Druck des Absolutwertes der Determinante angegeben. In vielen Fällen ist die Vorlage eines Programms in einer höheren Programmiersprache vorteilhafter als ein Programmablaufplan (PAP). Auch in der Praxis findet man als Vorlage eher ein Programm zur Algorithmusbeschreibung, beispielsweise aus der Literatur.

Gauß-Banachiewicz-Matrizen mit Pivots aus Zeilen:

```
Vorblock:
begin integer n; read (n);
Hauptblock:
begin array a[1 : n, 1 : n], v[1 : n];
      integer i, j, k; real f, g;
      read (a);
      for i := 1 step 1 until n do v[i] := i;
```

```
Matrizen L und R:
        for k := 1 step 1 until n − 1
        do
        begin
Pivot aus Zeile:
        f := abs (a[k, k]); g := k;
        for j := k + 1 step 1 until n
        do if f < abs(a[k, j])
           then begin f := abs(a[k, j]); g := j
                end Pivot aus Zeile;
Spaltentausch: if g ≠ k then
        begin j := v[k]; v[k] := v[g]; v[g] := j;
        for i := 1 step 1 until n
        do begin f := a[i, k];
                  a[i, k] := a[i, g]; a[i, g] := f
           end
        end Spaltentausch;
Spalte aus L:
        g := a[k, k];
        for j := k + 1 step 1 until n
        do begin f := a[j, k];
                  for i := 1 step 1 until k − 1
                  do f := f − a[j, i] × a[i, k]; a[j, k] := f/g
           end Spalte aus L;
Zeile aus R:
        for j := k + 1 step 1 until n
        do begin f := a[k + 1, j];
                  for i := 1 step 1 until k
                  do f := f − a[k + 1, i] × a[i, j]; a[k + 1, j] := f
           end Zeile aus R
        end Matrizen L und R;
Absolutwert der Determinante:
        f := 1;
        for k := 1 step 1 until n
        do f := f × a[k, k];
        print (f)
end Hauptblock
end Gauß-Banachiewicz Matrizen Vorblock
```

ALGOL-Programm für die Rückrechnungen zum Bestimmen der Werte für die Unbekannten (hier werden die bereits hergestellten Gauß-Banachiewicz-Matrizen L und R sowie der Vektor der Spaltenvertauschungen v und die Dimension n als globale Größen angenommen):

```
Rückrechnung Gauß-Banachiewicz und Druck:
begin array b[1 : n]; integer i, j, k; real f;
      read (b);
Vorbereitung rechte Seite:
      for i := 2 step 1 until n
      do begin f := b[i];
               for j := 1 step 1 until i − 1
               do f := f − b[j] × a[i, j];
                  b[i] := f
         end Vorbereitung rechte Seite;
eigentliche Rückrechnung:
      b[n] := b[n]/a[n, n];
      for i := n − 1 step −1 until 1
      do begin f := b[i];
               for j := i + 1 step 1 until n
               do f := f − a[i, j] × b[j];
               b[i] := f/a[i, i]
         end eigentliche Rückrechnung;
Beachtung der Spaltenvertauschung und Druck:
      for i := 1 step 1 until n
      do if i = v[i]
         then print (b[i])
         else begin for j := i, v[j] while i ≠ v[j]
                    do k := v[j];
                    print (b[k])
              end Druck
end Rückrechnung und Druck
```

Die Beachtung der Spaltenvertauschungen ist eine gute Übung im algorithmisch-dynamischen Denken. Beim Beispiel aus Tab. 2.33 ergab sich der Vertauschungsvektor

i	1	2	3	4
v_i	3	4	2	1

Es gilt also in keinem Fall $i = v_i$, und die Abarbeitung läuft in den ***else***-Zweig ein. Tab. 2.34 zeigt als Ablaufprotokoll die Werte der beteiligten Variablen. In b_k steht am Ende des Suchzyklus das verlangte x_i.

Tab. 2.34. Werteprotokoll für die am Suchprozeß für die Reihenfolge der Unbekannten beteiligten Variablen. Ausgangswerte für v siehe Rechenbeispiel Tab. 2.33 links, Programmteil siehe „Rückrechnung Gauß-Banachiewicz und Druck", Abschnitt „Beachtung der Spaltenvertauschung und Druck"

i	v_i	j	v_j	k
1	3	1	3	3
		3	2	2
		2	4	4
		4	1	
2	4	2	4	4
		4	1	1
		1	3	3
		3	2	
3	2	3	2	2
		2	4	4
		4	1	1
		1	3	
4	1	4	1	1
		1	3	3
		3	2	2
		2	4	

Inverse Matrix

Die „eigentliche Rückrechnung" für das System $Ax = b$, $x = A^{-1}b$ nach GAUSS-BANACHIEWICZ ist nichts anderes als die Auflösung des speziellen Dreieckssystems

$$Rx = r, \qquad x = R^{-1}r,$$

wobei r seinerseits aus dem gegebenen b durch

$$(E + L)\, r = b \qquad r = (E + L)^{-1}\, b$$

entstand. Dabei ist E die Einheitsmatrix, d. h., $(E + L)$ ist die mit Diagonaleinsen ergänzte linke Matrix L (vgl. Rechenbeispiel Tab. 2.33d).

Es gilt also

$$x = R^{-1}(E + L)^{-1}\, b$$

oder

$$A^{-1} = R^{-1}(E + L)^{-1} \tag{10}$$

und damit auch die interessante Zerlegung von A in ein Produkt:

$$A = (E + L) \cdot R. \tag{11}$$

Man kann demnach aus den L und R die inverse Matrix A^{-1} gewinnen. Am einfachsten gelingt das durch n-maliges Auflösen nach speziellen rechten Seiten, welche die Spalten der Einheitsmatrix E bilden. Die gefundenen Lösungen sind die Spalten der inversen Matrix A^{-1}, da dann

$$AA^{-1} = E \tag{12}$$

gilt. Das Programm dazu besteht lediglich noch aus dem Erzeugen der n rechten Seiten und dem Aufruf des Programmteils „Rückrechnung Gauß-Banachiewicz und Druck", und zwar mit Einsprung an der Stelle „Vorbereitung rechte Seite".

```
Inverse Matrix:
begin array b[1 : n]; integer i, j;
      for i := 1 step 1 until n
      do  begin for j := 1 step 1 until n
                  do b[j] := 0;
                  b[i] := 1;
                  goto Vorbereitung rechte Seite;
            rr:
          end
end Inverse Matrix
```

Für den nötigen Rücksprung muß der Programmteil der Rückrechnung am Ende modifiziert werden durch

```
                  ⋮
                  end druck;
          goto rr
end Rückrechnung und Druck
```

Es muß also ein Sprungbefehl eingefügt werden. (Natürlich darf bei echten ALGOL-Programmen nicht in Blöcke hineingesprungen werden. Die notierten Programme oder Programmteile dienen hier auch nur als Algorithmenbeschreibung und als Vorlage zur BASIC-Programmierung. Bei der Zusammenstellung eines echten ALGOL-Programms müssen die Sprachregeln beachtet werden. Beispielsweise wird man die Teile als Prozeduren umformen und dann aufrufen.)

Tab. 2.35. Inverse Matrix der Koeffizientenmatrix zum Zahlenbeispiel aus Tab. 2.32

a)
$$\begin{pmatrix} -16.20 & 6.73 & 4.87 & -0.60 \\ 4.80 & -1.60 & -1.30 & -0.10 \\ -5.80 & 2.60 & 1.80 & -0.40 \\ 1.40 & -0.80 & -0.40 & 0.20 \end{pmatrix}$$

b)
$$\frac{1}{30}\begin{pmatrix} -486 & 202 & 146 & -18 \\ 144 & -48 & -39 & -3 \\ -174 & 78 & 54 & -12 \\ 42 & -24 & -12 & 6 \end{pmatrix}$$

Tab. 2.35 enthält die errechneten Werte der Komponenten der inversen Matrix zum Zahlenbeispiel aus Tab. 2.32; sie zeigt

a) errechnete Werte, auf zwei Nachkommastellen gerundet;

b) exakte Werte; der Wert der Determinante ist −30, im Matrixfeld stehen also die negativen Adjunkten der Elemente der Koeffizientenmatrix; der Vergleich mit a) zeigt, daß nur bei a_{12}^{-1} und a_{13}^{-1} periodische Dezimalbrüche auftreten.

Durch analoge Überlegungen, wie sie zur Herstellung des Programms aus Tab. 2.31 durchgeführt wurden, kommt man — ausgehend von den ALGOL-Programm-Vorlagen — zu dem BASIC-Programm aus Tab. 2.36 für das Auflösen eines linearen Gleichungssystems nach GAUSS-BANACHIEWICZ mit Zeilenpivotisierung, evtl. mehreren rechten Seiten und Berechnung der inversen Matrix. Die bereits in Abb. 2.25 enthaltenen Bezeichnungen und einige andere, die man für ein PTR-Programm bzw. für eine BASIC-Version mit eindimensionalen Feldern braucht, haben folgende Bedeutung:

A	Anfangsadresse der Matrix oder Adresse von a_{11}
N	Anzahl der Gleichungen n
$TM = A(a_{11}) + n^2 - 1$	Matrixende, Adresse von a_{nn}
$D = A(n+1)\,TM$	Adressen der Diagonalelemente a_{kk}
$B = A(a_{11}) + n - 1$	Ende der ersten Zeile, Adresse von a_{1n}
$TZ = B(n)\,TM$	Zeilenenden, Adressen von a_{in}
$C = A(a_{11}) + n(n-1)$	Ende der ersten Spalte, Adresse von a_{n1}
$TS = C(1)TM$	Spaltenenden, Adressen von a_{nj}
$AS = A(1)B$	Spaltenanfänge, Adressen von a_{1j}
$AZ = A(n)C$	Zeilenanfänge, Adressen von a_{i1}
I	Indexwerte für Zeilen- oder Spaltenanfänge
I1, I2	Indexwerte für Zeilen- und Spaltenelemente

Das BASIC-Programm aus Tab. 2.36 zeigt zwar die relativ einfache Umsetzung aus einer Vorlage in einer höheren Programmiersprache, jedoch eben auch die Mängel dieser Sprache, wenn es sich um Verzweigungen mit ***then-else***-Elementen oder Zyklen mit ***while***- bzw. ***repeat***-Elementen (letztere gibt es in PASCAL) handelt. Man vergleiche dazu die Zeilen 680 bis 740.

Gibt man $n = 4$ ein (Zeile 20), so wird mit den Daten aus Zeile 900 die schon mehrfach gezeigte Lösung des Gleichungssystems geliefert. Diesmal wird die rechte Seite als letztes Wert-Quadrupel eingelesen.

Will man die inverse Matrix haben, so ändere man das Programm nach den Angaben unter dem Strich. Dadurch werden die Zeilen 515 bis 750 zu einer Subroutine.

Hat man schon eine Lösung des Systems errechnet, liegen also bereits die Gauß-Banachiewicz-Matrizen L und R vor, so startet man mit

```
GOTO 760
```

Hat man noch die ursprüngliche Matrix A vorliegen, so startet man mit

```
GOTO 105
```

Muß man auch A noch eingeben, so wird

```
RUN
```

richtig sein, aber dann muß das Datenfeld bei 900 entsprechend vorbereitet sein.

Tab. 2.36. BASIC-Programm zum Lösen linearer Gleichungssysteme nach Gauß-Banachiewicz mit Pivotisierung

```
  5 REM Gauss-Banachiewicz-Pivot:
 10 REM Vorblock:
 20 INPUT "n ="; n
 30 REM Hauptblock:
 40 DIM a(n, n) : DIM v(n)
 50 FOR i = 1 TO n
 60 LET v(i) = i
 70 FOR j = 1 TO n
 80 READ a(i, j)
 90 NEXT j
100 NEXT i
105 REM Matrizen L und R:
110 FOR k = 1 TO n - 1
120 REM Pivot aus Zeile:
130 LET f = ABS(a(k, k)) : LET g = k
140 FOR j = k + 1 TO n
150 IF f > = ABS(a(k, j)) THEN GOTO 170
160 LET f = ABS(a(k, j)) : LET g = j
170 NEXT j
180 REM Spaltentausch:
190 IF g = k THEN GOTO 240
200 LET j = v(k) : LET v(k) = v(g) : LET v(g) = j
210 FOR i = 1 TO n
220 LET f = a(i, k) : LET a(i, k) = a(i, g) : LET a(i, g) = f
230 NEXT i
240 REM Spalte aus L:
250 LET g = a(k, k)
260 FOR j = k + 1 TO n
270 LET f = a(j, k)
280 FOR i = 1 TO k - 1
290 LET f = f - a(j, i) * a(i, k)
300 NEXT i
310 LET a(j, k) = f/g
320 NEXT j
330 REM Zeile aus R:
340 FOR j = k + 1 TO n
350 LET f = a(k + 1, j)
360 FOR i = 1 TO k
370 LET f = f - a(k + 1, i) * a(i, j)
380 NEXT i
390 LET a(k + 1, j) = f
```

Tabelle 2.36. (Fortsetzung)

```
400  NEXT j
410  NEXT k
415  REM Absolutwert der Determinante:
420  LET f = 1
430  FOR k = 1 TO n
440  LET f = f * ABS(a(k, k))
450  NEXT k
460  PRINT „Det ="; f
470  REM Rueckrechnung und Druck:
480  DIM b(n)
490  FOR i = 1 TO n
500  READ b(i)
510  NEXT i
515  REM Vorbereitung rechte Seite:
520  FOR i = 2 TO n
530  LET f = b(i)
540  FOR j = 1 TO i - 1
550  LET f = f - b(j) * a(i, j)
560  NEXT j
570  LET b(i) = f
580  NEXT i
585  REM eigentliche Rueckrechnung:
590  LET b(n) = b(n)/a(n, n)
600  FOR i = n - 1 TO 1 STEP -1
610  LET f = b(i)
620  FOR j = i + 1 TO n
630  LET f = f - a(i, j) * b(j)
640  NEXT j
650  LET b(i) = f/a(i, i)
660  NEXT i
665  REM Beachtung der Spaltenvertauschung und Druck:
670  FOR i = 1 TO n
680  IF i<>v(i) THEN GOTO 700
690  PRINT b(i) : GOTO 740
700  LET j = i : LET k = v(j)
710  IF i = v(j) THEN GOTO 730
720  LET k = v(j) : LET j = v(j) : GOTO 710
730  PRINT b(k)
740  NEXT i
750  STOP
```

```
490
500
```

Tabelle 2.36. (Fortsetzung)

```
510  GOTO 760
515  REM Inverse Matrix:
750  RETURN
760  FOR p = 1 TO n
770  FOR q = 1 TO n
780  LET b(q) = 0
790  NEXT q
800  LET b(p) = 1
810  GOSUB 515
820  NEXT p
830  STOP
900  DATA    3, 4, -5, 1,
             6, 6, -13, -5,
             3, 6, -1, 10,
             9, 8, -19, -2,
            -35, -67, -31, -75
```

Das in Tab. 2.33 rechts gezeigte Verfahren des Nicht-Tauschens der Spalten spart den Aufwand der Zeilen 180 bis 230. Aber man muß natürlich in einem Vektor v die Reihenfolge der zu verarbeitenden Elemente notieren. Aus einem Matrixelement $a(i, j)$ in den Rechnungen wird dann $a(i, v(j))$.

Es muß noch vermerkt werden, daß die Anzahl der arithmetischen Operationen bei den Verfahren von Gauß und von Gauß-Banachiewicz gleich ist. Lediglich ihre Reihenfolgen sind anders. Das erlaubt bei Gauß-Banachiewicz in den Zeilen 290 und 370 den Einsatz einer nichtindizierten Hilfsgröße f. Dadurch wird mehrfach das Berechnen von Adressen aus Indizes vermieden, wodurch der Computer entlastet wird. Hierin liegt der Nutzen des Gauß-Banachiewicz-Verfahrens auch beim Computereinsatz, obwohl es ursprünglich für elektromechanische Tischrechner mit einem Speicherwerk erdacht wurde.

Das Herstellen der beim Gauß-Banachiewicz-Verfahren auftretenden Skalarprodukte gelingt mit einem elektronischen Rechner mit Hierarchie, z. B. SR1, einfach durch die Tastenfolge

$$a_1 \times b_1 + a_2 \times b_2 + \cdots + a_n \times b_n =$$

und, wenn keine Hierarchie vorliegt, z. B. MR 610, durch

$$(a_1 \times b_1) + (a_2 \times b_2) + \cdots + (a_n \times b_n) =$$

oder mit Verwendung eines Speichers

$$a_1 \times b_1 = \mathrm{M} + \mathrm{CL} \qquad a_2 \times b_2 = \mathrm{M} + \mathrm{CL} \qquad \ldots \qquad a_n \times b_n = +\mathrm{MR} =$$

2.4.4. Das Austauschverfahren

Einen anderen Lösungsgedanken, der sich aber als verwandt mit dem aus dem Gauß-Algorithmus erweist, verfolgt das Austauschverfahren. Es wurde schon von C. JORDAN (1832–1922) beschrieben und von E. STIEFEL 1960 näher untersucht sowie der praktischen Anwendung unter Einsatz von Computern wieder erschlossen.

Durch die Verwendung und Benutzung beim Aufgabenlösen in der linearen Optimierung (Simplexverfahren) erlangt das Austauschverfahren eine besondere Bedeutung (MfL Bd. 10, 7.4.).

Anstelle der Aufgabengleichung

$$Ax = -r \tag{13}$$

wird nun

$$y = Ax + r \tag{14}$$

betrachtet. (Damit hier ein „+"-Zeichen geschrieben werden kann, mußte die rechte Seite r in der Aufgabengleichung (13) negativ eingetragen werden.) Es werden nun fortlaufend gewisse Komponenten von y gegen Komponenten von x ausgetauscht. Dieses geschieht durch Auflösen einer der Gleichungen nach der x-Komponente und Substitution dieser Auflösung in die anderen Gleichungen.

Diese Grundidee findet auch im Schulunterricht bei der *Substitutionsmethode* zum Lösen linearer Gleichungssysteme Anwendung:

$$3x_1 - 4x_2 = 1,$$
$$2x_1 + 6x_2 = 18.$$

Aus der zweiten Gleichung erhält man

$$x_1 = 9 - 3x_2,$$

und dies, in die erste eingesetzt (substituiert), liefert

$$27 - 9x_2 - 4x_2 = 1,$$
$$13x_2 = 26,$$

also eine Gleichung mit einer Unbekannten. Die Lösung lautet in diesem Beispiel

$$x_2 = 2 \quad \text{und} \quad x_1 = 3.$$

Im hier betrachteten allgemeineren Fall entsteht aus (14) die Gleichung

$$x = By + s, \tag{15}$$

und für $y = 0$ (Nullvektor) wäre

$$x = s \tag{16}$$

die Lösung der ursprünglich gestellten Aufgabe (13):

$$0 = Ax + r. \tag{13'}$$

Ist dagegen $r = 0$, so erkennt man beim Austauschverfahren aus $y = Ax$ mit der Lösung $x = By$ die inverse Matrix

$$B = A^{-1} \tag{17}$$

und kann mit ihr durch eine einfache Matrix-Vektor-Multiplikation für beliebiges y, d. h. für beliebige rechte Seite eines linearen Gleichungssystems, die Lösung erhalten.

Beschreibung des Austauschverfahrens

Am einfachsten ist es, die y_i genau gegen die x_i auszutauschen, d. h. gegen die gleichnumerierten Komponenten. Um aber Null-Koeffizienten in der Matrix vorzubeugen oder bei sehr kleinen Koeffizienten unerwünschten numerischen Effekten zu entgehen, wird als *Austausch-* oder *Pivotelement* immer dasjenige a_{pq} genommen, welches absolut am größten ist. Es wird dann y_p gegen x_q getauscht. Rechentechnisch sollen die Elemente von $B = A^{-1}$ auf dem Platz von A und die Elemente von s auf dem Platz von r entstehen. Außerdem soll r als $(n+1)$-te Spalte von A zählen.

Schema vor dem Austausch:

Austauschspalte q

$$\begin{array}{ll} & \vdots \quad\quad \vdots \quad\quad \vdots \quad \vdots \\ \text{Austausch-} & \cdots + a_{pq}x_q + \cdots + a_{pj}x_j + \cdots + r_p = y_p \\ \text{zeile } p & \vdots \quad\quad \vdots \quad\quad \vdots \quad \vdots \\ & \cdots + a_{iq}x_q + \cdots + a_{ij}x_j + \cdots + r_i = y_i \\ & \vdots \quad\quad \vdots \quad\quad \vdots \quad \vdots \end{array} \tag{18}$$

Die Auflösung nach x_q in der Austauschzeile p ergibt

$$\cdots + \frac{1}{a_{pq}}\, y_p - \cdots - \frac{a_{pj}}{a_{pq}}\, x_j - \cdots - \frac{r_p}{a_{pq}} = x_q, \tag{19}$$

und dies, in den anderen Zeilen eingesetzt, führt auf das

Schema nach dem Austausch:

Austauschspalte q

$$\begin{array}{ll} & \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ \text{Austausch-} & \cdots + \dfrac{1}{a_{pq}}\, y_p - \cdots - \dfrac{a_{pj}}{a_{pq}}\, x_j \quad\quad - \cdots - \dfrac{r_p}{a_{pq}} \quad\quad = x_q \\ \text{zeile } p & \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ & \cdots + \dfrac{a_{iq}}{a_{pq}}\, y_p + \cdots + \dfrac{(a_{ij} - a_{pj}a_{iq})}{a_{pq}}\, x_j + \cdots + \left(r_i - \dfrac{r_p a_{iq}}{a_{pq}}\right) = y_i \\ & \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \end{array} \tag{20}$$

Dies liefert die folgende Rechenvorschrift in vier Schritten für einen Austausch:

Schritt 0: Austauschelement (Pivot) als absolut größtes Element aus A unter den Elementen suchen, die in noch nicht als Austauschzeilen und Austauschspalten benutzten Zeilen und Spalten stehen. Dies sei a_{pq}, dann wird y_p gegen x_q getauscht.

Schritt 1: Das Austauschelement erhält den neuen Wert

$$\text{pivot} := a_{pq} := 1/a_{pq}, \tag{21}$$

d. h. Kehrwert des Elements.

Schritt 2: Behandlung der Austauschzeile p

$$K_j := a_{pj} := -a_{pj} \times \text{pivot} \quad \text{für } j = 1(1)n+1, \quad \text{aber } j \neq q. \tag{22}$$

Schritt 3: Behandlung der anderen Elemente aus A und r

$$a_{ij} := a_{ij} + K_j \times a_{iq} \quad \text{und } j = 1(1)n+1,\ j \neq q, \quad \text{für } i = 1(1)n,\ i \neq p. \tag{23}$$

(Man beachte in diesem Schritt, daß a_{pj} bereits den neuen Wert hat. Man wird rechentechnisch diesen Schritt zeilenweise, d. h. für laufendes j, abarbeiten und kann dann im übergeordneten i-Zyklus jeweils ein Element der Austauschspalte (Schritt 4) nach Verarbeitung einer Zeile ändern.)

Schritt 4: Behandlung der Austauschspalte q

$$a_{iq} := a_{iq} \times \text{pivot} \quad \text{für } i = 1(1)n, \quad \text{aber } i \neq p. \tag{24}$$

Wenn man diese Vertauschung n-mal durchgeführt hat, steht in der letzten Spalte $(n+1)$ von A die Lösung s, und auf den restlichen Plätzen stehen die Elemente von A^{-1}. Man muß aber über die Vertauschungen Buch führen, um die richtige Numerierung von Zeilen und Spalten nicht zu verlieren.

Aus den Formeln (21) bis (24) erkennt man die Beziehung zwischen Gauß- und Austauschalgorithmus. Das Eliminieren durch (Auflösen und) Substituieren wird bei Gauß nur in noch nicht behandelten Zeilen durchgeführt, wodurch sich die bekannte Dreiecksgestalt ergibt und eine Rückrechnung zum Gewinn der Werte der Unbekannten nötig wird. Beim Austauschverfahren wird jeweils die gesamte Matrix behandelt und somit zwar sofort, jedoch in Teile zerlegt, die Gauß-Rückrechnung mit einbezogen. Bei Handrechnungen, d. h. Benutzung eines Taschen- oder Tischrechners mit häufigem Eintasten von Zahlen und Notieren von Zwischenresultaten, empfiehlt sich das Mitführen einer Probespalte mit dem Wert $1 - S$, wobei S die Zeilensumme ist. Diese Eigenschaft muß bei jedem Vertauschungsschritt

erhalten bleiben. Die Vorschrift für S lautet in der Pivotzeile

$$S_p := (1 - S_p)/a_{pq} + 1, \quad \text{also} \quad (1 - S_p) := -(1 - S_p)/a_{pq} \tag{25}$$

und in den anderen Zeilen $i = 1(1)n,\ i \neq p$

$$S_i := S_i + a_{iq} \times (S_p - 1), \quad \text{also} \quad (1 - S_i) := (1 - S_i) + a_{iq} \times (1 - S_p). \tag{26}$$

Ferner ist es praktisch, bei Handrechnungen die Austauschzeile oder Pivotzeile nach deren Neufestlegung in das alte Schema (vor dem Austausch) als sogenannte *Kellerzeile* K zu kopieren, natürlich ohne das Element a_{pq} in der Pivotspalte. Auf diese Weise hat man sie beim Berechnen der neuen Elemente ständig vor Augen. Im Schritt 2 oder (22) wurde dies mit dem Vektor K bereits demonstriert. Hier liegt sogar ein Ansatz zur Effektivitätssteigerung (zeitlich) beim Einsatz von Computern. Im Schritt 3 oder (23) braucht dann nämlich mit K_j anstelle von a_{pj} nur eine Größe mit einem Index aufgerufen zu werden, was schneller realisiert werden kann.

Beispiel

$$\begin{aligned} x_2 + 2x_3 &= 8, \\ x_1 \quad + x_3 &= 4, \\ x_1 + x_2 \quad &= 3. \end{aligned}$$

Schema 1:

	x_1	x_2	x_3	r	$1 - S$
y_1	0	1	(2)	−8	6
y_2	1	0	1	−4	3
y_3	1	1	0	−3	2
K	0	−1/2	*	4	−3

Pivot ist $a_{13} = 2$. Es wird also y_1 gegen x_3 getauscht.

Schema 2:

	x_1	x_2	y_1	r	$1 - S$	
x_3	0	−1/2	1/2	4	−3	← Diese Zeile kommt in K von Schema 1.
y_2	(1)	−1/2	1/2	0	0	
y_3	1	1	0	−3	2	
K	*	1/2	−1/2	0	0	

Das neue Pivotelement muß aus der linken unteren Ecke gesucht werden. Man hat die Auswahl unter drei Einsen. Es wird $a_{21} = 1$ gewählt. Es wird also y_2 gegen x_1 getauscht.

Schema 3:

	y_2	x_2	y_1	r	$1-S$	
x_3	0	−1/2	1/2	4	−3	
x_1	1	1/2	−1/2	0	0	← Diese Zeile kommt
y_3	1	(3/2)	−1/2	−3	2	in K von Schema 2.
K	−2/3	*	1/3	2	−4/3	

Nun bleibt $a_{22} = 3/2$ als Pivot übrig. Es muß noch y_3 gegen x_2 getauscht werden.

Schema 4:

	y_2	y_3	y_1	r	$1-S$	
x_3	1/3	−1/3	1/3	3	−1/3	
x_1	2/3	1/3	−1/3	1	−2/3	
x_2	−2/3	2/3	1/3	2	−4/3	← Diese Zeile kommt in K von Schema 3.

Es ist also

$$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{und} \quad A^{-1} = \frac{1}{3}\begin{pmatrix} -1 & 2 & 1 \\ 1 & -2 & 2 \\ 1 & 1 & -1 \end{pmatrix}.$$

Als Vorlage und Dokumentation einer programmierungstechnischen Realisierung des Austauschverfahrens wird ein ALGOL-Programm angegeben. Das Zur-Verfügung-Halten des Restfeldes, Merken der Indizes bei Vertauschungen und schließlich das Ordnen der inversen Matrix sind nicht einfach zu überschauen. Man verfolge den Ablauf des Programms.

Bemerkung zur Pivotsuche

Bei der Suche nach dem Pivotelement müssen alle bereits verwendeten Austausch- oder Pivotreihen (Spalten und Zeilen) übergangen werden. Dazu verwendet man zwei Indexvektoren *ri* und *rj* für die Angabe der Zeilen- und Spaltennummern im Restfeld. Anfangs stehen dort die Indizes 1 bis n. Bei jedem Pivot $a_{I,J}$ wird in *ri* das I-te Element und in *rj* das J-te Element gestrichen. Dies geschieht einfach durch Nachziehen der darauffolgenden Elemente. Die Pivotsuche erfolgt dann mit Hilfe der Indexvektoren *ri* und *rj*, in denen die Nummern der noch verbliebenen Reihen stehen.

Herstellung der korrekten Anordnung

Während des Austauschens führt man zwei Indexvektoren p und q mit, in denen die Tauschindexpaare vermerkt werden. Ist das Pivotelement $a_{I,J}$, so wird im Schritt $r = n(-1)1$ das p_r gleich I und das q_r gleich J gesetzt. Dabei ist aber zu beachten, daß diese Indizes aus ri_I und rj_J zu entnehmen sind.

Im Beispiel gelten zum Schluß die Vektoren $p = (3, 2, 1)$ und $q = (2, 1, 3)$ für den Spalten- bzw. Zeilentausch. Praktisch wird dieser Tausch über einen Hilfsvektor

vorgenommen, in welchen die Matrixreihen ausgelagert werden, um dann in der richtigen Reihenfolge zurückgespeichert zu werden. Beim Spaltentausch z. B. lagert man die Zeilen der Reihe nach entsprechend p und q aus und speichert sie dann zurück (siehe Beispiel unten nach dem Programm).

ALGOL-Programm

```
procedure Austausch (A, n); value n; integer n; array A;
comment Mit dem Austauschverfahren wird die Matrix A auf ihrem Platz invertiert. Es entsteht also die Kehrmatrix, wobei vorausgesetzt wird, daß die Determinante von A nicht verschwindet;
begin array ri, rj, p, q, K[1:n];
integer i, j, r, I, J, P, Q; real max, pivot, v;
comment Die Felder ri und rj merken die Zeilen- und Spaltennummern für das Restfeld bei der Maximum- oder Pivotsuche. Die Felder p und q merken die Indizes der Pivot- oder Austauschelemente für das spätere Ordnen;

Anfang: for i := 1 step 1 until n
        do ri[i] := rj[i] := i;

Austauschzyklus:
        for r := n step -1 until 1
        do
begin

Maxsuche: max := 0;
for i := 1 step 1 until r
do for j := 1 step 1 until r
   do
   begin
     v := abs(A[ri[i], rj[j]]);
     if max < v
     then begin
            max := v;
            P := p[r] := ri[i]; I := i;
            Q := q[r] := rj[j]; J := j
          end
   end Maxsuche, Austauschelement gefunden;

Indizesrestfeld:
        for i := I step 1 until r - 1
        do ri[i] := ri[i + 1];
        for i := J step 1 until r - 1
        do rj[i] := rj[i + 1];
```

```
Austausch:
Austauschelement:
        pivot := A[P, Q] := 1/A[P, Q];
Austauschzeile:
        for j := 1 step 1 until n
        do if j ≠ Q
            then K[j] := A[P, j] := −A[P, j] × pivot;
Restfeld:
        for i := 1 step 1 until n
            do if i ≠ P
            then
            begin v := A[i, Q]; for j := 1 step 1 until n
                                        do if j ≠ Q
              then A[i, j] := A[i, j] + K[j] × v;
              A[i, Q] := v × pivot
            end Restfeld und Austausch
end Austauschzyklus;
Ordnen:
Zeilen:   for j := 1 step 1 until n
          do
            begin
              for r := n step −1 until 1
              do K[q[r]] := A[p[r], j];
              for i := 1 step 1 until n
              do A[i, j] := K[i]
            end Zeilenordnen;
Spalten:  for i := 1 step 1 until n
          do
            begin
              for r := n step −1 until 1
              do K[p[r]] := A[i, q[r]];
              for j := 1 step 1 until n
              do A[i, j] := K[j]
            end Spaltenordnen
end Austauschprozedur
```

Zur Verfolgung des Programmablaufs und zur Erläuterung wird das bereits verwendete Beispiel wieder benutzt (das zugehörige BASIC-Programm, dessen Korrespondenz zur ALGOL-Programmiervorlage man leicht sieht, ist in Tab. 2.37 enthalten).

Anfangszustand:

$$A = \begin{pmatrix} 0 & 1 & 2 & -8 \\ 1 & 0 & 1 & -4 \\ 1 & 1 & 0 & -3 \end{pmatrix}$$

i	1	2	3
ri	1	2	3
rj	1	2	3

1. Zyklus: $r = 3$

$$\mathrm{K} = \begin{pmatrix} 0 & -\frac{1}{2} & * & 4 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & -\frac{1}{2} & \frac{1}{2} & 4 \\ 1 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 1 & 1 & 0 & -3 \end{pmatrix}$$

i	1	2	3	
ri	1	2	3	
rj	1	2	3	
p			1	$I = 1$
q			3	$J = 3$
ri	2	3	3	pivot $= \frac{1}{2}$
rj	1	2	3	

2. Zyklus: $r = 2$

$$\mathrm{K} = \begin{pmatrix} * & -\frac{1}{2} & \frac{1}{2} & 4 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & -\frac{1}{2} & \frac{1}{2} & 4 \\ 1 & \frac{1}{2} & -\frac{1}{2} & 0 \\ 1 & \frac{3}{2} & -\frac{1}{2} & -3 \end{pmatrix}$$

i	1	2	3	
ri	2	3	3	
rj	1	2	3	
p		2	1	$I = 2$
q		1	3	$J = 1$
ri	3	3	3	pivot $= 1$
rj	2	2	3	

3. Zyklus: $r = 1$

$$\mathrm{K} = \begin{pmatrix} -\frac{2}{3} & * & \frac{1}{3} & 2 \end{pmatrix}$$

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

i	1	2	3	
ri	3	3	3	
rj	2	2	3	
p	3	2	1	$I = 3$
q	2	1	3	$J = 2$
ri	3	3	3	pivot $= \frac{3}{2}$
rj	2	2	3	

Ende der Austauschzyklen, da $r = 0$.

Tab. 2.37. BASIC-Programm zum Austauschverfahren

```
10  REM "Austauschverfahren"
20  INPUT "n ="; n
30  DIM A(n, n)
40  FOR i = 1 TO n
50  FOR j = 1 TO n
60  READ A(i, j)
70  NEXT j
80  NEXT i
90  DIM s(n): DIM t(n): DIM p(n): DIM q(n): DIM k(n)
100 REM *** Anfang ***
110 FOR i = 1 TO n
120 LET s(i) = i: LET t(i) = i
130 NEXT i
140 REM *** Austauschzyklus ***
150 FOR r = n TO 1 STEP -1
160 REM *** Maxsuche ***
170 LET max = 0
180 FOR i = 1 TO r
190 FOR j = 1 TO r
200 LET v = ABS(A(s(i), t(j)))
210 IF max > = v THEN GOTO 250
220 LET max = v: LET gi = i: LET gj = j
230 LET p(r) = s(i): LET q(r) = t(j)
240 LET gp = p(r): LET gq = q(r)
250 NEXT j
260 NEXT i
270 REM *** Ende Maxsuche ***
280 REM *** Indizesrestfeld ***
290 FOR i = gi TO r - 1
300 LET s(i) = s(i + 1)
310 NEXT i
320 FOR i = gj TO r - 1
330 LET t(i) = t(i + 1)
340 NEXT i
350 REM *** Austausch ***
360 REM *** Austauschelement ***
370 LET v = 1/A(gp, gq): Let pivot = v: LET A(gp, gq) = v
380 REM *** Austauschzeile ***
390 FOR j = 1 TO n
400 IF j = gq THEN GOTO 420
```

Tabelle 2.37. (Fortsetzung)

```
410  LET v = -A(gp, j) * pivot: LET A(gp, j) = v: LET k(j) = v
420  NEXT j
430  REM *** Restfeld ***
440  FOR i = 1 TO n
450  IF i = gp THEN GOTO 520
460  LET v = A(i, gq)
470  FOR j = 1 TO n
480  IF j = gq THEN GOTO 500
490  LET A(i, j) = A(i, j) + k(j) * v
500  NEXT j
510  LET A(i, gq) = v * pivot
520  NEXT i
530  REM *** Ende Restfeld und Austausch ***
540  NEXT r
550  REM *** Ende Austauschzyklus ***
560  REM *** Ordnen ***
570  REM *** Zeilen ***
580  FOR j = 1 TO n
590  FOR r = n TO 1 STEP -1
600  LET k(q(r)) = A(p(r), j)
610  NEXT r
620  FOR i = 1 TO n
630  LET A(i, j) = k(i)
640  NEXT i
650  NEXT j
660  REM *** Ende Zeilenordnen ***
670  REM *** Spalten ***
680  FOR i = 1 TO n
690  FOR r = n TO 1 STEP -1
700  LET k(p(r)) = A(i, q(r))
710  NEXT r
720  FOR j = 1 TO n
730  LET A(i, j) = k(j)
740  NEXT j
750  NEXT i
760  REM *** Ende Spaltenordnen ***
770  REM *** Ende Ordnen ***
780  REM *** Ende Austauschverfahren ***
790  STOP
800  DATA 0,1,2,1,0,1,1,1,0
```

Es folgt nun das Ordnen zur inversen Matrix mit Hilfe der Vektoren p und q. Als Zwischenspeicher wird K benutzt. Beim Zeilenordnen entsteht z. B. bei Spalte 2 ($j = 2$)

$$K = \left(\frac{1}{2} \quad \frac{2}{3} \quad -\frac{1}{3}\right).$$

Erweiterung des Austauschverfahrens

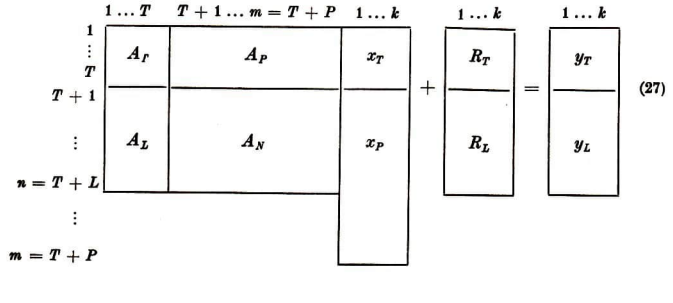
Das Austauschverfahren kann, wie auch der Gauß-Algorithmus, allgemein auf lineare Gleichungssysteme vom Typ $n \times m$ angewendet werden. Hat man mehr Gleichungen als Unbekannte, so werden sich spätestens nach m Schritten Nullzeilen in der Restmatrix (abgesehen von Rundungsfehlern) ergeben, so daß man kein Austauschelement mehr bestimmen kann. Das System ist lösbar, wenn dann auch die rechten Seiten in den Nullzeilen Nullen zeigen, anderenfalls nicht. Hat man mehr Unbekannte als Gleichungen, so können höchstens n Austauschzyklen ausgeführt werden. Die verbleibenden Unbekannten stellen dann für die Lösung frei wählbare Parameter dar.

Hat das Gleichungssystem einen Rangabfall, so treten die eben beschriebenen Verhältnisse eher ein. Das Austauschverfahren ist demnach für beliebige lineare Gleichungssysteme günstig einsetzbar. Man bemerkt den Rangabfall, wenn vor Erreichen des n kein Pivot $\neq 0$ mehr gefunden wird. Der Rang der Koeffizientenmatrix ist gleich der Anzahl der ausführbaren Austauschschritte, und diese Anzahl ist natürlich identisch mit der größten Zeilenzahl einer von Null verschiedenen Unterdeterminante. In der linearen Algebra wird der Rang einer Matrix durch diese größtmögliche Zeilenzahl definiert. Hier ergibt sich, wie auch schon beim Gauß-Algorithmus, die Möglichkeit, ihn real zu bestimmen.

Bei beliebiger Aufgabenstellung

$$y = Ax + R \tag{14'}$$

mit n Gleichungen, m Unbekannten und k rechten Seiten, wobei T Austauschschritte (Rang $(A) = T$) möglich sein sollen, ergibt sich folgende detaillierte Darstellung:



(27)

Dabei wurde angenommen, daß die austauschfähigen x- und y-Komponenten in der Numerierung vorne liegen, daß die Pivots also immer in der Hauptdiagonale stehen.

Durch Ausführen der T Austauschschritte entsteht daraus

$$\begin{pmatrix} A'_T & A'_P \\ A'_L & A'_N \end{pmatrix} \begin{pmatrix} y_T \\ x_P \end{pmatrix} + \begin{pmatrix} R'_T \\ R'_L \end{pmatrix} = \begin{pmatrix} x_T \\ y_L \end{pmatrix}, \tag{28}$$

und man muß sich Klarheit über den Aufbau der mit einem Strich versehenen, durch den Rechenvorgang veränderten Größen verschaffen. Zunächst ist

$$A'_N = 0, \tag{29}$$

da in dieser Restmatrix kein Pivot oder Austauschelement mehr gefunden werden konnte. Es sind somit L Zeilen von A linear abhängig. Weiter gilt

$$A'_T = A_T^{-1}, \tag{30}$$

was man aus dem Spezialfall $P = 0$, $L = 0$ und $R = 0$ ablesen kann, denn dann heißt die ursprüngliche Aufgabe $y_T = A_T x_T$, und es entsteht durch das Austauschen $x_T = A'_T y_T = A_T^{-1} y_T$. Wenn man dies jetzt weiß, erkennt man

$$R'_T = -A_T^{-1} R_T \tag{31}$$

aus dem Spezialfall $P = 0$, $L = 0$ und $y = 0$, denn dann hieß die Aufgabe

$$0 = A_T x_T + R_T,$$

was einerseits $0 = x_T + A_T^{-1} R_T$ und andererseits $R'_T = x_T$ ergibt. Die Teilmatrix A'_P wurde beim Austauschverfahren ebenso wie die rechten Seiten R_T behandelt. Daraus folgt

$$A'_P = -A_T^{-1} A_P. \tag{32}$$

Sie enthält die Koeffizienten der frei wählbaren x_P (Parameter) für die P-dimensionale Lösungsmannigfaltigkeit aus dem m-dimensionalen x-Raum.

Die Lösungsmannigfaltigkeit beschreibt einen P-dimensionalen Unterraum (anschaulich z. B. eine Ebene oder eine Gerade im dreidimensionalen Raum). Aus dem Spezialfall $L = 0$ und $y = 0$ erkennt man nämlich, daß aus

$$0 = (A_T A_P) \begin{pmatrix} x_T \\ x_P \end{pmatrix} + R_T$$

die allgemeine Lösung

$$x_T = R'_T + A'_P x_P = A_T^{-1} R_T + A_T^{-1} A_P x_P \tag{33}$$

entsteht. Was aber sind A'_L und R'_L? Speziell für $R = 0$ erhält man nach dem Austauschen für zunächst beliebiges y

$$A'_L y_T = y_L$$

in den unteren L Zeilen. Die y_L müssen also eine Linearkombination der T anderen sein, d. h., sie sind nicht ganz beliebig. Daraus folgt, daß A'_L in den Zeilen die Koeffizienten der Zeilen-Linearkombination enthält, welche die linearen Abhängigkeiten der $(A_L A_N)$-Teilmatrix beschreiben.

Es gilt also

$$A'_L(A_T A_P) = -(A_L A_N),$$

und aus dem Anteil $A'_L A_T = -A_L$ erhält man

$$A'_L = -A_L A_T^{-1}. \tag{34}$$

Die lineare Abhängigkeit zeigt sich darin, daß auch für den Anteil A'_L die Beziehung

$$A'_L A_P = -A_N$$

erfüllt ist. Für R_L gilt nun, was bereits für y_L gesagt wurde. Es muß $A'_L R_T = R_L$ sein bzw. beim Austausch muß sich

$$R'_L = 0 \tag{35}$$

wegen

$$A'_L y_T + R'_L = y_L$$

ergeben. Ist diese Bedingung nicht erfüllt, d. h., ist $R'_L \neq 0$, so ist das System (für die betreffende Spalte aus R als rechte Seite) nicht lösbar. Damit ist aus (27) nun durch das Austauschen das folgende Schema entstanden:

	$1 \dots T$	$T+1 \dots m = T+P$	$1 \dots k$		$1 \dots k$		$1 \dots k$
1							
$\vdots$	A_T^{-1}	$-A_T^{-1} A_P$	y_T	+	$-A_T^{-1} R_T$	=	x_T
T							
$T+1$							
$\vdots$	$-A_L A_T^{-1}$	0	x_P		0		y_L
$n = T+L$							
$\vdots$							
$m = T+P$							

(36)

Liegen konkret nicht so übersichtlich geordnete Verhältnisse vor, d. h., befindet sich A_T nicht in der linken oberen Ecke von A, so entstehen bei der Rechnung trotzdem dieselben Komponentenwerte, obwohl sie sich durch Zeilen- und Spaltentausch an anderen Plätzen befinden.

Beispiel

Im Beispiel wurde zur Wahrung einer übersichtlichen Gestalt das Pivot- oder Austauschelement stets aus dem nächsten Diagonalelement entnommen. Die Aufgabe

mit nur einer Spalte in R lautet

3	4	−5	1	3	−1	−2	x_1	35	y_1
6	6	−13	−5	2	−2	1	x_2	67	y_2
3	6	−1	10	−1	1	−2	x_3	31	y_3
9	8	−19	−2	−1	2	4	x_4	75	y_4
−3	−4	2	−12	7	−6	−3	x_5	−4	y_5
−3	6	20	29	3	0	−13	x_6	−10	y_6
							x_7		

Obwohl zu Beginn der Bearbeitung die Anzahl der Austauschmöglichkeiten (hier $T = 4$, also $\mathrm{Rang}(A) = 4$) noch nicht bekannt sein muß, sind hier die Grenzen der Teilmatrizen schon markiert worden. Das Austauschverfahren liefert nun

−16.2	6.7$\overline{3}$	4.8$\overline{6}$	−0.6	+39.4	−6.4	−27	y_1	10	x_1
4.8	−1.6	−1.3	−0.1	−12.6	+3.1	+9	y_2	−13	x_2
−5.8	2.6	1.8	−0.4	+13.6	−1.6	−9	y_3	3	x_3
1.4	−0.8	−0.4	0.2	−2.8	−0.2	+2	y_4	2	x_4
−1	−1	1	1	0	0	0	x_5	0	y_5
−2	1	2	1	0	0	0	x_6	0	y_6
							x_7		

Für $y = 0$ erhält man die dreiparametrige Lösungsmannigfaltigkeit

$$
\begin{aligned}
x_1 &= 10 + 39.4x_5 - 6.4x_6 - 27x_7,\\
x_2 &= -13 - 12.6x_5 + 3.1x_6 + 9x_7,\\
x_3 &= 3 + 13.6x_5 - 1.6x_6 - 9x_7,\\
x_4 &= 2 - 2.8x_5 - 0.2x_6 + 2x_7,
\end{aligned}
$$

x_5 beliebig, x_6 beliebig, x_7 beliebig.

Beispielsweise wird für $x_5 = x_6 = x_7 = 1$ die spezielle Lösung für $y = 0$

$$x_1 = 16, \quad x_2 = -13.5, \quad x_3 = 6, \quad x_4 = 1,$$

gewonnen. Man überzeuge sich, daß dieser Lösungsvektor das System wirklich befriedigt, natürlich einschließlich der linear abhängigen Gleichungen.

2.4.5. Inverse Matrix und verallgemeinerte inverse Matrix

Mit dem Gauß-Algorithmus und dem Austauschverfahren sind zwei Verfahren zur Berechnung der inversen Matrix beschrieben worden. Man kann die Inverse A^{-1} zu A berechnen, wenn die Determinante $\det(A) \neq 0$ ist. Es gilt dann im Reellen

$$A^{-1}A = AA^{-1} = E, \tag{37}$$

d. h., A^{-1} ... owohl Links- als auch Rechtsinverse zu A und liefert bei Multiplikation mit A die Einheitsmatrix E.

Bei lösbaren Gleichungssystemen $Ax = r$ liefert eine solche Multiplikation

$$x = A^{-1}r$$

sofort die Lösung. Bei allgemeinen linearen Gleichungssystemen mit einer Koeffizientenmatrix vom Typ $n \times m$, eventuellem Rangabfall und möglicherweise verletzter Lösbarkeitsbedingung sucht man nach einer *verallgemeinerten Inversen* oder *Pseudoinversen* (auch g-Inverse, von generalized (engl.) = verallgemeinert), die bei Multiplikation die gewünschten Resultate erkennen läßt. Dazu gehören: Trennung der Unbekannten in abhängige und frei wählbare (Parameter der Lösungsmannigfaltigkeit), Angabe der Koeffizienten in der Mannigfaltigkeit, Aussondern der linear abhängigen Gleichungen und möglicherweise Anzeige der Verletzung der Lösbarkeitsbedingung. Eine solche Inverse kann natürlich nicht durch die Gleichung (37) definiert werden. Derartige Fragen wurden von R. PENROSE (1955), E. H. MOORE u. a. untersucht (vgl. [38]). Bezeichnet man die verallgemeinerte Inverse mit A^g, so soll sie nach PENROSE die Bedingungen (im Reellen)

$$\begin{aligned} &\text{Bedingung 1:} && AA^gA = A, \\ &\text{Bedingung 2:} && A^gAA^g = A^g, \\ &\text{Bedingung 3:} && (AA^g)^T = AA^g, \\ &\text{Bedingung 4:} && (A^gA)^T = A^gA \end{aligned} \tag{38}$$

erfüllen. Das sind Bedingungen, die eine echte Inverse A^{-1} immer erfüllt. Bei A vom Typ $n \times m$ ist A^g vom Typ $m \times n$. Erfüllt eine verallgemeinerte Inverse die Bedingung 1, so nennt man sie *innere Inverse* (A^1 oder A^-). Bei Erfüllen der Bedingung 2 heißt sie *äußere Inverse* (A^2). Erfüllt sie die Bedingungen 1 und 2, so heißt sie *reflexive Inverse* ($A^{1,2}$). Die Bezeichnung A^- wird nicht nur für A^1, sondern auch allgemein für A^g benutzt. Bei der Pseudoinversen A^+ nach MOORE-PENROSE und bei Inversen $A^{1,3}$, $A^{1,4}$ u. a. ergeben sich gewisse Extremaleigenschaften, beispielsweise liefert die Linksmultiplikation eines Gleichungssystems mit einer solchen Inversen einen Lösungspunkt aus der Mannigfaltigkeit, der im Sinn einer gewissen Norm den geringsten Abstand vom Ursprung hat. Man könnte sogar, falls das System nicht lösbar ist, nach Inversen fragen, deren Verwendung bei Linksmultiplikation einen solchen Punkt liefert, dessen normierter Abstand (Defekt) von allen Gleichungen nach einer gewissen Norm am kleinsten ist oder ähnlich.

Derartige Forderungen sollen hier nicht untersucht werden, obwohl sie mitunter große praktische Bedeutung haben, vielmehr gehe es um die Beantwortung der weiter oben gestellten mehr algebraischen Fragen. Eine derartige verallgemeinerte (oder *erweiterte*; vgl. [29]) Inverse werde mit A^K (K = komplette Lösungsmannigfaltigkeit) bezeichnet.

Das Austauschverfahren bietet eine Möglichkeit zu ihrer Herstellung. Die erweiterte Inverse A^K ist eine Linksinverse vom Typ $n \times n$, wenn A vom Typ $n \times m$ ist. Ist $\text{Rang}(A) = T$ und befindet sich die den Rang bestimmende Teilmatrix A_T

in der linken oberen Ecke, so ist

$$A^K = \begin{pmatrix} A_T^{-1} & 0 \\ -A_L A_T^{-1} & E \end{pmatrix}. \tag{39}$$

Die linke Spalte stimmt also mit den Werten überein, die sich beim Austauschverfahren ergeben (vgl. (36)).

Liegt nun das Gleichungssystem

$$Ax + R = 0$$

oder ausführlich

$$\begin{pmatrix} A_T & A_P \\ A_L & A_N \end{pmatrix} \begin{pmatrix} x_T \\ x_P \end{pmatrix} + \begin{pmatrix} R_T \\ R_L \end{pmatrix} = 0 \tag{40}$$

vor, so liefert die Linksmultiplikation mit A^K

$$A^K Ax + A^K R = 0,$$

$$\begin{pmatrix} E & A_T^{-1} A_P \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_T \\ x_P \end{pmatrix} + \begin{pmatrix} A_T^{-1} R_T \\ 0 \end{pmatrix} = 0, \tag{41}$$

so daß jetzt bei R und der rechten Spalte der Produktmatrix $A^K A$ die vom Austauschverfahren bekannten Werte erscheinen (vgl. (36)).

Die Penrose-Bedingungen (38) werden von A^K nicht erfüllt, wie man leicht nachrechnet. Wird jedoch die Inverse A^K durch Nullzeilen zu A_0^K ergänzt, so daß sie vom Typ $m \times n$ ist, so gilt $AA_0^K A = A$, und sie ist innere Inverse.

Beispiel

Für die Aufgabe

$$Ax + R = 0$$

mit A und R vom obigen Beispiel ist

$$A^K = \left(\begin{array}{cccc|cc} -16.2 & 6.7\overline{3} & 4.8\overline{6} & -0.6 & 0 & 0 \\ 4.8 & -1.6 & -1.3 & -0.1 & 0 & 0 \\ -5.8 & 2.6 & 1.8 & -0.4 & 0 & 0 \\ 1.4 & -0.8 & -0.4 & 0.2 & 0 & 0 \\ \hline -1 & -1 & 1 & 1 & 1 & 0 \\ -2 & 1 & -2 & 1 & 0 & 1 \end{array}\right)$$

wobei die Trennlinien für die Teilmatrizen wieder eingezeichnet sind. Die Multiplikation der Aufgabe mit A^K von links liefert

$$A^K Ax + A^K R = 0,$$

$$\left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & -39.4 & 6.4 & 27 \\ 0 & 1 & 0 & 0 & 12.6 & -3.1 & -9 \\ 0 & 0 & 1 & 0 & -13.6 & 1.6 & 9 \\ 0 & 0 & 0 & 1 & 2.8 & 0.2 & -2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \hline x_5 \\ x_6 \\ x_7 \end{pmatrix} + \left(\begin{array}{c} -10 \\ 13 \\ -3 \\ -2 \\ \hline 0 \\ 0 \end{array}\right) = 0,$$

woraus natürlich dieselbe Lösung ablesbar ist.

2.4.6. Iterative Lösungsverfahren für lineare Gleichungssysteme

Außer den *direkten* Verfahren zur Lösung linearer Gleichungssysteme — zwei unterschiedliche Ansätze, nämlich der Algorithmus von GAUSS und das Austauschverfahren, wurden besprochen — kennt man auch noch *indirekte* oder *iterative*. Bei diesen wird, wie bereits beim Iterationsverfahren allgemein vorgeführt wurde, die Lösung schrittweise immer besser angenähert. Bei speziellen größeren Gleichungssystemen verhalten sich diese Verfahren sogar gelegentlich besser als die direkten, da sie während der Rechnung unvermeidbare Rundungsfehler und deren Auswirkung wieder eliminieren. Direkte Verfahren sind dazu nicht in der Lage und liefern mitunter trotz Rechnung ohne Rechenfehler (ohne „Verrechner") recht ungenaue Resultate. Es wird deshalb bei den direkten Verfahren oft eine zusätzliche Nachiteration empfohlen.

Bei den allgemeinen Iterationsverfahren wurde gezeigt, daß man aus der gegebenen Aufgabe eine iterierfähige Form herzustellen hat, und ferner, daß eine solche an sich iterierfähige Form nur unter gewissen Bedingungen beim Iterieren auch zur Lösung, zum Fixpunkt, konvergiert. Diese Eigenschaften findet man bei den iterativen Lösungsverfahren für lineare Gleichungssysteme wieder. Jetzt aber ist der Fixpunkt nicht mehr ein Punkt auf der reellen Zahlengeraden (eindimensional), sondern es ist ein Punkt im n-dimensionalen Raum. Seine n Komponenten oder Koordinaten sind die Lösungswerte der n Unbekannten des Gleichungssystems. Die hier angedeutete Verallgemeinerung anschaulicher geometrischer Sachverhalte ist gerade für lineare Gleichungssysteme günstig und bietet auch dem Lehrer bei einer Beschränkung auf den dreidimensionalen Raum $\mathbf{R}^3$ günstige Möglichkeiten, das Raumanschauungsvermögen bei den Schülern zu fördern.

2.4.7. Jacobi-Verfahren oder Gesamtschritt-Verfahren

C. G. J. JACOBI (1804—1851, Berlin und Königsberg) stellte eine iterierfähige Form aus dem System

$$Ax = r$$

durch additive Zerlegung der Matrix $A = L + D + R$ her. L ist dabei die linke (oder untere), D die diagonale und R die rechte (oder obere) Teilmatrix. Es ist dann

$$Dx = r - (L + R)\,x, \tag{42}$$

und die Isolierung des Vektors x gelingt leicht durch Division mit den Diagonalelementen

$$x = D^{-1}\big(r - (L + R)\,x\big), \tag{43}$$

die also verschieden von Null sein müssen. JACOBI hat das Gleichungssystem nach den x-Werten aus der Diagonale aufgelöst. Diese Strategie ist nicht zwingend, wenn auch naheliegend. Man kann jede Gleichung nach einer anderen Unbekannten auflösen. Allerdings kann man durch Gleichungsvertauschen erreichen, daß die „aufgelösten" Unbekannten in der Diagonale stehen.

In Komponenten schreibt sich die Gleichung (43)

$$x_i = \frac{r_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j \quad \text{für} \quad i = 1(1)n$$

als iterierfähige Form. Man startet mit

$$x_i^{(0)} = 0 \quad \text{für} \quad i = 1(1)n$$

oder bei Einsparung des sich daraus ergebenden ersten Iterationsschrittes gleich mit

$$x_i^{(0)} = \frac{r_i}{a_{ii}} \quad \text{für} \quad i = 1(1)n.$$

Die Iterationsvorschrift lautet

$$x_i^{(k+1)} = \frac{r_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} \quad \text{für} \quad i = 1(1)n. \tag{44}$$

Abgebrochen wird praktisch oft dann, wenn z. B.

$$\sum_{i=1}^{n} |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon \tag{45}$$

oder wenn

$$\max_i |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$$

wird. Als Abbruchbedingung sind außer diesen Formeln mit Absolutwerten auch Quadratsummen der Komponentendifferenzen denkbar, die dann das euklidische Betragsquadrat des Korrekturvektors pro Schritt bedeuten (vgl. dazu wieder die geometrische Anschauung). Die Abbruchbedingungen (45) können verbessert werden, indem nach 2.1. (12) der Konvergenzfaktor q berücksichtigt wird.

Beispiel

Das System

$$\begin{aligned} 3x_0 + 4x_1 - x_2 &= 6, \\ 2x_0 - 6x_1 + x_2 &= 7, \\ x_0 + 2x_1 + 4x_2 &= 9 \end{aligned}$$

wird iterierfähig umgeformt in

$$\begin{aligned} x_0 &= (6 \qquad -4x_1 + x_2)/3, \\ x_1 &= (-7 + 2x_0 \qquad + x_2)/6, \\ x_2 &= (9 - x_0 \quad -2x_1 \quad)/4. \end{aligned}$$

Gestartet wird mit $x^{(0)} = (0, 0, 0)$, so daß $x^{(1)} = (2, -1.17, 2.25)$ ist. Die Konvergenz ist langsam, nach anfänglichem stärkeren Schwanken der Werte pendeln sie sich nach etwa 30 Iterationsschritten ein bei

$$x^{(30)} = (2.588, -0.035, 1.621),$$

$$x^{(31)} = (2.587, -0.034, 1.620).$$

Die zehnstellig errechnete Lösung ist

$$x_0 = 2.586206900, \quad x_1 = -0.034482760, \quad x_2 = 1.620689655.$$

Tab. 2.38. Jacobi-Iteration für das im Text als Beispiel genannte lineare Gleichungssystem. Die hinreichenden Konvergenzkriterien *Zeilensumme* oder *Spaltensumme* sind auch nicht erfüllt. Das praktische Durchführen von 30 Iterationen mit zehnstelligen Zahlen ist nur mit einem programmierbaren Rechner vertretbar

k	x_0	x_1	x_2	k	x_0	x_1	x_2
0	0	0	0	8	2.194	−0.014	1.458
1	2	−1.17	2.25	9	2.504	−0.192	1.708
2	4.306	−0.125	2.33	10	2.826	−0.047	1.720
3	2.944	0.657	1.236	15	2.605	0.002	1.601
4	1.535	0.021	1.185	20	2.566	−0.033	1.612
5	2.367	−0.457	1.856	25	2.585	−0.038	1.622
6	3.228	−0.068	1.887	30	2.588	−0.035	1.621
7	2.720	0.224	1.477	31	2.587	−0.034	1.620

Konvergenzbedingung

Es ist die hinreichende Konvergenzbedingung herleitbar, daß das Jacobi-Verfahren für einen beliebigen Startvektor konvergiert, wenn das Maximum der durch das jeweilige Diagonalelement dividierten Summen der Beträge der Spaltenelemente aus der Matrix A kleiner als 1 ist oder wenn das Maximum der durch das Diagonalelement dividierten Summen der Beträge der Zeilenelemente kleiner als 1 ist, also

$$\max_j \sum_{\substack{i=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| < 1 \qquad (\textit{Spaltensummenkriterium}) \tag{46a}$$

oder

$$\max_i \sum_{\substack{j=1 \\ i \neq j}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| < 1 \qquad (\textit{Zeilensummenkriterium}), \tag{46b}$$

was hier auch als $\max_i \frac{1}{|a_{ii}|} \sum_{j=1}^{n} |a_{ij}| < 1$ geschrieben werden kann.

Auch das *Quadratsummenkriterium* (MfL Bd. 10, 6.1.3)

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} \left(\frac{a_{ij}}{a_{ii}}\right)^2 < 1 \tag{46c}$$

kann eingesetzt werden, welches keine Maximumbildung verlangt.

Als Faustregel für die Prüfung auf Konvergenz gilt, daß die Diagonalelemente betragsmäßig größer als die Betragssummen der anderen Elemente der Zeile sein sollen.

Das Zeilensummenkriterium ist leichter überprüfbar, denn bei ihm werden die Elemente einer Zeile durch dasselbe Diagonalelement dividiert. Am besten sind die Kriterien an der bereits hergestellten Iterierform prüfbar, wenn die Matrixelemente schon durch die Diagonalelemente dividiert wurden.

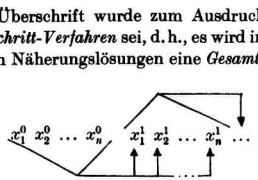
Beispiel

Das Beispiel von oben lieferte								Summe der Beträge der Zeilenelemente
$x_0 = 2$			$-$	1.33	$x_1 +$	0.33	x_2	1.67
$x_1 = -1.17 +$	0.33	x_0			$+$	0.17	x_2	0.5
$x_2 = 2.25 -$	0.25	$x_0 -$	0.5	x_1				0.75
Summe der Beträge der Spaltenelemente	0.58		1.83			0.5		

Die Kriterien sind also beide nicht erfüllt, denn die Maxima 1.83 und 1.67 sind beide größer als 1. Trotzdem konvergiert die Iteration, wenn auch recht schlecht. Es handelt sich eben um ein *hinreichendes* Kriterium. Es ist in seiner Forderung überhöht. Wenn es erfüllt ist, kann man mit Sicherheit Konvergenz erwarten. Wenn es nicht erfüllt ist, kann trotzdem noch Konvergenz vorliegen, braucht aber nicht. Es war genau die Absicht dieses Beispiels, einen solchen Sachverhalt „Konvergenz trotz verletzter (hinreichender) Konvergenzbedingung" zu zeigen. Oft kann man durch Linearkombination von Gleichungen ein äquivalentes System leicht herstellen, welches eines der Kriterien erfüllt. Beim Gauß-Seidel-Verfahren in 2.4.8. wird das vorgeführt werden.

Gesamtschritt-Verfahren

In der Überschrift wurde zum Ausdruck gebracht, daß das Jacobi-Verfahren ein *Gesamtschritt-Verfahren* sei, d. h., es wird in jedem Iterationsschritt aus der *Gesamtheit* der alten Näherungslösungen eine *Gesamtheit* neuer Näherungslösungen errechnet:



Tab. 2.39. BASIC-Programm zum Jacobi-Verfahren

```
10  REM **********************
20  REM *                    *
30  REM * Jacobi-Verfahren   *
40  REM *                    *
45  REM **********************
50  INPUT "Umfang:"; n
55  DIM a(n, n): DIM r(n): DIM x(n): DIM y(n)
60  FOR i = 1 TO n
65  FOR j = 1 TO n
70  PRINT "a(";i;", ";j;") =";
75  INPUT "a(i, j) ="; a(i, j)
80  PRINT a(i, j)
85  NEXT j
90  PRINT "r(";i;") =";
100 INPUT "r(i) ="; r(i)
110 PRINT r(i)
120 NEXT i
125 REM *** Vorbereitung ***
126 REM *** Kriterien ***
129 LET max = 0
130 FOR i = 1 TO n
140 LET f = a(i, i)
150 LET r(i) = r(i)/f
160 LET sum = -1
170 FOR j = 1 TO n
180 LET a(i, j) = a(i, j)/f
190 LET sum = sum + ABS a(i, j)
200 NEXT j
210 IF max < sum THEN LET max = sum
220 NEXT i
230 IF max < 1 THEN PRINT "Zeilensummenkriterium erfuellt"
240 IF max < 1 THEN GOTO 400
250 PRINT "Zeilensummenkriterium nicht erfuellt"
260 LET max = 0
270 FOR j = 1 TO n
280 LET sum = -1
290 FOR i = 1 TO n
300 LET sum = sum + ABS a(i, j)
310 NEXT i
320 IF max < sum THEN LET max = sum
```

Tabelle 2.39. (Fortsetzung)

```
330 NEXT j
340 IF max < 1 THEN PRINT "Spaltensummenkriterium erfuellt"
350 IF max < 1 THEN GOTO 400
360 PRINT "Spaltensummenkriterium nicht erfuellt"
370 PRINT "Soll trotzdem iteriert werden? Druecke Taste ENTER,
    sonst STOP"
375 LET b$ = INKEY$
380 IF b$ = " " THEN GOTO 375
390 IF b$ = " STOP " THEN STOP
400 REM **********************
401 REM *      Iteration       *
402 REM **********************
405 INPUT "abs. Genauigkeit: "; eps
406 CLS
410 FOR i = 1 TO n
420 LET x(i) = 0
430 NEXT i
435 LET sch = 1
438 LET max = 0: PRINT sch;". Iterationsschritt"
440 FOR i = 1 TO n
450 LET s = r(i)
460 FOR j = 1 TO i - 1
470 LET s = s - a(i, j) * x(j)
480 NEXT j
490 FOR j = i + 1 TO n
500 LET s = s - a(i, j) * x(j)
510 NEXT j
520 LET dif = ABS(x(i) - s)
530 PRINT "x0(";i;") =";x(i); "x1(";i;") =";s
540 IF max < dif THEN LET max = dif
550 LET y(i) = s
560 NEXT i
565 PRINT "max. Diff. =";max
570 IF max < eps THEN STOP
575 PAUSE 100
580 CLS
590 FOR i = 1 TO n
600 LET x(i) = y(i)
610 NEXT i
615 LET sch = sch + 1
620 GOTO 438
```

Die $x_1^0, x_2^0, \ldots, x_n^0$ werden alle benutzt, um der Reihe nach (Reihenfolge ist eigentlich gleichgültig, es könnte auch zugleich, parallel, insgesamt geschehen) die $x_1^1, x_2^1, \ldots, x_n^1$ zu berechnen. Diese Werte werden dann ihrerseits insgesamt benutzt, um den nächsten Iterationsschritt zu erledigen.

Dieses Vorgehen läßt sich auch geometrisch interpretieren. Leider können in einer überschaubaren Zeichnung nur zwei Dimensionen verwendet werden, also Systeme mit zwei Unbekannten.

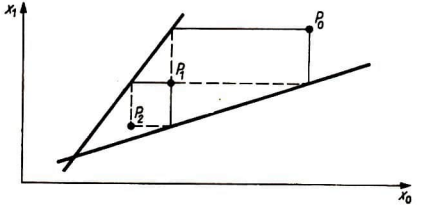


Abb. 2.26. Geometrische Veranschaulichung des Jacobi- oder Gesamtschritt-Verfahrens im $\mathbf{R}^2$

Eine lineare Gleichung stellt eine Hyperebene (Ebene, Gerade) dar. Ein lösbares System hat einen gemeinsamen Punkt im $\mathbf{R}^n$ ($\mathbf{R}^3$, $\mathbf{R}^2$). Ein Näherungspunkt für die Lösung liegt i. a. auf keiner der Hyperebenen (Ebenen, Geraden). Die iterierfähige Form des Systems nach dem Jacobi-Verfahren berechnet in jeder Zeile eine neue Punktkoordinate gerade so, daß unter Beibehaltung aller anderen Koordinatenwerte eine Gleichung erfüllt wird. Geometrisch bedeutet das, daß man parallel zu den Achsen bis zur nächsten Hyperebene (Ebene, Gerade) geht. Aus all diesen so gewonnenen Koordinaten wird dann ein neuer Näherungspunkt $P_1, P_2, \ldots$ hergestellt, wie Abb. 2.26 zeigt. Bei ungünstigen Winkellagen kann dadurch ein neuer Punkt weit von der Lösung abwandern. Besonders günstig ist es dagegen, wenn das System *orthogonal* ist, d. h. die Geraden senkrecht aufeinander stehen und parallel zu den Achsen ausgerichtet sind. Dies ist die geometrische Bedeutung der überwiegenden Diagonalelemente.

Das Programm zum Jacobi-Verfahren ist in Tab. 2.39 enthalten; es wurde als Demonstrationsprogramm gestaltet. Wählt man $n = 3$ („Umfang" genannt), so kann man als Beispiel die Koeffizienten des Gleichungssystems von S. 185 eingeben, die alle durch Anzeige auf dem Bildschirm bestätigt werden. Nach Eingabe des letzten Wertes werden automatisch die Konvergenzkriterien geprüft. Auch wenn keines von beiden erfüllt sein sollte (beim Beispiel ist das der Fall), kann mit ENTER trotzdem iteriert werden. Bei Eingabe einer gewünschten absoluten Genauigkeit eps = 0.001 werden die Werte der Tab. 2.38 (dort gerundet angegeben) geliefert. Der Befehl PAUSE 1ØØ sorgt nach jedem Zyklus für 2 s Anschauungszeit. Dazu wird in jedem Schritt die noch vorhandene maximale Differenz der Lösungskomponenten zum vorhergehenden Wert angezeigt.

Die Programmzeilen 440 bis 510, also lediglich acht Zeilen, enthalten die Iterationsvorschrift (44) des Jacobi-Verfahrens. Alles andere besteht aus Prüfung der Konvergenzbedingung, Prüfung zum Iterationsabbruch und — recht wesentlich — zum Anzeigekomfort.

2.4.8. Gauß-Seidel-Verfahren oder Einzelschritt-Verfahren

Es läßt sich denken, daß man die Iteration nach JACOBI beschleunigt, wenn man nicht erst alle neuen Punktkoordinaten abwartet, sondern vielmehr jede neue Koordinate sofort in die weitere Rechnung einbezieht. Die Jacobi-Formeln verändern sich nach diesem Vorgehen von GAUSS und L. SEIDEL (1874) in folgender Weise:

$$(L + D)\, x = r - Rx, \tag{47}$$

$$x = (L + D)^{-1}\,(r - Rx). \tag{48}$$

Tab. 2.40. Gauß-Seidel-Iteration für das im Text als Beispiel genannte Gleichungssystem. Sie ist im Beispiel ebenfalls schlecht konvergent, aber deutlich besser als das Jacobi-Verfahren. Die exakte Lösung ist immer zwischen zwei aufeinanderfolgende Näherungswerte eingeschlossen

k	x_0	x_1	x_2
0	0	0	0
1	2	−0.5	2
2	3.33	0.28	1.28
3	2.06	−0.27	1.87
4	2.98	0.14	1.44
5	2.29	−0.16	1.76
6	2.80	0.06	1.52
7	2.43	−0.11	1.70
8	2.71	0.02	1.56
9	2.50	−0.07	1.66
10	2.65	−0.01	1.59
15	2.571	−0.041	1.628
20	2.590	−0.033	1.619
25	2.5855	−0.0348	1.6210
30	2.5864	−0.0344	1.6206
31	2.5861	−0.0345	1.6207

Die Matrix $L + D$ ist eine Dreiecksmatrix mit Diagonale, so daß sich das Bilden einer Kehrmatrix auf eine *Rückrechnung* oder sukzessive Auflösung vereinfacht. Besonders deutlich wird das bei den komponentenweise geschriebenen Iterationsformeln

$$x_i^{(k+1)} = \frac{r_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}}\, x_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}}\, x_j^{(k)} \quad \text{für} \quad i = 1(1)n. \tag{49}$$

Durch den oberen Index (Iterationsindex) $k+1$ bei den Komponenten x_1 bis x_{i-1} wird deutlich, daß schon neu berechnete Werte zu verwenden sind.

Beispiel

Wieder wird das Beispiel verwendet, das bereits beim Jacobi-Verfahren diente. Startet man ebenfalls mit $x^{(0)} = (0, 0, 0)$, so erhält man nun jedoch $x^{(1)} = (2, -0.5, 2)$. Tab. 2.38 zeigt die weiteren iterierten Werte.

Konvergenzbedingung

Eine Konvergenzbedingung für das Gauß-Seidel-Verfahren herzuleiten ist wegen des komplizierten Rechenganges schwieriger als beim Jacobi-Verfahren. Die dort genannten Kriterien (46) sind auch für das Gauß-Seidel-Verfahren hinreichend. Aber sie sind nun erst recht unnötig grob, d. h. noch mehr Gleichungssysteme liefern beim Gauß-Seidel-Verfahren konvergierende Iterationen (obwohl Spalten- oder Zeilensummenkriterien nicht erfüllt sind), als das beim Jacobi-Verfahren der Fall ist.

Ein praktisch besser brauchbares hinreichendes Kriterium im allgemeinen Fall zur Prüfung auf hinreichend gute Konvergenz ist nach L. COLLATZ

$$\max_i \frac{S_i}{|a_{ii}| - S_i} < 2 \quad \text{mit} \quad S_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|. \tag{50}$$

Die S_i sind also Zeilensummen ohne Diagonalelement. Im vorliegenden Beispiel ist

$$\max\left(\frac{5}{3-5}, \frac{3}{6-3}, \frac{3}{4-3}\right) = 3 > 2$$

und damit das Kriterium auch nicht erfüllt. Die Konvergenz ist ja auch nicht gut.

Nach H. SASSENFELD (1951, also erst relativ spät) kennt man ein ebenfalls hinreichendes und dem Gauß-Seidel-Verfahren angepaßtes Konvergenzkriterium (vgl. [32] S. 57 und [72] S. 150): Das Gauß-Seidel-Verfahren konvergiert für beliebigen Startvektor $x^{(0)}$, falls

$$\max_i k_i < 1$$

gilt; dabei ist

$$k_1 = \sum_{j=2}^{n} \left|\frac{a_{1j}}{a_{11}}\right|, \quad k_i = \sum_{j=1}^{i-1} \left|\frac{a_{ij}}{a_{ii}}\right| k_j + \sum_{j=i+1}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| \quad \text{für} \quad i = 2(1)n. \tag{51}$$

Im vorliegenden Beispiel ist das iterierfähige System

$$\begin{aligned} x_0 &= 2 - 1.33x_1 + 0.33x_2, \\ x_1 &= -1.17 + 0.33x_0 + 0.17x_2, \\ x_2 &= 2.25 - 0.25x_0 - 0.5x_1, \end{aligned}$$

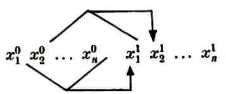
also ergibt sich

$$k_1 = 1.67, \quad k_2 = 0.72, \quad k_3 = 0.78,$$

und durch k_1 wird das Erfülltsein des Kriteriums auch hier gestört.

Einzelschritt-Verfahren

Wie bereits am Anfang dieses Abschnitts gesagt wurde, besteht der Unterschied des Gauß-Seidel-Verfahrens zum Jacobi-Verfahren darin, daß jeder neu errechnete Wert sofort wieder in die Rechnung einbezogen wird. Man nennt deshalb dieses Verfahren auch *Verfahren in Einzelschritten* im Gegensatz zum Gesamtschritt-Verfahren. Man kann sich den Rechengang wie folgt veranschaulichen:



Die zur Berechnung jeweils benutzten $n-1$ Komponenten schieben sich sukzessive vorwärts. Es kommt dabei nun natürlich auf die gewählte Reihenfolge, auf eine Rechenstrategie, an. Die zufällig vorliegende Numerierungsfolge zu nehmen ist nicht zwingend. Wählt man beispielsweise im bereits mehrfach vorgeführten Beispiel die Reihenfolge

$$x_1 \quad x_2 \quad x_0,$$

so ergeben sich im Sassenfeld-Kriterium die Zahlen

$$k_1 = 0.48, \qquad k_2 = 0.49, \qquad k_0 = 0.80,$$

deren Maximum kleiner als 1 ist und somit das Konvergenzkriterium erfüllen.

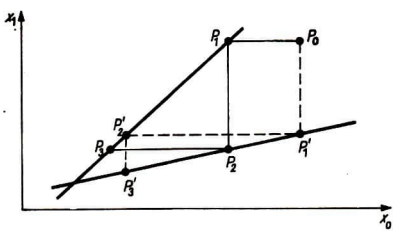


Abb. 2.27. Geometrische Veranschaulichung des Gauß-Seidel- oder Einzelschrittverfahrens im $\mathbf{R}^2$

Das Einzelschritt-Verfahren läßt sich ebenso wie das Gesamtschritt-Verfahren geometrisch interpretieren. Von einem Punkt für die Näherungslösung ausgehend, lotet man den „gesamten" Punkt (nicht nur jeweils eine Koordinate ändernd) parallel zu den Achsen zur nächsten Geraden (Ebene, Hyperebene) und dann von dort sofort weiter zur nächsten Geraden (Ebene, Hyperebene) gemäß der gewählten Rechenstrategie. Dies zeigt im $\mathbf{R}^2$ die Abb. 2.27. Auch hier ist es für die Konvergenz günstig, wenn die Geraden möglichst senkrecht aufeinander stehen und parallel

zu den Achsen verlaufen. Es werden zwei Strategien gezeigt: P_0, P_1, P_2', P_3 und P_0, P_1', P_2', P_3'.

Aus dem Programm zum Jacobi-Verfahren (Tab. 2.39) erhält man durch die neu einzugebenden Programmzeilen, wodurch die alten überschrieben werden,

```
30 REM * Gauss-Seidel *
550 LET x(i) = s
```

und Löschung der Zeilen 590 bis 610 ein Programm zum Gauß-Seidel-Verfahren. Dieses liefert, angewendet auf dasselbe Beispiel, die Werte aus Tab. 2.40 (dort gerundet angegeben).

2.4.9. Andere Iterationsverfahren

Beim Gesamtschritt-Verfahren von JACOBI und beim Einzelschritt-Verfahren nach GAUSS-SEIDEL wurde beim Hinweis auf eine geometrische Deutung von „Lotungen" gesprochen, die man natürlich auch als Projektionen ansehen kann. Diesen Gedanken ausbauend und verallgemeinernd, entstanden eine ganze Reihe von *Projektionsverfahren*, die man zusammenfassend bei HOUSEHOLDER [23] studieren kann. Auch

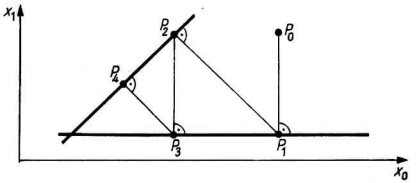


Abb. 2.28. Geometrische Veranschaulichung eines Projektionsverfahrens im $\mathbf{R}^2$. Es ist günstig, wenn die Geraden möglichst senkrecht aufeinander stehen, aber man ist frei von der Forderung, daß die Geraden achsenparallel sein sollen

in [29] und [32] findet man dazu Erläuterungen. Für eine Darstellung im $\mathbf{R}^2$ dient Abb. 2.28. Es wird dabei jeweils senkrecht auf die nächste Gerade (Ebene, Hyperebene) projiziert. Bei der zeichnerischen Beschränkung auf den $\mathbf{R}^2$ in Abb. 2.28 geht die Möglichkeit verloren, die Projektion auf *Unterräume* (z. B. Schnittebenen, Schnittgeraden) höherer Dimension vorzuzeigen. Im $\mathbf{R}^3$ ist das für das menschliche Raumvorstellungsvermögen noch erfaßbar (drei Schnittgeraden dreier Ebenen, Kanten einer Dreikantfeile, auf die reihum projiziert wird), in höheren Dimensionen muß es analytisch erschlossen und abstrahiert werden.

Ein anderes Verfahren ist das des *steilsten Abstiegs*. Es wird aus der Überlegung gewonnen, daß das Minimum der Funktion (quadratische Form)

$$Q(x) = \frac{1}{2}\, x^{\mathrm{T}} A x - xr \tag{52}$$

identisch ist mit der Lösung des linearen Gleichungssystems

$$Ax = r.$$

Auf der Fläche Q strebt man in Richtung des steilsten Abstiegs immer ein passendes Stück zum Minimum hin. Die Größe dieses passenden Stückes wird durch das Verfahren bestimmt ([29] S. 147). Diesen Gedanken eines passenden Stückes kann man auch bei anderen Iterationsverfahren, z. B. Projektionsverfahren, aber auch Gauß-Seidel, benutzen.

Durch einen passenden Faktor, den man Relaxationsfaktor ω (Lockerung, Entspannung) nennt, verkürzt bzw. verlängert — verringert bzw. verstärkt — man die vom Verfahren eigentlich errechnete Korrektur pro Iterationsschritt. Im Fall $\omega < 1$ spricht man von *Unter-*, im Fall $\omega > 1$ von *Überrelaxation*. Die Bestimmung eines günstigen Relaxationsfaktors ist wieder ein besonderes Problem ([16]; [32] S. 20), das auf eine Eigenwertbestimmung für die Koeffizientenmatrix A hinausläuft.

2.4.10. Erzwingen der Konvergenzbedingung

Gegeben ist das System

$$\begin{aligned} \text{(I)}\quad & 4x_1 + 3x_2 + x_3 = 7.9, \\ \text{(II)}\quad & x_1 + x_2 + 2x_3 = 5.6, \\ \text{(III)}\quad & 7x_1 + 3x_2 + 4x_3 = 17.2. \end{aligned}$$

Die Faustregel für das Zeilensummenkriterium, daß die Diagonalelemente überwiegen, ist nicht erfüllt. Durch Linearkombination der drei Gleichungen läßt sich ein äquivalentes System herstellen, welches das Kriterium erfüllt. Geometrisch bedeutet dies Übergang auf neue Basis- (Hyper-) Ebenen, die fast senkrecht aufeinander stehen und fast parallel zu den Koordinatenebenen sind. Beispielsweise ist erreichbar:

$$\begin{aligned} \text{(I')} &= -\text{(I)} - \text{(II)} + 2\text{(III)} = 9x_1 + 2x_2 + 5x_3 = 20.9, \\ \text{(II')} &= 3\text{(I)} + 3\text{(II)} - 2\text{(III)} = x_1 + 6x_2 + x_3 = 6.1, \\ \text{(III')} &= -\text{(I)} + 6\text{(II)} \qquad\quad = 2x_1 + 3x_2 + 11x_3 = 25.7. \end{aligned}$$

Jetzt ist das Zeilensummenkriterium erfüllt; man kann sicher sein, daß das Gauß-Seidel-Verfahren konvergieren wird. Die iterierfähige Form ist

$$\begin{aligned} x_1 &= (20.9 \qquad - 2x_2 - 5x_3)/9, \\ x_2 &= (\ 6.1 - x_1 \qquad - x_3)/6, \\ x_3 &= (25.7 - 2x_1 - 3x_2 \qquad)/11, \end{aligned}$$

und Tab. 2.41 zeigt die Rechenergebnisse. Für solch kleine Systeme, wie sie in den bisherigen Beispielen vorgeführt wurden, ist es vielleicht ratsam, die Koeffizienten gleich in das Programm eines Kleinstrechners einzuarbeiten. In Tab. 2.41 ist dies an einem BASIC-Programm für Demonstrationszwecke durchgeführt.

Tab. 2.41. Die exakte Lösung $x_1 = 1.1$, $x_2 = 0.5$, $x_3 = 2$ wird bei zehnstelliger Rechnung, auf fünf Stellen gerundet, nach sechs Iterationsschritten erreicht, nachdem die Erfüllung des Konvergenzkriteriums erzwungen wurde

Programm

```
10 REM "Demonstration"
20 LET x1 = 0 : LET x2 = 0 : LET x3 = 0
30 LET x1 = (20.9 - 2 * x2 - 5 * x3)/9
40 LET x2 = (6.1 - x1 - x3)/6
50 LET x3 = (25.7 - 2 * x1 - 3 * x2)/11
60 PRINT x1, x2, x3
70 STOP
80 GOTO 30
```

Fortsetzung nach jedem STOP durch CONTINUE

Resultat

	x_1	x_2	x_3
0	0	0	0
1	2.3222	0.6296	1.7424
2	1.2143	0.5239	1.9727
3	1.1099	0.5029	1.9974
4	1.1008	0.5003	1.9998
5	1.1001	0.5000	2.0000
6	1.1000	0.5000	2.0000

2.4.11. Numerische Betrachtungen — Kondition von Gleichungssystemen

Fragt man nach der Auflösung eines linearen Gleichungssystems

$$Ax = r,$$

wobei A natürlich als nichtsingulär (d. h. $\det(A) \neq 0$) angenommen ist, nach der Güte der eigentlich in jedem Fall berechneten Näherungslösung $\bar{x}$, gleichgültig, ob ein direktes oder ein iteratives Verfahren benutzt wurde, so stützt man sich gewöhnlich auf den Restfehler oder das *Residuum*, auch *Defekt* genannt,

$$\text{res} = r - A\bar{x}. \tag{53}$$

Man könnte aber, wenn die exakte Lösung x bekannt wäre, auch vom *Lösungsfehler*

$$\varepsilon = x - \bar{x} \tag{54}$$

ausgehen. Daß die beiden Forderungen *kleines Residuum* und *kleiner Lösungsfehler* nicht gleichbedeutend sind, hat C. B. Moler (1969) an einem schönen kleinen Beispiel gezeigt [30]: Für das System

$$0.780x_1 + 0.563x_2 = 0.217,$$
$$0.913x_1 + 0.659x_2 = 0.254,$$

das bereits in der Schule behandelt werden kann, lautet die exakte Lösung

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ mit dem Residuum } \text{res} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Das Wertepaar

$$\bar{x} = \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \end{pmatrix} = \begin{pmatrix} 0.341 \\ -0.087 \end{pmatrix},$$

welches also den großen Lösungsfehler $\varepsilon = \begin{pmatrix} 0.659 \\ -0.913 \end{pmatrix}$ hat, liefert das kleine Residuum

$$\text{res} = \begin{pmatrix} 0.000001 \\ 0 \end{pmatrix}.$$

Dagegen liefert das Wertepaar $\bar{x} = \begin{pmatrix} 0.999 \\ -1.001 \end{pmatrix}$, welches den kleinen Lösungsfehler $\varepsilon = \begin{pmatrix} 0.001 \\ 0.001 \end{pmatrix}$ hat, das relativ große Residuum

$$\text{res} = \begin{pmatrix} 0.001243 \\ 0.001572 \end{pmatrix}.$$

Wenn man aber, was meistens der Fall ist, die exakte Lösung nicht kennt, weiß man nicht, inwiefern man der errechneten Lösung vertrauen darf. Das Anschauen des Residuums allein genügt offenbar nicht. Hier hilft nun ein Kriterium von Prager-Oettli [50] weiter, welches das verwendete Rechenhilfsmittel einbezieht. Je nach der verwendeten Stellenzahl der benutzten Arithmetik hat man es mit einer begrenzten Genauigkeit zu tun. Ein Lösungsangebot ist akzeptierbar, wenn es im gelieferten Rundungsbereich liegt. Abb. 2.29 veranschaulicht diesen Sachverhalt. Ihr entnimmt man auch, daß dafür offenbar die „fast lineare Abhängigkeit" verantwortlich ist, die sich zeichnerisch in einem „schleifenden Schnittpunkt" äußert. Allgemein nennt man solche Systeme „schlecht konditioniert" (schlecht veranlagt). Sie haben auch die unerwünschte Eigenschaft, daß die Lösung sich stark ändert, wenn sich die Daten (Koeffizientenwerte und rechte Seite) nur wenig ändern. Solch eine geringfügige Änderung kann schon eintreten, wenn der exakte Wert 1/3 beim Einsatz eines Taschenrechners durch 0.333... ersetzt werden muß. Die beiden Gleichungen liefern Geraden, die im Rahmen der Zeichengenauigkeit nicht getrennt werden können.

Die exakte Lösung ● liegt bei (1, −1). Die Näherungslösung ○ liegt im Rundungsstreifen und kann akzeptiert werden. Die Näherungslösung + dagegen liegt außerhalb und muß verworfen werden. Zur Verdeutlichung ist ein vergröberter Ausschnitt rechts gezeichnet worden.

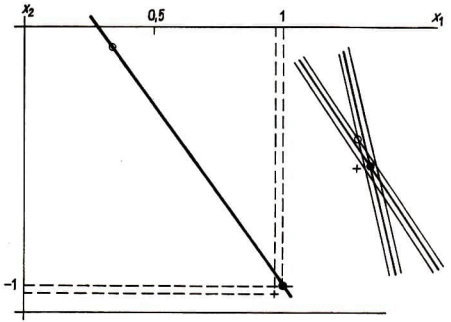


Abb. 2.29. Anwendung des Kriteriums von PRAGER-ÖTTLI zur Akzeptierung bzw. Ablehnung eines Lösungsangebotes linearer Gleichungssysteme. Vorgelegt ist das im Text genannte Beispiel von MOLER

Beispiel [60]

Die Systeme

$$\begin{aligned} (\mathrm{I}) \qquad x_1 + 0.99x_2 &= 1.99, \\ 0.99x_1 + 0.98x_2 &= 1.97 \end{aligned}$$

und, bei geringer Abänderung der rechten Seite durch $r = \begin{pmatrix} -0.000097 \\ 0.000106 \end{pmatrix}$,

$$\begin{aligned} (\mathrm{II}) \qquad x_1 + 0.99x_2 &= 1.989903, \\ 0.99x_1 + 0.98x_2 &= 1.970106 \end{aligned}$$

haben die exakten Lösungen

$$(\mathrm{I}) \quad x_1 = 1, \quad x_2 = 1$$

und

$$(\mathrm{II}) \quad x_1 = 3, \quad x_2 = -1.0203.$$

Es ist also die enorme Lösungsänderung

$$\Delta x = \begin{pmatrix} 2.0000 \\ -2.0203 \end{pmatrix}$$

verursacht worden. Der relative Fehler $\frac{|\Delta r|}{|r|}$ ist zu $\frac{|\Delta x|}{|x|}$ um rund gerechnet den Faktor 40000 vergrößert worden. Dieser Faktor wird *Konditionszahl* des Gleichungs-

systems genannt. Hier ist er groß, und die Kondition ist schlecht. Gute Kondition liegt vor, wenn der Faktor klein ist. Es ist schwierig, die Kondition eines Gleichungssystems von vornherein zu bestimmen. Dazu muß die Kehrmatrix berechnet werden, denn $\operatorname{cond}(A)$ wird durch

$$\operatorname{cond}(A) = \|A\| \cdot \|A^{-1}\| \tag{55}$$

definiert. Die Doppelstriche geben ein besonderes Maß, die Norm, für die Matrix an. Man kann als Norm die maximale Summe der Beträge der Zeilenelemente nehmen. Da im Beispiel

$$A^{-1} = \begin{pmatrix} -9800 & 9900 \\ 9900 & -10000 \end{pmatrix}$$

ist, ergibt sich $\|A\| = 1.99$ und $\|A^{-1}\| = 19900$, so daß

$$\operatorname{cond}(A) = 39601 \approx 40000$$

ist, was gut mit obiger Abschätzung der Fehlervergröberung zusammenpaßt.

Wie sich der Übergang auf eine Computer-Arithmetik auswirkt, wird in [60] S. 91 noch einmal demonstriert.

Beispiel

Das System

$$x_1 + \frac{x_2}{2} + \frac{x_3}{3} = 1,$$

$$\frac{x_1}{2} + \frac{x_2}{3} + \frac{x_3}{4} = 0,$$

$$\frac{x_1}{3} + \frac{x_2}{4} + \frac{x_3}{5} = 1$$

liefert bei dreistelliger dezimaler Rechnung (z. B. Rechenstab) die Näherungslösung $\bar{x}^{(0)}$ mit den Komponenten

$$\bar{x}_1 = 42.1, \quad \bar{x}_2 = -233, \quad \bar{x}_3 = 225,$$

während

$$x_1 = 39, \quad x_2 = -216, \quad x_3 = 210$$

die exakte Lösung ist. Als Residuum von $\bar{x}$ erhält man (hier muß zwischenzeitlich genauer als dreistellig gerechnet werden)

$$\operatorname{res} = (0.475, 0.298, 0.230).$$

Iteriert man nun zur Lösungsverbesserung $\bar{x}^{(0)}$ einmal nach, so erhält man $\bar{x}^{(1)}$ mit den Komponenten

$$\bar{x}_1 = 42.9, \quad \bar{x}_2 = -236, \quad \bar{x}_3 = 228,$$

also gegenüber der exakten Lösung scheinbar eine Verschlechterung. Tatsächlich ist aber doch eine Verbesserung eingetreten, denn in dem während der Rechnung verwendeten System werden die Koeffizienten 1/3 durch 0.333 ersetzt, und dieses System hat die Lösung (sechsstellig)

$$x_1' = 52.9542, \quad x_2' = -236.459, \quad x_3' = 229.055.$$

Auch das genaueste Verfahren kann nur die Lösung der im Computer tatsächlich vorhandenen Aufgaben liefern! Inwieweit diese mit der beabsichtigten übereinstimmt, muß anderweitig überprüft werden.

Die Computer-Arithmetik hat auch den folgenden Effekt der Vervielfachung von Lösungen.

Beispiel [33]
Das System

$$\begin{aligned} 0.89x_1 - 0.87x_2 &= \ \ 3.4, \\ -0.96x_1 + 0.97x_2 &= -3.6 \end{aligned}$$

ist eindeutig lösbar und hat bei zehnstelliger Rechnung zunächst die durch Eliminieren gewonnene Lösung

$$x_1 = 5.907473311, \quad x_2 = 2.135231317.$$

Noch fünfmaliges Iterieren gemäß

$$x := \begin{pmatrix} 0.11 & 0.87 \\ 0.96 & 0.03 \end{pmatrix} x + \begin{pmatrix} 3.4 \\ -3.6 \end{pmatrix}$$

ist nötig, um den Fixpunkt

$$x_{F1} = 5.907473312, \quad x_{F2} = 2.135231320$$

zu erhalten. Rechnet man jedoch in einer groben zweistelligen Arithmetik (zur Demonstration), so erhält man 11 (!) Fixpunkte, die sogar weit vom eben angegebenen entfernt liegen. Zum Beispiel sind zwei davon

$$x_1' = 6.3, \quad x_2' = 2.5 \quad \text{und} \quad x_1'' = 7.9, \quad x_2'' = 4.1,$$

während der zweistellig gerundete „echte" Fixpunkt

$$x_1 = 5.9, \quad x_2 = 2.1$$

bei der Iteration den Punkt (5.9, 2.2) liefert.

2.5. Anwendungen aus der Statistik

Die Wahrscheinlichkeitsrechnung und die mathematische Statistik sind feste Bestandteile des Lehrprogramms für angehende Mathematiklehrer (vgl. MfL Bd. 11). Statistische Auswertungen sind nicht nur in Wirtschaft und Technik von Bedeutung,

sondern auch bei pädagogischen und psychologischen Versuchen [10]. In diesem Kapitel kann natürlich weder die zugehörige Theorie noch einmal vermittelt noch kann ein vollständiger Überblick gegeben werden. Es werden lediglich einige ausgewählte statistische Methoden und Verfahren unter Benutzung kleiner Rechner erläutert. Dadurch soll auch eine gewisse Unsicherheit und Befangenheit in der Anwendung statistischer Methoden abgebaut werden. Der Einsatz schon kleiner Rechenhilfsmittel beseitigt außerdem die Scheu vor der zwar in der Art einfachen, aber sehr aufwendigen arithmetischen Arbeit.

2.5.1. Lineare Regression und Trendvorhersage

Es seien zwei (als zufällig ansehbare) Merkmale von Objekten gegeben, z. B. die Mathematik- und Physikzensuren (Merkmale) von Schülern (Elemente). Es soll untersucht werden, ob und wie stark ein Zusammenhang zwischen beiden besteht — wie stark also beide *korrelieren* —, und bei Kenntnis des einen Merkmals (oder der zufälligen Variablen), z. B. der Mathematikzensur, soll auf den Wert des anderen, nämlich der Physikzensur, rückgeschlossen bzw. dieser Wert geschätzt oder vorhergesagt werden. Diese Schätzung nennt man *Regression* (Rückgriff, Rückschluß).

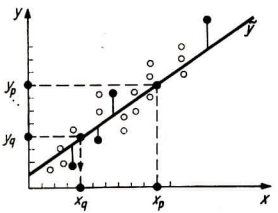


Abb. 2.30. Die Merkmale (x_i = Punktzahlen für Mathematikarbeiten, y_i = Punktzahlen für Physikarbeiten) gruppieren sich offenbar um eine Gerade

Dabei spielt noch die Art des funktionalen Zusammenhangs zwischen beiden Merkmalen eine Rolle. Es kann sich um eine lineare Abhängigkeit handeln. Natürlich sind auch andere möglich, jedoch kann oft das bei der linearen Regression entwickelte mathematische Verfahren durch geeignete Maßnahmen (Transformationen) wieder angewendet werden.

Bei den betrachteten Merkmalen muß es sich um *meßbare Merkmale* handeln. Es muß für sie ein Maß geben. Insofern ist das erwähnte Beispiel der Zensuren ungünstig, da Zensuren i. a. sogenannte *ordinalskalierte Daten* sind, d. h., es gibt wie im Sport 1., 2., 3. Plätze. Aber was ist der Unterschied zwischen dem 1. und dem 2. Platz?

In Abb. 2.30 erkennt man einen linearen Zusammenhang zwischen den beiden Merkmalen. Es soll die in einem gewissen Sinn beste Gerade mit den Parametern m (Anstieg) und b (Schnittpunkt mit der y-Achse) bestimmt werden. Zunächst ist eine beliebige Gerade

$$\tilde{y} = mx + b$$

eingezeichnet worden. Es wird $\bar{y}$ geschrieben, um den Funktionswert $\bar{y}_i$, der sich beim Einsetzen von x_i ergibt, vom zugehörigen zufälligen Variablenwert y_i unterscheiden zu können. Nach der Methode der kleinsten Quadrate von GAUSS wird nun diejenige Gerade als beste bezeichnet, für die

$$F = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \Rightarrow \text{Min} \tag{1}$$

gilt. In Abb. 2.30 sind vier solche Differenzen eingetragen. Aus (1) erhält man mit $\bar{y}_i$

$$F = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \Rightarrow \text{Min}$$

für die Meßwertpaare (x_i, y_i), und die notwendigen Bedingungen für das Minimum (eigentlich nur für einen Extremwert) sind

$$\begin{aligned} &\frac{\partial F}{\partial m} = \frac{\partial F}{\partial b} = 0, \\ &\sum_{i=1}^{n} (y_i - mx_i - b)\, x_i = 0, \quad \sum_{i=1}^{n} (y_i - mx_i - b) = 0. \end{aligned} \tag{2}$$

Daraus erhält man das lineare Gleichungssystem

$$\begin{aligned} m \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i, \\ m \sum_{i=1}^{n} x_i + bn \quad &= \sum_{i=1}^{n} y_i. \end{aligned} \tag{3}$$

Dividiert man beide Gleichungen durch n, so nimmt insbesondere die zweite wegen der Mittelwerte $\bar{x}$ und $\bar{y}$ eine einfache Form an,

$$\begin{aligned} m \cdot \frac{1}{n} \sum_{i=1}^{n} x_i^2 + b\bar{x} &= \frac{1}{n} \sum_{i=1}^{n} x_i y_i, \\ m \cdot \bar{x} \quad + b \quad &= \bar{y}, \end{aligned}$$

und daraus ergibt sich sofort

$$m = \frac{\frac{1}{n} \sum_{i=1}^{n} x_i y_i - \bar{x}\bar{y}}{\frac{1}{n} \sum_{i=1}^{n} x_i^2 - \bar{x}^2}, \quad b = \bar{y} - m\bar{x}. \tag{4}$$

Diese bezüglich der Auswertung wirklich leicht abschreckend aussehenden Formeln verlieren diese Eigenschaft bei Benutzung eines geeigneten, d. h. mit statistischen Funktionen ausgerüsteten, oder programmierbaren Rechners. Manche für statistische Anwendungen vorbereitete Rechner haben eine „$\Sigma +$"- und eine „$\Sigma -$"-Taste. Nach dem Eintasten eines (x_i, y_i)-Paares, wobei beide Werte je nach Typ des Rech-

ners laut Gebrauchsanweisung getrennt werden, drücke man die Taste „$\Sigma +$“. Darauf bildet der Rechner in Registern R_k bis R_{k+5}:

$R_k := R_k + x_i$	Summe der x_i-Werte,
$R_{k+1} := R_{k+1} + y_i$	Summe der y_i-Werte,
$R_{k+2} := R_{k+2} + x_i^2$	Summe der x_i^2-Werte,
$R_{k+3} := R_{k+3} + y_i^2$	Summe der y_i^2-Werte (vorläufig noch nicht benötigt),
$R_{k+4} := R_{k+4} + y_i x_i$	Summe der gemischten Produkte $x_i y_i$,
$R_{k+5} := R_{k+5} + 1$	Anzahl der Paare.

Es werden also sechs Register für diese Zwecke benötigt. Hat man ein falsches Wertepaar schon auf diese Weise verarbeitet, so wird es noch einmal eingetastet, aber die Taste „$\Sigma -$“ gedrückt. Dadurch werden in allen Registern infolge einer Subtraktion die aufgelaufenen Summen korrigiert. Sind alle Werte auf diese Weise eingegeben, so berechnet eine Taste REG (Regression) oder mit ähnlicher Bezeichnung die Werte m und b, die im x- und y-Register angezeigt und außerdem in zwei Registern aufbewahrt werden. Tastet man nun einen weiteren x_p-Wert ein und drückt eine Taste YREG (y-Regression) oder mit entsprechender Bezeichnung (oft einfach y'), so liefert der Rechner die Vorhersage, d. h. den Trend, für y_p:

$$y_p = mx_p + b. \tag{5}$$

Das ist natürlich beliebig oft wiederholbar. Ebenso kann man bei Eintasten eines y_q-Wertes den x_q-Wert vorhersagen:

$$x_q = \frac{1}{m}(y_q - b). \tag{6}$$

Es wird bei der linearen Regression vorausgesetzt (das müßte eigentlich überprüft werden), daß die beiden Merkmale *normalverteilt* sind. Wie unten noch deutlich gemacht wird, liefert die Regression von x auf y eine andere Regressionsgerade als bei der Regression von y auf x. Das Verwenden derselben Geraden für beide Regressionen in (5) und (6) geschieht in der Richtung „von y auf x“ lediglich zur Beantwortung der Frage: „Von welchem x aus würde die Regression auf y führen?“

In Abb. 2.30 sind derartige Vorhersagen eingezeichnet unter der Voraussetzung, daß $\hat{y}$ bereits die Regressionsgerade ist.

Beispiel

i	x_i	y_i	x_i^2	y_i^2	$x_i y_i$
1	1.5	2.25	2.25	5.06	3.38
2	3.0	3.0	9.0	9.0	9.0
3	4.25	5.5	18.06	30.25	23.37
4	6.0	3.5	36.0	12.25	21.0
5	8.0	7.0	64.0	49.0	56.0
Σ	22.75	21.25	129.31	105.56	112.75

Mit $n = 5$ ergeben sich daraus

$$m = 0.62\,, \quad b = 1.42\,,$$

jeweils alle Werte auf zwei Nachkommastellen gerundet. Gibt man $x_p = 12$ vor, so errechnet man

$$y_p = 8.89\,,$$

und gibt man $y_q = 11.25$ vor, so errechnet man

$$x_q = 15.79\,.$$

Derartige lineare statistische Zusammenhänge zweier Merkmale sind sehr häufig. Der oft beobachtete Zusammenhang zwischen Mathematik- und Physikzensuren wurde schon erwähnt. In Körperdaten von Schülern korreliert z. B. die Größe mit

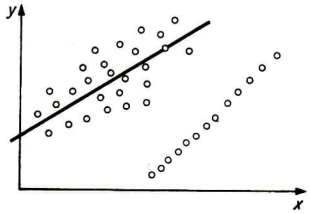


Abb. 2.31. Beispiele für weniger gute und extrem gute Korrelation

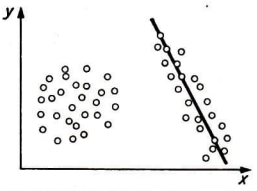


Abb. 2.32. Beispiele für nicht vorhandene und für negative Korrelation

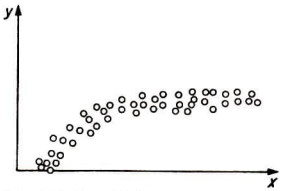


Abb. 2.33. Beispiel für nichtlineare Korrelation, hier vermutlich logarithmisch: $y = m \lg x + b$

dem Gewicht, in der Biologie oder Landwirtschaft korrelieren Düngergaben mit dem Ertrag, im Sport die Resultate in Kurzstreckenlauf und Weitspringen. Auch umgekehrte oder negative Korrelation (Abb. 2.32) beobachtet man: Je bessere Resultate in einer Disziplin, desto schlechtere in der anderen (z. B. Gewichtheben und Leichtathletik).

Es erhebt sich ferner die Frage nach der Güte der gemäß linearer Regression getroffenen Vorhersagen. Eine Regressionsgerade erhält man nach den Formeln (4) aus jedem Datenmaterial, es sei denn, der Nenner für m aus (4) verschwindet. Wenn die Meßpunkte wie in der Abb. 2.31 „gut" oder „eng" an der Geraden liegen, wird die Vorhersage auch gut sein. Liegen aber die Punkte nahezu gleichverteilt im Feld (Abb. 2.32) oder lassen sie einen ganz anderen Kurvenverlauf vermuten (Quadrat, Wurzel, Logarithmus) (Abb. 2.33), so wird auch die Vorhersage nach der linearen Regression schlecht sein. Ein Maß für die Stärke der linearen Abhängigkeit zwischen den x- und y-Werten ist der sogenannte *empirische Korrelationskoeffizient* (genauer: *Maßkorrelationskoeffizient*, MfL Bd. 11, 8.4.). Man erhält ihn durch folgende Überlegung.

Bei der Berechnung der Regressionsgeraden werden die y-Differenzen der Geradenpunkte zu den Meßpunkten benutzt. Extrem gute Korrelation liegt vor, wenn alle Meßpunkte auf der Regressionsgeraden liegen. Vertauscht man nun die Rolle der x- und y-Werte, so muß sich genau dieselbe Gerade ergeben, d. h., zunächst war

$$y = mx + b,$$

und nun wird

$$x = My + B,$$

aber es ist

$$M = \frac{1}{m}, \quad B = -\frac{b}{m}.$$

Also ist in diesem extrem guten Fall

$$mM = 1, \tag{7}$$

oder, anders gesagt, der Korrelationskoeffizient

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\,\overline{y}}{\sqrt{\left(\frac{1}{n}\sum_{i=1}^{m} x_i^2 - \overline{x}^2\right)\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2 - \overline{y}^2\right)}} \tag{8}$$

gibt gute Korrelation bei oder in der Nähe von $+1$ bzw. -1 an. Ist es dagegen ein Wert in der Nähe von 0, so korrelieren die untersuchten Werte nicht. Größere absolute Werte als 1 kann r nicht annehmen.

Man nennt übrigens die Werte

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{x}^2 &= \sigma_x^2, \\ \frac{1}{n}\sum_{i=1}^{n} y_i^2 - \overline{y}^2 &= \sigma_y^2 \end{aligned} \tag{9}$$

Tab. 2.42. Programm zur linearen Regression

```
 5 REM Lineare Regression
10 PRINT "Lineare Regression"
20 PRINT "Werteeingabe"
30 PRINT "Abbruch der Eingabe durch e statt einer Zahl.
          Dann Weiterrechnen mit GOTO 200"
40 LET n = 0
50 LET sx = 0
60 LET sy = 0
70 LET sx2 = 0
80 LET sy2 = 0
90 LET sxy = 0
100 INPUT "x ="; x, "y ="; y
105 PRINT AT 5, 0; "
110 LET n = n + 1
120 LET sx = sx + x
130 LET sy = sy + y
140 LET sx2 = sx2 + x * x
150 LET sy2 = sy2 + y * y
160 LET sxy = sxy + x * y
170 PRINT AT 5, 0; "x("; n;") = "; x, "y("; n;") ="; y
180 GOTO 100
190 REM Ende der Eingabe
200 PRINT AT 2, 0; "Berechnung von m und b der Regressionsgeraden"
210 PRINT AT 2, 0; "Berechnung von m und b der Regressionsgeraden"
220 LET sx = sx/n
230 LET sy = sy/n
240 LET sx2 = sx2/n - sx * sx
260 LET sy2 = sy2/n - sy * sy
270 LET sxy = sxy/n
280 LET m = (sxy - sx * sy)/sx2
290 LET b = sy - m * sx
300 PRINT "m ="; m, "b ="; b
310 PRINT "m ="; m, "b ="; b
320 PRINT "Berechnung des Korrelationskoeffizienten"
330 LET r = m * SQR (sx2/sy2)
340 PRINT "r ="; r
```

Tabelle 2.42 (Fortsetzung)

```
350  PRINT "r ="; r
360  PRINT "Trendvorhersage bei gegebenem x: GOTO 380
                    oder bei gegebenem y: GOTO 420"
370  STOP
380  INPUT "x ="; x
390  LET y = m * x + b
400  PRINT "x = "; x, "y =" y
410  STOP
420  INPUT "y ="; y
430  LET x = (y - b)/m
440  PRINT "y ="; y, "x ="; x
450  STOP
```

Varianzen oder *Streuungen* (in der Stichprobe) der x- bzw. y-Werte, so daß auch

$$r = m \frac{\sigma_x}{\sigma_y} \tag{10}$$

ist. (Oft wird in der Literatur die Stichprobenstreuung mit s_n^{*2} bezeichnet, während σ_x^2 bzw. σ_y^2 für die Normalverteilung als Streuungsangabe vorbehalten bleibt; vgl. aber (22) bezüglich der *empirischen Streuung*.) Ein BASIC-Programm ist in Tab. 2.42 enthalten.

Das Programm aus Tab. 2.42 erklärt sich eigentlich bei der Anwendung durch eingefügte Textstellen selbst. Es hat dadurch bei der Werteeingabe und bei den Trendvorhersagen einen gewissen Dialogcharakter. Auch die Variablennamen gestatten einen schnellen Bezug zu den Formeln (3), (4) und (8). Es wurde sx für Summe der x, $sx2$ für Summe der x^2 usw. gewählt.

Während der Werteeingabe werden die eingegebenen Werte als Bestätigung gedruckt. Hier bietet ein Heimcomputer bessere Möglichkeiten als ein Taschenrechner, auch wenn dieser die schon erwähnte spezielle Statistikausstattung besitzt. (Spezielle Statistikbefehle hat BASIC nicht, man muß ein Programm anfertigen.) Stellt man bei der Bestätigung einen Eingabefehler fest, so muß man bei diesem Programm mittels sofort ausführbarer Befehle nach einer Unterbrechung mittels *e*

```
LET n = n - 1: LET x = falsche Eingabe
LET y = falsche Eingabe: LET sx = sx - x
LET sy = sy - y: LET sx2 = sx2 - x ↑ x
LET sy2 = sy2 - y ↑ y: LET sxy = sxy - x ↑ y
```

die Werte zurückstellen und dann mit GOTO 100 fortsetzen. Wenn das öfter passiert, ist eine Programmergänzung mit diesen Befehlen anzuraten.

Am Ende der Rechnung, d. h. eigentlich schon ab Zeile 260, befinden sich unter den Variablennamen *sx2* bzw. *sy2* die Varianzen oder Streuungen (9).

Beispiel

Für das bereits oben angegebene kleine Zahlenbeispiel ergibt sich

$$\sigma_x^2 = \frac{129.31}{5} - \frac{22.75^2}{25} = 5.16,$$

$$\sigma_y^2 = \frac{105.56}{5} - \frac{21.25^2}{25} = 3.05,$$

$$r = 0.62 \sqrt{\frac{5.16}{3.05}} = 0.81.$$

Diese Korrelation erweist sich somit als recht gut. Aber wie sicher ist eine solche Aussage?

Sicherheit des Korrelationskoeffizienten

Ein psychologischer Versuch testet beispielsweise manuelle Fähigkeiten und mathematische Kenntnisse von Schülern. Es zeigt sich vielleicht zwischen beiden eine gute Korrelation. (Die Problematik in diesem Beispiel liegt natürlich auch in der Frage nach dem Maß der beiden Merkmale.)

Bei genauerer Betrachtung des Datenmaterials stellt sich aber heraus, daß das Alter der Schüler der beherrschende Faktor ist. Ältere Schüler sind offenbar sowohl manuell gewandter als auch mit besseren mathematischen Kenntnissen ausgerüstet. Wählt man für den Test gleichaltrige Schüler, so kann sich ein gänzlich anderer (vielleicht auch kein) Zusammenhang zwischen den untersuchten Merkmalen ergeben. Es ist also Vorsicht geboten, wenn man die Resultate der linearen Regression kritiklos verwendet. Die Substanz des Datenmaterials ist von Bedeutung, und man muß sorgfältig überlegen, was man messen und testen will.

Wenn man nur eine kleine Anzahl von Meßwerten zur Berechnung des Korrelationskoeffizienten verwendet, wird man sich der gefundenen Aussage nicht sehr sicher sein können. Im obigen Beispiel lagen nur fünf Meßpaare vor. Der Korrelationskoeffizient war 0.81, ist das eine sichere Aussage? Es gibt eine Möglichkeit, diese Sicherheit zu prüfen. Ohne auf die theoretische Grundlage einzugehen, wird die Vorgehensweise mitgeteilt:

- Man wähle eine wünschenswerte Sicherheit für den berechneten Korrelationskoeffizienten, z. B. 95%; diese Angabe nennt man *Konfidenz-* oder *Vertrauenszahl* (auch *Konfidenzniveau* oder *Sicherheitswahrscheinlichkeit*).
- In Tab. 2.4 sucht man in der Spalte der gewählten Konfidenzzahl und in der Zeile, welche die Anzahl der Meßpaare (Proben) angibt, eine Zahl r_{Test}; für das Beispiel steht dort 0.878.

– Ist der errechnete Korrelationskoeffizient r größer als r_{Test}, so kann man bei der gewählten Konfidenzzahl sicher sein, daß der Korrelationskoeffizient gültig ist. Im Beispiel ist aber $0.81 < 0.878$, somit ist $r = 0.81$ nicht mit 95% sicher.

Wählt man die Vertrauenszahl nur mit 90%, so findet man in Tab. 2.43 den Wert 0.805, und es ist $0.805 < 0.81$, so daß man sagen kann: Mit 90% Sicherheit ist der Korrelationskoeffizient 0.81 für die Gesamtheit, aus der das Datenmaterial stammt, korrekt.

Die Werte in Tab. 2.43 werden nach (11)

$$r_{\text{Test}} = \sqrt{\frac{t^2}{t^2 + a - 2}} \tag{11}$$

Tab. 2.43. Konfidenzwerte oder Testtabelle für den Korrelationskoeffizienten

Anzahl der Meßpaare	Konfidenzwerte				
	80%	90%	95%	99%	99.9%
3	0.951	0.988	0.997	1.000	1.000
4	0.800	0.900	0.950	0.990	0.999
5	0.687	0.805	0.878	0.959	0.991
6	0.608	0.729	0.811	0.917	0.974
7	0.551	0.669	0.755	0.875	0.951
8	0.507	0.621	0.707	0.834	0.925
9	0.472	0.582	0.666	0.798	0.898
10	0.443	0.549	0.632	0.765	0.872
11	0.419	0.521	0.602	0.735	0.847
12	0.398	0.479	0.576	0.708	0.823
13	0.380	0.476	0.553	0.684	0.801
14	0.365	0.457	0.532	0.661	0.780
15	0.351	0.441	0.514	0.641	0.760
16	0.338	0.426	0.497	0.623	0.742
17	0.327	0.412	0.482	0.606	0.725
18	0.317	0.400	0.468	0.590	0.708
19	0.308	0.389	0.456	0.575	0.693
20	0.299	0.378	0.444	0.561	0.679
21	0.291	0.369	0.433	0.549	0.665
22	0.284	0.360	0.423	0.537	0.652
23	0.277	0.352	0.413	0.526	0.640
24	0.271	0.344	0.404	0.515	0.629
25	0.265	0.337	0.396	0.505	0.618
26	0.260	0.330	0.388	0.496	0.607
27	0.255	0.323	0.381	0.487	0.597
28	0.250	0.317	0.374	0.479	0.588
29	0.245	0.311	0.367	0.471	0.579
30	0.241	0.306	0.361	0.463	0.570
31	0.237	0.301	0.355	0.456	0.562
32	0.233	0.296	0.349	0.449	0.554
42	0.202	0.257	0.304	0.393	0.490
62	0.165	0.211	0.250	0.325	0.408
122	0.117	0.150	0.178	0.232	0.294
∞	0	0	0	0	0

errechnet. Dabei ist d die Anzahl der Meßpaare, welche zur Bestimmung des Korrelationskoeffizienten dienten, und t ist ein Wert, der aus der sogenannten Student-Verteilung (vgl. (26)) abgeleitet ist.

Lineare Regression und Approximation

Die Aufgabe der linearen Regression ist eine *Ausgleichsaufgabe* oder *Approximationsaufgabe*: Nach der Methode der kleinsten Quadrate (GAUSS), und zwar der Quadrate der Ordinaten- bzw. y-Differenzen, wird die sich am besten anschmiegende Gerade gesucht. Es erweist sich aber dann, daß die sich ergebende Gerade nicht unabhängig vom Koordinatensystem ist (Abb. 2.34). Mit anderen Worten: Transformiert man die aus n Punkten bestehende „Punktwolke" so, daß die Punkte ihre gegenseitige Lage nicht ändern — etwa durch eine Drehung —, so ist dann deren beste Approxi-

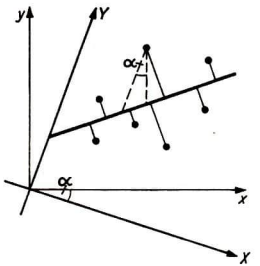


Abb. 2.34. Der senkrechte Abstand zwischen Punkten und einer Geraden ist von der Lage der Punkte in einem Koordinatensystem unabhängig. Die y-Differenzen, welche zur Bestimmung der Regressionsgeraden benutzt werden, ändern sich

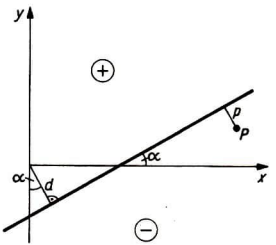


Abb. 2.35. Gerade in Hessescher Normalform

mationsgerade nicht durch dieselbe Transformation (Drehung) aus der ehemaligen erhältlich. Im Sinne der Approximation einer Geraden an n Punkte ist das unbefriedigend. Bei der linearen Regression jedoch wird dieses Verhalten gerade zur Bestimmung der Korrelation ausgenutzt. Die Transformation ist dabei eine Spiegelung an der Geraden $y = x$ (Vertauschen der x- und y-Koordinaten der n Punkte).

Wenn man eine Gerade sucht, die sich unabhängig vom Koordinatensystem an eine gegebene „Punktwolke" anschmiegt, muß man eine geometrische Größe finden, die zu minimieren ist und die vom Koordinatensystem unabhängig ist. Im Fall der Geraden bietet sich in einfacher Weise der senkrechte Abstand der Punkte von der Geraden an (Abb. 2.35). Diesen senkrechten Abstand p eines Punktes P (mit Vorzeichen) erhält man bei Geraden leicht aus der *Hesseschen Normalform* (vgl.

MfL Bd. 8, 2.5.1.)

$$\frac{ax + by + d}{\sqrt{a^2 + b^2}} = 0, \tag{12}$$

nämlich

$$p = \frac{ax_p + bx_p + d}{\sqrt{a^2 + b^2}}. \tag{13}$$

Je nach Lage von P in einer der durch die Gerade bestimmten Halbebenen ist p positiv oder negativ. Der Koordinatenursprung hat den Abstand d.

Nun verlangt man

$$\sum_{i=1}^{n} p_i^2 = \text{Min}, \quad F = \sum_{i=1}^{n} (ax_i + by_i + d)^2 = \text{Min} \tag{14}$$

mit der Nebenbedingung

$$G = a^2 + b^2 - 1 = 0.$$

Nach der Methode der Lagrangeschen Multiplikatoren wird verlangt:

$$F - \lambda G = \text{Min}.$$

Daraus ergibt sich das Gleichungssystem

$$\begin{pmatrix} \overline{x_i y_i} - \lambda & \overline{x_i y_i} & \overline{x_i} \\ \overline{x_i y_i} & \overline{y_i y_i} - \lambda & \overline{y_i} \\ \overline{x_i} & \overline{y_i} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ d \end{pmatrix} = 0 \tag{15}$$

als Eigenwertproblem, wobei zur einfacheren Schreibweise beispielsweise

$$\overline{y_i y_i} = \frac{1}{n} \sum_{i=1}^{n} y_i^2$$

bedeutet. Die charakteristische Gleichung zur Bestimmung der Eigenwerte ist somit

$$\lambda^2 - \lambda(\sigma_x^2 + \sigma_y^2) + \sigma_x^2 \sigma_y^2 - (\overline{x_i y_i} - \overline{x}_i \overline{y}_i)^2 = 0; \tag{16}$$

also ist

$$\lambda_{1,2} = \frac{1}{2}(\sigma_x^2 + \sigma_y^2) \pm \frac{1}{2}\sqrt{(\sigma_x^2 - \sigma_y^2)^2 + 4(\overline{x_i y_i} - \overline{x}_i \overline{y}_i)^2}.$$

Für das bereits verwendete Beispiel ergibt sich

$$\lambda_{1,2} = \frac{1}{2}(5.16 + 3.05) \pm \frac{1}{2}\sqrt{(5.16 - 3.05)^2 + 4(22.55 - 4.55 \cdot 4.25)^2} \cdot$$

$$= 4.11 \pm \frac{1}{2}\sqrt{4.45 + 41.28} = 4.11 \pm 3.38$$

und daraus

$$\begin{pmatrix} 25.86 - 0.73 & 22.55 & 4.55 \\ 22.55 & 22.11 - 0.73 & 4.25 \\ 4.55 & 4.25 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ d \end{pmatrix} = 0.$$

Die Lösung bei Erfülltsein der Nebenbedingung ist

$$-0.5949x + 0.8038y - 0.6979 = 0,$$

und zum Vergleich mit der Regressionsgeraden, deren m- und b-Wert 0.62 bzw. 1.42 ist, ergibt sich nun

$$m = 0.7401, \quad b = 0.8683,$$

wenn die Gerade in der Form $y = mx + b$ angegeben wird. (Die Gerade, welche sich aus dem anderen Eigenwert $\lambda_2 = 7.49$ ergibt, steht auf der angegebenen senkrecht.)

Einfache Verallgemeinerungen der linearen Regression

Bei der Berechnung der Regressionsgeraden ging man von der näherungsweise linearen Beziehung zwischen zwei Zufallsvariablen Y und X aus,

$$Y = mX + b,$$

und zur Bestimmung von m und b wurde

$$\sum_{i=1}^{n} (y_i - mx_i - b)^2 = \text{Min}$$

benutzt. Allgemein kann es sich um die Abhängigkeit des zufälligen Y von mehreren anderen zufälligen Größen X_k handeln,

$$Y = \sum_{k=1}^{K} m_k X_k + b,$$

und es können sogar die X_k anders als linear in den Zusammenhang eingehen:

$$Y = \sum_{k=0}^{M} m_k f_k(X_1, \ldots, X_k).$$

Beispielsweise könnte auch sein:

$$Y = \sum_{k=0}^{M} m_k X^k,$$

also nur eine einzige andere Zufallsvariable X, aber in polynomialem Zusammenhang mit Y, oder

$$Y = \sum_{k=0}^{M} m_k f_k(X),$$

wobei nun beliebige Funktionen $f_k(X)$ eine Rolle spielen könnten. Nach Abb. 2.33 würde man auf einen logarithmischen Zusammenhang tippen und für die lineare Regression

$$Y = m_1 \ln X + m_2$$

ansetzen. In allen Fällen kann man nach dem beschriebenen Verfahren durch Auflösung eines linearen Gleichungssystems die gesuchten m_k bestimmen. Dagegen ist das Berechnen oder das Schätzen der Gestalt eines funktionalen Zusammenhangs problematisch und erst recht das Festlegen der Vertrauensgrenzen, d. h. von Testgrößen entsprechend Tab. 2.42, für einen solchen Zusammenhang.

Das Rechenverfahren versagt dagegen oft, wenn der Zusammenhang zwischen Y und den X_k nicht explizit vorliegt, weil dann meist ein nichtlineares Gleichungssystem entsteht. Hat man beispielsweise eine Punktwolke, die eine *Ausgleichsellipse* oder einen anderen Kegelschnitt vermuten läßt, so wäre aus der allgemeinen Form

$$F(X, Y, a) = Y^2 + a_1XY + a_2X^2 + a_3X + a_4Y + a_5 = 0 \tag{17}$$

zwar noch

$$Y_{1,2} = -\frac{a_1X + a_4}{2} \pm \sqrt{\left(\frac{a_1X + a_4}{2}\right)^2 - (a_2X^2 + a_3X + a_5)}$$

explizit zu erhalten, aber die Koeffizienten a_1, a_2, a_3, a_4 und a_5 gehen nicht mehr linear ein. Hier hilft wieder eine geometrische Überlegung weiter, die in den Grundzügen bereits bei der Ausgleichsgeraden unter Beachtung der senkrechten Abstände vorgeführt wurde [31].

Werden die Koordinaten (x', y') eines Punktes P' (außerhalb des Kegelschnittes gelegen) in (17) eingesetzt, so ergibt sich ein Wert $w \neq 0$:

$$F(x', y', a) = w. \tag{18}$$

Das Vorzeichen von w charakterisiert ein inneres und ein äußeres Gebiet bezüglich des Kegelschnittes. Die Größe w erweist sich als unabhängig von Orthonormaltransformationen T des Koordinatensystems. Sie ist eine invariante Größe:

Ist $(u, v) = T(x, y)$ eine solche Transformation, also etwa

$(u', v') = T(x', y')$,

dann ist

$F\big(T^{-1}(u, v), a\big) = 0$

die transformierte Gleichung, und es gilt

$$F\big(T^{-1}(u', v'), a\big) = F\big(T^{-1}(T(x', y')), a\big) = F(x', y', a) = w,$$

d. h., Einsetzen des transformierten Punktes TP' (außerhalb des Kegelschnittes) in die transformierte Gleichung liefert denselben Wert $w \neq 0$ wie Einsetzen des

Punktes P' in die ursprüngliche Gleichung. Allerdings geht bei einer solchen Transformation die Normierung des Koeffizienten von y^2 verloren.

Ist nun ein Kegelschnitt durch $n \geqq 5$ Punkte gesucht, so löst man das Ausgleichsproblem

$$FQ = \sum_{k=1}^{n} w_k^2 = \text{Min},$$

um sich vom Vorzeichen des w zu befreien. Dies führt auf das folgende lineare Gleichungssystem vom Typ 5×5 in den Unbekannten a_j:

$$\frac{\partial FQ}{\partial a_j} = \sum_{k=1}^{n} w_k \frac{\partial w_k}{\partial a_j} = 0, \quad j = 1(1)5.$$

(Es muß bei der vorliegenden Normierung gesichert sein, daß das Glied mit y^2 existiert.)

Durch sehr einfache Rechnung erhält man aus (17) und (18):

j	1	2	3	4	5
$\frac{\partial w_k}{\partial a_j}$	$x_k y_k$	x_k^2	x_k	y_k	1

Dies liefert das symmetrische Gleichungssystem ausführlich (die Summationen laufen über $k = 1(1)n$, die Summationszeichen sind der Übersichtlichkeit halber weggelassen):

$$\begin{pmatrix} (x_k^2 y_k^2) & (x_k^3 y_k) & (x_k^2 y_k) & (x_k y_k^2) & (x_k y_k) \\ (x_k^3 y_k) & (x_k^4) & (x_k^3) & (x_k^2 y_k) & (x_k^2) \\ (x_k^2 y_k) & (x_k^3) & (x_k^2) & (x_k y_k) & (x_k) \\ (x_k y_k^2) & (x_k^2 y_k) & (x_k y_k) & (y_k^2) & (y_k) \\ (x_k y_k) & (x_k^2) & (x_k) & (y_k) & n \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} -(x_k y_k^3) \\ -(x_k^2 y_k^2) \\ -(x_k y_k^2) \\ -(y_k^3) \\ -(y_k^2) \end{pmatrix}. \tag{19}$$

Seine Lösung liefert in einem Schritt die Koeffizienten des gesuchten Ausgleichskegelschnittes. Ist $n = 5$, so erkennt man die Gleichungen aus (19) als Linearkombinationen der fünf Gleichungen

$$F(x_k, y_k, a) = 0;$$

beispielsweise ist die letzte Gleichung aus (19) die einfache Summe dieser fünf Gleichungen. Der Ausgleichskegelschnitt geht dann durch die gegebenen fünf Punkte. Ist $n < 5$, so hat (19) einen Rangabfall, und die Lösungsmannigfaltigkeit bietet alle Kegelschnitte durch diese n Punkte.

Das skizzierte Verfahren ist erweiterungsfähig auf Flächen zweiter Ordnung und überhaupt auf alle Gleichungen der Bauart

$$F(x, y, \ldots, a) = 0$$

mit linear enthaltenen Komponenten von a und sonst beliebigem Einbau von $x, y, \ldots$

Dieses Verfahren arbeitet besser als eine auf Quadrate der y-Differenzen bezogene Ausgleichsrechnung, da dann das entstehende Gleichungssystem, wie bereits oben gesagt wurde, meist nicht linear ist und iterativ gelöst werden muß, und außerdem sind die y-Differenzen nicht unabhängig von der Lage des Koordinatensystems.

Beispiel

Für die zehn Punkte der Tab. 2.44 wurde der Kegelschnitt mit den Koeffizienten a_i errechnet:

i	a_i
1	−0.78024333
2	1.1910942
3	−6.4466148
4	−4.4277246
5	10.928104

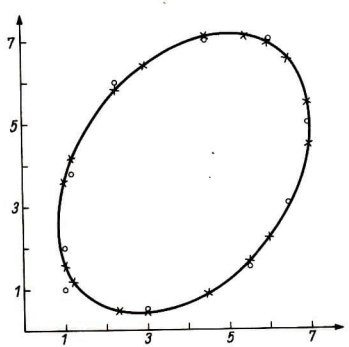


Abb. 2.36. Beste Ellipse durch zehn Punkte entsprechend Tab. 2.43

Es handelt sich offenbar um eine Ellipse, wie Abb. 2.36 ausweist. In Abb. 2.36 sind die Meßwerte mit o und die zu denselben Abszissenwerten gehörenden Ellipsenpunkte mit × markiert. Die Koeffizienten des linearen Gleichungssystems (19) können leicht mit dem ersten Teil des Programms aus Tab. 2.42 berechnet werden. Das System selbst kann mit dem Programm aus Tab. 2.31 gelöst werden.

Tab. 2.44. Meßwerte und errechnete Ausgleichswerte einer besten Ellipse durch zehn Punkte (vgl. Abb. 2.36)

	Meßwert	Kegelschnitt (Ellipse)	
x	y	y_1	y_2
1.0	1.0	3.6566706	1.5512974
1.2	3.8	4.1939034	1.1701132
4.5	7.0	7.0868156	0.85200400
7.0	5.0	5.4783618	4.4110662
5.5	1.5	7.0987270	1.6203360
1.0	2.0	3.6566706	1.5512974
2.3	6.0	5.8088129	0.41347150
6.0	7.0	6.9245090	2.1846756
6.5	3.0	6.5414028	2.9579034
3.0	0.5	6.4082787	0.36017590

2.5.2. Das Problem der Stichproben

Bei statistischen Untersuchungen großer Grundgesamtheiten ist man oft auf Stichproben angewiesen, und das Problem besteht darin, aus der Untersuchung der Stichprobe auf die Grundgesamtheit zu schließen sowie eine Sicherheit dieser Schlußweise anzugeben. Wir betrachten zur Veranschaulichung eine völlig überschaubare Grundgesamtheit von lediglich fünf Elementen. Die erreichten Punktzahlen der Mathematikarbeiten von fünf Schülern seien

4, 5, 6, 7 und 8.

Die maximale Punktzahl wäre 10 gewesen. Sofort errechenbar sind der Mittelwert μ und die Streuung σ^2 der Grundgesamtheit (Bezeichnungen μ und σ unter der Voraussetzung, daß Normalverteilung vorliegt):

$$\mu = (4 + 5 + 6 + 7 + 8)/5 = 6,$$

$$\sigma = \sqrt{(2^2 + 1^2 + 0^2 + 1^2 + 2^2)/5} = 1.41.$$

Entnimmt man aber der Grundgesamtheit eine kleine Stichprobe, beispielsweise hier zwei Elemente — natürlich rein zufällig, und bestimmt deren Mittelwert $\bar{x}$, mit welchen Aussichten kann man dann den wahren Mittelwert μ der Grundgesamtheit treffen? (Die Entnahme der Stichprobe geschieht durch Entnahme eines Elementes, Zurücklegen und Entnahme eines zweiten Elementes. So ist z. B. auch die Stichprobe (8, 8) erklärbar.)

Tab. 2.45 und Abb. 2.37 zeigen, daß der Mittelwert μ der Grundgesamtheit fünfmal getroffen wird. Er hat bei den Stichproben in diesem Beispiel die größte Wahrscheinlichkeit. Für größere Grundgesamtheiten, d. h. mehr als 100 Elemente, und größere Stichproben, d. h. mehr als 30 Probeelemente, gilt dies dann auch allgemein (Abb. 2.38). Würde man für 100 und 30 eine entsprechende Tabelle und Abbildung entwerfen, so erhielte man $100^{30} = 10^{60}$ (statt $5^2 = 25$) Positionen, und die Verteilung

der Mittelwerte $\bar{x}$ der Stichproben nähme die Form der Normalverteilungskurve an. Diese Erkenntnis kann man als Faustregel annehmen.

Betrachtet man die Kästchen in Abb. 2.37, so erkennt man als Chance, mit der Stichprobe den echten Mittelwert erhalten zu haben, $\frac{5}{25} = 20\%$. Das ist zunächst nicht sehr gut. Fragt man aber nach der Chance, einen Mittelwert $\bar{x}$ zu erhalten, der

Tab. 2.45. Zweierstichproben aus einer Grundgesamtheit von fünf Elementen und deren Mittelwerte. Die Stichprobenpaare werden durch *Ziehen, Zurücklegen und wieder Ziehen* gewonnen

8	6	6.5	7	7.5	8
7	5.5	6	6.5	7	7.5
6	5	5.5	6	6.5	7
5	4.5	5	5.5	6	6.5
4	4	4.5	5	5.5	6
	4	5	6	7	8

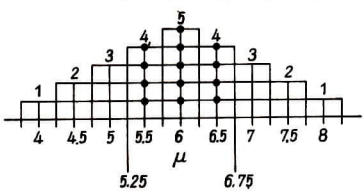


Abb. 2.37. Häufigkeiten der Stichprobenmittelwerte bei Zweierstichproben aus einer Menge von fünf Grundelementen

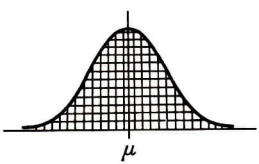


Abb. 2.38. Normalverteilung der Stichprobenmittelwerte $\bar{x}$ bei großen Stichproben (> 30) aus großen Grundgesamtheiten (> 100)

im Bereich $\mu \pm 0.75$, also im Intervall (5.25, 6.75), liegt, so ergibt sich durch Abzählen $\frac{13}{25} = 52\%$, und das ist schon mehr als die Hälfte aller Fälle.

Bei der Normalverteilung (hier der Stichprobenmittelwert $\bar{x}$) — sie genügt übrigens der Formel

$$\varphi(\bar{x}; \mu, \sigma^2) = \frac{1}{\sigma_{\bar{x}}\sqrt{2\pi}} \cdot e^{-\frac{(\bar{x}-\mu)^2}{2\sigma_{\bar{x}}^2}} \tag{20}$$

für die Dichte — gilt, daß im Bereich $(\mu - \sigma_{\bar{x}}, \mu + \sigma_{\bar{x}})$ sogar $68.26\% = 1 - \frac{1}{e}$ aller Werte liegen, die $\bar{x}$ annehmen kann (Abb. 2.39). Das heißt: Wenn man eine größere Stichprobe aus einer größeren Grundgesamtheit entnimmt, so stehen die Chancen 68.26 zu 100, daß der Stichprobenmittelwert im Bereich $\pm\sigma_{\bar{x}}$ des Grundbereichmittelwertes μ liegt. Mit anderen Worten: Man kann zu 68.26% sicher sein, daß der Mittelwert der Grundgesamtheit μ irgendwo im Bereich des Stichprobenmittelwertes $\pm\sigma_{\bar{x}}$ liegt. Dabei kann man $\sigma_{\bar{x}}$ ungefähr mit $\sigma_x/\sqrt{n}$ gleichsetzen, wobei σ_x

die Standardabweichung der Stichprobe ist (bei vielen Rechnern mit der Taste „S.DEV", standard deviation, erhältlich), vorausgesetzt, es war eine größere Stichprobe (> 30). In (20) wäre statt μ korrekt $\mu_{\bar{x}}$ zu schreiben. Aber es gilt $\mu = \mu_{\bar{x}}$, d. h., der Mittelwert der Grundgesamtheit ist gleich dem Mittelwert aller Stichprobenmittelwerte.

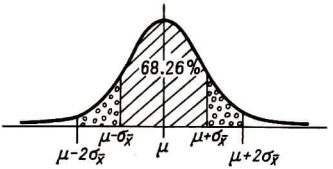


Abb. 2.39. Die Gesamtfläche unter der Kurve der Normalverteilung ist 1 (100%). Im Bereich $\mu \pm \sigma_{\bar{x}}$ liegen 68.26%, im Bereich $\mu \pm 2\sigma_{\bar{x}}$ etwa 95% der Fläche

Die Streuung aller Stichprobenmittelwerte (engl. variance) ist

$$\sigma_{\bar{x}}^2 = \frac{1}{N} \sum_{i=1}^{N} (\bar{x}_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^{N} \bar{x}_i^2 - \mu^2. \tag{21}$$

Die Standardabweichung einer Stichprobe oder *empirische Streuung* (engl. standard deviation) ist

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} \tag{22}$$

und wird auch mit s_n (vgl. (9) und (10)) bezeichnet. Für große Grundgesamtheiten ist deren Streuung recht gut durch die Standardabweichung einer größeren Stichprobe von n Elementen zu schätzen:

$$\sigma_{\bar{x}} \approx \frac{\sigma_x}{\sqrt{n}}. \tag{23}$$

Mittelwertanalyse mit großen Stichproben

Anstelle der hier aus den Eigenschaften der Normalverteilung abgeleiteten Sicherheit von 68.26% kann man für den aus der Stichprobe erhältlichen Mittelwert andere Sicherheiten oder *Vertrauenszahlen* (*Konfidenzzahlen*) wählen und erhält dann das Intervall, in welchem der Mittelwert der Grundgesamtheit (auch *Populationsmittelwert*) liegt. Für eine bequeme Anwendung sind sogenannte z-Werte als Faktoren für die Grenzen des 68.26%-Intervalls in Abhängigkeit von den Konfidenzzahlen k aufgestellt worden. Man wählt eine Konfidenzzahl k und erhält mit dem zugeordneten z, daß der Populationsmittelwert μ im sogenannten *zentralen Konfidenzintervall* liegt:

$$\mu \in \left(\bar{x} \pm \frac{\sigma_x}{\sqrt{n}} z\right). \tag{24}$$

Dabei ist $\bar{x}$ der Mittelwert der Stichprobe aus n Elementen und σ_x deren Standardabweichung gemäß (22). Bei $k = 95\%$, also für hohe Wahrscheinlichkeiten, ist $z \approx 2$ (vgl. Abb. 2.39 für $\mu + 2\sigma_{\bar{x}}$). Bei höherer Sicherheit wird das Intervall natürlich größer. Die zweite Spalte in Tab. 2.46 liefert z-Werte für den Fall, daß man nicht

Tab. 2.46. Konfidenz- oder Vertrauenszahlen k und zugehörige Faktoren z für das 68.26%-Intervall $(\mu \pm \sigma_{\bar{x}})$ in Spalte 1 *Test auf obere und untere Grenze*. In Spalte 2 stehen die Faktoren z für den *Test auf eine Grenze*

k	z-Werte	
	Spalte 1	Spalte 2
60	0.84	0.26
65	0.94	0.39
68.26	1	0.48
70	1.04	0.53
75	1.15	0.68
80	1.28	0.84
85	1.44	1.04
90	1.65	1.28
95	1.96	1.65
99	2.58	2.33

nach dem Intervall fragt, welches das μ enthält (man sagt, *Test auf obere und untere Grenze*), sondern man fragt nur nach der oberen (oder unteren) Grenze, so daß μ im Intervall

$$\left(-\infty, \bar{x} + \frac{\sigma_x}{\sqrt{n}} z\right) \quad \text{bzw.} \quad \left(\bar{x} - \frac{\sigma_x}{\sqrt{n}} z, +\infty\right) \tag{25}$$

liegt (man sagt, *Test auf nur eine Grenze*). Natürlich sind diese z-Werte kleiner, da die andere Intervallseite unbegrenzt ist, d. h., die dort liegende Fläche unter der Verteilungskurve geht insgesamt in die Rechnung ein. Solche Tests mit Stichproben werden vorgenommen, wenn man prüfen will, ob der Mittelwert einer Grundgesamtheit in einem vorgegebenen Intervall liegt bzw. eine Grenze nicht über- oder unterschreitet.

Mittelwertanalyse mit kleinen Stichproben

Ist der Stichprobenumfang $n < 30$, so gilt die Normalverteilung für die Mittelwerte der Stichproben nicht mehr genau genug. Man verwendet dann die sogenannten t-Kurven oder die *Student-Verteilung*, welche die *Gamma-Funktion* $\Gamma(x)$ enthält, die für ganzzahlige Argumente der Fakultät entspricht:

$$T(\bar{x}) = \frac{1}{\sqrt{f\pi}} \frac{\Gamma\left(\frac{f+1}{2}\right)}{\Gamma\left(\frac{f}{2}\right)\left(1 + \frac{\bar{x}^2}{f}\right)^{\frac{f+1}{2}}}; \tag{26}$$

Tab. 2.47. Konfidenz- oder Vertrauenszahlen und zugehörige Faktoren t für obere und untere Grenze $\mu \pm \sigma_x \cdot t/\sqrt{n}$ des Intervalls für den Mittelwert der Grundgesamtheit bei kleinen Stichprobengrößen n

f / $n-1$	t-Werte Konfidenzzahlen				
	80%	90%	95%	99%	99.9%
1	3.078	6.314	12.706	63.657	636.619
2	1.886	2.920	4.303	9.925	31.598
3	1.638	2.353	3.182	5.841	12.941
4	1.533	2.132	2.776	4.604	8.610
5	1.476	2.015	2.571	4.032	6.859
6	1.440	1.943	2.447	3.707	5.959
7	1.415	1.895	2.365	3.499	5.405
8	1.397	1.860	2.306	3.355	5.041
9	1.383	1.833	2.262	3.250	4.781
10	1.372	1.812	2.228	3.169	4.587
11	1.363	1.796	2.201	3.106	4.437
12	1.356	1.782	2.179	3.055	4.318
13	1.350	1.771	2.160	3.012	4.221
14	1.345	1.761	2.145	2.977	4.140
15	1.341	1,753	2.131	2.947	4.073
16	1.337	1.746	2.120	2.921	4.015
17	1.333	1.740	2.110	2.898	3.965
18	1.330	1.734	2.101	2.878	3.922
19	1.328	1.729	2.093	2.861	3.883
20	1.325	1.725	2.086	2.845	3.850
21	1.323	1.721	2.080	2.831	3.819
22	1.321	1.717	2.074	2.819	3.792
23	1.319	1.714	2.069	2.807	3.767
24	1.318	1.711	2.064	2.797	3.745
25	1.316	1.708	2.060	2.787	3.725
26	1.315	1.706	2.056	2.779	3.707
27	1.314	1.703	2.052	2.771	3.690
28	1.313	1 701	2.048	2.763	3.674
29	1.311	1.699	2.045	2.756	3.659
30	1.310	1.697	2.042	2.750	3.646
40	1.303	1.684	2.021	2.704	3.551
60	1.296	1.671	2.000	2.660	3.460
120	1.289	1.658	1.980	2.617	3.373
∞	1.282	1.645	1.960	2.576	3.291

dabei ist f der sogenannte *Freiheitsgrad* und hier um 1 kleiner als die Stichprobengröße, $f = n - 1$, da ein Parameter der Verteilung geschätzt wird:

$$T(\bar{x}) = \frac{1}{\sqrt{(n-1)\,\pi}} \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-1}{2}\right)\left(1 + \frac{\bar{x}^2}{n-1}\right)^{n/2}} \tag{26'}$$

(vgl. MfL Bd. 11, 5.6.2.; bezüglich $\Gamma(x)$ vgl. MfL Bd. 5, 4.3.2.).

Tab. 2.48. Wie Tab. 2.47, aber nur für Test auf eine Grenze (obere oder untere) des Mittelwerts der Grundgesamtheit bei kleinen Stichprobengrößen n

f / $n-1$	t-Werte Konfidenzzahlen			
	90%	95%	99%	99.5%
1	3.078	6.314	31.821	63.657
2	1.886	2.920	6.965	9.925
3	1.638	2.353	4.541	5.841
4	1.533	2.132	3.747	4.604
5	1.476	2.015	3.365	4.032
6	1.440	1.943	3.143	3.707
7	1.415	1.895	2.998	3.499
8	1.397	1.860	2.896	3.355
9	1.383	1.833	2.821	3.250
10	1.372	1.812	2.764	3.169
11	1.363	1.796	2.718	3.106
12	1.356	1.782	2.681	3.055
13	1.350	1.771	2.650	3.012
14	1.345	1.761	2.624	2.977
15	1.341	1.753	2.602	2.947
16	1.337	1.746	2.583	2.921
17	1.333	1.740	2.567	2.898
18	1.330	1.734	2.552	2.878
19	1.328	1.729	2.539	2.861
20	1.325	1.725	2.528	2.845
21	1.323	1.721	2.518	2.831
22	1.321	1.717	2.508	2.819
23	1.319	1.714	2.500	2.807
24	1.318	1.711	2.492	2.797
25	1.316	1.708	2.485	2.787
26	1.315	1.706	2.479	2.779
27	1.314	1.703	2.473	2.771
28	1.313	1.701	2.467	2.763
29	1.311	1.699	2.462	2.756
30	1.310	1.697	2.457	2.750
40	1.303	1.684	2.423	2.704
60	1.296	1.671	2.390	2.660
120	1.289	1.658	2.358	2.617
∞	1.282	1.645	2.326	2.576

Je größer der Freiheitsgrad, d. h. je größer die Stichprobe, desto besser schmiegt sich die Student-Verteilung der Normalverteilung an (Abb. 2.40). Im Grenzfall unendlich großer Stichproben geht die Student-Verteilung in die Normalverteilung über. Praktisch erreicht sie diese bereits bei 30.

Für solche kleinen Stichproben sind ähnlich zu den z-Werten entsprechende t-Werte, aber natürlich in Abhängigkeit von den Freiheitsgraden, tabelliert worden. Die Arbeit mit ihnen verläuft in gleicher Weise.

Die letzte Zeile in Tab. 2.47 enthält einige der z-Werte aus Spalte 1 der vorherigen Tab. 2.46. Auch die letzte Zeile in Tab. 2.48 entspricht wieder den z-Werten, aber denen aus Spalte 2 der Tab. 2.46.

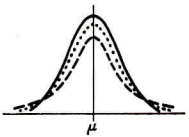


Abb. 2.40. Qualitatives Verhalten der Normalverteilung (durchgezogen) zu Student-Verteilungen mit verschiedenen kleinen Freiheitsgraden (gestrichelt, punktiert)

2.5.3. Anwendungen

Gütetest mit großer Stichprobe und zweiseitigem Test

Eine Schule erhält für Schülerversuche eine größere Anzahl Packungen mit Versuchsmaterial (Chemikalien, Sämereien o. ä.), die nach Angaben des Herstellers 510 g enthalten sollen. Für eine Probe werden 40 Packungen nachgewogen (Packung entnehmen, prüfen und wieder zurücklegen!), und es stellt sich ein Mittelwert

$$\bar{x} = 508.75 \text{ g}$$

sowie eine Standardabweichung (22)

$$\sigma_x = 19.97 \text{ g}$$

heraus. Sofern diese überaus große Streuung nicht schon Anlaß für eine Reklamation ist, will die Schulleitung zu 95% sicher sein, daß die festgestellte Abweichung vom Mittelwert zu einer Reklamation berechtigt. Man entnimmt der Tab. 2.46 für zweiseitigen Test und für die Konfidenzgröße 95% den Faktor 1.96, und damit liegt mit 95% Sicherheit der Mittelwert μ der Grundgesamtheit im Intervall

$$\left(508.75 \pm \frac{19.97}{\sqrt{40}} \cdot 1.96\right) = (502.56,\ 514.94).$$

Da der vom Hersteller angegebene Mittelwert 510 g in diesem Intervall tatsächlich enthalten ist, liegt kein Anlaß für eine Reklamation vor, obwohl der Mittelwert der Stichprobe einen kleineren Wert zeigt.

Gütetest mit großer Stichprobe und einseitigem Test

Für den Physikunterricht werden zu Schulversuchen $N = 5000$ Batterien geliefert, für die der Hersteller eine Betriebszeit von 180 Stunden angibt. Zur Überprüfung werden 50 Stück getestet. Es ergibt sich ein Mittelwert

$$\bar{x} = 175 \text{ h}$$

und eine Standardabweichung (22)

$$\sigma_x = 18 \text{ h}.$$

Hier ist die Streuung relativ sogar noch größer als im ersten Beispiel und könnte für sich bereits Anlaß zu einer Reklamation sein. (Man sieht, daß zur Gütecharakteristik vom Hersteller nicht nur ein Mittelwert, sondern auch eine Streuungsangabe erforderlich wäre.) Man will nun bezüglich der gefundenen Abweichung vom Mittelwert wieder 95% Sicherheit für eine Reklamation haben. Die Schulleitung beauftragt den Physiklehrer, dies zu prüfen. Es liegt hier im Vergleich zur oben geschilderten Theorie aber noch eine Abweichung vor. Der Test zerstört die Batterien. Die Stichprobe konnte nicht durch „Entnehmen, Prüfen, Zurücklegen, Entnehmen, . . ." gewonnen werden. Es ist also ein Korrekturfaktor nötig. Es ist außerdem ein Test mit nur einer Grenze, denn eine größere Lebensdauer als 180 Stunden ist kein Grund für eine Reklamation. Der Tab. 2.46, Spalte 2, entnimmt man nun den Faktor $z = 1.65$. Damit wird

$$\mu \in \left(\bar{x} \pm \frac{N-n}{N-1}\,\frac{\sigma_x}{\sqrt{n}}\,z\right) = (172.06,\ 177.94), \tag{27}$$

und dieser Bereich enthält nicht die angegebenen 180 Stunden. Eine Reklamation ist mit 95% Sicherheit berechtigt.

Gütetest mit kleiner Stichprobe und zweiseitigem Test

In einem chemischen Betrieb, in dem Schüler zum „Unterricht in der Produktion (UTP)" arbeiten, wird Farbe hergestellt, die pro Kilogramm 15.5 g der farbgebenden Substanz enthalten soll. Die Schüler führen zur Kontrolle die etwas aufwendige Analyse für diese 15.5 g durch und erhalten bei den entnommenen acht Proben die Werte

15.2, 15.7, 15.8, 15.6, 15.0, 15.9, 16.1, 15.9.

Der Mittelwert ist mit

$$\bar{x} = 15.65\ \text{g}$$

etwas zu hoch und die Standardabweichung (22)

$$\sigma_x = 0.37\ \text{g}$$

weist auf eine kleine Streuung, so daß von den Schülern angenommen wird, der automatische Produktionsprozeß sei falsch eingestellt, nämlich auf zu hohe Beimengung der Farbsubstanz. Der Lehrer prüft, ob mit diesen geringen Daten (acht Proben) eine so schwerwiegende Mitteilung an den Betrieb zu geben ist. Es ist hier $f = n - 1 = 7$, er wählt 90% Sicherheit und entnimmt der Tab. 2.47 den Faktor $t = 1.895$; damit ist

$$\mu \in \left(\bar{x} \pm \frac{\sigma_x}{\sqrt{n}}\,t\right) = (15.40,\ 15.90),$$

und dieses Intervall enthält auch 15.5 g. Damit kann mit 90% Sicherheit der Fertigungsprozeß mit der geltenden Einstellung weiterlaufen.

Gütetest mit kleiner Stichprobe und einseitigem Test

Für biologische Schülerversuche soll ein dabei verwendetes Präparat höchstens 8% Alkohol enthalten. Aus fünf Flaschen werden Stichproben entnommen, und die Analyse ergibt

$$7.85\%,\quad 8.31\%,\quad 8.33\%,\quad 7.76\%,\quad 7.97\%;$$

zwei Werte liegen recht deutlich über 8%. Soll beim Hersteller des Lehrmittels reklamiert werden? Eine Reklamation soll auf Wunsch der Schulleitung zu 95% sicher sein. Der Biologielehrer errechnet als Mittelwert der Stichprobe (diese Mittelwertbildung aus Prozentzahlen ist berechtigt, da in allen Flaschen die gleiche Menge als Füllung vorausgesetzt werden kann)

$$\bar{x} = 8.04\%$$

und als Standardabweichung (22)

$$\sigma_x = 0.26\%.$$

Für $n = 5$, d. h. $f = 4$, und die geforderte Konfidenz 95% entnimmt er der Tab. 2.48 den Wert $t = 2.132$ (einseitiger Test). Daraus ergibt sich

$$\mu \in (7.79\%,\ 8.29\%),$$

und da die angegebenen 8% in diesem Intervall liegen, kann keine Reklamation mit 95% Sicherheit angebracht werden. Für eine berechtigte Reklamation müßte der gesamte μ-Bereich über 8% liegen.

Gütetest mit Alternative

Es kommt öfter vor, daß ein Gütetest auf der Grundlage einer ja-nein-Entscheidung (*funktionieren* oder *versagen* bei Bauelementen) vorgenommen werden muß. Es gibt also keine Maßzahl für einen Mittelwert. Es ist vielmehr ein Test für eine unbekannte Wahrscheinlichkeit (hier des Versagens) durchzuführen (vgl. MfL Bd. 11, 11.4.5.). Der Hersteller z. B. von Mikroelektronikchips gibt an, daß nicht mehr als 12% Versager in der Lieferung sind. Das ist ein Mengenverhältnis oder eine relative Häufigkeit als Schätzung für die Wahrscheinlichkeit. Nun ergab etwa eine Kontrolle von $n = 250$ Chips 43 Versager, deutlich mehr als 12%, nämlich $\bar{p} = 17.2\%$. Wenn eine Reklamation zu 90% berechtigt sein soll, müssen die angegebenen $p = 12\%$ im Intervall

$$\left(\bar{p} \pm \sqrt{\frac{\bar{p}(1-\bar{p})}{n}}\, z\right) \tag{28}$$

liegen (es ist dies eine modifizierte Formel (24) für Mengenverhältnisse, $\bar{p} = \frac{43 \cdot 100}{250}$).

Der Wert z wird aus Tab. 2.46 für einseitigen Test (große Probe, Spalte 2) mit $z = 1.28$ entnommen. Es ergibt sich

$$p \notin (14.1\%,\ 20.3\%).$$

Mit 90% Sicherheit ist eine Reklamation berechtigt.

Es könnte nun sein, daß der Werkleitung 90% Sicherheit nicht genügen. Die Chips stammen etwa aus Importen, und eine Reklamation ist aufwendig und verursacht auch Ärger, wenn sie sich doch als unberechtigt herausstellt. Es werden also 95% verlangt. Das Konfidenzintervall vergrößert sich natürlich, denn nun gilt der Faktor $z = 1.65$, und damit wird

$$p \notin (13.3\%,\ 21.1\%),$$

so daß auch jetzt noch mit 95% Sicherheit die Herstellerangabe von $p = 12\%$ Versagern übertrieben günstig ist. Eine Reklamation ist berechtigt.

Test auf Veränderungen

Statistische Untersuchungen haben oft das Ziel, die Auswirkung einer Maßnahme zu prüfen, d. h. eine Veränderung festzustellen. Das kann z. B. die Auswirkung einer neuen Lehrmethode sein, gemessen in den Ergebnissen von Schülerarbeiten. Das kann die Änderung im Heilerfolg bei Anwendung eines neuen Arzneimittels sein. Das kann die Ertragssteigerung bei Anwendung eines neuen Düngers, einer neuen Bearbeitungsmethode, einer neuen Saatgutsorte sein. Das kann eine Produktivitätssteigerung auf der Basis eines Verbesserungsvorschlages sein. Viele andere Beispiele ließen sich aus vielen Bereichen anschließen. Für solche Fragestellungen hat die Statistik den *F-Test* (FISHER) und die *Konfidenzintervall-Methode* entwickelt. Es gibt dafür verschiedene Prüfverteilungen (χ^2-, t-, F-Verteilung; vgl. MfL Bd. 11, 5.6.), und die entsprechenden Tests führen dann die zugeordneten Namen (χ^2-, t-, F-Test; vgl. MfL Bd. 11, 11.4. und 11.5.). In den folgenden Beispielen könnte auch ein sogenannter χ^2-Homogenitätstest (vgl. Mfl Bd. 11, 11.5.3.) oder ein anderer verteilungsfreier Test angewendet werden. Hier aber sollen normalverteilte Versuchsreihen angenommen werden.

Beispiele

1. Zur Prüfung eines Verbesserungsvorschlages werden $n = 6$ Experimente nach der alten und der neuen Variante vorgenommen. Es ergab sich:

alt	neu
3.68	2.68
1.28	0.45
1.84	0.92
3.68	1.69
1.83	0.05
6.00	0.16
$\bar{x} = 3.05$	$\bar{x} = 0.99$
$\sigma_x = 1.76$	$\sigma_x = 1.02$

Aus den Mittelwerten leitet der Einreicher eine Senkung (etwa des Energieverbrauches) auf 1/3 bei niedrigerer Standardabweichung (22) ab (Tab. 2.49).

Der F-Test verlangt Unterscheidung der Daten in *hohe* und *niedrige*, je nach den Werten der Standardabweichung. Die Kennzeichnung erfolgt durch H = hoch,

Tab. 2.49a. Testwerte für den Fisher-Test auf Änderung für 95% Sicherheit

Freiheitsgrade des Nenners	Freiheitsgrade des Zählers																
	1	2	3	4	5	6	7	8	9	10	12	15	20	30	60	120	∞
1	161.4	199.5	215.7	224.6	230.2	234.0	236.8	238.9	240.5	241.9	243.9	245.9	248.0	250.1	252.2	253.3	254.3
2	18.51	19.00	19.16	19.25	19.30	19.33	19.35	19.37	19.38	19.40	19.41	19.43	19.45	19.46	19.48	19.49	19.50
3	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81	8.79	8.74	8.70	8.66	8.62	8.57	8.55	8.53
4	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6.00	5.96	5.91	5.86	5.80	5.75	5.69	5.66	5.63
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77	4.74	4.68	4.62	4.56	4.50	4.43	4.40	4.36
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10	4.06	4.00	3.94	3.87	3.81	3.74	3.70	3.67
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68	3.64	3.57	3.51	3.44	3.38	3.30	3.27	3.23
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39	3.35	3.28	3.22	3.15	3.08	3.01	2.97	2.93
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18	3.14	3.07	3.01	2.94	2.86	2.79	2.75	2.71
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02	2.98	2.91	2.85	2.77	2.70	2.62	2.58	2.54
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90	2.85	2.79	2.72	2.65	2.57	2.49	2.45	2.40
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80	2.75	2.69	2.62	2.54	2.47	2.38	2.34	2.30
13	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.77	2.71	2.67	2.60	2.53	2.46	2.38	2.30	2.25	2.21
14	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.70	2.65	2.60	2.53	2.46	2.39	2.31	2.22	2.18	2.13
15	4.54	3.68	3.29	3.06	2.90	2.79	2.71	2.64	2.59	2.54	2.48	2.40	2.33	2.25	2.16	2.11	2.07
16	4.49	3.63	3.24	3.01	2.85	2.74	2.66	2.59	2.54	2.49	2.42	2.35	2.28	2.19	2.11	2.06	2.01
17	4.45	3.59	3.20	2.96	2.81	2.70	2.61	2.55	2.49	2.45	2.38	2.31	2.23	2.15	2.06	2.01	1.96
18	4.41	3.55	3.16	2.93	2.77	2.66	2.58	2.51	2.46	2.41	2.34	2.27	2.19	2.11	2.02	1.97	1.92
19	4.38	3.52	3.13	2.90	2.74	2.63	2.54	2.48	2.42	2.38	2.31	2.23	2.16	2.07	1.98	1.93	1.88
20	4.35	3.49	3.10	2.87	2.71	2.60	2.51	2.45	2.39	2.35	2.28	2.20	2.12	2.04	1.95	1.90	1.84
21	4.32	3.47	3.07	2.84	2.68	2.57	2.49	2.42	2.37	2.32	2.25	2.18	2.10	2.01	1.92	1.87	1.81
22	4.30	3.44	3.05	2.82	2.66	2.55	2.46	2.40	2.34	2.30	2.23	2.15	2.07	1.98	1.89	1.84	1.78
23	4.28	3.42	3.03	2.80	2.64	2.53	2.44	2.38	2.32	2.27	2.20	2.13	2.05	1.96	1.86	1.81	1.76
24	4.26	3.40	3.01	2.78	2.62	2.51	2.42	2.36	2.30	2.25	2.18	2.11	2.03	1.94	1.84	1.79	1.73
25	4.24	3.39	2.99	2.76	2.60	2.49	2.40	2.34	2.28	2.24	2.16	2.09	2.01	1.92	1.82	1.77	1.71
26	4.23	3.37	2.98	2.74	2.59	2.47	2.39	2.32	2.27	2.22	2.15	2.07	1.99	1.90	1.80	1.75	1.69
27	4.21	3.35	2.96	2.73	2.57	2.46	2.37	2.31	2.25	2.20	2.13	2.06	1.97	1.88	1.79	1.73	1.67
28	4.20	3.34	2.95	2.71	2.56	2.45	2.36	2.29	2.24	2.19	2.12	2.04	1.96	1.87	1.77	1.71	1.65
29	4.18	3.33	2.93	2.70	2.55	2.43	2.35	2.28	2.22	2.18	2.10	2.03	1.94	1.85	1.75	1.70	1.64
30	4.17	3.32	2.94	2.69	2.53	2.42	2.33	2.27	2.21	2.16	2.09	2.01	1.93	1.84	1.74	1.68	1.62
40	4.08	3.23	2.84	2.61	2.45	2.34	2.25	2.18	2.12	2.08	2.00	1.92	1.84	1.74	1.64	1.58	1.51
60	4.00	3.15	2.76	2.53	2.37	2.25	2.17	2.10	2.04	1.99	1.92	1.84	1.75	1.65	1.53	1.47	1.39
120	3.92	3.07	2.68	2.45	2.29	2.17	2.09	2.02	1.96	1.91	1.83	1.75	1.66	1.55	1.43	1.35	1.25
∞	3.84	3.00	2.60	2.37	2.21	2.10	2.01	1.94	1.88	1.83	1.75	1.67	1.57	1.46	1.32	1.22	1.00

Tab. 2.49b. Testwerte für den Fisher-Test auf Änderung für 99% Sicherheit

Freiheitsgrade des Nenners	Freiheitsgrade des Zählers																
	1	2	3	4	5	6	7	8	9	10	12	15	20	30	60	120	∞
1	4052	5000	5403	5625	5764	5859	5928	5982	6022	6056	6106	6157	6209	6261	6313	6339	6366
2	98.50	99.00	99.17	99.25	99.30	99.33	99.36	99.37	99.39	99.40	99.42	99.43	99.45	99.47	99.48	99.49	99.50
3	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49	27.35	27.23	27.05	26.87	26.69	26.50	26.32	26.22	26.13
4	21.20	18.00	16.69	15.98	15.52	15.21	14.98	14.80	14.66	14.55	14.37	14.20	14.02	13.84	13.65	13.56	13.46
5	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29	10.16	10.05	9.89	9.72	9.55	9.38	9.20	9.11	9.02
6	13.75	10.92	9.78	9.15	8.75	8.47	8.26	8.10	7.98	7.87	7.72	7.56	7.40	7.23	7.06	6.97	6.88
7	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84	6.72	6.62	6.47	6.31	6.16	5.99	5.82	5.74	5.65
8	11.26	8.65	7.59	7.01	6.63	6.37	6.18	6.03	5.91	5.81	5.67	5.52	5.36	5.20	5.03	4.95	4.86
9	10.56	8.02	6.99	6.42	6.06	5.80	5.61	5.47	5.35	5.26	5.11	4.96	4.81	4.65	4.48	4.40	4.31
10	10.04	7.56	6.55	5.99	5.64	5.39	5.20	5.06	4.94	4.85	4.71	4.56	4.41	4.25	4.08	4.00	3.91
11	9.65	7.21	6.22	5.67	5.32	5.07	4.89	4.74	4.63	4.54	4.40	4.25	4.10	3.94	3.78	3.69	3.60
12	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.50	4.39	4.30	4.16	4.01	3.86	3.70	3.54	3.45	3.36
13	9.07	6.70	5.74	5.21	4.86	4.62	4.44	4.30	4.19	4.10	3.96	3.82	3.66	3.51	3.34	3.25	3.17
14	8.86	6.51	5.56	5.04	4.69	4.46	4.28	4.14	4.03	3.94	3.80	3.66	3.51	3.35	3.18	3.09	3.00
15	8.68	6.36	5.42	4.89	4.56	4.32	4.14	4.00	3.89	3.80	3.67	3.52	3.37	3.21	3.05	2.96	2.87
16	8.53	6.23	5.29	4.77	4.44	4.20	4.03	3.89	3.78	3.69	3.55	3.41	3.26	3.10	2.93	2.84	2.75
17	8.40	6.11	5.18	4.67	4.34	4.10	3.93	3.79	3.68	3.59	3.46	3.31	3.16	3.00	2.83	2.75	2.65
18	8.29	6.01	5.09	4.58	4.25	4.01	3.84	3.71	3.60	3.51	3.37	3.23	3.08	2.92	2.75	2.66	2.57
19	8.18	5.93	5.01	4.50	4.17	3.94	3.77	3.63	3.52	3.43	3.30	3.15	3.00	2.84	2.67	2.58	2.49
20	8.10	5.85	4.94	4.43	4.10	3.87	3.70	3.56	3.46	3.37	3.23	3.09	2.94	2.78	2.61	2.52	2.42
21	8.02	5.78	4.87	4.37	4.04	3.81	3.64	3.51	3.40	3.31	3.17	3.03	2.88	2.72	2.55	2.46	2.36
22	7.95	5.72	4.82	4.31	3.99	3.76	3.59	3.45	3.35	3.26	3.12	2.98	2.83	2.67	2.50	2.40	2.31
23	7.88	5.66	4.76	4.26	3.94	3.71	3.54	3.41	3.30	3.21	3.07	2.93	2.78	2.62	2.45	2.35	2.26
24	7.82	5.61	4.72	4.22	3.90	3.67	3.50	3.36	3.26	3.17	3.03	2.89	2.74	2.58	2.40	2.31	2.21
25	7.77	5.57	4.68	4.18	3.85	3.63	3.46	3.32	3.22	3.13	2.99	2.85	2.70	2.54	2.36	2.27	2.17
26	7.72	5.53	4.64	4.14	3.82	3.59	3.42	3.29	3.18	3.09	2.96	2.81	2.66	2.50	2.33	2.23	2.13
27	7.68	5.49	4.60	4.11	3.78	3.56	3.39	3.26	3.15	3.06	2.93	2.78	2.63	2.47	2.29	2.20	2.10
28	7.64	5.45	4.57	4.07	3.75	3.53	3.36	3.23	3.12	3.03	2.90	2.75	2.60	2.44	2.26	2.17	2.06
29	7.60	5.42	4.54	4.04	3.73	3.50	3.33	3.20	3.09	3.00	2.87	2.73	2.57	2.41	2.23	2.14	2.03
30	7.56	5.39	4.51	4.02	3.70	3.47	3.30	3.17	3.07	2.98	2.84	2.70	2.55	2.39	2.21	2.11	2.01
40	7.31	5.18	4.31	3.83	3.51	3.29	3.12	2.99	2.89	2.80	2.66	2.52	2.37	2.20	2.02	1.92	1.80
60	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82	2.72	2.63	2.50	2.35	2.20	2.03	1.84	1.73	1.60
120	6.85	4.79	3.95	3.48	3.17	2.96	2.79	2.66	2.56	2.47	2.34	2.19	2.03	1.86	1.66	1.53	1.38
∞	6.63	4.61	3.78	3.32	3.02	2.80	2.64	2.51	2.41	2.32	2.18	2.04	1.88	1.70	1.47	1.32	1.00

N = niedrig. Der F-Test (vgl. MfL Bd. 11, 11.4.4.) verlangt unter der Voraussetzung, daß normalverteilte Versuchsreihen vorliegen, dann weiter den Vergleich des Quotienten

$$Q = \frac{\text{Zähler}}{\text{Nenner}} = \frac{\sigma_{xH}^2}{\sigma_{xN}^2} \approx \frac{3.12}{1.04} \approx 2.99 \tag{29}$$

mit vorgefertigten F-Werten in Abhängigkeit von Konfidenzangaben. Es spielen auch die Freiheitsgrade $f = n - 1$ in den beiden Versuchsreihen eine Rolle. Für $k = 95\%$ und $f = 5$ in Zähler und Nenner findet man in Tab. 2.49a

$$F = 5.05.$$

Ist $F > Q$, dann ist der F-Test positiv ausgefallen, was also hier der Fall ist. Man kann nun sofort den Verbesserungsbereich errechnen (doppelter t-Test; vgl. MfL Bd. 11, 11.4.2.):

$$\Delta\bar{x} = (\bar{x}_H - \bar{x}_N) \pm \sqrt{\left(\frac{f_H\sigma_{xH}^2 + f_N\sigma_{xN}^2}{f_H + f_N}\right)\left(\frac{1}{n_H} + \frac{1}{n_N}\right)} \cdot t. \tag{30}$$

Dabei ist $t = 2.228$ der Testwert aus der Student-Tabelle (Tab. 2.47) für $f = f_H + f_N = 10$ und $k = 95\%$. Es ergibt sich somit

$$\Delta\bar{x} \in (2.06 \pm 1.86) = (0.20,\ 3.92),$$

und das heißt, daß das neue Verfahren mindestens eine Verbesserung um 0.20 und höchstens eine von 3.92 bei 95% Sicherheit bringen wird. (Der Maximalwert 3.95 ist hier physikalisch ohne Sinn.)

2. Zwei Versuchsreihen ergaben:

	neu	alt
n	13	9
$\bar{x}$	110.02	101.58
σ_x	9.91	2.86
σ_x^2	98.24	8.16
	„hoch"	„niedrig"

Ist eine Verbesserung (Steigerung) in $\bar{x}$ erzielt worden? Es sieht ganz so aus. Der neue Mittelwert liegt um fast 10% höher, allerdings ist auch die Standardabweichung größer.

Es wird $k = 99\%$ gewählt, d. h., man will sehr sicher sein. Dann ist

$$Q = \frac{98.24}{8.16} \approx 12.04,$$

und in der F-Tabelle 2.49b für $k = 99\%$ steht bei zwölf Freiheitsgraden im Zähler und acht im Nenner

$$F = 5.67.$$

Hier ist nun $F < Q$ und damit der Test nicht erfüllt, d. h., es muß bei der Konfidenzintervall-Berechnung noch eine Korrektur angebracht werden. Diese Korrektur besteht in der Festlegung einer neuen Anzahl von Freiheitsgraden f_K nach (31):

$$f_K = \frac{1}{\dfrac{K^2}{f_H} + \dfrac{(1-K)^2}{f_N}} \quad \text{mit} \quad K = \frac{\sigma_{xH}^2/n_H^2}{\sigma_{xH}^2/n_H + \sigma_{xN}^2/n_N}. \tag{31}$$

Im Zahlenbeispiel ergibt sich

$$K = 0.89 \quad \text{und} \quad f_K = 14.73,$$

und dieses f_K wird statt $13 + 9 - 2 = 20$ zum Suchen des t-Wertes in Tab. 2.47 verwendet. Man erhält durch Interpolation in der Spalte für $k = 99\%$

$$t = 2.96.$$

Damit ergibt sich dann nach (30)

$$\Delta\bar{x} \in (8.44 \pm 10.12) = (-1.67,\ 18.55).$$

Der Unterschied 0 zwischen den Versuchsreihen ist darin enthalten, also kann nicht mit 99%iger Sicherheit auf eine bessere neue Reihe geschlossen werden.

Da aber der Wert 0 nahe am Rand des Intervalls liegt, kann man bei Verkürzung des Intervalls auf einen deutlichen oder *signifikaten* Unterschied hoffen. Solche Verkürzung erhält man durch kleineres t und damit Verzicht auf große Sicherheit.

Verzichtet man also auf die Schärfe 99% und ist mit 95% oder gar nur 90% zufrieden, so ergeben sich t-Werte von

$$t_{95} = 2.13, \quad t_{90} = 1.76$$

(wegen der F-Werte von $F_{95} = 3.28$ — und F_{90} noch kleiner — wird an der Korrektur zu einem f_K nichts geändert). Diese t-Werte verkleinern das Intervall,

$$\Delta\bar{x}_{95} \in (8.44 \pm 7.28) = (1.16,\ 15.72),$$

$$\Delta\bar{x}_{90} \in (8.44 \pm 6.01) = (2.43,\ 14.45),$$

und man erhält mit der verringerten Sicherheit eine signifikante Veränderung des Mittelwertes.

3. Ergänzungen zu Kleinstrechnern und einige mikroelektronische Grundlagen

3.1. Näherungsformeln für elementare transzendente Funktionen für den Vierspeziesrechner

Für den Gebrauch des Vierspeziesrechner, eines Taschenrechners also, der nur das Addieren, Subtrahieren, Multiplizieren und Dividieren kann, ergibt sich für den Einsatz zur Berechnung elementarer transzendenter Funktionen ein besonderes Approximationsproblem. Es sollen nicht etwa die transzendenten Funktionen Sinus, Kosinus, Tangens, Arcusfunktionen, Logarithmus und Exponentialfunktion nur möglichst genau berechnet werden — das ist selbstverständlich möglich —, sondern es sollen diese Funktionswerte mit *möglichst wenig Tastendrücken* im Rahmen einer bestimmten Genauigkeit zur Verfügung gestellt werden. Während man sonst bei ähnlichen Approximationsaufgaben, etwa in Großcomputern, die gewünschte Genauigkeit durch Hinzunahme weiterer Glieder in der Approximationsfunktion erreicht, wobei die Koeffizienten stets in der Stellenzahl vorhanden sind, die die Approximation verlangt, ist es gerade diese Stellenzahl, welche den Einsatz derselben Approximationsformel im Taschenrechner verbietet. Man kann nicht pro Koeffizient zehn oder mehr Tastendrücke zulassen. Dabei müßte man außerdem ständig die Approximationsformel auch vor Augen haben, weil man sich drei oder vier zehnstellige Zahlen einfach nicht merken kann. Es ergibt sich somit durch den *Vierspeziesrechner* eine ganz neue Approximationsaufgabe: Konstruktion von Ersatzformeln mit ein-, höchstens zweistelligen ganzen (Vermeiden der Komma- oder Dezimalpunkttaste) Koeffizienten darin und möglichst wenig Tastendrücke! Da man kaum mit einmaligem Verwenden des Funktionsarguments in der Formel auskommen wird, empfiehlt sich wenigstens ein Speicher, d. h., Rechner mit der Funktion M+, M− u. ä. (einer Speicherzelle *memory*) sind für diese Aufgabe besser gerüstet als die ganz primitiven. Ein noch größerer Vorteil ergibt sich für Geräte mit der Wurzel- und Kehrwertfunktion (reziproker Wert). Das trifft beispielsweise für SR1, MR411 und MR610 zu.

Ein weiterer Grund für notwendige Kenntnis über und den Einsatz von Ersatzformeln ist in der bedauerlichen Tatsache zu sehen, daß viele Taschenrechner, darunter leider auch der SR1 und der MR610, elementare transzendente Funktionen in ganz bestimmten kritischen Argumentbereichen nicht nur ungenau, sondern schlicht falsch berechnen. Ein Beispiel dafür ist die Logarithmusfunktion in der Nähe von 1, und zwar dicht unterhalb 1. Nach 3.1.6., (23) muß im Rahmen der Genauigkeit

$$\ln(1 - x) = -x$$

gelten. Wenn man 9.9999999_{10}-1 eintippt und eine Ziffer $(1 \ldots 9)_{10}$-9 addiert, kann man sich davon mittels der auf S. 47/48 beschriebenen Methode überzeugen. Statt der nun erwarteten ln-Werte $-(9 \ldots 1)_{10}$-09 werden falsch die Werte $-(2.0723 \ldots$

Tab. 3.1. Die Tabelle zeigt die Abweichungen der Ersatzformel für sin (a) gegenüber den exakten Werten. Absolute und relative Fehler sind angegeben. Im Bereich $0 \leqq a \leqq \pi/6$ (30°) ist der Fehler kleiner als 1‰, und selbst bei $\pi/2$ (90°) wird 0.5% noch nicht überschritten

a (Bogen)	a (Grad)	$\cos(\pi/2 - a)$ / $\sin(a)$ Ersatzformel (1)	$\cos(\pi/2 - a)$ / $\sin(a)$ exakt (2)	$\sin(a)$ (Tastenfunktion)	$\cos(\pi/2 - a)$ (Tastenfunktion)	absoluter Fehler (2)−(1)	relativer Fehler %
0	0	0	0	0	0	0	0
10^{-10}	$5.7295779 \cdot 10^{-9}$	10^{-10}	10^{-10}	10^{-10}	0	0	0
$5 \cdot 10^{-9}$	$2.8647889 \cdot 10^{-7}$	$5 \cdot 10^{-9}$	$5 \cdot 10^{-9}$	$5 \cdot 10^{-9}$	$5.119720032 \cdot 10^{-9}$	0	0
$5 \cdot 10^{-8}$	$2.8647889 \cdot 10^{-6}$	$5 \cdot 10^{-8}$	$5 \cdot 10^{-8}$	$5 \cdot 10^{-8}$	$5.017325631 \cdot 10^{-8}$	0	0
$5 \cdot 10^{-7}$	$2.8647889 \cdot 10^{-5}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$5.000942523 \cdot 10^{-7}$	0	0
$5 \cdot 10^{-6}$	$2.8647889 \cdot 10^{-4}$	$5 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	$5.000055109 \cdot 10^{-6}$	0	0
$1 \cdot 10^{-5}$	$5.7295779 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1.000017848 \cdot 10^{-5}$	0	0
$2 \cdot 10^{-5}$	$1.1459156 \cdot 10^{-3}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2.000001565 \cdot 10^{-5}$	0	0
$5 \cdot 10^{-5}$	$2.8647889 \cdot 10^{-3}$	$5 \cdot 10^{-5}$	$4.999999998 \cdot 10^{-5}$	$4.999999996 \cdot 10^{-5}$	$5.000014163 \cdot 10^{-5}$	$-2 \cdot 10^{-14}$	$4 \cdot 10^{-8}$
$1 \cdot 10^{-4}$	$5.7295779 \cdot 10^{-3}$	$9.999999967 \cdot 10^{-5}$	$9.999999983 \cdot 10^{-5}$	$9.999999950 \cdot 10^{-5}$	$1.000001469 \cdot 10^{-4}$	$16 \cdot 10^{-14}$	$16 \cdot 10^{-8}$
$2 \cdot 10^{-4}$	$1.1459156 \cdot 10^{-2}$	$1.999999987 \cdot 10^{-4}$	$1.999999987 \cdot 10^{-4}$	$1.999999980 \cdot 10^{-4}$	$2.000001585 \cdot 10^{-4}$	0	0
$5 \cdot 10^{-4}$	$2.8647889 \cdot 10^{-2}$	$4.999999783 \cdot 10^{-4}$	$4.999999792 \cdot 10^{-4}$	$4.999998875 \cdot 10^{-4}$	$5.000001303 \cdot 10^{-4}$	$9 \cdot 10^{-12}$	$2 \cdot 10^{-7}$
0.001	$5.7295779 \cdot 10^{-2}$	$9.999998333 \cdot 10^{-4}$	$9.999998333 \cdot 10^{-4}$	$9.999998000 \cdot 10^{-4}$	$9.999999850 \cdot 10^{-4}$	0	0
0.002	0.11459156	$1.999998667 \cdot 10^{-3}$	$1.999998667 \cdot 10^{-3}$	$1.999998600 \cdot 10^{-3}$	$1.999998787 \cdot 10^{-3}$	0	0
0.005	0.28647890	$4.999979167 \cdot 10^{-3}$	$4.999979167 \cdot 10^{-3}$	$4.999979100 \cdot 10^{-3}$	$4.999979278 \cdot 10^{-3}$	0	0
0.01	0.57295780	$9.999833333 \cdot 10^{-3}$	$9.999833333 \cdot 10^{-3}$	$9.999833318 \cdot 10^{-3}$	$9.999833468 \cdot 10^{-3}$	0	0
0.02	1.14591560	$1.999866669 \cdot 10^{-2}$	$1.999866670 \cdot 10^{-2}$	$1.999866664 \cdot 10^{-2}$	$1.999866680 \cdot 10^{-2}$	$1 \cdot 10^{-11}$	$2 \cdot 10^{-8}$
0.05	2.8647890	$4.997916927 \cdot 10^{-2}$	$4.997916927 \cdot 10^{-2}$	$4.997916923 \cdot 10^{-2}$	$4.997916933 \cdot 10^{-2}$	0	0
0.1	5.7295780	$9.983341663 \cdot 10^{-2}$	$9.983341663 \cdot 10^{-2}$	$9.983341660 \cdot 10^{-2}$	$9.983341690 \cdot 10^{-2}$	0	0
0.2	11.459156	$1.986693280 \cdot 10^{-1}$	$1.986693309 \cdot 10^{-1}$	$1.986693307 \cdot 10^{-1}$	$1.986693308 \cdot 10^{-1}$	$29 \cdot 10^{-10}$	$15 \cdot 10^{-7}$
0.3	17.188734	0.29552015 93	0.29552020 67	exakte Werte	exakte Werte	$41 \cdot 10^{-8}$	0.00012
0.4	22.918311	0.38941789 93	0.38941834 23	↓	↓	$36 \cdot 10^{-8}$	0.00009
0.52359878	30	0.49999769 77	0.5			$23 \cdot 10^{-7}$	0.00046
0.6	34.377468	0.56463654 23	0.56464247 34			$59 \cdot 10^{-7}$	0.0011
0.7	40.107046	0.64420042 30	0.64421767 83			$17 \cdot 10^{-6}$	0.0027
0.78539816	45	0.70706853 33	0.70710678 10			$38 \cdot 10^{-6}$	0.0053
0.9	51.566202	0.78322921 67	0.78332690 95			$10 \cdot 10^{-5}$	0.013
1.0471976	60	0.86574976 77	0.86602540 40			$28 \cdot 10^{-5}$	0.032
1.1	63.025357	0.89082193 93	0.89120735 98			$38 \cdot 10^{-5}$	0.043
1.2	68.754935	0.93134328 37	0.93203908 60			$70 \cdot 10^{-5}$	0.075
1.3	74.484513	0.96236360 83	0.96355818 54			$12 \cdot 10^{-4}$	0.13
1.4	80.214091	0.98348512 47	0.98544973 00			$20 \cdot 10^{-4}$	0.20
1.5	85.943669	0.99438202 27	0.99749498 63			$31 \cdot 10^{-4}$	0.31
1.5707963	90	0.99577290 13	1			$42 \cdot 10^{-4}$	0.42

0.23025) in den Ziffern geliefert. Die Exponentenangabe und damit die Größenordnung des Resultats stimmt. Das Ergebnis ist aber im Betrag etwa um den Faktor 4.343 zu klein. (Vom MR 610 gibt es sogar eine Produktionsserie — Nr. 022... —, bei der in dieser Rechnung die Exponentenangabe gänzlich fehlt.) Rechnet man mit diesen „Resultaten" weiter, so entstehen böse Folgen.

3.1.1. Sinus

Für die Sinusfunktion ist folgende Formel bekannt geworden [62]:

$$\sin(a) \approx \left(\left(\frac{a^2}{20} + 1\right)^{-1} * 10 - 7\right) * a/3. \tag{1}$$

Diese Formel gilt für a im Bogenmaß und hat besonders günstige Genauigkeit im Bereich

$$0 \leqq a \leqq \pi/4,$$

wie Tab. 3.1 zeigt. Die Tabelle gibt außerdem den Verlauf über Argumenten in Grad an, da dies dem Nutzer meist geläufiger sein wird. Besonderer Wert ist auf den Bereich in der Nähe von Null gelegt worden, da dort oft auch Taschenrechner, welche die elementaren transzendenten Funktionen zur Verfügung stellen, gewisse Fehler haben. So werden zwar acht bis zehn Stellen korrekt angegeben, aber führende Nullen nicht berücksichtigt, so daß der relative oder prozentuale Fehler mitunter, d. h. je nach Fabrikat, erheblich werden kann.

Die gleiche Formel kann für

$$\sin(a) = \cos(\pi/2 - a)$$

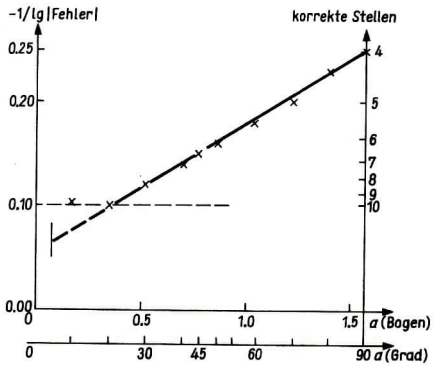


Abb. 3.1. Fehlerkurve zu der verbesserten Näherungsformel (2) für die Sinusfunktion. Der Wert 0.2 bedeutet beispielweise, daß 1/0.2 = 5 Stellen korrekt sind (weitere Erläuterungen im Text)

angewendet werden. Die von Taschenrechnern für die Kosinusfunktion in der Nähe von $\pi/2$ gelieferten Werte sind jedoch mitunter so beschaffen, daß diese Identität nicht gilt. Man prüfe also den jeweils vorliegenden Taschenrechner diesbezüglich nach!

Tabelle 3.1 weist den Sachverhalt genauer aus. Im oberen Teil der Tabelle sind die von einem Taschenrechner in der Nähe von Null gelieferten Sinuswerte und die in der Nähe von $\pi/2$ gelieferten Kosinuswerte aufgenommen worden.

Man erkennt oft, daß die Konstrukteure Wert auf mehrstellige absolute Genauigkeit legten und nicht auf relative Genauigkeit. Bei einem solchen Taschenrechner ist Vorsicht bei der Multiplikation und Division mit diesen Funktionswerten geboten, da sich dabei die relativen Fehler addieren. Eine etwas aufwendigere Ersatzformel — dafür mit erheblich höherer Genauigkeit — ist

$$\sin(a) \approx \left(\left(\frac{a^2}{42}+1\right)^{-1} \times 21 - 11\right) \times \frac{a^2}{-60} + 1\Big) \times a. \tag{2}$$

Der Fehlerverlauf wird in Abb. 3.1 gezeigt und weist aus, daß bis etwa 25° zehn Stellen richtig wiedergegeben werden und selbst bei 90° noch vier Stellen korrekt sind. Der maximale relative Fehler wird nur 0.1‰.

Die Wahl der Formel für die Abszissenwerte in Abb. 3.1 erfolgte nach dieser Überlegung: Der absolute Fehler (Näherung minus exakter Wert) beispielsweise bei $\pi/4$ ist 0.000000218. Der dekadische Logarithmus dazu ist −6.66. Würde dieser als Abszissenfunktion gewählt werden, so hätte man Werte von $-\infty$ (bei 0) bis −4.0 (bei $\pi/2$) aufzutragen. Nimmt man dagegen $-1/\lg|\text{Fehler}|$, so liegen die Werte zwischen 0 und 0.25. Dabei weisen größere Werte auch optisch auf einen größeren Fehler. Für $\pi/4$ erhält man 0.15, und $1/0.15 = 6.67$ gibt bei der Größenordnung des Sinus an, daß sechs Stellen korrekt sind.

3.1.2. Kosinus

Obwohl die Formeln für den Sinus wegen der Beziehung

$$\sin(a) = \cos(\pi/2 - a)$$

auch für den Kosinus genommen werden können, ist dies infolge des Fehlerverhaltens für a-Werte in der Nähe von $\pi/2$ ungünstig. Mit den Näherungsformeln für den Sinus erhält man gerade in der Umgebung von Null (d. h. $\pi/2 - a \approx 0$) verhältnismäßig ungenaue Werte für den Kosinus (vgl. Tab. 3.1). Zwei bessere Formeln, d. h. mit besserem Fehlerverhalten für kleine Argumente, sind

$$\cos(a) \approx \left(\left(\frac{a^2}{30}+1\right)^{-1} \times 5 - 3\right) \times \frac{a^2}{-4} + 1, \tag{3}$$

$$\cos(a) \approx \left(\left(\frac{a^2}{56}+1\right)^{-1} \times 28 - 13\right) \times \frac{a^2}{360} - .5\Big) \times a^2 + 1. \tag{4}$$

Ihr Fehlerverhalten wird in Abb. 3.2 dargestellt. Da

$$\cos(a) = \sin(\pi/2 - a)$$

gilt, können für größere Argumente bei der Berechnung des Sinus besser die Formeln (3) und (4) statt der Formeln (1) und (2) verwendet werden und umgekehrt.

3.1.3. Tangens

Hier werden die Formeln

$$\tan(a) \approx \left(\left(-\frac{2}{5}a^2+1\right)^{-1} \times 5 + 1\right) \times \frac{a}{6}, \tag{5}$$

$$\tan(a) \approx \left(\left(\left(-\frac{17}{42}a^2+1\right)^{-1} \times 84 + 1\right) \times \frac{a^2}{255} + 1\right) \times a \tag{6}$$

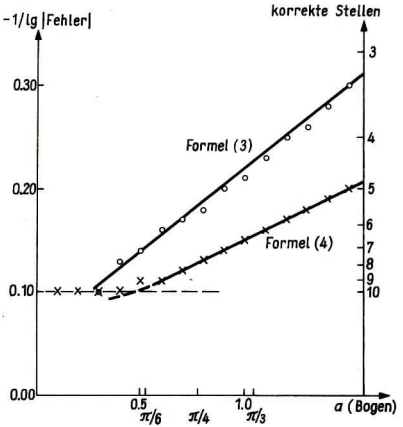


Abb. 3.2. Fehlerkurven zu den Näherungsformeln (3) und (4) für die Kosinusfunktion

angeboten. Wegen

$$\tan(a) = \cot(\pi/2 - a) \quad \text{und} \quad \cot(a) = 1/\tan(a)$$

sind sie auch für die Kotangensfunktion verwendbar. Da die Tangensfunktion bei $\pi/2$ einen Pol hat, sind die Formeln nicht im gesamten Intervall $(0, \pi/2)$ zu gebrauchen. Das Fehlerverhalten in Abb. 3.3 zeigt, daß man bei $a = 1.2$ (68.75°) bereits einen Fehler von 1% oder nur noch zwei korrekte Stellen erreicht.

Viele Taschenrechner liefern, wenn sie die Tangensfunktion zur Verfügung stellen, in der Nähe von $\pi/2$ inkorrekte Werte. Man prüfe also jeweils nach. Zur Berechnung exakter Werte in diesem Bereich verwende man

$$\tan(a) \approx \frac{1}{x} \frac{1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720}}{1 - \frac{x^2}{6} + \frac{x^4}{120} - \frac{x^6}{5040}}, \tag{7}$$

d. h.

$$\tan(a) \approx \frac{1}{x}\left(1 - \frac{x^2}{3} - \frac{x^4}{45} - \frac{2x^6}{945}\right)$$

mit $a = \pi/2 - x$; dabei gilt sogar bei zehnstelliger Rechnung

$$\tan(a) = \frac{1}{x} \quad \text{für} \quad 0 < x \leqq 10^{-5},$$

mit x^2-Gliedern für $10^{-5} < x \leqq 0.006$,
mit x^4-Gliedern für $0.006 < x \leqq 0.06$,
mit x^6-Gliedern für $0.06 < x \leqq 0.4$.

Die Grenzen errechnen sich aus der Bedingung, daß das jeweils nächsthöhere Glied noch keinen Beitrag in der zehnten Stelle liefert.

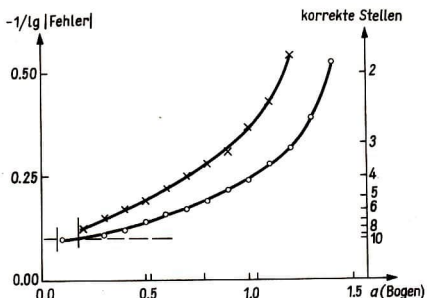


Abb. 3.3. Fehlerkurven zu den Näherungsformeln (5) und (6) für die Tangensfunktion

In der Nähe von Null büßt der Taschenrechner ein bis zwei Stellen an Genauigkeit ein. Ein größeres Fehlverhalten zeigt er bei 0.0005 (0.03°), wo der Tangenswert kleiner als das Argument wird. Der Genauigkeitsverlust beträgt dort sogar vier Stellen. Die Näherungsformel (5) liefert exakte Werte. In der Nähe von $\pi/2$ (90°) büßen die Taschenrechnerwerte rasch an Genauigkeit ein. Der Verlust beginnt bei $x = 0.05$ ($\pi/2 - x$; 87.1°), und bei 10^{-9} (90° − 5.10^{-8}) ist keine Stelle mehr korrekt, jedoch stimmt noch die Größenordnung. Ab 10^{-10} liefert der Taschenrechner die größte darstellbare Zahl, so daß dann auch die Größenordnungen falsch werden.

Viele Taschenrechner, die die Tangensfunktion zur Verfügung stellen, liefern außerdem für kleine Argumente nur eine absolute Genauigkeit bezüglich der Stellenzahl, nicht aber relative Genauigkeit. Bei Verwendung solcher Funktionswerte in weiteren Rechnungen ist dann Vorsicht geboten. Exakte Werte für kleine Argumente erhält

man aus der Beziehung

$$\tan(a) \approx a + \frac{1}{3}a^3 + \frac{2}{15}a^5 + \frac{17}{315}a^7; \tag{8}$$

dabei gilt bei zehnstelliger Rechnung

$\tan(a) \approx a$	für $0 \leqq a \leqq 10^{-5}$,
mit dem a^3-Glied	für $10^{-5} < a \leqq 0.001$,
mit dem a^5-Glied	für $0.001 < a \leqq 0.03$,
mit dem a^7-Glied	für $0.03 < a \leqq 0.1$.

Tab. 3.2. Vergleich von Taschenrechnerwerten für den Tangens in der Nähe von Null und in der Nähe von $\pi/2$ (90°) mit exakten zehnstelligen Werten

a (Bogen)	$\tan(a)$ (8) exakt	$\tan(a)$ (Tastenfunktion)	$\tan(a)$ (5) Näherung
0	0	0	0
10^{-12}	10^{-12}	10^{-12}	10^{-12}
10^{-11}	10^{-11}	10^{-11}	10^{-11}
10^{-5}	10^{-5}	10^{-5}	10^{-5}
10^{-4}	$0.1000000003 \cdot 10^{-3}$	10^{-4}	$0.1000000003 \cdot 10^{-3}$
0.0002	$0.2000000027 \cdot 10^{-3}$	$0.2000000020 \cdot 10^{-3}$	$0.2000000027 \cdot 10^{-3}$
0.0005	$0.5000000417 \cdot 10^{-3}$	$0.4999999500 \cdot 10^{-3}$	$0.5000000417 \cdot 10^{-3}$
0.001	$0.1000000333 \cdot 10^{-2}$	$0.1000000300 \cdot 10^{-2}$	$0.1000000333 \cdot 10^{-2}$
0.002	$0.2000002667 \cdot 10^{-2}$	$0.2000002600 \cdot 10^{-2}$	$0.2000002667 \cdot 10^{-2}$
0.005	$0.5000041667 \cdot 10^{-2}$	$0.5000041600 \cdot 10^{-2}$	$0.5000041667 \cdot 10^{-2}$
0.01	$0.1000033334 \cdot 10^{-1}$	$0.1000033333 \cdot 10^{-1}$	$0.1000033335 \cdot 10^{-1}$
0.02	$0.2000266710 \cdot 10^{-1}$	$0.2000266703 \cdot 10^{-1}$	$0.2000266710 \cdot 10^{-1}$
0.05	$0.5004170838 \cdot 10^{-1}$	$0.5004170836 \cdot 10^{-1}$	$0.5004170838 \cdot 10^{-1}$
0.1	0.1003346720	0.1003346720	0.1003346720

x (Bogen)	$\tan(\pi/2 - x)$ (7) exakt	$\tan(\pi/2 - x)$ (Tastenfunktion)
0	∞	$9.999999990 \cdot 10^{99}$
10^{-12}	10^{12}	$9.999999990 \cdot 10^{99}$
10^{-11}	10^{11}	$9.999999990 \cdot 10^{99}$
10^{-10}	10^{10}	$9.999999990 \cdot 10^{99}$
10^{-9}.	10^{9}	$8.617199147 \cdot 10^{8}$
10^{-5}	10^{5}	$9.999821523 \cdot 10^{4}$
10^{-4}	$9.999999967 \cdot 10^{3}$	$9.999985255 \cdot 10^{3}$
0.0002	$4.999999934 \cdot 10^{3}$	$4.999995938 \cdot 10^{3}$
0.0005	$1.999999833 \cdot 10^{3}$	$1.999999229 \cdot 10^{3}$
0.001	$9.999996667 \cdot 10^{2}$	$9.999995153 \cdot 10^{2}$
0.002	$4.999993334 \cdot 10^{2}$	$4.999993034 \cdot 10^{2}$
0.005	$1.999983333 \cdot 10^{2}$	$1.999983289 \cdot 10^{2}$
0.01	99.99666665	99.99666527
0.02	49.99333316	49.99333291
0.05	19.98333056	19.98333052
0.1	9.966644424	9.966644406

Tabelle 3.2 zeigt Tangenswerte in der Nähe von 0 und in der Nähe von $\pi/2$, errechnet nach den Formeln (8) bzw. (7). In der Nähe von 0 sind außerdem Werte nach Formel (5) angegeben, und in beiden Bereichen werden die gelieferten Werte eines Taschenrechners notiert.

3.1.4. Arcussinus und Arcuskosinus

Die Formeln

$$\arcsin(a) \approx \left(\left(-\frac{9}{20}a^2+1\right)^{-1} \times 10 + 17\right) \times \frac{a}{27}, \tag{9}$$

$$\arcsin(a) \approx \left(\left(\left(-\frac{25}{42}a^2+1\right)^{-1} \times 189 + 61\right) \times \frac{a^2}{1500} + 1\right) \times a \tag{10}$$

können verwendet werden. Wegen

$$\arccos(a) = \pi/2 - \arcsin(a)$$

gelten sie prinzipiell auch für den Arcuskosinus.

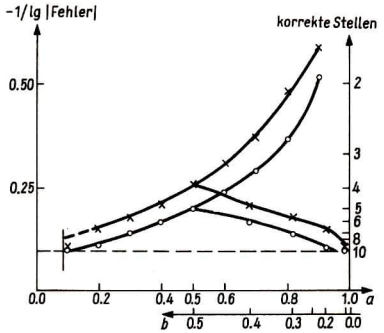


Abb. 3.4. Fehlerkurven zu den Näherungsformeln (9) und (10) für die Arcussinusfunktion

Abb. 3.4 zeigt die Fehlerkurven für diese Näherungsformeln.

Mehr als vier richtige Ziffern erhält man nur für $a \leqq 0.5$; für größere a, d. h. für $0.5 < a \leqq 1$, sollte man die Transformation

$$\arcsin(a) = \pi/2 - 2\arcsin(b) \tag{11}$$

anwenden, woraus

$$b = \sqrt{\frac{1-a}{2}}$$

folgt. Diese Transformation erzeugt für

$$0.5 < a \leqq 1$$

das b-Intervall

$$0.5 > b \geqq 0,$$

so daß die Formeln (9) und (10) wieder mit besserer Genauigkeit verwendbar sind. In Abb. 3.4 sind die b-Werte auf der a-Achse vermerkt und die entsprechenden Fehlerwerte eingetragen worden.

Tab. 3.3. Vergleich von Taschenrechnerwerten für den Arcussinus in der Nähe von Null mit exakten zehnstelligen Werten

a (Bogen)	arc sin (a) (12) exakt	arc sin (a) (Tastenfunktion)	arc sin (a) (9) Näherung
0	0	0	0
10^{-12}	10^{-12}	10^{-12}	10^{-12}
10^{-11}	10^{-11}	10^{-11}	10^{-11}
10^{-5}	10^{-5}	10^{-5}	10^{-5}
10^{-4}	$0.1000000002 \cdot 10^{-3}$	$0.9999999900 \cdot 10^{-4}$	$0.1000000002 \cdot 10^{-3}$
0.0002	$0.2000000013 \cdot 10^{-3}$	$0.1999999990 \cdot 10^{-3}$	$0.2000000013 \cdot 10^{-3}$
0.0005	$0.5000000208 \cdot 10^{-3}$	$0.4999999580 \cdot 10^{-3}$	$0.5000000211 \cdot 10^{-3}$
0.001	$0.1000000166 \cdot 10^{-2}$	$0.1000000166 \cdot 10^{-2}$	$0.1000000167 \cdot 10^{-2}$
0.002	$0.2000001333 \cdot 10^{-2}$	$0.2000001333 \cdot 10^{-2}$	$0.2000001333 \cdot 10^{-2}$
0.005	$0.5000020833 \cdot 10^{-2}$	$0.5000020833 \cdot 10^{-2}$	$0.5000020833 \cdot 10^{-2}$
0.01	$0.1000016668 \cdot 10^{-1}$	$0.1000016664 \cdot 10^{-1}$	$0.1000016667 \cdot 10^{-1}$
0.02	$0.2000133357 \cdot 10^{-1}$	$0.2000133357 \cdot 10^{-1}$	$0.2000133357 \cdot 10^{-1}$
0.05	$0.5002085680 \cdot 10^{-1}$	$0.5002085678 \cdot 10^{-1}$	$0.5002085681 \cdot 10^{-1}$
0.09	$0.9012194501 \cdot 10^{-1}$	$0.9012194494 \cdot 10^{-1}$	$0.9012194452 \cdot 10^{-1}$
0.1	0.1001674212	0.1001674211	0.1001674201

Taschenrechner, welche die Arcusfunktionen liefern, haben mitunter die Eigenschaft, daß für kleine Argumentwerte der Arcussinus und für Argumente in der Nähe von 1 der Arcuskosinus nicht in der vollen Stellenzahl des betreffenden Gerätes wiedergegeben werden. Dies liegt gewöhnlich daran, daß die verwendeten Näherungsformeln für die Mikroprogrammierung des Rechners zwar absolut, aber nicht relativ die Genauigkeitsbedingung erfüllen. Tab. 3.3 zeigt ein Beispiel für dieses Verhalten. Die korrekten Werte können mit der Reihenentwicklung

$$\arcsin(a) \approx a + \frac{a^3}{6} + \frac{3a^5}{40} + \frac{5a^7}{112} \tag{12}$$

gewonnen werden; dabei gilt bei zehnstelliger Rechnung

$$\begin{aligned} \arcsin(a) &= a && \text{für } 0 \leqq a \leqq 10^{-5}, \\ \text{mit dem } a^3\text{-Glied} &&& \text{für } 10^{-5} < a \leqq 0.005, \\ \text{mit dem } a^5\text{-Glied} &&& \text{für } 0.005 < a \leqq 0.03, \\ \text{mit dem } a^7\text{-Glied} &&& \text{für } 0.03 < a \leqq 0.09. \end{aligned}$$

Der in Tab. 3.3 verwendete Taschenrechner befolgt bis $a = 10^{-5}$ die Regel $\arcsin(a) = a$. Dann weist er bis $a = 0.0005$ einen prinzipiellen Fehler auf, da er $\arcsin(a) < a$ gibt. Der absolute Fehler liegt bei 10^{-10}. Die weiteren Werte haben dann nur noch relative Fehler bei 10^{-10}. Die Näherungsformel (9) liefert in der Nähe von Null bessere Werte als der Taschenrechner. Ein Taschenrechner mit zehnstelligen Zahlen kann die Nähe von 1 erst für $b > 10^{-10}$ mit unterschiedlichen Werten erfassen:

$$a = 1 - b.$$

Tab. 3.4. Vergleich von Taschenrechnerwerten für den Arcuskosinus mit exakten Werten. Die exakten Werte sind nach Formel (13) bis $b = 0.0001$ und nach (13′) für $b = 0.001$ und $b = 0.01$ errechnet. Für $b = 0.1$ liefert keine der Formeln (13), (13′) und (14) mehr die exakten Werte

b	$1 - b$	$\arccos(1-b)$ (Tastenfunktion)	$\arccos(1-b)$ (13), (13′) exakt
0	1	0	0
10^{-11}	$1-10^{-11}$	0	$0.4472135955 \cdot 10^{-6}$
10^{-10}	$1-10^{-10}$	$0.1414200000 \cdot 10^{-4}$	$0.1414213562 \cdot 10^{-5}$
10^{-9}	$1-10^{-9}$	$0.4472100000 \cdot 10^{-4}$	$0.4472135955 \cdot 10^{-4}$
10^{-8}	$1-10^{-8}$	$0.1414210000 \cdot 10^{-3}$	$0.1414213563 \cdot 10^{-3}$
10^{-7}	$1-10^{-7}$	$0.4472140000 \cdot 10^{-3}$	$0.4472135991 \cdot 10^{-3}$
10^{-6}	$1-10^{-6}$	$0.1414214000 \cdot 10^{-2}$	$0.1414213679 \cdot 10^{-2}$
10^{-5}	0.99999	$0.4472140000 \cdot 10^{-2}$	$0.4472139680 \cdot 10^{-2}$
10^{-4}	0.9999	$0.1414225400 \cdot 10^{-1}$	$0.1414225347 \cdot 10^{-1}$
10^{-3}	0.999	$0.4472508700 \cdot 10^{-1}$	$0.4472508716 \cdot 10^{-1}$
10^{-2}	0.99	0.1415394730	0.1415394724
0.1	0.9	0.4510258120	

Bessere Werte für $\arccos(a)$ kann man aus Formel (3) erhalten:

$$\cos(x) \approx 1 - b = -\left(\frac{150}{x^2 + 30} - 3\right) \times \frac{x^2}{4} + 1.$$

Die Auflösung nach dem x in dieser Formel liefert im Bereich $0 \leqq x \leqq 0.2$ (vgl. Abb. 3.2) zehnstellig korrekte Werte für $\arccos(1 - b)$. Zunächst erhält man für x^2 die quadratische Gleichung

$$x^4 - \left(20 - \frac{4}{3}b\right) x^2 + 40b = 0,$$

deren kleinere Wurzel zur Vermeidung der Auslöschung von Stellen durch Subtraktion in der Form

$$x^2 = 40b \Big/ \left(10 - \frac{2}{3}b + \sqrt{\left(10 - \frac{2}{3}b\right)^2 - 40b}\right)$$

geschrieben wird. Diese Formel gibt bei Entwicklung nach b bis zum linearen Glied, d. h. bei zehnstelliger Rechnung verwendbar für $b \leqq 10^{-4}$, den Ausdruck

$$x = \sqrt{2b}\left(1 + \frac{b}{12}\right). \tag{13}$$

Tabelle 3.4 zeigt den Unterschied dieser exakten Werte zu den von einem Taschenrechner gelieferten (Formel (13) bis zur Trennlinie).

Aus der nach dem Glied vierter Ordnung abgebrochenen Reihe für den Kosinus,

$$\cos(x) \approx 1 - b = 1 - \frac{x^2}{2} + \frac{x^4}{24},$$

die bei zehnstelliger Rechnung für $|x| \leqq 0.07$ verwendbar ist, erhält man bei Auflösung nach x

$$x^2 = 24b \Big/ \left(6 + 6\sqrt{1 - \frac{2}{3}b}\right)$$

bzw. bei Entwicklung bis zum quadratischen Glied

$$x = \sqrt{2b}\left(1 + \frac{b}{12} + \frac{7b^2}{288}\right). \tag{14}$$

Während (13) nur für $b \leqq 10^{-4}$ korrekte zehnstellige Werte liefert, müßte man (14) bis $b \leqq 10^{-3}$ verwenden können. Dann wird aber schon $x = 0.04$ geliefert, so daß man in der Nähe der Gültigkeitsgrenzen der abgebrochenen Reihenentwicklung ist.

Führt man (13) bis zum quadratischen Glied weiter, so erhält man

$$x = \sqrt{2b}\left(1 + \frac{b}{12} + \frac{3}{160}b^2\right). \tag{13'}$$

3.1.5. Arcustangens

Hier sind die Formeln

$$\arctan(a) \approx ((0.6a^2 + 1)^{-1} \lessdot 5 + 4) \lessdot \frac{a}{9}, \tag{15}$$

$$\arctan(a) \approx \left(\left(\left(\frac{5a^2}{7} + 1\right)^{-1} \lessdot 21 + 4\right) \lessdot \frac{a^2}{-75} + 1\right) \lessdot a \tag{16}$$

bekannt geworden. Sie können nur gut verwendet werden für

$$0 \leqq a \leqq 0.5,$$

wie es auch Abb. 3.5 zeigt.

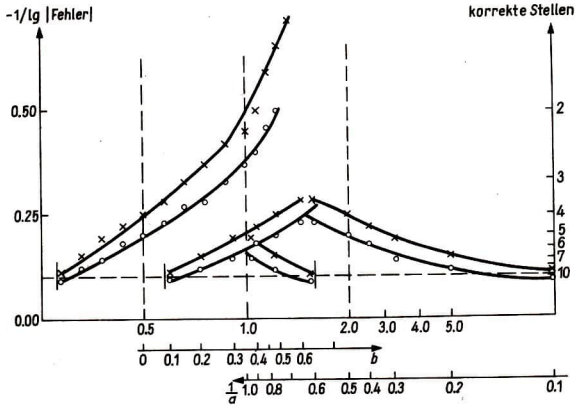


Abb. 3.5. Fehlerkurven zu den Näherungsformeln (15) und (16) für die Arcustangensfunktion. Die Formeln sind nicht besonders gut. Für eine Genauigkeit von drei bis vier Stellen muß man mit a unter 0.5 bleiben

Für Argumente $a > 0.5$ spaltet man nach der Additionsformel den Wert

$$c = \arctan(0.5) = 0.463647090$$

ab und verwendet für $0.5 < a \leqq 1$

$$\arctan(a) = \arctan(b) + c$$

mit

$$b = \left(\left(\frac{2}{a} + 1\right)^{-1} \times 5 - 1\right)\Big/2. \tag{17}$$

Für $a > 1$ nehme man

$$\arctan(a) = \frac{\pi}{2} - \arctan\left(\frac{1}{a}\right). \tag{18}$$

Die Formeln (17) und (18) sind exakte Umformungen und keine Näherungen. Bei (18) ist dies sofort zu sehen, und (17) folgt aus der Additionsformel

$$\arctan(x) - \arctan(y) = \arctan\left(\frac{x-y}{1+xy}\right)$$

für $x = a$ und $y = 0.5$, so daß

$$b = \frac{a - 0.5}{1 + 0.5a} = \frac{2a - 1}{a + 2}$$

folgt. Diese Beziehung erhält man aus (17). Die dort gegebene Gestalt ist aber gleich auf eine Tastendruckfolge eines Taschenrechners abgestimmt.

Das Verhalten der Formeln (15) und (16) zeigt Abb. 3.5. Für den Bereich (0.5, 1.0) kombiniere man (17) mit den Formeln, für (1.0, 2.0) verwendet man (18) und (17) zusätzlich, und für (2, ∞) kommt man mit (18) als Zusatz aus. Die als „besser" bezeichnete Formel (16) liefert meist nur eine Stelle mehr. Oberhalb von $a = 1$ ist sie sogar schlechter als (15).

Die Anzeigen von Taschenrechnern mit der Arcustangens-Taste sind mitunter für kleine Argumente nicht völlig korrekt, wenn man volle Ausschöpfung der Zahlendarstellung für die Ziffern der Ergebnisse erwartet. Dann liefert schon (15) bessere Resultate.

Exakte Werte in diesem Bereich kann man aus der Potenzreihe erhalten,

$$\arctan(a) = a - \frac{a^3}{3} + \frac{a^5}{5} - \frac{a^7}{7} + \cdots, \tag{19}$$

und es gilt für zehnstellige Rechnung

$\arctan(a) = a$	für $0 \leqq a \leqq 5 \cdot 10^{-5}$,
mit dem a^3-Glied	für $5 \cdot 10^{-5} \leqq a \leqq 0.009$,
mit dem a^5-Glied	für $0.009 \leqq a \leqq 0.05$,
mit dem a^7-Glied	für $0.05 \leqq a \leqq 0.1$.

Beispielsweise zeigt ein Taschenrechner

$$\arctan(0.0001) = 0.999999990 \cdot 10^{-4},$$

während

$$0.999999967 \cdot 10^{-4}$$

der zehnstellige Näherungswert ist. Er hat also zwei Stellen an Genauigkeit verloren.

Für große Argumente gilt

$$\arctan(a) = \frac{\pi}{2} - \frac{1}{a} + \frac{1}{3a^3} - \frac{1}{5a^5} + \frac{1}{7a^7} - \cdots, \tag{20}$$

und für zehnstellige Rechnung gilt

$\arctan(a) = \frac{\pi}{2}$	für $10^9 < a$,
mit dem a-Glied	für $1000 \leqq a \leqq 10^9$,
mit dem a^3-Glied	für $50 \leqq a \leqq 1000$,
mit dem a^5-Glied	für $15 \leqq a \leqq 50$,
mit dem a^7-Glied	für $8 \leqq a \leqq 15$.

In diesen Bereichen liefern Taschenrechner gewöhnlich auch die exakten Werte ohne Stellenverluste.

3.1.6. Logarithmus

In enger Beziehung zu der Reihenentwicklung

$$\ln(a) = 2\left(\frac{a-1}{a+1} + \frac{1}{3}\left(\frac{a-1}{a+1}\right)^3 + \frac{1}{5}\left(\frac{a-1}{a+1}\right)^5 + \cdots\right) \tag{21}$$

für $a > 0$ steht die Näherungsformel

$$\ln(a) \approx \left((-0.6b^2 + 1)^{-1} \times 5 + 4\right) \times \frac{2b}{9} \tag{22}$$

mit $b = \dfrac{a-1}{a+1} = (a+1)^{-1} \times (-2) + 1$, welche für $0.7 < a < 1.6$ im ungünstigen Fall mit fünf bis sechs korrekten Stellen arbeitet.

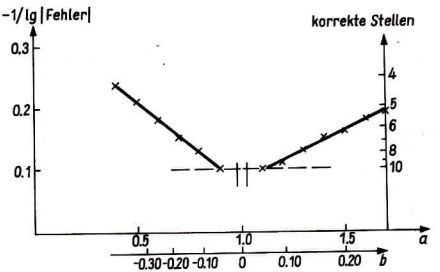


Abb. 3.6. Fehlerkurve zu der Näherungsformel (22) für die Logarithmusfunktion

Die Formel (22) ist bemerkenswert stark. Noch bei $a = 10$ liegt der Fehler erst in der dritten Stelle.

Liegt a außerhalb des angegebenen Intervalls und will man die Zahl korrekter Stellen nicht zu rasch sinken lassen, so spaltet man eine Potenz von 2 ab,

$$\ln(a) = \ln(2^n \times c) = n \ln(2) + \ln(c) \approx 0.693147\,1806 \times n + \ln(c),$$

so daß nun c im Intervall (0.7, 1.6) liegt. Den dekadischen Logarithmus erhält man einfach aus der Beziehung

$$\lg(a) = \ln(a)/\ln(10) \approx \ln(a)/2.302\,585\,093.$$

In der Nähe von $a = 1$ liefern Taschenrechner nicht immer korrekte Werte unter Ausnutzung der vollen Stellenzahl. Genaue Werte in diesem Bereich erhält man durch

$$\ln(a) = \ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots, \tag{23}$$

und es gilt bei zehnstelliger Rechnung

$\ln(1 + x) = x$	für $0 \leqq x < 10^{-10}$,
mit dem x^2-Glied	für $10^{-10} \leqq x < 1.2 \cdot 10^{-5}$,
mit dem x^3-Glied	für $1.2 \cdot 10^{-5} \leqq x < 0.0005$,
mit dem x^4-Glied	für $0.0005 \leqq x < 0.003$.

Beispielsweise liefert ein Taschenrechner

$$\ln(1.00001) = 0.9999900000 \cdot 10^{-5},$$

während der zehnstellige exakte Wert

$$\ln(1.00001) = 0.9999950000 \cdot 10^{-5}$$

ist, was einen fünfstelligen Genauigkeitsverlust erkennen läßt. Die Näherungsformel (22) liefert in der Nähe von $a = 1$ (d. h. $b = 0$) bessere Werte als der Taschenrechner.

3.1.7. Exponentialfunktion

Als Näherungsformel verwendbar ist

$$\exp(a) \approx \left(\left(\frac{a^2}{60} + 1\right)^{-1} \times (-5) + 6\right) \Big/ a - 0.5\Big)^{-1} + 1, \tag{24}$$

und zwar für $0 < a < 1$.

Abb. 3.7 zeigt das Fehlerverhalten. Bis $a = 0.6$ werden mindestens fünf korrekte Stellen geliefert. Will man für $a > 1$ genauere Werte haben, so muß eine e-Potenz mit ganzem Exponenten abgespalten werden:

$$\exp(a) = \exp(n + b) = \exp(n) \cdot \exp(b).$$

Als Näherungswert für e kann man

$$e \approx 193/71 = 2.718309859$$

verwenden (der zehnstellige Wert ist $e = 2.718281828$), so daß der Fehler nur zwei Einheiten der fünften Ziffer (0.001%) beträgt. Der Näherungsbruch für e entsteht aus (24) für $a = 1$, jedoch statt der in ihm vorkommenden Ziffernfolge 1-9-3-7-1 (Zähler und Nenner) aus fünf Ziffern kann man sich auch gleich 2-7-1-8-3 für fünf Ziffern

von e merken. Obwohl die Formel (24) auch außerhalb des Intervalls (0, 1) noch recht gute Werte liefert, versagt sie wegen Division durch Null bei $a = 0$. In der Nähe von $a = 0$ erhält man exakte Werte durch die bekannte Reihe für exp (a):

$$\exp(a) = 1 + a + \frac{a^2}{2!} + \frac{a^3}{3!} + \frac{a^4}{4!} + \cdots. \tag{25}$$

Für zehnstellige Rechnung gilt dabei

$\exp(a) = 1$	für $\lvert a\rvert < 5 \cdot 10^{-11}$,
mit dem a-Glied	für $5 \cdot 10^{-11} \leqq \lvert a\rvert < 10^{-5}$,
mit dem a^2-Glied	für $10^{-5} \leqq \lvert a\rvert < 6 \cdot 10^{-4}$,
mit dem a^3-Glied	für $6 \cdot 10^{-4} \leqq \lvert a\rvert < 0.005$,
mit dem a^4-Glied	für $0.005 \leqq \lvert a\rvert < 0.02$.

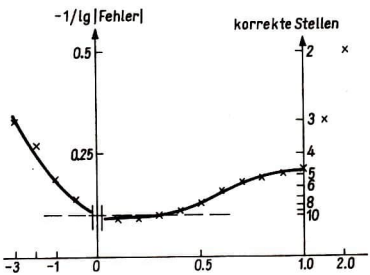


Abb. 3.7. Fehlerkurve zur Näherungsformel (24) für die Exponentialfunktion. An der Stelle $a = 1$ werden noch fünf korrekte Stellen für e geliefert

3.2. Schnelle Berechnung transzendenter Funktionen und der Quadratwurzel

Bereits in den ersten Jahren der Existenz programmgesteuerter Rechenanlagen stellte man dem Anwender neben den arithmetischen Grundoperationen auch einige transzendente Funktionen (z. B. die Logarithmus- und Exponentialfunktion, die trigonometrischen Funktionen, Arcus-, Hyperbel- und Areafunktionen) und die Quadratwurzel zur Verfügung. Dies geschah mitunter durch „feste Verdrahtung", wie man den gerätemäßigen elektronischen Einbau (in der Hardware) nannte, oder durch Unterprogramme. Besonders die Quadratwurzelberechnung wurde öfter fest eingebaut. Dies erfolgte beispielsweise bereits in dem Relaisrechner OPREMA (Optik-Rechenmaschine) des VEB Carl Zeiss Jena im Jahre 1954. Ein gerätetechnischer

Einbau oder eine Realisierung durch Bauelemente setzt immer eine *Mikroprogrammierung* voraus, d. h., die Operation wird in elementare Aktionen wie Registerumläufe, Verschiebungen, Elementaradditionen ganzer Zahlen, Bitabfragen und Schaltersteuerungen zerlegt. Bei Unterprogrammen werden dagegen die Berechnungen in Befehlsfolgen des jeweiligen Computers (z. B. Assembler) umgesetzt, in Befehle also, wie sie dem Nutzer auch zur Verfügung stehen. In beiden Fällen, d. h., in der Mikroprogrammierung und in der *Programmierung*, ist man an möglichst schneller Berechnung interessiert. Sehr verbreitet und üblich ist die Berechnung transzendenter Funktionen und der Quadratwurzel durch Reihenentwicklungen, durch Ersatz- bzw. Ausgleichsfunktionen und durch Iterationsformeln. Bei Reihenentwicklungen, d. h. *Entwicklungen an einer Stelle* — weshalb man diese auch *Punktformeln* nennt, ist ein Genauigkeitstest in der Befehlsfolge eingebaut. Somit ist bei einer aktuellen Anwendung der Zeitaufwand nicht von vornherein bekannt, da der Test manchmal früher (in der Nähe der Entwicklungsstelle) und manchmal später erfüllt ist. Bei den Ersatzfunktionen hat man zwischen zwei Arten zu unterscheiden:

Erstens kann ihre Konstruktion auf der Grundlage der Methode der kleinsten Quadrate in einem Intervall erfolgen. Dies ist das Verfahren nach GAUSS oder FOURIER oder die *Approximation im Mittel*. Bei dieser Methode ist der Fehler zwischen Original- und Ersatzfunktion im Mittel über ein Intervall möglichst klein. Das schließt nicht aus, daß in einzelnen Punkten oder kleinen Teilintervallen ein großer Fehler existiert. Meist werden als Ersatzfunktion nach dieser Methode Polynome konstruiert, da deren Auswertung mit der Methode nach HORNER sehr schnell erfolgen kann. Der Nutzer benötigt dann eine Information über die erzielbare Genauigkeit.

Zweitens kann die Konstruktion der Ersatzfunktion nach dem Prinzip der *gleichmäßigen Approximation* erfolgen, das mit dem Namen ČEBYŠEV (TSCHEBYSCHEFF) verknüpft ist. Hierbei wird in einem Intervall garantiert, daß die maximale Abweichung der Ersatzfunktion von der Originalfunktion eine gewisse Größe nicht übersteigt. Auch hier werden gerne Polynome als Ersatzfunktionen konstruiert, deren Koeffizienten also aus der Bedingung „maximale Abweichung minimal" im Intervall von x_1 bis x_2 gewonnen werden. Die gleichmäßige Approximation oder ČEBYŠEV-Approximation darf nicht mit der „Entwicklung nach ČEBYŠEV-Polynomen" verwechselt werden, da letztere nach dem Verfahren von FOURIER erfolgt, also eine Approximation im Mittel ist. Allerdings schafft eine Entwicklung nach ČEBYŠEV-Polynomen eine gute Ausgangssituation für eine echte ČEBYŠEV-Approximation.

Bei Iterationsformeln, die in einem Unterprogramm realisiert werden, wie es z. B. oft bei der Berechnung der Winkelfunktionen der Fall ist, muß wie bei der Reihenentwicklung im allgemeinen eine Genauigkeitsschranke eingebaut werden, so daß ebenfalls die Arbeitszeit von Fall zu Fall unterschiedlich ist.

Wie aufwendig die genannten Verfahren sind, kann man bei elektronischen Taschen- und Tischrechnern optisch wahrnehmen. Während die Resultate der arithmetischen Grundoperationen praktisch sofort sichtbar werden, dauert die Berechnung der transzendenten Funktionen, die ja sehr häufig direkt über Tastendruck zur Verfügung gestellt werden, deutlich länger und manchmal „sehr" lange, d. h. einige Sekunden.

Aber das gilt eben nur manchmal, denn bei einigen Fabrikaten werden auch die transzendenten Funktionen sehr schnell — in zwei bis drei Grundoperationszeiten geliefert. Das bei diesen eingesetzte Berechnungsverfahren ist seit 1961 (MEGGITT [41]) bekannt und kann jetzt voll eingesetzt werden, da die elektronischen Kleinrechner auf der Basis der mikroprogrammierbaren Mikroprozessoren arbeiten. Jede Grundoperation des Kleinrechners wird durch ein Mikrounterprogramm in den Befehlen (Mikrobefehlen) eines Mikroprozessors realisiert. Die schnelle Berechnung der transzendenten Funktionen und der Quadratwurzel basiert nun auf einer modifizierten Division oder einer modifizierten Multiplikation. Es entsteht demnach für das Mikroprogramm einer Division oder Multiplikation keine sehr große Belastung, wenn in den einzelnen Subtraktionsrunden des Divisionsablaufs (der Divisor wird ständig subtrahiert) oder Additionsrunden der Multiplikation (der Multiplikand wird ständig addiert) der Dividend und Divisor bzw. der Multiplikand und das Zwischenresultat geringfügig geändert werden. Die Art dieser Änderung bringt es mit sich, daß das Resultat der Division oder Multiplikation der eine oder andere transzendente Funktionswert (oder eine Vorstufe dazu) ist. Das erklärt, wieso transzendente Funktionen in Zeiten berechnet werden können, die sich aus wenigen Divisions- und Multiplikationszeiten zusammensetzen. Das Verfahren ist eigentlich nur für Mikroprogrammierung geeignet; seine Realisierung als Unterprogramm in Nutzerbefehlen (Assembler) bedeutet eine Zerlegung der Grundoperationen Division und Multiplikation in die Grundoperationen Subtraktion, Addition und Verschiebung, d. h. ein Nachspielen der Mikroprogrammierung auf dem höheren Niveau der Nutzerbefehle. Diese *Emulation* (Simulation eines Computers mit einem Computer) ist bekanntlich wenig effektiv (vgl. auch 3.3.5.). Somit kommt ein altes Verfahren erst jetzt richtig zur Wirkung, und es ist angebracht, es aus dem Dunkel der Vergangenheit herauszuholen und neu vorzustellen.

3.2.1. Normale Division und Multiplikation

Für das Verständnis der später zu erläuternden Modifikationen werden zunächst an zwei Beispielen die Mikroabläufe der Division und Multiplikation vierstelliger ganzer Dezimalzahlen vorgestellt.

Multiplikation

Die letzte Multiplikatorziffer steuert die Anzahl der Additionen des Multiplikanden

Multiplikand	Produkt	Multiplikator
4158	00000000	3024
	+ 4158	3
	4158	
	+ 4158	2
	8316	
	+ 4158	1
	12474	

	12474	
	+ 4158	0
1. Ziffer verarbeitet, Multiplikand verschieben (* 10)	16632	würde negativ werden, Verschiebung (: 10) 0302
	+ 00041580	1
	58212	
	+ 4158	0
2. Ziffer verarbeitet, Multiplikand verschieben (* 10)	99792	würde negativ werden, Verschiebung (: 10) 0030
3. Ziffer verarbeitet, Multiplikand verschieben (* 10)		würde negativ werden, Verschiebung (: 10) 0003
	+ 04158000	2
	4257792	
	+ 4158	1
	8415792	
	+ 4158	0
4. Ziffer verarbeitet, Ende	12573792	würde negativ werden, Ende

Das in doppelter Länge entstehende Resultat von 4148 · 3024 ist 12573792.

Bei den Verschiebungen des Multiplikators nach rechts wird die letzte Ziffer jeweils abgestoßen.

Division

Der Divisor wird vom Dividenden (von vorn beginnend) subtrahiert. Die Anzahlen der möglichen Subtraktionen geben die Quotientenziffern. Eine nötige Verschiebung des Vorganges nach rechts bringt den Übergang zur nächsten Quotientenziffer. Sobald beim Dividenden infolge der Verschiebung am rechten Registerende Nullen eingezogen werden, ist beim Quotienten der Anfang des gebrochenen Anteils zu markieren (Dezimalpunkt).

Divisor	Dividend	Quotient
0416	1257	0000
	− 0416	
	841	1
	− 416	
	425	2
	− 416	
	9	0003
1. Ziffer erarbeitet		würde negativ werden, Dividend und Quotient verschieben (* 10)
	0090	003.0
2. Ziffer erarbeitet		würde negativ werden, Dividend und Quotient verschieben (* 10)

	0900	03.00
	− 0416	
	484	1
	− 0416	
	68	03.02
3. Ziffer erarbeitet		würde negativ werden, Dividend und Quotient verschieben (* 10)
	0680	3.020
	− 0416	
	264	3.021
4. Ziffer erarbeitet		würde negativ werden
Ende		Ende

Das Resultat der Division 1257/416 ist 3.021. Der Abbruch bedingt, daß die letzte Ziffer nicht gerundet wird.

3.2.2. Schnelle Berechnung des Logarithmus

Man geht aus von der Darstellung

$$f(z) = \ln(1 + az) = \sum_{k=0}^{\infty} q_k \ln\left(1 + \left(\frac{z}{10}\right)^k\right).$$

Für $z = 1$, $a = \dfrac{y}{x}$ wird daraus

$$\ln\left(1 + \frac{y}{x}\right) = \sum_{k=0}^{\infty} q_k \ln(1 + 10^{-k}), \tag{1}$$

$$1 + \frac{y}{x} = \prod_{k=0}^{\infty} (1 + 10^{-k})^{q_k}. \tag{2}$$

Es sollen nun x und y als n-stellige ganze positive Zahlen gegeben sein, und für die q_k soll gelten

$$0 \leqq q_k \leqq 9,$$

ebenfalls ganzzahlig, d. h., die q_k sind Ziffern des Dezimalsystems. Dann können die q_k aus (2) durch eine modifizierte Division errechnet werden:

$$y + x = x \prod_{k=0}^{\infty} (1 + 10^{-k})^{q_k} \tag{3}$$

Das größte mögliche Produkt, d. h., alle q_k sind gleich 9, ist bei spezieller zehnstelliger Auswertung

$$p_{\max} = 512 \cdot 2.604738725 = 1333.636467.$$

Also muß gelten

$$1 + \frac{y}{x} \leqq p_{\max},$$

und das ist eine Bedingung, die im praktischen Einsatz stets gut erfüllt werden kann.

Ist etwa die Differenz D_{j-1} bereits berechnet,

$$D_{j-1} = y - x\left(\prod_{k=0}^{j-1}(1 + 10^{-k})^{q_k} - 1\right), \tag{4}$$

wobei die q_k so gewählt seien, daß D_{j-1} möglichst klein wird, so sieht man aus (3),

$$y - x\left(\prod_{k=0}^{\infty}(1 + 10^{-k})^{q_k} - 1\right) = 0,$$

daß

$$D_{j-1} = x\prod_{k=j}^{\infty}(1 + 10^{-k})^{q_k} \tag{5}$$

ist. Subtrahiert man von D_{j-1} die Größe $10^{-j}D_{j-1}$, so erhält man, falls dies möglich ist,

$$D_j^{(1)} = (1 - 10^{-j})\,D_{j-1} = y - x\left(\prod_{k=0}^{j-1}(1 + 10^{-k})^{q_k}(1 + 10^{j}) - 1\right). \tag{6}$$

Hat man bereits

$$D_j^{(a)} = y - x\left(\prod_{k=0}^{j-1}(1 + 10^{-k})^{q_k}(1 + 10^{-j})^{a} - 1\right),$$

so subtrahiert man $10^{-j}D_{j-1}(1 + 10^{-j})^a$ und erhält, falls dies noch möglich ist,

$$D_j^{(a+1)} = y - x\left(\prod_{k=0}^{j-1}(1 + 10^{-k})^{q_k}(1 + 10^{-j})^{a+1} - 1\right). \tag{7}$$

Für den größtmöglichen Exponenten $a_{\max} \leqq 9$ gelte dann

$$D_j^{a_{\max}} = D_j,$$

und der Prozeß wird fortgesetzt. Zur Beschreibung ist es angebracht,

$$x_j^{(a)} = D_{j-1}(1 + 10^{-j})^a, \qquad y_j^{(a)} = y - D_{j-1}(1 + 10^{-j})^a + x$$

zu setzen. Dann erhält man aus (6) und (7) die Rekursion

$$y_j^{(a+1)} = y_j^{(a)} - 10^{-j}x_j^{(a)}, \qquad x_j^{(a+1)} = x_j^{(a)} + 10^{-j}x_j^{(a)}. \tag{8}$$

Um keinen Stellenverlust zu erleiden, ist es besser,

$$y_j^{(a)} := 10^j y_j^{(a)}$$

zu setzen, wodurch

$$y_j^{(a+1)} = y_j^{(a)} - x_j^{(a)}, \quad x_j^{(a+1)} = x_j^{(a)} + 10^{-j}x_j^{(a)} \tag{9}$$

entsteht. Für den Beginn der Rechnung an der Stelle j gilt

$$y_j^{(0)} = 10y_{j-1}^{(q_{j-1})} \qquad x_j^{(0)} = x_{j-1}^{(q_{j-1})}, \tag{10}$$

so daß der letzte Subtraktionsrest $y_{j-1}^{(q_{j-1})}$ bei der Berechnung der Ziffer q_{j-1} einmal nach links zu verschieben ist und der letzte Subtrahend bei dieser Berechnung unver-

ändert übernommen wird. Der Anfang der gesamten Rechnung erfolgt mit

$$y = y_0^{(0)}, \quad x = x_0^{(0)}.$$

Aus den Rekursionen (9) und den Bedingungen, daß die $y_j^{(q_j)}$ gerade noch positiv bleiben sollen sowie die q_j den Wert 9 nicht übersteigen dürfen, erkennt man die Tatsache einer modifizierten Division. Die Modifikation erfolgt beim Divisor x,

Tab. 3.5. Zur schnellen Berechnung des Logarithmus

k	x	y	Q und Zähler
0	2000000000	6267956312	0000000000
	+2000000000	− 2000000000	
	4000000000	4267956312	1
	+4000000000	− 4000000000	
	8000000000	0267956312	0000000002
		zu klein, Verschiebung	
1	8000000000	2679563120	0000000020
		zu klein, Verschiebung	
2	8000000000	26795631200	0000000200
	+ 80000000	− 8000000000	
	8080000000	18795631200	1
	+ 80800000	− 8080000000	
	8160800000	10715631200	2
	+ 81608000	− 8160800000	
	8242408000	2554831200	0000000203
		zu klein, Verschiebung	
3	8242408000	25548312000	0000002030
	+ 8242408	− 8242408000	
	8250650408	17305904000	1
	+ 8250650	− 8250650408	
	8258901058	9055253692	2
	+ 8258901	− 8258901058	
	8267159959	796352634	0000002033
		zu klein, Verschiebung	
4	8267159959	7963526340	0000020330
		zu klein, Verschiebung	
5	8267159959	79635263400	0000203300
	+ 82672	− 8267159959	
	8267242631	71368103441	1
	+ 82672	− 8267242631	
	8267325303	63100860810	2
	+ 82673	− 8267325303	
	8267407976	54833535507	3
	+ 82674	− 8267407976	
	8267490650	46566127531	4
	+ 82675	− 8267490650	
	8267573325	38298636881	5
	+ 82676	− 8267573325	
	8267656001	30031063556	6
	+ 82677	− 8267656001	
	8267738678	21763407555	7
	+ 82677	− 8267738678	
	8267821355	13495668877	8
	+ 82678	− 8267821355	
	8267904033	5227847522	0000203309
		zu klein, Verschiebung	
6	8267904033	52278475220	0002033090
	+ 8268	− 8267904033	
	8267912301	44010571187	1
	8267912301	44010571187	1
	+ 8268	− 8267912301	
	8267920569	35742658886	2
	+ 8268	− 8267920569	
	8267928837	27474738317	3
	+ 8268	− 8267928837	
	8267937105	19206809480	4
	+ 8268	− 8267937105	
	8267945373	10938872375	5
	+ 8268	− 8267945373	
	8267953641	2670927002	0002033096
		zu klein, Verschiebung	
7	8267953641	26709270020	0020330960
	+ 827	− 8267953641	
	8267954488	18441316379	1
	+ 827	− 8267954468	
	8267955295	10173361911	2
	+ 827	− 8267955295	
	8267956122	1905406616	0020330963
		zu klein, Verschiebung	
8	8267956122	19054066160	0203309630
	+ 83	− 8267956122	
	8267956205	10786110038	1
	+ 83	− 8267956205	
	8267956288	2518153833	0203309632
		zu klein, Verschiebung	
9	8267956288	25181538330	2033096320
	+ 8	− 8267956288	
	8267956296	16913582042	1
	+ 8	− 8267956296	
	8267956304	8645625746	2
	+ 8	− 8267956304	
		377669442	2033096323
		zu klein, Ende	

Es hat sich also $Q = 2.033096323$ ergeben. Die Multiplikation liefert nun

j	q_j	$\ln(1 + 10^{-j})$
0	2	0.693147181
1	0	0.095310180
2	3	0.009950331
3	3	0.000999500
4	0	0.000099950
5	9	0.000010000
6	6	0.000001000
7	3	0.000000100
8	2	0.000000010
9	3	0.000000001

$\ln 4.133978156 = 1.419240177$

der vor jeder Subtraktion zur Berechnung der Stelle j mit $1 + 10^{-j}$ multipliziert wird. Dies geschieht praktisch durch Verschiebung um j Stellen und eine Addition. Sind n Stellen des „Quotienten" berechnet, so kann mit vorbereiteten Zahlen $\ln(1 + 10^{-j})$ gemäß (1) mittels einer modifizierten Multiplikation der Wert $\ln\left(1 + \frac{y}{x}\right)$ berechnet werden. Diese „Multiplikation" von Q benötigt als Multiplikand für die Quotientenziffer q_j den Wert $\ln\left(1 + \frac{y}{x}\right)$ anstelle des sonst zu verschiebenden konstanten Multiplikanden. Ist beispielsweise ln 4.133978156 und 10stellig zu berechnen, dann bildet man $1 + 3.133978156$ und daraus etwa $y = 626795312$ (noch kleiner als 10^{10}) mit $x = 2000000000$. Die Rechnung verläuft wie in Tab. 3.5.

Der dort errechnete Wert ist bis auf 10^{-9} korrekt. Für die Berechnung des Zehnerlogarithmus sind entweder einfach die Werte $\lg(1 + 10^{-j})$ für die Zusatzmultiplikation bereitzustellen, oder es ist eine echte Multiplikation mit 1/ln 10 zu ergänzen. Ist das Paar (x, y) unter der Bedingung $y/x < 2^{10} - 1$ (diese Bedingung ist schärfer als die aus (3) erhältliche und resultiert aus der Rekursion für $q_0 \leqq 9$; denn dann wird maximal $(2^{10} - 1)$-mal das x subtrahiert) nicht so einfach wie im Beispiel für ln 4.133978156 herstellbar, so sind einige einfache Reduktionen der Größenordnungen durchzuführen. Beispielsweise ist der Exponent in der Gleitkommadarstellung einer zu logarithmierenden Zahl sofort ein additiver Beitrag zum Dezimallogarithmus der Mantisse. Bei der Berechnung des natürlichen Logarithmus dagegen wird man vor Addition des Exponenten diesen mit $1/\lg e$ multiplizieren.

Man hätte übrigens bei $k = 6$ auf eine echte Division überleiten können, da die Änderungen am Divisor dann nur noch die letzten vier Stellen betreffen, so daß die ersten vier Ziffern des Quotienten davon nicht beeinflußt werden:

$$\frac{52278475220}{8267904033} = 6.323\ldots$$

3.2.3. Schnelle Berechnung des Arcustangens

Für eine komplexe Zahl z gilt die Eulersche Beziehung

$$z = x + iy = Re^{i\varphi}, \quad \varphi = \arctan\frac{y}{x}, \quad R = \sqrt{x^2 + y^2}, \tag{11}$$

wenn z im ersten Quadranten liegt. Es gilt offenbar nun

$$\begin{aligned} &\ln(x + iy) = \ln R + i\varphi, \quad \varphi = \operatorname{Im}\left(\ln(x + iy)\right), \\ &\operatorname{Im}\left(\ln(x + iy)\right) = \arctan\frac{y}{x}, \end{aligned} \tag{12}$$

so daß der Imaginärteil des komplexen Logarithmus der gesuchte Arcustangens ist. In ähnlicher Weise wie bei der Berechnung des Logarithmus im vorigen Abschnitt

wird angesetzt:

$$\ln(x+iy) = \ln R - \sum_{k=0}^{\infty} q_k \ln(1 - i \cdot 10^{-k}), \tag{13}$$

$$\operatorname{Im}\left(\ln(x+iy)\right) = -\sum_{k=0}^{\infty} q_k \operatorname{Im}\left(\ln(1 - i \cdot 10^{-k})\right) = -\sum_{k=0}^{\infty} q_k \varphi_k,$$

wobei $\varphi_k = \arctan(-10^{-k}) = -\arctan 10^{-k}$ ist, so daß

$$\arctan\frac{y}{x} = \sum_{k=0}^{\infty} q_k \arctan 10^{-k} \tag{14}$$

gilt. Aus (13) erhält man für das reelle R

$$R = (x+iy)\prod_{k=0}^{\infty}(1 - i \cdot 10^{-k})^{q_k}. \tag{15}$$

Hat man etwa bereits

$$R_j^{(a)} = x_j^{(a)} + iy_j^{(a)} = (x+iy)\prod_{k=0}^{j-1}(1 - i \cdot 10^{-k})^{q_k}(1 - i \cdot 10^{-j})^a$$

berechnet, und zwar so, daß der Imaginärteil möglichst klein wird, so erhält man q_j als das maximal mögliche a aus der Rekursion

$$y_j^{(a+1)} = y_j^{(a)} - 10^{-j}x_j^{(a)}, \quad x_j^{(a+1)} = x_j^{(a)} + 10^{-j}y_j^{(a)}, \tag{16}$$

die einfach aus der Multiplikation

$$(x_j^{(a)} + iy_j^{(a)})(1 - i \cdot 10^{-j}) = x_j^{(a+1)} + iy_j^{(a+1)}$$

folgt und die angesetzt wird, um möglichst oft den Winkel

$$\varphi_j = +\arctan 10^{-j}$$

zu subtrahieren, damit der Imaginärteil möglichst klein wird. Um keine Genauigkeitseinbußen zu erleiden, wird

$$y_j^{(a)} := 10^j y_j^{(a)}$$

gesetzt, so daß die endgültige Rekursion

$$y_j^{(a+1)} = y_j^{(a)} - x_j^{(a)}, \quad x_j^{(a+1)} = x_j^{(a)} + 10^{-2j}y_j^{(a)} \tag{17}$$

entsteht. Dies ist wiederum die Rekursion einer modifizierten Division, wobei nun der Divisor anders zu ändern ist. Zu Beginn ist

$$y_0^{(0)} = y, \quad x_0^{(0)} = x,$$

und beim Übergang zur Berechnung der nächsten „Quotientenziffer" ist

$$y_{j+1}^{(0)} = 10y_j^{(q_j)}, \quad x_{j+1}^{(0)} = x_j^{(q_j)}.$$

Tab. 3.6. Zur schnellen Berechnung des Arcustangens

k	x	y	Q	
0	10000000000	5200713348	0000000000	
		zu klein, Verschiebung		
1	10000000000	5200713348	0	
	+ 5200713350	− 10000000000		
	10520071335	4200713348		01
	+ 4200713350	− 10520071335		
	10940142670	3148706215		02
	+ 3148706215	− 10940142670		
	11255013291	2054569194		03
	+ 2054569194	− 11255013291		
	11460482485	999906184		04
		zu klein, Verschiebung		
2	11460482485	9299061840	0000000040	
	+ 9299061	− 11460482485		
	11469774391	8145857955		1
	+ 8145858	− 11469774391		
	11477920249	6998804964		2
	+ 6998880	− 11477920249		
	11484919129	5851088715		3
	+ 5851088	− 11484919129		
	11490770217	4702596586		4
	+ 4702597	− 11490770217		
	11495472814	3553519536		5
	+ 3553520	− 11495472814		
	11499026334	2403972255		6
	+ 2403972	− 11499026334		
	11501430306	1254069622		7
	+ 1254070	− 11501430306		
	11502684376	1039265915	0000000048	
		zu klein, Verschiebung		
3	11502684376	10392659150	0000000480	
		zu klein, Verschiebung		
4	11502684376	1039265915	0000004800	
	+ 1039	− 11502684376		
	11502685415	924239071		1

k	x	y	Q	
	11502685415	924239071		1
	+ 924	− 11502685415		
	11502686339	809212217		2
	+ 809	− 11502686339		
	11502687148	694185353		3
	+ 694	− 11502687148		
	11502687842	579158482		4
	+ 579	− 11502687842		
	11502688421	464131603		5
	+ 464	− 11502688421		
	11502688885	349104719		6
	+ 349	− 11502688885		
	11502689234	234077830		7
	+ 234	− 11502689234		
	11502689468	119056938		8
	+ 119	− 11502689468		
	11502689587	4024043720		9
		zu klein, Verschiebung		
5	11502689587	4024043720	0000048090	
		zu klein, Verschiebung		
6	11502689587	4024043720	0000480900	

k	q_k	arc tan 10^{-k}
0	0	45.000000000
1	4	5.710593137
2	8	0.572938698
3	0	0.057295760
4	9	0.005729578
5	0	0.000572958
6	3	0.000057296
7	4	0.000005730
8	9	0.000000573
9	8	0.000000057
10	4	0.000000006

Es sei z. B. arctan 0.5200713348 und 10stellig zu berechnen. Dann bildet man

$$y = 5200713348, \qquad x = 1000000000,$$

und die Rechnung verläuft wie in Tab. 3.6 gezeigt. Von der Stelle $k = 6$ an wird der Divisor nicht mehr verändert, da $10^{-12} \cdot 10^{10} < 1$ ist. Der Prozeß wird zu einer echten Division und liefert nun die weiteren Ziffern

$$\frac{40240437200}{11502689587} = 3.4984,$$

und somit entsteht $Q = 0.480903\,4984$.

Man hätte sogar bereits ab $k = 4$ auf eine echte Division überleiten können, denn von da an ändert sich der Divisor nur in den letzten vier Stellen, so daß die ersten sechs Stellen des Quotienten nicht beeinflußt werden:

$$\frac{1039265915\,00}{11502684376} = 9.034986.$$

Mit vorbereiteten Werten von arctan 10^{-k} ergibt sich somit gemäß Formel (14) das Resultat

$$\arctan 0.5200713348 = 27.47764878 \text{ in Grad.}$$

Es ist korrekt mit einem Relativfehler bei 10^{-10}.

3.2.4. Berechnung der Quadratwurzel

Hier werden die Ziffern der Quadratwurzel sofort geliefert. Die Methode entspricht weitgehend der bis 1958 im Schulunterricht der Klasse 8 gelehrten.

Zur Berechnung von

$$\sqrt{\frac{y}{x}} = \sum_{k=0}^{\infty} q_k \, 10^{-k} \tag{18}$$

wird

$$y = x\left(\sum_{k=0}^{\infty} q_k \, 10^{-k}\right)^2 \tag{19}$$

gebildet. Sind bereits die Ziffern $q_0, \ldots, q_{j-1}$ bekannt, so findet man q_j durch Suchen des größtmöglichen a in

$$y_j^{(a)} = y - x\left(\sum_{k=0}^{j-1} q_k \, 10^{-k} + a \cdot 10^{-j}\right)^2, \tag{20}$$

so daß $y_j^{(a)}$ möglichst klein wird. Mit der Abkürzung

$$x_j^{(a)} = 2x\left(\sum_{k=0}^{j-1} q_k \, 10^{-k} + a \cdot 10^{-j}\right) + x \cdot 10^{-j} \tag{21}$$

erhält man leicht die Rekursion aus (20)

$$y_j^{(a+1)} = y_j^{(a)} - 10^{-j} \cdot x_j^{(a)} \tag{22}$$

und ebenso aus (20)

$$x_j^{(a+1)} = x_j^{(a)} + 2x \cdot 10^{-j}.$$

Dies entspricht wiederum einer modifizierten Division, wobei der Divisor vor jeder Subtraktion um $2x \cdot 10^{-j}$ vergrößert wird. Zur Vermeidung von Genauigkeitsverlusten setzt man besser

$$y_j^{(a)} := 10^j y_j^{(a)}$$

und erhält endgültig

$$\begin{aligned} y_j^{(a+1)} &= y_j^{(a)} - x_j^{(a)}, \\ x_j^{(a+1)} &= x_j^{(a)} + 2x \cdot 10^{-j}. \end{aligned} \tag{23}$$

Beim Übergang zu einer neuen Ziffer ist zunächst einfach

$$y_{j+1}^{(0)} = 10 y_j^{(q_j)}.$$

Aber nach (21) wird

$$\begin{aligned} x_{j+1}^{(0)} &= 2x\left(\sum_{k=0}^{j} q_k \cdot 10^{-k}\right) + x \cdot 10^{-(j+1)} = x_j^{(q_j)} - x \cdot 10^{-j} + x \cdot 10^{-(j+1)}, \\ x_{j+1}^{(0)} &= x_j^{(q_j)} - 0.9x \cdot 10^{-j}. \end{aligned} \tag{24}$$

Zu Beginn der gesamten Rechnung ist

$$y_0^{(0)} = y, \quad x_0^{(0)} = x. \tag{25}$$

Als Beispiel wurde $\sqrt{2}$ errechnet. Es ist

$$\begin{aligned} y &= 200000000, \\ x &= 100000000, \\ 2x &= 200000000, \\ 0.9x &= 90000000. \end{aligned}$$

Die Rechnung verläuft wie in Tab. 3.7 gezeigt. Somit erhält man den korrekten Wert

$$\sqrt{2} = 1.414213562.$$

Tab. 3.7. Zur schnellen Berechnung der Quadratwurzel

k	z	y	Q
0	100000000	200000000	000000000
	+ 200000000	− 100000000	
	300000000	100000000	1
	− 90000000	zu klein, Verschiebung	
1	210000000	1000000000	000000010
	+ 20000000	− 210000000	
	230000000	790000000	1
	+ 20000000	− 230000000	
	250000000	560000000	2
	+ 20000000	− 250000000	
	270000000	310000000	3
	+ 20000000	− 270000000	
	290000000	40000000	4
	− 9000000	zu klein, Verschiebung	
2	281000000	400000000	000000140
	+ 2000000	− 281000000	
	283000000	119000000	1
	− 900000	zu klein, Verschiebung	
3	282100000	1190000000	000001410
	+ 200000	− 282100000	
	282300000	907900000	1
	+ 200000	− 282300000	
	282500000	625600000	2
	+ 200000	− 282500000	
	282700000	343100000	3
	+ 200000	− 282700000	
	282900000	60400000	4
	− 90000	zu klein, Verschiebung	
	282810000	604000000	000014140
	+ 20000	− 282810000	
	282830000	321190000	1
	+ 20000	− 282830000	
	282850000	38360000	2
	− 9000	zu klein, Verschiebung	
5	282841000	383600000	000141420
	+ 2000	− 282841000	
	282843000	100759000	1
	− 900	zu klein, Verschiebung	
6	282842100	1007590000	001414210

k	z	y	Q
6	282842100	1007590000	001414210
	+ 200	− 282842100	
	282842300	724747900	1
	+ 200	− 282842300	
	282842500	441905600	2
	+ 200	− 282842500	
	282842700	159063100	3
	− 90	zu klein, Verschiebung	
7	282842610	1590631000	014142130
	+ 20	− 282842610	
	282842630	1307788390	1
	+ 20	− 282842630	
	282842650	1024945760	2
	+ 20	− 282842650	
	282842670	742103110	3
	+ 20	− 282842670	
	282842690	459260440	4
	+ 20	− 282842690	
	282842710	176417750	5
	− 9	zu klein, Verschiebung	
8	282842701	1764177500	141421350
	+ 2	− 282842701	
	282842703	1481334799	1
	+ 2	− 282842703	
	282842705	1198492096	2
	+ 2	282842705	
	282842707	915649391	3
	+ 2	− 282842707	
	282842709	632806684	4
	+ 2	− 282842709	
	282842711	349963975	5
	+ 2	− 282842711	
	282842713	67121264	6
	− 1	zu klein, Verschiebung	
9	282842712	671212640	1414213560
		− 282842712	
		388369928	1
		− 282842712	
		105527216	2
		zu klein, Ende	

Man erkennt am Verlauf der Rechnung, daß man bereits ab $k = 5$ mit einer echten Division hätte rechnen können, denn die Änderungen am Divisor liegen dann in den letzten vier Stellen, so daß sie auf die ersten Ziffern des Resultates keinen Einfluß haben:

$$\frac{383\,600\,000}{282\,841\,000} = 1.3562.$$

Diese fünf Ziffern sind tatsächlich die noch fehlenden zu den bei $k = 5$ bereits vorhandenen 1.4142.

3.2.5. Schnelle Berechnung der Exponentialfunktion

Man geht von der Beziehung

$$e^x = \prod_{k=0}^{\infty} (1 + 10^{-k})^{q_k} \tag{26}$$

aus. Diese führt auf

$$x = \sum_{k=0}^{\infty} q_k \ln (1 + 10^{-k}), \tag{27}$$

und die „Ziffern" q_k müssen für gegebenes x als erstes berechnet werden. Da alle q_k, also auch q_0, kleiner als 10 sein sollen, erhält man als Beschränkung

$$x < 10 \ln 2 = 6.93 = x_{\max}.$$

Bei größeren x wird also eine Reduktion nötig sein, die aber nur das Speichern weniger Konstanten erfordert, da bereits

$$e^{230.25} \approx (1 - 10^{-10})\, 10^{99}$$

die größte Zahl ergibt, die in Taschenrechnern dargestellt werden kann. Man wird also von einem gegebenen $x \leqq 230.25$, so oft es geht, $6.93^2 = 48.02$ subtrahieren und ebenso oft den Faktor

$$e^{48.02} = 7.16 \cdot 10^{20}$$

in ein Produkt einbauen und danach den Vorgang mit 6.93 bzw. mit

$$e^{6.93} = 1022.49$$

wiederholen. Ein anderer Weg, die Schranke zu umgehen, wäre die Erweiterung der Formel (26) bzw. (27) auf ein geeignetes k ($k_{\min} = -12$), jedoch könnte dies auch in der Mikroprogrammierung aufwendiger sein.

Die Ziffern q_k erhält man durch eine modifizierte Division, bei der der Divisor ständig gewechselt wird. Bei der Berechnung der Ziffer q_j ist er $\ln (1 + 10^{-j})$. Diese Größen können aus demselben (read only memory, ROM) Speicher genommen werden, der bereits in der Beschreibung der schnellen Berechnung des Logarithmus erwähnt wurde.

Ein Beispiel wird mit $x = 2.000\,000\,000$ in Tab. 3.8 durchgerechnet. Bereits ab $k = 3$ erkennt man, daß die weiteren Ziffern von Q mit denen aus der Spalte x übereinstimmen. Es ist also bei 10stelliger Rechnung

$$Q = 2.642\,044\,236.$$

Tab. 3.8. Zur schnellen Berechnung der Exponentialfunktion

k	Divisor	x	Q
0	ln 2 =	2.000000000	000000000
	0.693147181	− 693147181	1
		− 693147181	2
		0.613705639	
		zu klein,	
		Verschiebung	
1	ln 1.1 =	613705390	00000020
	0.095310180	− 953101798	1
		− 953101798	2
		− 953101798	3
		− 953101798	4
		− 953101798	5
		− 953101798	6
		418445602	
		zu klein,	
		Verschiebung	
2	ln 1.01 =	418456020	000000260
	0.009950331	− 995033080	1
		− 995033080	2
		− 995033080	3
		− 995033080	4
		204323700	
		zu klein,	
		Verschiebung	
3	ln 1.001 =	2043237000	000002640
	0.000999500	− 999500300	1
		− 999500300	2
		44236400	
		zu klein,	
		Verschiebung	
4	ln 1.0001 =	442364000	000026420
	0.000099995	− 999950000	
		zu klein,	
		Verschiebung	
5	ln 1.00001 =	4423640000	000264200
	0.000010000	− 999990000	1
		− 999990000	2
		− 999990000	3
		− 999990000	4
		423680000	
		zu klein,	
		Verschiebung	
6	ln 1.000001 =	4236800000	002642040
	0.000001000	−1000000000	1
		−1000000000	2
		−1000000000	3
		−1000000000	4
		236800000	
		zu klein,	
		Verschiebung	
7	ln 1.0000001 =	23680000000	026420440
	0.000000100		

j	y	Q
0	1000000000	2642044236
	1	1
	2	0
1	4000000000	642044236
	4	5
	44	4
	484	3
	5324	2
	58564	1
	644204	0
2	7086244000	42044236
	70862440	3
	71571064	2
	72286775	1
	73009643	0
3	7373973922	2044236
	7373974	1
	7381348	0
4	7388729244	044236
5	7388729244	44236
	73887	3
	73888	2
	73889	1
	73890	0
6	7389024798	4236
	7389	3
	7389	2
	7389	1
	7389	0
7	7389054354	236
	739	1
	739	0
8	7389055832	36
	74	2
	74	1
	74	0
9	7389056054	6
	7	5
	7	4
	7	3
	7	2
	7	1
	7	0
	7389056096	

Der nächste Schritt besteht in der Realisierung der Formel (26). Es sei etwa das Teilprodukt

$$y_j^{(0)} = \prod_{k=0}^{j-1} (1 + 10^{-k})^{q_k}$$

schon berechnet. Setzt man

$$y_j^{(a)} = \prod_{k=0}^{j-1} (1 + 10^{-k})^{q_k} (1 + 10^{-j})^a, \tag{28}$$

so erhält man

$$y_j^{(a+1)} = y_j^{(a)} + 10^{-j} y_j^{(a)} \tag{29}$$

durch Rekursion.

Beim Übergang zu einer neuen Ziffer q_{j+1} ist

$$y_{j+1}^{(0)} = y_j^{(q_j)}, \tag{30}$$

und ganz zu Beginn ist

$$y_0^{(0)} = 1. \tag{31}$$

Die Rekursionsvorschrift (29), (30) und (31) hat große Ähnlichkeit mit der Multiplikation.

Das Ergebnis ist somit (Tab. 3.8)

$$e^2 = 7.389056096,$$

und dies ist infolge der Auf- und Abrundungen um $\sqrt{3} \cdot 10^{-9}$ zu klein. Diesen Fehler kann man verringern, wenn in der Mikroprogrammierung zwei oder drei Schutzstellen (Registerlänge vergrößern) mitgeführt werden.

3.2.6. Schnelle Berechnung des Tangens

Zur Berechnung von $\tan x$ wird

$$0 < x < \frac{\pi}{2}$$

vorausgesetzt. Mit vorbereiteten Zahlen $\arctan 10^{-k}$ (sie können aus demselben Konstantenspeicher genommen werden, der bereits bei der Berechnung des Arcustangens verwendet wurde) wird mit einer modifizierten Division

$$x = \sum_{k=0}^{N} q_k \arctan 10^{-k} \tag{32}$$

etwa bis $N = 10$ gebildet. Mit diesen q_k werden entsprechend der Formel (13) aus dem Abschnitt „Schnelle Berechnung des Arcustangens", die zu

$$u + iv = R \prod_{k=0}^{\infty} (1 + i \cdot 10^{-k})^{q_k} \tag{33}$$

umgeformt ist, die Größen u und v errechnet, und schließlich ist

$$\tan x = v/u. \tag{34}$$

Dabei ist die Größe R beliebig und wird passend als $R = 1$ angesetzt.

Ein Beispiel wird mit $x = 1.047197551 \approx \frac{\pi}{3}$ in Tab. 3.9 durchgerechnet. Bei $k = 5$ bilden die Ziffern in der x-Spalte bereits die noch fehlenden Ziffern des „Quotienten". Somit ist

$$Q = 1.2624640826.$$

Tab. 3.9. Zur schnellen Berechnung des Tangens

k	Divisor	x	Q
0	$\arctan 1 =$	1.047197551	000000000
	0.785398164	−0.785398164	1
		261799387	
		zu klein, Verschiebung	
1	$\arctan 0.1 =$	261799386 0	000000010
	0.099668653	− 99668653	1
		− 99668653	2
		62620820	
		zu klein, Verschiebung	
2	$\arctan 0.01 =$	6246208200	000000120
	0.009999667	− 9999669	1
		− 9999669	2
		− 9999669	3
		− 9999669	4
		− 9999669	5
		− 9999669	6
		2468198	
		zu klein, Verschiebung	
3	$\arctan 0.001 =$	2464081980	000001260
	0.000999999	− 999999666	1
		− 999999666	2
		464082648	
		zu klein, Verschiebung	
4	$\arctan 0.0001 =$	4640826480	000012620
	0.000100000	− 999999990	1
		− 999999990	2
		− 999999990	3
		− 999999990	4
		640826520	
		zu klein, Verschiebung	
5	$\arctan 0.00001 =$	6408265200	000126240
	0.000010000		

j	u	v	Q
9	1000000000	0000000000	1262464083
	− 0	+ 1	2
	− 0	+ 1	1
	− 0	+ 1	0
	1000000000	3000000000	
8	1000000000	300000000	126246408
	− 0	+ 1000000000	7
	− 0	+ 1	6
	− 0	+ 1	5
	− 0	+ 1	4
	− 0	+ 1	3
	− 0	+ 1	2
	− 0	+ 1	1
	− 0	+ 1	0
	1000000000	8300000000	

j	u	v	Q
7	1000000000	830000000	12624640
6	1000000000	83000000	1262464
	− 0	+ 1	3
	− 0	+ 1	2
	− 0	+ 1	1
	− 0	+ 1	0
	1000000000	4083000000	
5	1000000000	408300000	126246
	− 0	+ 1	5
	− 0	+ 1	4
	− 0	+ 1	3
	− 0	+ 1	2
	− 0	+ 1	1
	− 0	+ 1	0
	1000000000	6408300000	
4	1000000000	640830000	12624
	− 6	+ 1000000000	3
	999999994	1640830000	
	− 16	+ 999999994	2
	999999978	2640829994	
	− 26	+ 999999978	1
	999999952	3640829972	
	− 36	+ 999999952	0
	999999916	4640829924	
3	999999916	464082992	1262
	− 464	+ 999999916	1
	999999452	1464082908	
	− 1464	+ 999999452	0
	999997988	2464082360	
2	999997988	246408236	126
	− 24641	+ 999997988	5
	999973347	1246406224	
	− 124641	+ 999973347	4
	999848706	2246379571	
	− 224638	+ 999848706	3
	999624068	3246228277	
	− 324623	+ 999624068	2
	999299445	4245852345	
	− 424585	+ 999299445	1
	998874860	5245151790	
	− 524515	+ 998874860	0
	998350345	6244026650	
1	998350345	624402665	12
	− 6244027	+ 998350345	1
	992106318	1622753010	
	− 16227530	+ 992106318	0
	975878788	2614859328	
0	975878788	261485933	1
	− 261485933	+ 975878788	0
	0.714392855	1.237364721	

Zur Bearbeitung des Produktes aus (33) bildet man die Zwischenwerte

$$u_j^{(a)} + iv_j^{(a)} = \prod_{k=j+1}^{N} (1 + i \cdot 10^{-k})^{q_k} \, (1 + i \cdot 10^{-j})^a , \tag{35}$$

d. h., man beginnt mit der letzten Ziffer q_N von Q. Mit

$$v_j^{(a)} := 10^j v_j^{(a)}$$

zur Vermeidung von Stellenverlusten erhält man die Rekursion

$$v_j^{(a+1)} = v_j^{(a)} + u_j^{(a)}, \quad u_j^{(a+1)} = u_j^{(a)} - 10^{-2j} v_j^{(a)}, \tag{36}$$

und dies sind Formeln für eine modifizierte Multiplikation. Der Multiplikand $u_j^{(a)}$ wird laufend geändert. Beim Übergang auf eine neue Ziffer gilt

$$v_{j-1}^{(0)} = 10^{-1} v_j^{(q_j)}, \quad u_{j-1}^{(0)} = u_j^{(q_j)}, \tag{37}$$

und zu Beginn ist

$$v_N^{(0)} = 0, \quad u_N^{(0)} = 1 \ (= R).$$

Im Beispiel der Tab. 3.9 ändert sich u wegen des Faktors 10^{-2j} erst ab $j = 4$. Es ergibt sich

$$\frac{v}{u} = \frac{1.237364721}{0.714392855} = 1.732050807.$$

Dieser Wert ist bis auf 10^{-9} übereinstimmend mit $\sqrt{3}$, dem Tangens von 60° oder $\pi/3$.

3.2.7. Berechnung der üblichen sonstigen transzendenten Funktionen

Wenn mittels der Mikroprogrammierung die Funktionen

$$\ln(x), \quad \exp(x) = e^x, \quad \arctan(x), \quad \tan(x) \quad \text{und} \quad \operatorname{sqrt}(x) = \sqrt{x}.$$

berechnet wurden, genügen wenige Formeln, um die auf Taschenrechnern üblichen sonstigen transzendenten Funktionen ebenfalls zu erhalten. Für den dekadischen Logarithmus und die Exponentialfunktion mit 10 als Basis ist es bereits erwähnt worden:

$$\lg(x) = \frac{\ln(x)}{\ln(10)}, \quad 10^x = \exp(x/\lg e).$$

Die Hyperbelfunktionen erhält man leicht aus

$$\sinh(x) = \frac{1}{2}\left(\exp(x) - 1/\exp(x)\right),$$

$$\cosh(x) = \frac{1}{2}\left(\exp(x) + 1/\exp(x)\right).$$

Die trigonometrischen Funktionen ergeben sich aus

$$\sin(x) = \frac{\tan(x)}{\sqrt{1 + \tan^2(x)}}, \quad \cos(x) = \frac{1}{\sqrt{1 + \tan^2(x)}}, \quad \cot(x) = 1/\tan(x)$$

oder, wenn man schon auf die Werte u und v aus dem Abschnitt „Schnelle Berechnung des Tangens" (34) zurückgreift,

$$\sin(x) = \sqrt{\frac{v^2}{u^2 + v^2}}, \quad \cos(x) = \sqrt{\frac{u^2}{u^2 + v^2}}.$$

Ebenso einfach findet man die Arcusfunktionen

$$\arcsin(x) = \arctan\sqrt{\frac{x^2}{1-x^2}}, \quad \operatorname{arccos}(x) = \arctan\sqrt{\frac{1-x^2}{x^2}}.$$

Die Areafunktionen oder inversen Hyperbelfunktionen erhält man aus

$$\operatorname{Arsinh}(x) = \ln\left(x + \sqrt{x^2+1}\right),$$

$$\operatorname{Arcosh}(x) = \pm\ln\left(x + \sqrt{x^2-1}\right) \quad \text{für } x \geqq 1$$

oder besser

$$\operatorname{Arcosh}(x) = \pm\ln\left(1 + \sqrt{x-1}\left(\sqrt{x-1} + \sqrt{x+1}\right)\right) \quad \text{für} \quad x \geqq 1.$$

3.3. Mikroelektronik

3.3.1. Physikalische Grundlagen

Die Mikroelektronik basiert auf der Physik der Festkörper oder genauer gesagt der elektronischen Halbleiter. Deren Erforschung begann schon vor etwa 100 Jahren mit der Entdeckung des Sperrschichteffektes und setzte sich seit der Entwicklung des Transistors im Jahre 1948 stürmisch fort. Während zunächst der Transistor als funktioneller Ersatz schaltender Elektronenröhren (Triode: Anode, Katode, Gitter) noch eine Baugröße von 0.25 bis 1 cm³ einnahm, konnte in der Folge (Beginn der Entwicklung 1960, Beherrschung der Planar-Technologie Ende der fünfziger Jahre) das Bauvolumen drastisch gesenkt werden. Dies gelang durch komplizierte technologische Prozesse wie Oxidation, Ätzen, Diffusion, Ionenimplantation in dünnen oberflächennahen Bereichen mit Dicken zwischen 10 und 300 nm. Das heißt, man arbeitet mit Schichtdicken, bei denen bereits Interferenzerscheinungen im sichtbaren Licht auftreten, wie man sie bei dünnen Ölfilmen auf Wasser oder bei Seifenblasen in Form von Farbeffekten kennt, d. h., die Schichtdicken liegen im Bereich der Wellenlänge sichtbaren Lichtes, und an sich undurchsichtige Stoffe, wie z. B. Aluminium, werden lichtdurchlässig (Wellenlänge sichtbaren Lichtes von 360 nm (violett) bis 780 nm (rot)). So erreicht man heute Dichten von 10000 bis 180000 Bauelementen pro Chip. Dabei ist ein Chip ein Trägerplättchen in der Größe von 12 bis 40 mm². Mit wenigen Chips kann man Schaltungskomplexe realisieren, die den früheren großen Rechenanlagen entsprechen.

In der überwiegenden Zahl der Taschenrechner befindet sich beispielsweise ein einziges Chip, ebenso in einer digitalen Armbanduhr. Solche Chips tragen also bereits Funktionen, welche nicht nur arithmetische Grundfunktionen, sondern höhere Funktionen und die gesamte Rechnersteuerung beinhalten. Bei der Armbanduhr sind dies Taktgenerator mit Schwingquarz und periodische Zähler für Monat, Monatstag, Wochentage, Stunden, Minuten, Sekunden, wobei noch Beziehungen zwischen den Zählern beachtet werden müssen (Monatstage 28, 30, 31 in Abhängigkeit vom Monat).

Da diese Chips in einer Massenfertigung, also verhältnismäßig billig entstehen, eröffnen sich neben den beiden erwähnten Anwendungsgebieten zahlreiche weitere Einsatzmöglichkeiten, von denen man sich gegenwärtig noch kaum eine Vorstellung machen kann. Die niedrigen Preise der Herstellung dürfen aber nicht darüber hinwegtäuschen, daß die Investitionen für die Beherrschung der Technologie enorm sind.

3.3.2. Grenzschichteffekt und Siliziumhalbleiterdiode

Die Halbleiter besitzen, wie der Name schon sagt, eine elektrische Leitfähigkeit, die bei Zimmertemperatur zwischen derjenigen für Leiter (z. B. Metalle) und derjenigen für Isolatoren (z. B. Marmor oder Bernstein) liegt. Schalttafeln in älteren Physik-Kabinetten der Schulen sind auf Marmor unter Ausnutzung der Isolationseigenschaften montiert. Noch besser isoliert Bernstein, weshalb auch eine durch Reibung aufgebrachte Oberflächenladung sich nicht ausgleichen und abfließen kann. So aufgeladener Bernstein zieht kleine leichte Körper, etwa Papierschnipselchen, an bzw. stößt sie ab. Aus dem Namen *elektron*, den die Griechen in der Antike dem Bernstein gaben, ist (im Zusammenhang mit den elektrostatischen Eigenschaften dieses Stoffes) der Name *Elektrizität* entstanden. Die elektrische Leitfähigkeit ist der reziproke Wert des spezifischen Widerstandes (Ohm · cm):

Kupfer	$1.75 \cdot 10^{-6}$,	Germanium	$10^{-2} \ldots 10^{2}$,
Aluminium	$2.82 \cdot 10^{-6}$,	Marmor	10^{10},
Silizium	$10^{-1} \ldots 10^{6}$,	Bernstein	10^{20}.

Die großen Intervalle bei den Halbleitern rühren daher, daß ihre spezifischen Widerstände sehr stark von der Temperatur, von Lichteinwirkung und von Verunreinigungen abhängen. Die Elemente Germanium und Silizium sind typische Vertreter der Halbleiter und werden auch technisch benutzt. Im periodischen System der Elemente stehen sie in der vierten Gruppe, da ihre äußerste Schale vier Elektronen enthält: .

III	IV	V
B	C	N
Al	Si	P
Ga	Ge	As
In	Sn	Sb

Die relativen Atommassen von Silizium und Germanium sind 28.06 bzw. 72.60, ihre Dichten 2.4 bzw. 5.35 g cm^{-3}. In einem Kubikmillimeter befinden sich demnach $5.15 \cdot 10^{19}$ Atome bei Silizium und $4.44 \cdot 10^{19}$ bei Germanium.

Im reinen Halbleiterkristall (der nur in besonderer Technologie der Einkristallherstellung [65] bei kristallographischer Perfektion, d. h. keine Versetzungen u. a.

gefertigt werden kann), also in durch Fremdatome ungestörtem Zustand und außerdem bei tiefen Temperaturen, sorgen die vier sogenannten Valenzelektronen des chemisch vierwertigen Siliziums (bei Germanium entsprechend) für die feste Bindung der Atome (Abb. 3.8.).

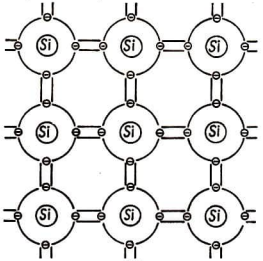


Abb. 3.8. Kristallgitter eines Siliziumkristalls in schematischer Darstellung. Alle Elektronen sind fest eingebaut. Der Kristall ist elektrisch nicht leitend

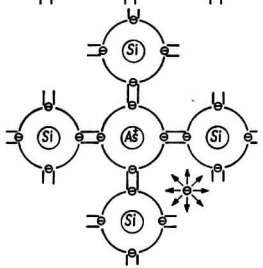


Abb. 3.9. Kristallgitter eines mit einem Arsenatom dotierten Siliziumkristalls in schematischer Darstellung. Das fünfte Elektron aus der äußersten Schale des Arsens ist frei beweglich. Der Kristall ist elektrisch leitend durch negative Ladungsträger

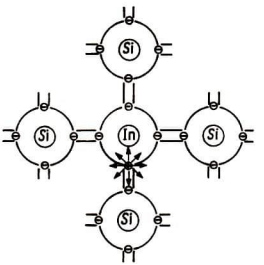


Abb. 3.10. Kristallgitter eines mit einem Indiumatom dotierten Siliziumkristalls in schematischer Darstellung. Die drei Elektronen in der äußersten Schale des Indiums lassen ein Loch in der Struktur, das sich durch Hineinspringen benachbarter Elektronen wie ein positiver Ladungsträger bewegt

Die elektronischen Halbleiter zeichnen sich dadurch aus, daß bei Zimmertemperatur die Energie der Gitterschwingungen ausreicht, um einige Elektronen aus der Gitterbildung zu befreien, so daß bei Anlegen eines elektrischen Feldes eine Leitfähigkeit (Eigenleitung) auftritt.

Durch Einbau von fünf- bzw. dreiwertigen Fremdatomen (z. B. Phosphor, Bor) in das Wirtsgitter mit vierwertigen Atomen (Germanium, Silizium) kann man die Leitfähigkeit um Größenordnungen erhöhen (Abb. 3.9 und 3.10).

Den Einbau von Fremdatomen in das Kristallgitter nennt man *Dotieren* (dotare (lat.) = hinzugeben). Wird ein Atom der fünften Gruppe, z. B. Arsen, eingebaut,

so bleibt ein freies Elektron (Abb. 3.9). Der Halbleiter wird *n-leitend* (n von negativ). Atome, die Elektronen abgeben, nennt man *Donatoren* (donatio (lat.) = Schenkung). Das Donatoratom selbst zeigt sich durch die Elektronenabgabe als positive ortsfeste Ladung.

Bei einer Dotierung von einem As-Atom auf etwa je 10^7 Si-Atome spricht man von einer normalen *n-Dotierung* (n von negativ) und erreicht einen spezifischen Widerstand von etwa 5 Ohm · cm. Wird stark dotiert, d. h. mit einem As-Atom auf bereits 10^4 Si-Atome, also tausendmal stärker, so wird dies n^+-*Dotierung* genannt. Der spezifische Widerstand fällt auf etwa den hundertsten Teil, 0.05 Ohm · cm.

Wird ein Atom der dritten Gruppe, z. B. Indium, in das Kristallgitter eingebaut, so reichen dessen drei Außenelektronen nicht zum Absättigen der Bindungen in der Kristallstruktur aus. Es bleibt eine Lücke oder ein *Loch*. Springen nun Elektronen

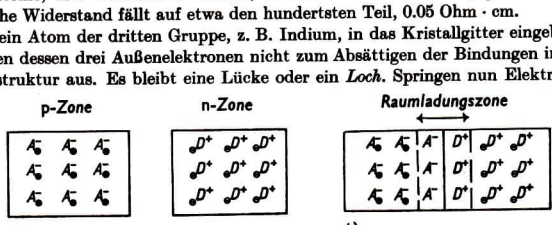


Abb. 3.11. Ausbildung und schematische Darstellung einer Raumladungszone am pn-Übergang eines dotierten Halbleiters (b)

a) In der p-Zone liegt ein elektrisch neutraler Zustand vor, jedoch gibt es im Kristallgefüge „Löcher". Auch in der n-Zone liegt ein elektrisch neutraler Zustand vor. Es gibt dort frei bewegliche Elektronen. In der Zeichnung sind die Löcher durch ein „+" gekennzeichnet, so daß zum Ausgleich die Akzeptoren (A) ein „−" tragen. Ebenso sind die frei beweglichen Elektronen durch „−" und zum Ausgleich die Donatoren (D) durch „+" markiert

b) Elektronen aus der n-Zone haben einige Löcher aus der p-Zone besetzt. Die „−" haben sich mit den „+" ausgeglichen. Durch die ohne Löcher verbleibenden Akzeptoren A^- und die ohne Elektronen verbleibenden Donatoren D^+ entsteht an der Grenzschicht eine Raumladungszone

von benachbarten Atomen in solch eine Lücke, so wandert das Loch unter dem Einfluß eines elektrischen Feldes wie ein positiver Ladungskörper durch den Kristall. Der Halbleiter wird *p-leitend*. Die Atome, die Elektronen aufnehmen, nennt man *Akzeptoren* (acceptare (lat.) = aufnehmen). Das Akzeptoratom wird durch die Elektronenaufnahme negativ geladen (Abb. 3.10).

Bei einer Dotierung von einem In-Atom auf etwa je 10^6 Si-Atome spricht man von einer normalen *p-Dotierung* (p von positiv) und erreicht einen spezifischen Widerstand von etwa 2 Ohm · cm. Wird stark dotiert, d. h. mit einem In-Atom auf bereits 10^4 Si-Atome, so sinkt der spezifische Widerstand auf 0.05 Ohm · cm. Die starke Dotierung mit Indium wird mit p^+-*Dotierung* bezeichnet.

Grenzen eine n-leitende und eine p-leitende Kristallzone aneinander, so diffundieren die frei beweglichen Elektronen aus dem n-Gebiet in das p-Gebiet, wo eine verstärkte Rekombination auftritt und daher die Löcher und die Elektronen nach der Rekombination nicht mehr als bewegliche Ladungsträger zur Verfügung stehen. Die negativ geladenen Akzeptoren werden nun nicht mehr durch die positiven Löcher

(Defektelektronen) kompensiert, und es entsteht eine negative Raumladung im p-Gebiet der Übergangszone.

Analog wandern die Löcher in das n-Gebiet und rekombinieren mit den Elektronen, so daß diese nicht mehr die ortsfesten positiv geladenen Donatoren kompensieren und eine positive Raumladung im n-Gebiet der Übergangszone entsteht. Zwischen positiver und negativer Raumladung besteht ein elektrisches Feld und somit eine Spannung, die *Diffusionsspannung*. Die verstärkte Rekombination im Übergangsgebiet bedeutet eine Verarmung an beweglichen Ladungsträgern, so daß dieses Gebiet hochohmig wird. Das Einströmen der Elektronen in die Löcher des Kristallgefüges der p-Zone kommt zum Stillstand. An dem *pn-Übergang* entsteht eine an frei beweglichen Ladungsträgern verarmte Grenzschicht, die *Raumladungszone* (Abb. 3.11).

Legt man nun an einen Siliziumkristall mit einem derartigen pn-Übergang eine Spannung nach Abb. 3.12a an, so verbreitert sich einfach die Raumladungszone. Die

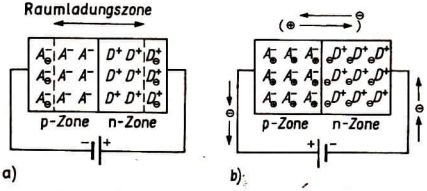


Abb. 3.12. Zur Veranschaulichung der Sperr- (a) und Durchlaßrichtung (b) einer Halbleiterdiode

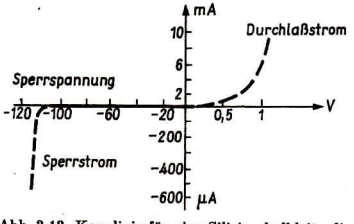


Abb. 3.13. Kennlinie für eine Siliziumhalbleiterdiode. Man beachte die unterschiedliche Skalierung auf den positiven und negativen Achsen. Bei etwa −110 V schlägt diese Diode durch (Sperrspannungen gibt es von −5 V bis −1000 V)

Elektronen können nicht durchwandern. Der Strom ist (bis auf einen geringen Sperrstrom, der durch das Befreien von Elektronen in der Raumladungszone durch thermische Anregung zustande kommt) gesperrt: *Sperrichtung*.

Legt man dagegen eine positive Spannung (Abb. 3.12b) an die p-Zone und eine negative Spannung an die n-Zone, so baut diese die Sperrschicht der Raumladungszone ab, und die Elektronen können fließen: *Durchlaßrichtung*. Es ist dadurch eine *Diode* entstanden, die zur Gleichrichtung verwendet werden kann. Der Durchlaß-

strom beginnt erst dann zu fließen, wenn die angelegte Spannung in Durchlaßrichtung einen gewissen Schwellenwert, welcher der Diffusionsspannung entspricht, übersteigt. Bei Silizium ist dieser Schwellenwert etwa 0.7 V bei Zimmertemperatur. Auch in Sperrichtung ist stets ein wenn auch kleiner Strom vorhanden. Der *Sperrstrom* kann etwa den 10^7ten Teil des Durchlaßstromes betragen. Jedoch verschwindet er nie völlig, da immer Ladungsträger an Störstellen des Kristalls vorhanden sind und in der Realität nicht die idealisierten und schematischen Verhältnisse aus Abb. 3.11 und 3.12 vorliegen. Da bei einer Erwärmung das Kristallgefüge energetisch angeregt wird, gibt es mehr Ladungsträger ab, so daß der Sperrstrom temperaturabhängig ist und mit steigender Temperatur auch steigt. Bei einem Umschalten von Durchlaß in Sperrichtung tritt eine Verzögerung auf, da der pn-Übergang, d. h. die Raumladungszone, erst von Ladungsträgern ausgeräumt werden muß, was in einem Halbleiter eine gewisse Zeit, die *Sperrerholzeit*, beansprucht. Bei speziellen Schaltdioden kann diese Zeit auf 10^{-9} s gedrückt werden. Solche Dioden können demnach im Mega- oder sogar Gigahertzbereich arbeiten. Bei besonders hoher Sperrspannung schlägt eine Halbleiterdiode durch. Abb. 3.13 zeigt eine typische Kennlinie für eine Siliziumhalbleiterdiode.

3.3.3. Transistor

Bipolarer Transistor

Der Transistor ist ein elektronisches Halbleiterbauelement. Ein Transistor (von transfer und resistor, d. h. „übertragener" (gesteuerter) Widerstand) kann die Funktion einer Triode (Vakuumröhre mit Katode, Anode und Gitter) übernehmen. Er kann also als Steuerelement oder Schalter verwendet werden. Seine Funktion als Verstärker ist in den logischen Schaltkreisen ohne große Bedeutung.

Der bipolare Transistor trägt seinen Namen, da beim Zustandekommen seiner Wirkung sowohl die (fiktiven) positiven Ladungsträger, die Löcher, als auch die

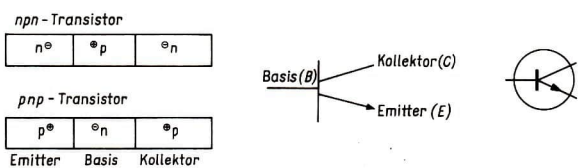


Abb. 3.14. Schematische Darstellung der Leitungszonen eines npn- und eines pnp-Transistors sowie das Schaltsymbol für einen Transistor. Erläuterung der Begriffe Basis, Kollektor und Emitter sowie der Funktion im Text

negativen, die Elektronen, beteiligt sind. Ein bipolarer Transistor besteht aus drei Zonen unterschiedlichen Leitungstyps *pnp* oder *npn* (Abb. 3.14).

Im folgenden wird anhand von Abb. 3.15 die Wirkungsweise eines *npn*-Transistors erläutert. Für einen *pnp*-Transistor gilt alles analog. Ein Transistor besitzt zwei

pn-Übergänge. Einer liegt zwischen Emitter (E) und Basis (B), der andere zwischen Basis (B) und Kollektor (C). Es wird nun eine Spannung U_{CB} in Sperrichtung zwischen Basis und Kollektor angelegt. Im Kollektorkreis fließt ein kleiner Strom I_C, der Kollektor-Sperrstrom. Wird an den Emitter-Basis-pn-Übergang eine Spannung in Durchlaßrichtung U_{EB} angelegt, so werden Elektronen vom Emitter in die Basiszone geliefert (emittiert). Es entsteht der Emitterstrom I_E. Die vom Emitter in die Basis injizierten Elektronen durchwandern das feldfreie Gebiet der Basis, bedingt durch das Konzentrationsgefälle. Da die Basis sehr dünn und wesentlich geringer dotiert ist als Emitter und Kollektor, ist die Rekombination der Elektronen in der Basis sehr gering, so daß der größte Teil der in die Basis injizierten Elektronen vom elektrischen Feld des Kollektor-Basis-Übergangs erfaßt wird und damit den Kollektorstrom erhöht. Das Anlegen der Spannung U_{EB} hat wie ein Schalter für den Strom I_C gewirkt.

Die Wirkungsweise des bipolaren Transistors beruht also auf der Injektion von Ladungsträgern vom Emitter in die Basis. Er ist *stromgesteuert*. In den Leitungszonen bewegen sich sowohl die Löcher als auch die freien Elektronen.

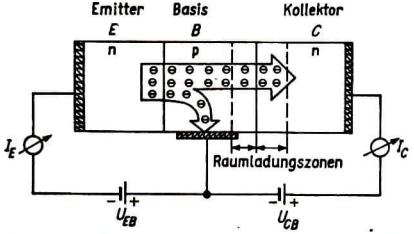


Abb. 3.15. Wirkungsprinzip eines npn-Transistors. Nähere Erläuterung im Text. Dargestellt ist nur die Injektion von Elektronen als Ladungsträger

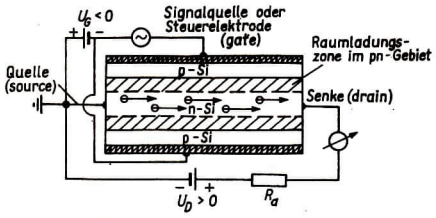


Abb. 3.16. Wirkungsprinzip eines n-Kanal-Sperrschicht-Feldeffekt-Transistors (SFET). Erhöhte negative Spannung am Gate (Steuerelektrode) verhindert durch Verbreiterung der Raumladungsfelder den Durchfluß der Elektronen von der Quelle zur Senke (R_a ist ein Außenwiderstand)

Unipolar- oder Feldeffekt-Transistor

Im Gegensatz zum bipolaren Transistor arbeitet der unipolare nur mit Ladungsträgern einer einzigen Sorte, Elektronen oder Löchern (Abb. 3.16). An den Enden einer n-leitenden Zone wird eine Spannung angelegt. Elektronen fließen von der Quelle (source) zur Senke (drain = Drainage, Abfluß). An den Seiten der n-Zone sind zwei p-Zonen angebracht, und an diesen liegt eine negative Spannung, so daß sich an den pn-Übergängen Sperrschichten oder Raumladungszonen bilden. Wird die anliegende Spannung erhöht, so vergrößern sich die Raumladungsgebiete und schnüren den Durchgang der Elektronen schließlich ab. Die Steuerelektrode an den p-Zonen wirkt wie ein Schalter (Tor = gate).

Da hier die Steuerung des Leitungsquerschnittes über das elektrische Feld der Raumladung des in Sperrichtung vorgespannten pn-Überganges erfolgt, bezeichnet man diesen Unipolar-Transistor als *pn-Feldeffekt-Transistor* (pn-FET field effect transistor). Wird dagegen die Ladungsträgerdichte eines Kanals in der Halbleiteroberfläche über eine Metallelektrode, die durch eine dünne hochisolierende SiO_2- oder Si_3N_4-Schicht von der Halbleiteroberfläche getrennt ist, gesteuert, so erhält man den *MOSFET* (metal-oxide-semiconductor-FET oder Metall-Oxid-Halbleiter-FET), auch MISFET (Isolator), der insbesondere bei der Herstellung hochintegrierter Schaltungen eingesetzt wird.

MOS-Transistor (FET) (metal oxide semiconductor transistor = Metall-Halbleiter-Transistor)

Bei diesem für die Großintegration sehr bedeutsamen Typ beeinflußt eine Steuerspannung die Leitfähigkeit einer dünnen Oberflächenschicht im Halbleiterkristall. Es gibt zwei physikalische Grundarten des MOSFET-Transistors:

p-Kanal MOSFET,

n-Kanal MOSFET,

und zwei Grundarten der Schaltzustände:

Anreicherungs-(enhancement)Transistor,

Verarmungs-(depletion)Transistor,

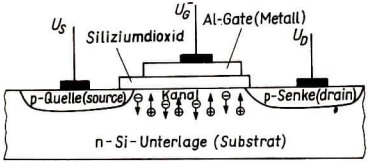


Abb. 3.17. Ein p-Kanal MOSFET-Transistor vom Anreicherungstyp im Schnitt (nähere Erläuterung im Text). Die negative Gatespannung reichert den Kanal mit positiven Ladungsträgern, d. h. Löchern, an

welche zwar beliebig kombiniert werden können, jedoch sind aus technologischen Gründen die p-Kanal MOSFET-Transistoren vorwiegend vom Anreicherungstyp und die n-Kanal MOSFET-Transistoren ebenso überwiegend vom Verarmungstyp. Was steckt hinter den genannten vier Begriffen?

Der p-Kanal MOSFET hat eine p^+-dotierte Quelle (source) und eine Senke (drain) auf einem n-dotierten Silizium-Substrat (Unterlage). Zwischen der Quelle und der Senke liegt die Spannung U_{DS} (Abb. 3.17). Zwischen Quelle und Senke befindet sich der Kanal, der von einer Aluminium (Al)-Brücke, dem Tor (gate), zwischen Quelle und Senke überspannt wird. Damit dadurch Quelle und Senke nicht leitend verbunden werden, wird durch SiO_2 (gate-oxid, auch Si_3N_4) eine Isolationsschicht zwischengelegt.

Zwischen Gate und Quelle liegt eine Steuerspannung, U_{GS}. Beim n-Kanal MOSFET sind die Dotierungen der Quelle, der Senke und der Unterlage gegenüber dem p-Kanal MOSFET vertauscht.

Ist ein MOSFET-Transistor ohne Gatespannung, d. h. nicht leitend, und tritt die Leitfähigkeit erst bei bestimmter Gatespannung ein, nämlich dann, wenn durch das Gatefeld genügend Ladungsträger in den Kanal gezogen werden, so liegt der *Anreicherungstyp* vor. Ist dagegen der MOSFET-Transistor im Ruhezustand, d. h. ohne Gatespannung, leitend und kann er bei bestimmter Gatespannung abgeschaltet werden, so ist er vom *Verarmungstyp* (das Feld der Gatespannung schnürt den Kanal ab).

Als *Schwellenspannung* wird die Gatespannung bezeichnet, welche mindestens benötigt wird, um einen Kanal zu erzeugen oder zu beseitigen. Sie liegt je nach Technik zwischen 1 V und 4 V.

3.3.4. Technologie

Züchtung von Silizium-Einkristallen

Die bereits erläuterten physikalischen Grundlagen der Halbleiterelektronik ließen deutlich werden, daß diese Effekte keine Störungen in der kristallinen Struktur des

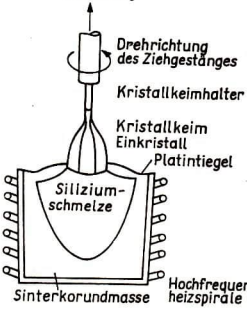


Abb. 3.18. Schematische Darstellung der Einkristallzüchtung aus einer Schmelze nach CZOCHRALSKI

Siliziumsubstrats erlauben. Die erste Forderung der Mikroelektronik ist deshalb die Herstellung chemisch hochreinen, d. h. definiert „verunreinigten" oder dotierten Siliziums in sogenannten *Einkristallen*. In solchen Kristallen haben dann alle Atome über die geforderten geometrischen Abmessungen hinweg, d. h. in Zentimeterbereichen, die in den Abb. 3.8 bis 3.10 skizzierten Plätze.

In speziellen Betrieben zur Herstellung von Reinmetallen, von Edelsteinen (ebenfalls Einkristalle), in optischen Betrieben (diese benötigen ebenfalls Einkristalle) oder eben Elektronikbetrieben werden Halbleiter-Einkristalle gezüchtet (Fototafel, Bild 1). Die größte Bedeutung hat dabei gegenwärtig für Si das tiegelfreie Zonenschmelzverfahren nach KELK und GOLAY. Das Silizium in der Schmelze ist hochrein, was (siehe [65]) nach der chemisch rein hergestellten Grundsubstanz noch durch weitere physikalische Methoden (Zonenschmelzverfahren) erreicht wird. Im Schmelztiegel (Abb. 3.18) befindet sich dieses Silizium mit den Dotierungszusätzen (etwa 1 mg Arsen oder Indium auf 1 kg Silizium, d. h. $1:10^6$ für das Masseverhältnis, oder auch anders je nach Dotierung). An den Kristallkeim am Ziehgestänge lagern sich weitere Atome geordnet an, so daß man nach einigen Stunden einen Silizium-Einkristall von etwa 125 mm Durchmesser und bis zu 100 cm Länge und den gewünschten Leitungseigenschaften erhält. Anschließend werden diese Stäbe mit Diamantsägen senkrecht zur Stabachse und damit etwa parallel zu einer Kristallebene in etwa 0.5 mm dicke Scheiben zersägt. Durch Polier-, Läpp- (Feinstpolierung) und auch Ätzvorgänge werden die vom Sägen herrührenden kristallinen Störungen an der Oberfläche beseitigt. Damit hat man die Siliziumunterlage (das Substrat), eine runde dünne Scheibe von 5 bis 8 cm Durchmesser zur weiteren Bearbeitung vorliegen [65].

Dotierungstechniken und fotolithografisches Verfahren

Die für die Halbleiterelektronik benötigten kleinen Gebiete in oder auf der Siliziumunterlage werden durch

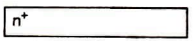
Diffusion,
kristallines Aufwachsen,
Aufdampfen,
Oxidation

hergestellt (Abb. 3.19).

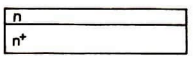
Bei der Diffusion wird das Substrat bei etwa 1100 °C dampfförmigen Verbindungen des Dotierungselements ausgesetzt. Relativ langsam diffundieren die entsprechenden Atome in den Siliziumkristall in eine Tiefe von 3 bis $10 \cdot 10^{-3}$ mm ein, je nach Temperatur und Dauer. Diese Technik kann wiederholt auf derselben Stelle im Substrat angewendet werden, so daß Schichten von p- bzw. n-leitenden Zonen entstehen. Beim kristallinen Aufwachsen bildet sich auf der Einkristalloberfläche bei etwa 1200 °C in einer Wasserstoffatmosphäre aus einer gasförmigen Siliziumverbindung eine weitere Kristallschicht.

Enthält die Atmosphäre Dotierungsstoffe, so entstehen neue Schichten vom gewünschten Leitungsverhalten.

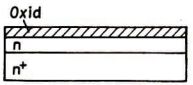
Durch Aufdampfen im Vakuum werden an den gewünschten Stellen dünne Metallfilme aus Aluminium, Gold oder Legierungen als Kontaktstellen aufgebracht. Die Oxidation, welche bei 900 bis 1200 °C in einer Sauerstoff- und Wasserdampfatmosphäre stattfindet, überzieht Silizium mit einer dünnen, aber sehr dichten und widerstandsfähigen Oxidschicht SiO_2. In ungefähr einer Stunde wird eine Schichtdicke



Hochdotierte n^+-Silizium-Einkristallscheibe, Substrat, ⌀ 50···75 mm, Dicke 0.3 mm

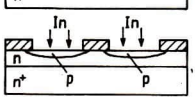


Aufwachsen einer n-dotierten Silizium-Schicht von etwa 0.01 mm Dicke (Kollektorschicht)

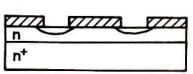


Oxidation 1; Erzeugen einer SiO_2-Schicht von etwa 0.002 mm Stärke

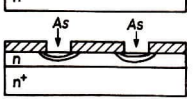
Fotolithografie 1; Ätzen von Fenstern in die Oxidschicht



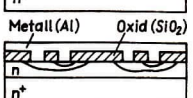
Diffusion zur Erzeugung der Transistorbasis; Indiumatome diffundieren durch die Oxidfenster in den Kristall und erzeugen die p-leitende Basiszone



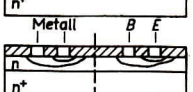
Oxidation und Fotolithografie 2; die Oberfläche wurde wieder voll oxidiert, und neue Fenster für die Emitter-Diffusion wurden geätzt



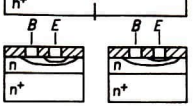
Diffusion zur Erzeugung des Transistoremitters; Arsenatome diffundieren durch die Oxidfenster in den Kristall und erzeugen die n-leitende Emitterzone



Oxidation und Fotolithografie 3; die Oberfläche wurde wieder voll oxidiert, und neue Fenster für die Metallkontakte wurden geätzt, die gesamte Oberfläche ist bereits mit Metall (Al) bedampft



Das Metall wurde abgeätzt, bis nur noch die Kontaktstellen stehen bleiben; das Substrat wurde auf 0.1 mm Dicke abgeschliffen



Trennung der Chips (hier nur je einen Transistor enthaltend) und Gehäuseeinbau (hier Kollektoranschluß *C*)

Abb. 3.19. Schematische Folge der Produktionsvorgänge einer integrierten Schaltung, vereinfacht auf die Herstellung zweier Transistoren in Planartechnik

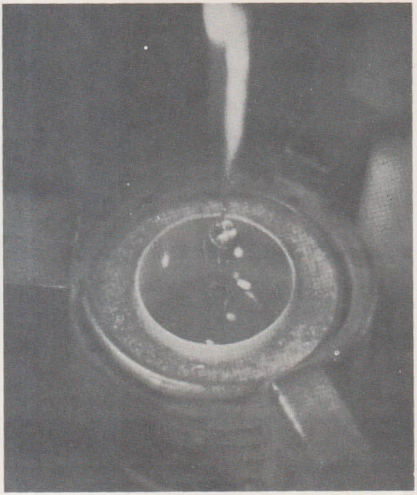


Bild 1. Ziehen eines Einkristalls aus der Schmelze nach CZOCHRALSKI [ADN-ZB/Kürsten]

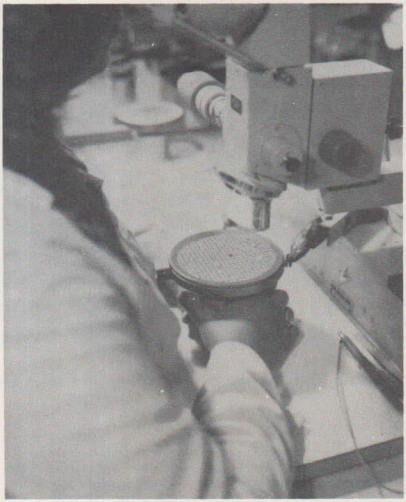


Bild 2. Chips für die Mikroelektronik werden auf runden Siliziumscheiben simultan zu etwa 200 Stück hergestellt. Das Trennen der Chips erfolgt auf einer Gummimembran, auf der bei Dehnung die Chips manipulierbar auseinanderrücken [DEWAG-Leipzig/Mokanski; Neg.-Nr. 140 231]

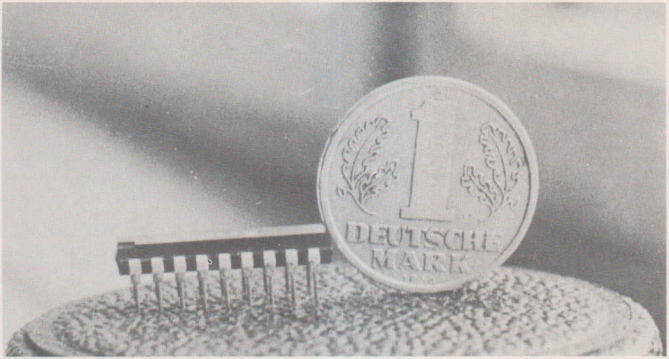


Bild 3. Mikroprozessor U 808 D im Gehäuse mit Anschlüssen, fertig zur Verwendung. Nur die Anschließbarkeit der Kontakte bedingt die gezeigte Größe (Vergleich mit 1-Mark-Stück); der Mikroprozessorchip selbst ist nur 6 mm × 6 mm groß [ADN-ZB/Ludwig 1977]

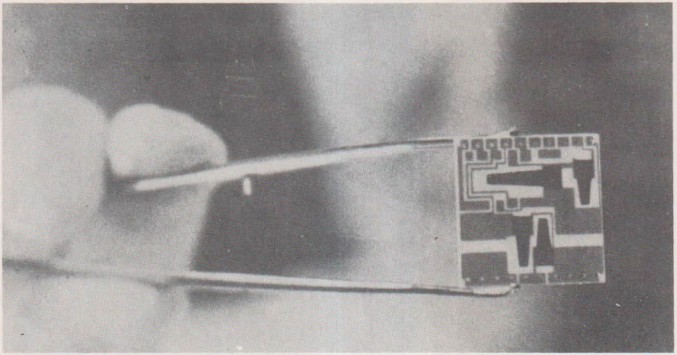


Bild 4. Chip mit kleinintegrierter Schaltung SSI (small scale integration)

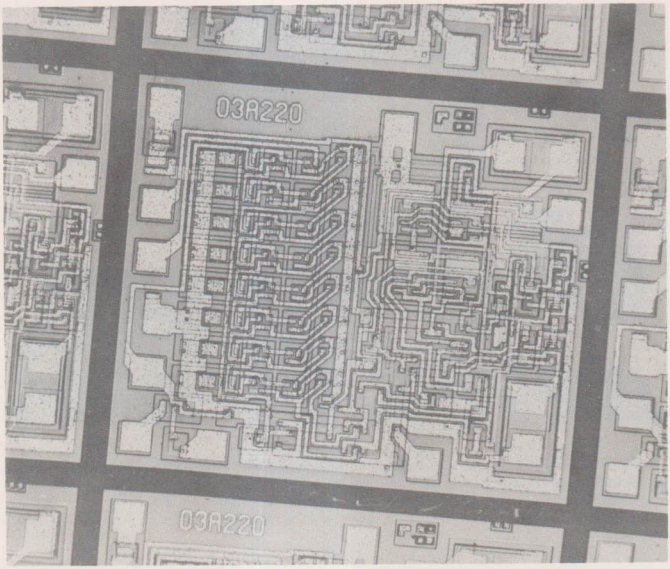


Bild 5. Mikrofoto eines Mikroprozessorchips 16 mm² [ADN-ZB/Werkfoto 1980]

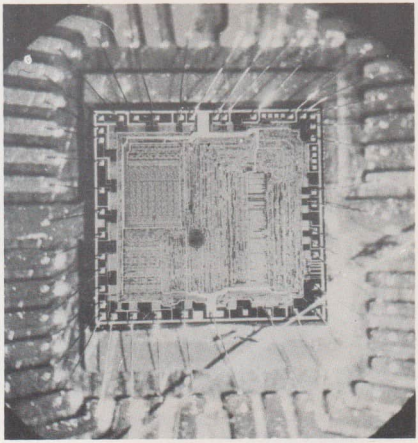


Bild 6. Blick durch ein Mikroskop auf das „Herz" eines hochintegrierten Schaltkreises mit den Anschlußverbindungen (im Vergleich dazu ein Haar) [ADN-ZB/Häßler 1981]

Bild 7. Mikrocomputer Z 9001 mit Zubehör: Farbfernsehgerät, Kasettenrecorder, BASIC-Modul, ROM und RAM Zusatzspeicher und Steuerhebel (mit Genehmigung des Herstellers VEB Robotron Meßelektronik „Otto Schön" Dresden, Bildautor Foto-Darre Dresden)

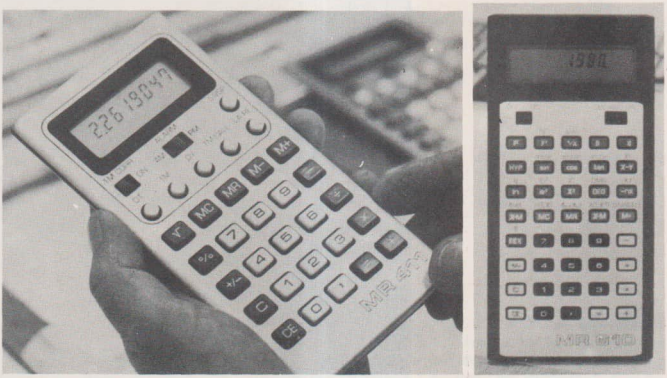


Bild 8. Moderne Taschenrechner MR 411 und MR 610 vom VEB Kombinat Mikroelektronik [ADN-ZB/Ludwig 1981 bzw. 1980]

von etwa 10^{-3} mm erreicht. Diese Oxidschicht ist bereits für die Dotieratome undurchlässig. Es besteht somit die Möglichkeit, mit Hilfe der an bestimmten Stellen abgelösten (oder an bestimmten Stellen am Entstehen verhinderten) Oxidschicht die Dotiersubstanzen nur an diesen gewünschten oxidfreien Stellen der Siliziumoberfläche einwirken zu lassen. Durch ein *fotolithografisches* Verfahren können die erforderlichen, geometrisch genau festgelegten Stellen auf dem Siliziumsubstrat fixiert werden. Zunächst wird die oxidierte Siliziumscheibe mit einem Fotolack überzogen. Dieser Lack wird durch eine Maske hindurch belichtet. Die Maske kann als Spezialwerkzeug zur Herstellung von mikroelektronischen Schaltungen angesehen werden. Sie ist wiederverwendungsfähig, aber ihre Herstellung ist außerordentlich aufwendig und teuer. Man benötigt nämlich für sie Platten- oder Filmmaterial mit extrem hohem Auflösungsvermögen, ferner optische Geräte hoher Güte und feinmechanische Einrichtungen sehr großer Genauigkeit. Es müssen nämlich auf einem Siliziumträger von etwa 5 · 6 mm² einige Tausend Schalttransistoren, Widerstände und Kondensatoren erzeugt werden. Die Oxidfenster haben also lediglich

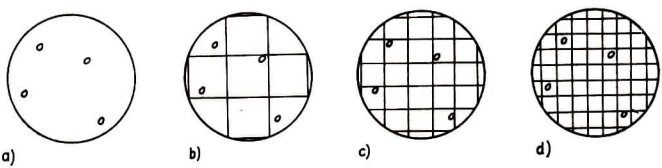


Abb. 3.20. Der Einfluß der Chipgröße auf die Ausbeute an funktionstüchtigen Schaltungen (schematisch)
a) Siliziumscheibe mit unvermeidlichen Fehlstellen. Es sei angenommen, daß jede Fehlstelle auch zum Ausfall eines Chips führt, wenn sie auf dem Chip liegt; Durchmesser der Scheibe 50 mm
b) Chipgröße etwa 16 · 16 mm², 5 Chips, Ausbeute 3 Chips, 60%
c) Chipgröße etwa 9 · 9 mm², 14 Chips, Ausbeute 12 Chips, 86%
d) Chipgröße etwa 6 · 6 mm², 42 Chips, Ausbeute 38 Chips, 90%

Durchmesser einiger tausendstel Millimeter. Nach der Belichtung kann der Fotolack an den nicht belichteten Stellen entfernt werden. An diesen Stellen wird danach die Oxidschicht weggeätzt. Anschließend wird auch der belichtete Fotolack entfernt, und nun ist das Substrat für *einen* Diffusions-, Aufwachs- oder Aufdampfvorgang vorbereitet. Zur Herstellung einer Schaltung sind jedoch mehrere Diffusions- oder Aufdampfvorgänge nötig. Das erfordert mehrfaches Anwenden der Fototechnik. Die Masken — es sind natürlich Masken mit unterschiedlicher geometrischer Struktur — müssen auf 10^{-4} Millimeter genau aufgepaßt werden. Dies geschieht unter dem Mikroskop mit Hilfe von Mikromanipulatoren, die definierte, reproduzierbare und kontrollierte Bewegungen auszuführen gestatten.

Ein Vorteil der Fototechnik ist, daß die Masken leicht kopiert werden können. Somit werden auf einer Siliziumscheibe zugleich jeweils je nach Chipgröße zwischen 80 und 1600 integrierte Schaltungen stufenweise bearbeitet (Fototafel, Bild 2). Alle Schaltungen werden auf ihre Funktion hin geprüft. Durch Ritzen mit einem Diamanten gewinnt man die einzelnen Chips von etwa $6 \cdot 6$ mm² Größe. An die Anschlußpunkte der Schaltung werden feine Gold- oder Aluminiumdrähte von 0.02 bis 0.03 mm Durchmesser gesetzt und an die späteren Lötfahnen geführt. Ein Gehäuse mit Kunstharz- oder Plastverguß bzw. aus Keramik für höhere Anforderungen komplettiert schließlich den integrierten Baustein, der nun etwa 3 bis 5 cm lang ist (Fototafel, Bild 3).

Wie bereits gesagt, ist die Herstellung der Fotomasken recht aufwendig. Man setzt dazu computergesteuerte Zeichen- und Entwurfshilfsmittel ein. Gewöhnlich wird eine Zeichnung im Maßstab 1000 : 1, d. h. für den $6 \cdot 6$ mm²-Chip eine 36 m²-Zeich-

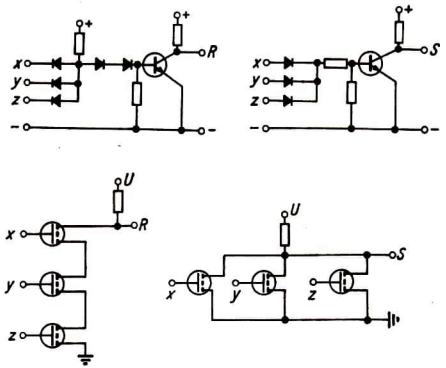


Abb. 3.21. Die NAND- und NOR-Schaltung in der DTL- und der MOS-Technik

nung, angefertigt. Diese wird dann optisch verkleinert und vervielfältigt (Fototafel, Bild 4 und 5). Die 1985 in der Großintegration VLSI (very large scale integration) erreichte Schaltungsdichte liegt bei 10^6 Funktionselementen pro Chip. Dabei werden Masken durch computergesteuertes Elektronenstrahlätzen erzeugt oder ersetzt. Auch andere oder weiterentwickelte Techniken des physikalischen Aufbaus und der Schalttechnik werden zum Einsatz gelangen (vgl. I^2L = integrated injection logic, CCD = ladungsgekoppelte Schaltungen u. a. m. in der Elektronikliteratur). Als erreichbare Kosten werden jetzt 1 bis 0.1 cent/gate angegeben, so daß in den nächsten Jahren mit weiteren drastischen Kostensenkungen der mikroelektronischen Geräte gerechnet werden kann.

Logische Schaltungen in der Halbleitertechnik werden in verschiedenen Versionen

hergestellt [65]. Nur zur Information seien einige der „Logiken" für die Bausteine aufgezählt:

DTL Dioden-Transistor-Logik,
TTL Transistor-Transistor-Logik,
RTL Widerstands(resistor)-Transistor-Logik,
ECL Emitter-gekoppelte(coupled)-Logik,
DCTL direkt-gekoppelte(coupled)-Transistor-Logik,
MOS und C-MOS.

Gewöhnlich werden nur wenige logische Funktionen realisiert, um die Variationsbreite der Fertigung klein zu halten. Bekanntlich (vgl. MfL Bd. 9, 2.4., und [29] Bd. 1, 2.3.) kann man mit der NAND- (negierte Konjunktion oder Sheffer-) oder der NOR-

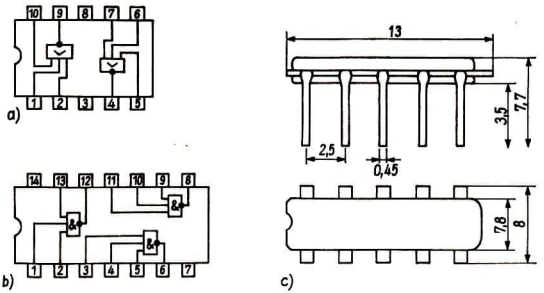


Abb. 3.22. Zwei kleine integrierte Bausteine, wie sie von der Elektronikindustrie der DDR (VEB Kombinat Mikroelektronik) angeboten werden. Der Mikroprozessor U 8080 D ist als Baustein nur unwesentlich größer
a) U 102 D in MOS mit zwei NOR-Schaltungen zu je drei Eingängen,
b) D 210 C in TTL mit drei NAND-Schaltungen zu je drei Eingängen,
c) Abmessungsübersicht — hier für einen 10beinigen Baustein

(negierte Disjunktion oder Peirce-) Funktion alle logischen Grundfunktionen darstellen. Abb. 3.21 zeigt die Schaltung für

$$R = \overline{xyz} \quad \text{und} \quad S = \overline{x + y + z}$$

in der DTL- und der MOS-Version (schaltalgebraische Schreibweise vgl. [29] Teil 1, 2.2.).

Solche einfachen Schaltungen werden in SSI (small scale integration) auch von der Industrie angeboten. In MOS-Technik wird von der Elektronikindustrie der DDR beispielsweise der Baustein U 102 D mit zwei NOR-Schaltungen mit je drei Eingängen und in TTL der Baustein D 210 C mit drei NAND-Schaltungen mit je drei Eingängen angeboten (Abb. 3.22).

3.3.5. Mikroprozessor und Mikrocomputer

Seit 1971 gibt es in der Halbleiterelektronik ein neues Bauelement, den Mikroprozessor (Abb. 3.23). Es ist ein hochintegriertes Bauelement. Von der Praxis ist seine Entwicklung angeregt worden oder sogar in Spezialgebieten zwingend erforderlich gewesen (Weltraumtechnik). Seinerseits hat er dann eine überaus große Anwendbarkeit und Einsatzbreite offenbart, so daß er wiederum in Wechselwirkung nun die Praxis stark beeinflußt. So ist seit seinem Erscheinen der Aufbau altbekannter elektronischer Meßgeräte, aber auch ganz profaner Geräte, wie z. B. eines Waschvollautomaten, für den Ingenieur kein schaltungstechnisches Problem mehr, sondern eine programmierungstechnische Aufgabe. Das wird nun auch wieder Einfluß auf

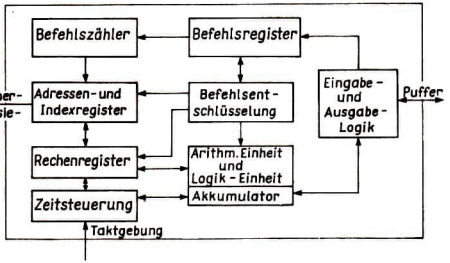


Abb. 3.23. Die wichtigsten Funktionseinheiten eines Mikroprozessors. Alle Strukturen des schematischen Bildes sind normalerweise in einem Chip integriert, z. B. auch beim U 808 D

viele Berufe und die dafür notwendige Ausbildung haben. Es sind wiederum mehr Kenntnisse in der Computertechnik und Computer-Programmierung nötig. Im Prinzip ist der Mikroprozessor das Herz- und Kernstück eines Digitalrechners, er ist ein elektronisches Bauelement mit Systemcharakter. Ein Mikroprozessor ist eine integrierte Standardschaltung, die die zentrale Steuer- und Recheneinheit in einem Mikrocomputer (Taschenrechner oder programmierbarer Tischrechner) oder einer anderen Automatisierungseinheit bildet. Die Einsatzmöglichkeiten sind sehr vielfältig. Sie reichen von der Steuerung für Werkzeugmaschinen, Belichtungsrechner in Kameras bis zur Haushaltelektronik, wie beispielsweise der Steuerung von Waschvollautomaten. Ein Mikroprozessor ist ein kleiner (16 mm² großer) Elektronenrechner mit Zahl- und Befehlswörtern von 8 bit = 1 Byte Länge. Der Befehlsvorrat ist sehr begrenzt. An arithmetischen Operationen können meist nur die Addition und die Subtraktion ausgeführt werden. Aber die Struktur ist so eingerichtet, daß die Wortverlängerungen, andere Zahldarstellungen (andere als nur ganze Zahlen) und auch höhere Operationen *mikroprogrammiert* werden können.

Als Folge des Einsatzes von Mikroprozessoren ergeben sich zunächst

— drastische Verkleinerung der Geräte,
— drastische Verringerung des Gerätegewichts.

Die Verkleinerung der Gerätegröße kann durch die Reihe vom Geräteeinschub in Tischkastengröße über einen Schaltungsmodul von Postkartenformat bis zu einem Mikroprozessor von Daumennagelgröße in einem würfelzuckergroßen Gehäuse veranschaulicht werden, und alle könnten von derselben Funktion sein.

Es gibt heute Mikroprozessoren, die mit 4 bit, 8 bit oder sogar 16 bit Wörtern (Zahlen, Befehlen) arbeiten. Es existieren weiter als abgerüstete Bestandteile sogenannte 2- oder 4-bit *slices* (Scheiben), die zu 8-, 16- und 32 bit-Mikroprozessoren kaskadiert, d. h. gekoppelt werden können.

Auch Kleinrechenanlagen, wie Taschen- und Tischrechner, werden nicht mehr wie früher die größeren Computer individuell von A bis Z entworfen und gefertigt, sondern man konzipiert sie unter Verwendung vorhandener Mikroprozessoren, für die ein

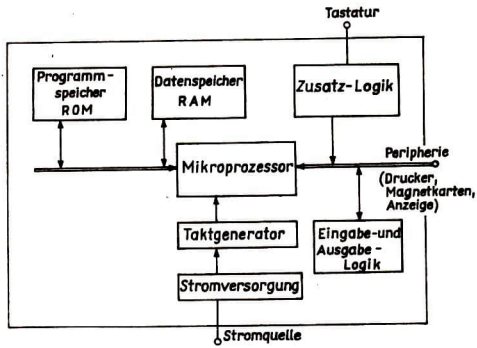


Abb. 3.24. Schematisches Blockbild eines Mikrocomputers. Das Zentrum bildet der Mikroprozessor. Seine Stellung ist so angeordnet, daß die Anschlüsse nach Abb. 3.23 etwa an der entsprechenden Stelle liegen

Anwendungsprogramm angefertigt wird, welches eben die Funktion eines solchen Kleinrechners *emuliert* (emulate = nacheifern (wörtlich), *Emulation* hat die Bedeutung der Nachbildung eines Computers auf einem anderen Computer, also einer speziellen Simulation; vgl. auch 3.2.). Die Entwicklung von derartigen Taschenrechnern oder Tischrechnern ist heute weitgehend als Programmieraufgabe, insbesondere auch für schnelle Resultatlieferung der mathematischen Standardfunktionen, in die Hände von Computerspezialisten und Informatikern übergegangen; Elektronikingenieure gehören selbstverständlich zum Entwicklungskollektiv, aber ihr Anteil bezieht sich mehr und mehr auf Impulstechnik und Stromversorgung als auf die logischen Strukturen, es sei denn, sie haben sich die entsprechenden Kenntnisse in Informationsverarbeitung aneignen können.

Ein Mikrocomputer ist eine programmierbare Kleinrechnerschaltung aus LSI-(large scale integrated)-Bausteinen, von denen einer ein Mikroprozessor ist (Abb.

3.24). Mikrocomputer können die Funktion größerer Rechenanlagen ausführen. Die Steuer- und Rechenfunktionen übernimmt dabei der Mikroprozessor. Außerdem enthält ein gesonderter Halbleiterbaustein das für die Arbeit des Mikroprozessors nötige Programm. Dieser Speicher ist ein nur lesbarer (read only memory ROM) Speicher. Er kann vom Nutzer des Mikrorechners nicht beeinflußt werden. Aber oft ist dieser Baustein auswechselbar.

Der Datenspeicher ist ein Baustein, dessen Elemente beliebig ansteuerbar sind und deren Inhalte natürlich gelesen und gespeichert werden können. Man bezeichnet diese Speicher als RAM (random access memory = Speicher mit beliebigem (zufälligen) Zugriff). Alle Teile des Mikrorechners werden durch Datenleitungen verbunden, die *Busse* genannt werden. In Abb. 3.24 sind die Busse durch doppelte Linien markiert.

Zur Entwicklung eines Mikrocomputers aus einem Mikroprozessor gehört die gezielte Herstellung des programmspeichernden ROM. Aber auch andere Anwender eines Mikroprozessors benötigen einen ROM für ihre speziellen Zwecke. Die Elektronikindustrie liefert dazu die allerdings auch aufwendigeren und teureren PROM (programmierbare ROM), bei denen mit einem Programmiergerät in der ROM-Schaltung gewisse Diodenstrecken gezielt durchgebrannt werden, und sogar die noch komplizierteren EPROM (löschbare PROM). Letztere können mit ultraviolettem Licht wieder völlig gelöscht und einige Male neu programmiert werden, aber nicht beliebig oft. Da ein Anwender nicht in unnötiger Weise PROM-Bausteine durch Fehlprogrammierung verderben will, gibt es für Großrechner geschriebene Emulatoren, welche Mikroprozessor-Programme simulierend abarbeiten und prüfen. Die Herstellung kleinerer Mikroprozessorprogramme (etwa 1000 Befehle) bis zur fehlerfreien Funktion benötigt durchschnittlich 10 Minuten pro Befehl und bei größeren Programmen mit 10000 Befehlen das Doppelte.

Als Maß für die Güte oder Leistungsfähigkeit eines Mikrocomputers verwendet man seinen *Durchsatz*, d. h., es wird die Abarbeitungszeit für einen genormten Mix von Befehlen (ein Benchmark-Programm = Einschätzungsprogramm) gemessen. Für Taschenrechnerentwürfe kann man die angebotene Leistung recht gut an den mathematischen Standardfunktionen abschätzen. Diese können mit einer Reihenentwicklung (sehr langsam, meist einige Sekunden) durch ein Ersatzpolynom (etwa durch Čebyšev-Approximation erhalten und dann nur noch Programmierung des Hornerschema, schon schneller, aber noch erkennbar langsamer als die arithmetischen Grundfunktionen) oder durch eine modifizierte Division (vgl. 3.2.) gewonnen werden. Die letzte Methode berechnet die Funktionswerte ziffernweise, arbeitet genau so schnell wie die Grundfunktionen und garantiert bei rationalen bzw. ganzzahligen Werten innerhalb des Zahlenbereiches zifferngenaue und nicht nur approximative Resultate.

3.3.6. Entwicklungsgeschichte und Prognose

Seit 1960 kennt man die bipolare Mikroelektronik aus der Fachliteratur und auch in den ersten Einsätzen. Als zuverlässiges Trägersubstrat hat sich das Element Silizium erwiesen, das heute weitgehend und beinahe ausschließlich benutzt wird. Die

relativ komplizierte Bipolartechnik wurde über MOS (im Text beschrieben) zur CCD-Technik (charge coupled device, im Text nicht beschrieben) und I^2L-Technik (im Text nicht beschrieben, etwa ab 1974), d. h. mit relativ einfachem technischen Aufbau und damit einfacheren technologischen Prozessen in der Fertigung weiterentwickelt. Dadurch wurde eine Verfeinerung der Strukturen und dichtere Packung der Elemente möglich. Die LSI (large scale integration = Großintegration) erreicht gegenwärtig 60000 Elemente pro Chip, wobei die Feinstruktur auf 10^{-6} m für die Leiterbahnen gedrückt wurde. Die MOS-Technik wurde Ende der sechziger Jahre vervollkommnet, und etwa um diese Zeit oder danach kann man von LSI sprechen. Fachleute (vgl. [65] und [69]) rechnen mit Beibehaltung dieser Tendenz der dichteren Packung und Verkleinerung, wobei bis 1985 eine Sättigung von 10^6 (1 Million!) Bauelementen pro Chip erreicht wurde. Diese Großintegration (VLSI very large scale integration) läßt noch eine interessante Bemerkung zu: Mit 10^6 Funktionselementen pro Chip und einer Produktion von jährlich Millionen von VLSI-Chips erreicht man Produktionszahlen von Milliarden von Schaltfunktionen! Das hat ohne Zweifel größte volkswirtschaftliche Einsatzbedeutung. Die Kostenentwicklung im internationalen Vergleich weist eine Reduktion auf 1/1000 des Preises für eine Schaltfunktion innerhalb von 10 Jahren aus.

Literatur

[1] Adam, J., Einführung in die medizinische Statistik, VEB Verlag Volk und Gesundheit, Berlin 1963.

[2] Albrecht, P., Die numerische Behandlung gewöhnlicher Differentialgleichungen, Akademie-Verlag, Berlin 1979.

[3] Ahlberg, J. H.; Nilson, E. N.; Walsh, J. L., The Theory of Splines and Their Application, Academic Press, New York 1967.

[4] Brehmer, S.; Apelt, H., Analysis I und II, Studienbücherei, Mathematik für Lehrer, Band 4 und 5, 3. Aufl., VEB Deutscher Verlag der Wissenschaften, Berlin 1982 bzw. 1980.

[5] Blomeyer-Bartenstein, H.-P., Mikroprozessoren und Mikrocomputer, Siemens-Bauelemente, Firmenschrift, München 1976.

[6] Blomeyer-Bartenstein, H.-P., Halbleiterbauelemente für die Elektronik (Ein Kompendium), Siemens-Bauelemente Vertrieb, Firmenschrift, München 1976.

[7] Bohle, G.; Hofmeister, E., Ein Neues Mikrocomputer-Konzept, Elektronik **25** (1976) 11, 83–87.

[8] Bromm, G., Programmierbare Taschenrechner in Schule und Ausbildung, Verlag Vieweg, Braunschweig 1979.

[9] Bronstein, I. N.; Semendjajew, K. A., Taschenbuch der Mathematik, 20. Aufl., BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1981.

[10] Clauss, G.; Ebner, H., Grundlagen der Statistik für Psychologen, Pädagogen und Soziologen, Volk und Wissen, Volkseigener Verlag, Berlin 1978.

[11] Collatz, L.; Albrecht, J., Aufgaben aus der Angewandten Mathematik I, Akademie-Verlag, Berlin 1972.

[12] Engel, A., Elementarmathematik vom algorithmischen Standpunkt, Ernst Klett, Stuttgart 1977.

[13] Führig, A.; Kaiser, H., Numerische Mathematik und Rechentechnik. In: Böhm, J., u. a.: Aufgabensammlung II, Studienbücherei, Mathematik für Lehrer, Band 15, VEB Deutscher Verlag der Wissenschaften, Berlin 1982, Kap. VI.

[14] Gilde, W.; Altrichter, S., Mehr Spaß mit dem Taschenrechner, VEB Fachbuchverlag, Leipzig 1978.

[15] Gloistehn, A., Programmierung von Taschenrechnern (2. Lehr- und Übungsbuch für den TI-57), Verlag Vieweg, Braunschweig 1978.

[16] Gram, Ch., Selected Numerical Methods, Regnecentralen, Copenhagen 1962.

[17] RFT electronic, Halbleiter-Bauelemente, Katalog 1975, VEB Halbleiterwerk, Frankfurt (Oder) 1975/76.

[18] Henrici, P., Elements of Numerical Analysis, J. Wiley and Sons, Inc., New York 1964.

[19] Hewlett-Packard, Bedienungshandbuch HP-25, Palo Alto, Calif., 1975.

[20] Hewlett-Packard, Programming Handbook HP-65, Palo Alto, Calif., 1975.

[21] Hilbert, A., Matrizenrechnung, Volk und Wissen Verlag, Berlin 1977.

[22] Höhne, M., Der Mikroprozessor U 808 D, radio fernsehen elektronik **26** (1977) 5, 145–150; 6, 187–188, 197–198.

[23] Householder, A. S., Principles of Numerical Analysis, McGraw-Hill, New York 1953.

[24] Isaacson, E.; Keller, H. B., Analysis of Numerical Methods, J. Wiley and Sons, New York (deutsche Übersetzung: Analyse numerischer Verfahren, Edition, Leipzig 1972).

[25] KAHAN, W.; PARLETT, B. N., Können Sie sich auf Ihren Rechner verlassen?, Jahrbuch Überblicke Mathematik 1978, Hrsg. S. VON FUCHSSTEINER, Wissenschaftsverlag, Mannheim/Wien/Zürich 1978.
[26] KAISER, H., Numerische Mathematik und Rechentechnik I, Studienbücherei, Mathematik für Lehrer, Band 9. VEB Deutscher Verlag der Wissenschaften, Berlin 1977.
[27] KAISER, H., Numerische Mathematik und Rechentechnik II, Studienbücherei, Mathematik für Lehrer, Band 10, VEB Deutscher Verlag der Wissenschaften, Berlin 1980.
[28] KHAMBATA, A. J., Einführung in die Mikroelektronik, VEB Verlag Technik, Berlin 1966 (Übersetzung aus dem Englischen).
[29] KERNER, I. O., Numerische Mathematik und Rechentechnik, Teile 1 und 2/1—2, BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1970, 1973.
[30] KERNER, I. O., Diskrete Arithmetik, Rostocker Math. Koll. **10** (1978), 71—81.
[31] KERNER, I. O., Bester Kegelschnitt durch n Punkte, ZAMM **59** (1979), 396—397.
[32] KIESEWETTER, H.; MAESS, G., Elementare Methoden der numerischen Mathematik, Akademie-Verlag, Berlin 1974.
[33] KLATTE, R.; ULLRICH, CHR., Zur mathematischen Struktur von Rechnerarithmetiken, Math.- und naturw. Unterricht **30** (1977) 1, 1—8.
[34] Kleine Enzyklopädie Mathematik, VEB Bibliographisches Institut, Leipzig 1965.
[35] KNOPP, K., Theorie und Anwendung unendlicher Reihen, Springer, Berlin 1947 (erste Auflage 1931).
[36] KREUL, H., Was kann mein elektronischer Taschenrechner? VEB Fachbuchverlag, Leipzig 1979.
[37] KRONROD, A. S., Uzly i wesa kwadraturnych formul, Moskwa 1964.
[38] KUHNERT, F., Pseudoinverse Matrizen und die Methode der Regularisierung, BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1976.
[39] LEVIN, M.; GIRSHOVICH, J., Optimale Quadrature Formulas, BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1979.
[40] MAIBAUM, G., Wahrscheinlichkeitstheorie und mathematische Statistik, Studienbücherei, Mathematik für Lehrer, Band 11, 2. Aufl., VEB Deutscher Verlag der Wissenschaften, Berlin 1980.
[41] MEGGITT, J. E., Pseudo Division and Pseudo Multiplication Processes, IBM Journal (1961) August, 210—226.
[42] MEILING, W., Mikroprozessor-Mikrorechner. Funktion und Anwendung, Akademie-Verlag, Berlin 1979.
[43] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrpläne Klasse 1, Volk und Wissen, Volkseigener Verlag, Berlin 1975.
[44] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrpläne Deutsch und Mathematik Klasse 2, Volk und Wissen, Volkseigener Verlag, Berlin 1968.
[45] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrplan Mathematik Klasse 3, Volk und Wissen, Volkseigener Verlag, Berlin 1970.
[46] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrplan Mathematik Klasse 4, Volk und Wissen, Volkseigener Verlag, Berlin 1970.
[47] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrplan Mathematik Klassen 5 bis 10, Volk und Wissen, Volkseigener Verlag, Berlin 1977.
[48] Ministerrat der Deutschen Demokratischen Republik, Ministerium für Volksbildung, Lehrplan Mathematik Abiturstufe, Volk und Wissen, Volkseigener Verlag, Berlin 1979.
[49] NATANSON, I. P., Konstruktive Funktionentheorie, Akademie-Verlag, Berlin 1955 (Übersetzung aus dem Russischen).
[50] PRAGER, W.; ÖTTLI, W., Computability of approximate solution of linear equations with given error bounds for coefficients and right-hand sides, Num. Math. **6** (1964), 405 bis 409.
[51] RALSTON, A.; WILF, H. S., Mathematical Methods for Digital Computers, John Wiley and Sons, Inc., 1960.

[52] Bedienungsanleitung PKR K 1001/2/3, VEB Kombinat Robotron Dresden, Karl-Marx-Stadt 1977.
[53] Bedienungsanleitung MR 610, VEB Röhrenwerk Mühlhausen 1979.
[54] Schauer, H.; Barta, G., Methoden der Programmerstellung für Tisch- und Taschenrechner, Springer-Verlag, Wien/New York 1979.
[55] Schäf, W.; u. a., Programmieren mit dem Taschenrechner TI-57, Oldenbourg-Verlag, München 1978.
[56] Schumny, H., Taschenrechner Handbuch, Verlag Vieweg, Braunschweig 1977/BGB B. G. Teubner Verlagsgesellschaft, Leipzig 1979.
[57] Schwetlick, H., Numerische Lösungen nichtlinearer Gleichungen, VEB Deutscher Verlag der Wissenschaften, Berlin/Oldenbourg-Verlag, München—Wien 1979.
[58] Smith, J. M., Scientific Analysis on the Pocket Calculator, J. Wiley and Sons, New York 1975.
[59] Stroud, D. H.; Don Secrest, Gaussian Quadrature Formulas, Prentice Hall, Englewood Cliffs 1966.
[60] Stetter, H. J., Numerik für Informatiker (Computergerechte numerische Verfahren — Eine Einführung), Oldenburg-Verlag, München 1976.
[61] Texas Instuments, TI-51-III Bedienungsanleitung, Austin, Texas, Firmenschrift, 1978.
[62] Texas-Instruments, Bedienungsanleitung für den Taschenrechner SR-11, Austin, Texas, Firmenschrift, 1973.
[63] Texas-Instruments, Making Tracks into Programming, Austin, Texas, Firmenschrift, 1977.
[64] Texas-Instruments, Der Weg zum Programmieren, Austin, Texas, Firmenschrift, 1977.
[65] Völz, H., Elektronik für Naturwissenschaftler, Akademie-Verlag, Berlin 1974 (3. Halbleiterelemente, 272—488; 3.5. Technologie 404—427; 3.6. Mikroelektronik 427—488).
[66] Weber, E., Grundriß der biologischen Statistik für Naturwissenschaftler, Landwirte und Mediziner, 8. Aufl., VEB Gustav Fischer Verlag, Jena 1980.
[67] Weber, S.; Altmann, L.; Scrupski, S. E., Microprocessors-Special Issue, Electronics 49 (1976) 8, 74—174.
[68] Wensley, J. H., A Class of Non-Analytical Iterative Processes, Computer Journal 1 (1959) 4, 163.
[69] Wilhelmy, H. J., Ein schöpferisches Vierteljahrhundert (Rückblick auf 25 Jahre Elektronik-Entwicklung), Elektronik 25 (1976) 10, 36—45.
[70] Wilkinson, J. H., Rounding Errors in Algebraic Processes, Her Majesty's Stationery Office, London 1963.
[71] Zielinski, R., Erzeugung von Zufallszahlen. Programmierung und Test auf Digitalrechnern, VEB Fachbuchverlag, Leipzig 1978.
[72] Zurmühl, R., Praktische Mathematik für Ingenieure und Physiker, Springer-Verlag, Berlin 1953.
[73] VEB Kombinat Mikroelektronik, Bedienungsanleitung SR 1, Mühlhausen 1984.
[74] Fanghänel, G., Der Schulrechner SR 1 — Ein Taschenrechner für den Unterricht in der Abiturstufe, Mathematik in der Schule 22 (1984), 7/8, 515—527.
[75] Walsch, W.; Flade, L., Taschenrechner in der Schule, Wiss. Beiträge der Martin-Luther-Universität Halle—Wittenberg 1984, 10.
[76] Kleinstrechner-TIPS, Herausg. von H. Kreul, W. Leupold, Th. Horn, Heft 1 und 2, VEB Fachbuchverlag, Leipzig 1984.
[77] Kerner, I. O.; Winkler, U., Taschenrechner in der Schule, Lehrmaterial zur Aus- und Weiterbildung von Mathematiklehrern an der PH Dresden. Dresdner Reihe zur Lehre 6/85.
[78] Vickers, St., BASIC Programming/ZX Spectrum Sinclair, Sinclair Research Ltd., Cambridge 1982.
[79] VEB ROBOTRON, Bedienungshandbuch für den Heimcomputer Z 9001, Dresden 1984.
[80] Kerner, I. O.; Wagenknecht, Ch.: Einführung in das strukturierte Programmieren mit BASIC, VE Verlag Volk und Wissen, Berlin 1986.

Namen- und Sachverzeichnis