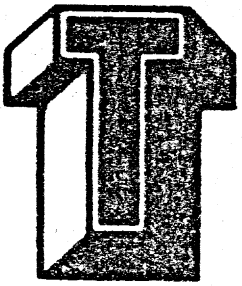
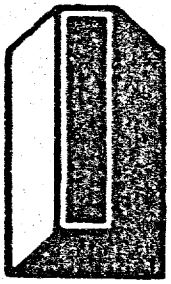
Kleinstrechner



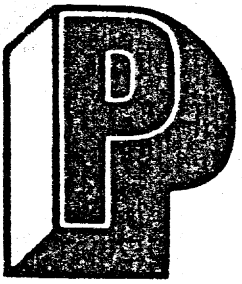
Tendenzen
und Theorien

Informatik
in der Schule



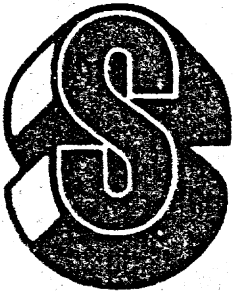
Informationen
und Ideen

Bildschirmspiele



Programme
und Projekte

Grafikbefehle



Spaß
und Spiel

Kleinstrechner-TIPS

Heft 8

Mit 26 Bildern

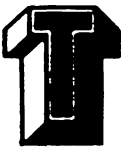
Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Doz. Dr.-Ing. Wilhelm Leupold
Doz. Dr. sc. techn. Thomas Horn

VEB Fachbuchverlag Leipzig

Inhalt

ISBN 3-343-00385-9

Zum Inhalt



Das außerordentlich starke Interesse, das die bisher erschienenen Hefte der Schriftenreihe Kleinstrechner-TIPS gefunden haben, ermutigt die Herausgeber und den Verlag, die ursprüngliche Konzeption der Reihe beizubehalten und fortzuführen. Für nicht als Informatiker ausgebildete Laien, für Schüler und Lehrer, für Arbeitsgemeinschaften und Computerclubs sollen Anregungen gegeben werden, wie mit den modernen Hilfsmitteln der Informationsverarbeitungstechnik allgemein interessierende Probleme gelöst werden können. Wir wollen mit unseren Beiträgen vor allem das Gefühl vermitteln, daß der Umgang mit Computern durchaus auch Freude und Entspannung bereiten kann.

In diesem Heft gibt KERNER einen ausführlichen Überblick über die Aspekte einer künftigen Informatik-Ausbildung an den allgemeinbildenden Schulen.

Als Vorstufe für eine Informatik-Ausbildung kann der Einsatz des Taschenrechners in der Schule betrachtet werden. WALSCH und FLADE berichten in ihrem Artikel darüber, welche Ergebnisse dabei erzielt werden konnten.

Hinweise darauf, wie man bereits beim Programmieren die Rechenzeit eines Programms günstig beeinflussen kann, will der Beitrag von KREUL geben.

FISCHER bringt ein Programm zur Lösung quadratischer Gleichungen und geht dabei besonders auf die Frage der erzielbaren Genauigkeit der Berechnungen ein.

Von KERN und M. KREUL wird gezeigt, wie man mit Hilfe eines Computers Extremwertaufgaben lösen kann, ohne dabei auf Kenntnisse aus der Differentialrechnung zurückgreifen zu müssen.

Die exakte Lösung algebraischer Gleichungen steht im Mittelpunkt des darauffolgenden Artikels von OELSCHLÄGEL und GROCHOW. Für Freunde der Computerspiele bringt GUMPERT Hinweise darauf, wie Zickzack-Bewegungen von Objekten auf dem Bildschirm erzeugt werden können.

KOSSOW und PREUSS stellen ein Programm vor, mit dessen Hilfe die Verfügbarkeit technischer Systeme am Bürocomputer A 5120 ermittelt werden kann. Aus diesem Artikel wird ersichtlich, daß auch kompliziertere technische Aufgabenstellungen bereits mit »einfachen« Computern gelöst werden können.

Schließlich folgt noch ein Beitrag von WENDT, in dem gezeigt wird, wie die relativ langsam arbeitenden Grafikbefehle in BASIC durch schnell ablaufende Maschinenprogramm-Routinen ersetzt werden können.

Hans Kreul

Aspekte der Informatik-Ausbildung im Volksbildungswesen der DDR

1. Beitrag der Informatik zur Allgemeinbildung

Weit verbreitet ist die Argumentation, daß Informatik bzw. Computerkenntnisse nunmehr zur Allgemeinbildung gehören, da jeder Bürger sowohl im Beruf als auch im Freizeitbereich oder Alltag zunehmend mit der informationsverarbeitenden Technik in Kontakt kommt. Obwohl die Konsequenz korrekt, ist doch der Ausgangspunkt anfechtbar. Es gibt zahlreiche Beispiele für Sachverhalte, mit denen jeder Bürger in Berührung kommt und die kein Bestandteil des Schulunterrichts sind, der ja die *Basis für die Allgemeinbildung* aufbaut. Beispielsweise hat das Verkehrs- und Transportwesen unser Leben durch die Entwicklung der Eisenbahn und des Autos tiefgehend beeinflußt. Aber es gibt in keinem Land ein Schulfach »Verkehrswesen« oder »Eisenbahn und Auto«. Dagegen sind die zentralen Wirkungsprinzipien dieser Verkehrsmittel Gegenstand des Schulunterrichts und Bestandteil der Ausbildung: Dampfmaschine, Elektromotor und Ottomotor.

Was also ist das zentrale Wirkungsprinzip der Computer?

Nach dem Analogon
Lokomotive : Dampfmaschine
oder
Auto : Ottomotor
kann man leicht zu der Antwort »der mikroelektronische Zentralprozessor« verleitet werden.

Eine eingehendere Analyse jedoch kommt zu einem anderen Resultat:

- Der *mikroelektronische* Zentralprozessor kann nicht das Wesen des Computers bedeuten, denn Computer gab es schon auf der Basis anderer Techniken, wie Relais, Elektronenröhren und Transistoren.
- Der *Aufbau* und das *Zusammenspiel* der Baugruppen in der *Arbeitsweise* eines Zentralprozessors können ebenfalls nicht zum Wirkungsprinzip gehören, denn auch die Struktur hat sich wesentlich in der Entwicklung der inzwischen bereits fünf Generationen von Computern geändert.

Nach diesen beiden negativen Aussagen, die aber notwendig für das Finden des richtigen Weges sind, muß man diesen betreten.

Die Menschheit hat sich im Verlauf der Geschichte bisher *drei Typen von Maschinen zur Unterstützung der Arbeit und Verbesserung der Lebensbedingungen* geschaffen:

- Die *energie- bzw. kraftwandelnden Maschinen*, angefangen vom Hebel und der schiefen Ebene bis zu modernsten Motoren und Nuklearreaktoren, bei denen die *Physik* die wissenschaftliche Basis bildet.

- Die *stoff- bzw. materialwandelnden Maschinen*, angefangen von den Geräten zur Beherrschung der urtümlichen Koch-, Brat- und Gärungsprozesse bis hin zu den Großanlagen der Petrolchemie und Pharmazie, bei denen die *Chemie* die wissenschaftliche Basis bildet.
- Die *informationwandelnden Maschinen*, deren Anfänge bei den ersten primitiven Anlagen zur Nachrichtenübermittlung und Signalgebung liegen und die sich über vielfältige Geräte und Methoden der Steuerungs- und Regeltechnik zu den Computern entwickelten.

Das Wesen der Computer besteht also darin, daß sie die *Information* als eine der möglichen Erscheinungsformen der Materie *verarbeiten*. Da die Verarbeitungsvorschriften selbst Informationen darstellen, kann man ihnen diese variabel und vielfältig aufprägen. Diese Maschinen sind *programmierbar*. Das zentrale Wirkungsprinzip der Computer ist die *Programmierbarkeit*. Dieses Wissen und grundlegende *Kenntnisse im Programmieren* stellen einen *Beitrag der Informatik zur Allgemeinbildung* dar. Kenntnisse zur Programmierung zu erwerben wird sogar als *zweite Alphabetisierung* bezeichnet und damit den entscheidenden Grundprozessen der Bildung, dem Schreiben und dem Lesen, an die Seite gestellt (A. P. Ershov, Novosibirsk). Hier ist die Stelle erreicht, an der der sehr vordergründigen Schlußkette: »Informatik gleich Mikroelektronik – Taschenrechner haben Mikroelektronik – Einführung der Taschenrechner in der Schule gleich Einführung der Informatik in der Schule« der Boden entzogen wird.

Zwei weitere Aspekte liefert die Informatik noch als Beitrag zur Allgemeinbildung. *Es muß jedem Bürger klar sein, daß auch diese Technik Einsatz- und Wirkungsgrenzen hat.* Falsche Meinungen über unbegrenzte Möglichkeiten des Lösens von Problemen, wenn man nur genügend viel und genügend schnelle Computer hat, *dürfen nicht entstehen und dürfen nicht gefördert werden.* So wie es z. B. in der Mathematik unter Beachtung der eingesetzten Hilfsmittel unlösbare Probleme gibt [Dreiteilung des Winkels oder Rektifizierung des Kreisbogens sind auf konstruktive Weise mit Zirkel und Lineal unmöglich, formelmäßiges Auflösen allgemeiner algebraischer Gleichungen vom fünften Grad an ist nicht möglich (Galois und Abel, 1830), Beweis der Widerspruchsfreiheit einer Theorie aus ihr selbst ist nicht möglich], haben die theoretischen Grundlagen der Informatik erwiesen, daß es Fragestellungen und Probleme der Informatik gibt, welche algorithmisch – und das heißt mit Computerprogrammen – *nicht lösbar* und in zugespitzter Fragestellung *nicht einmal entscheidbar* sind. Letzteres bedeutet: Es ist mit mathematischer Strenge und Schlüssigkeit erwiesen, daß es mit dem Hilfsmittel Algorithmus keine allgemeine Entscheidungsmöglichkeit gibt, ob die Antwort auf gewisse Fragen »ja« oder »nein« lauten wird. Das sind Begrenzungsaussagen für den Einsatz von Computern, die zur Allgemeinbildung gehören. Wenn auch die Theorie der Informatik nicht Gegenstand des Schulunterrichts sein kann, soll doch jedermann wissen, daß es eine solche Theorie gibt, die Gegenstand eines weiterführenden Studiums ist und mit der sich Fachleute befassen. Die *Allgemeinbildung darf nicht in den Anfangsgründen der Programmierung stecken bleiben* und den Eindruck hinterlassen, daß der junge Mensch nach Fertigstellung eines ersten Programms nun die Informatik gemeistert hat. So wie man Wege gefunden hat, das

Wissen über Grundergebnisse der Kernphysik u.a. in die Allgemeinbildung zu integrieren, ohne die theoretischen Grundlagen (z.B. SCHRÖDINGER, 1928) vermitteln zu können, wird dies auch für die Informatik gelingen. Man wird aber wohl mehr Theoriegehalt in die Allgemeinbildung einzubringen haben als das im gewählten Parallelfall Kernphysik der Fall zu sein braucht, denn dort akzeptiert jedermann den wissenschaftlichen Hintergrund Physik, während beim Computer vordergründig »Tastenbedienung« als das Wesentliche erscheint. Die Auswahl der Theorieanteile wird man wählen aus der Automatentheorie (TURING, 1936), Algorithmen und Theorie der rekursiven Funktionen (GÖDEL, 1930), formale Sprachen und Automaten (CHOMSKY, GINSBURG, 1955) und der Komplexitätstheorie.

Gerade die Komplexitätstheorie liefert den zweiten der beiden angekündigten Aspekte zur Einsatzbegrenzung der Computer. Es ist jedermann einsichtig, daß die Lösung eines »großen« Problems »mehr« Aufwand erfordert als die Lösung eines »kleinen«, daß »schwierigere« Probleme mehr Aufwand brauchen als »einfachere«. Es gibt Probleme, deren Lösungsaufwand (Computerressourcen Speicher oder Zeit) linear, quadratisch oder allgemein polynomial mit der Problemgröße ansteigt. Das sind i.allg. Probleme, deren Lösungsalgorithmus klar vorbestimmt »deterministisch» abläuft. Diese Probleme trugen zu der Meinung bei: Computer können nur das, was der Mensch ihnen eingibt. Im Gegensatz dazu gibt es Lösungsverfahren, die nichtdeterministisch ablaufen, d.h., bei ihnen spielen stochastische Elemente der Datenauswahl und auch der Wahl der Operationsabfolge eine Rolle. Solche Algorithmen können nach Evolutionsgrundlage von »Versuch und Irrtum« sehr wohl Resultate liefern, an die kein Programmierer denken konnte. Nun gibt es aber auch Probleme, deren Lösungsverfahren in einem systematischen Durchprobieren aller Varianten besteht. Der Aufwand steigt dann exponentiell oder sogar noch stärker mit der Problemgröße. Man erreicht sehr schnell Anforderungen an die Ressourcen, die alle z.Z. möglichen und alle zukünftig denkbaren Quellen übersteigen. Derartige Probleme entstammen leider oft dem ökonomischen Bereich – wir werden sie mit Computerhilfe nie vollständig lösen können – und auch dem militärischen Bereich – mit Computerhilfe kann der Sieger eines Konflikts nicht sicher vorherbestimmt werden; ein den Frieden mit bewahrender Faktor und ein Appell an die Vernunft der Menschheit. *Dieses Wissen ist ein bedeutsamer Beitrag der Informatik zur Allgemeinbildung.*

2. Verhältnis Volksbildung/ Berufsbildung

Die Nutzung der Computer im Beruf ist anwendungsorientiert.

Fertige Programme oder Programmpakete werden genutzt. Der Computer am Arbeitsplatz ist ein Werkzeug, und im übertragenen Sinne ist das Programmpaket ein Werkzeug. Das Bedienen und Verwenden dieser Werkzeuge ist Gegenstand der Berufsbildung, und es ist deutlich abzuheben von der Allgemeinbildung, obwohl einige Bestandteile der Vorbereitungen auf die Berufsbildung in den Bildungsumfang der polytechnischen Oberschule der DDR gehören. Beispielsweise wird das Nutzen einfacherer Programmpakete zur Textverarbeitung, zur grafischen Datenverarbeitung (Einblick in CAD), zu Datenbasen und zu Expert- und Auskunftssystemen in der Schule vorbereitet, denn einen *Ausblick auf den*

Beruf zu geben, ist Aufgabe der Schule. Aber alleiniger Inhalt des Informatikunterrichts kann dieser pragmatische Teil nicht sein, wie unter 1. gezeigt wurde. In der DDR wurden – beginnend mit dem Ausbildungsjahr 1986/87 – zwei Drittel aller Ausbildungsberufe mit einem Lehrgang Automatisierungstechnik ergänzt, der direkte Computerberührung für alle Lehrlinge verlangt. Die technischen Voraussetzungen dafür wurden geschaffen. Das Zeitvolumen ist gestaffelt von etwa 50 Ausbildungsstunden bis 120. Darüber liegen natürlich noch diejenigen Berufe, deren Inhalt das Bedienen von Computern und das Herstellen von Softwarewerkzeugen ist. Da trotz sorgfältiger Beratungen der konzeptionellen Grundlagen noch Erfahrungen gesammelt werden müssen, sind alle genannten Zahlen als vorläufig zu charakterisieren.
(Zur Hoch- und Fachschulausbildung vergleiche man andere Quellen.)

3. Grundpositionen zur Einführung eines Informatikunterrichts

Ausgehend von der unter 1. dargelegten konzeptionellen Basis, die durch Beratungen von Arbeitsgruppen an der Akademie der Pädagogischen Wissenschaften der DDR formuliert wurde, und auf der Grundlage von Beschlüssen des ZK der SED und des Ministerrates der DDR vom November 1985 und April 1986 wurde eine Vorgehensweise festgelegt, welche für die DDR im Bereich der Volksbildung die zwei Jahre zuvor im Rat für Gegenseitige Wirtschaftshilfe gemeinsam beratene und empfohlene Förderung und Entwicklung der Mikroelektronik in allen Bereichen der Volkswirtschaft realisiert.
Eine Grundposition war von vornherein und wird strikt eingehalten: *Keine Informatikausbildung ohne zugleich Computervertrautheit zu vermitteln.* Deshalb wird der Informatikunterricht stufenweise nach Bereitstellung der gerätetechnischen Basis eingeführt. (An Hoch- und Fachschulen gibt es in der DDR etwa seit 1960 Unterricht in Rechentechnik, später Datenverarbeitung, später Informationsverarbeitung, jetzt Informatik.) Bei dieser stufenweisen Einführung werden Lehrprogramme erprobt, und es entstehen parallel dazu Lehrbücher und andere Lehrmaterialien, in denen dann bereits ein Grundstock von Erfahrungen enthalten ist. Ferner werden immer die Lehrer für die nächste umfangreichere Etappe geschult.
Bereits im Februar 1985 wurde damit begonnen, die pädagogischen Hochschulen der DDR, an denen Mathematiklehrer ausgebildet werden, mit Kleincomputerkabinetten auszurüsten. Damit wurde deren Informatikausbildung, die bereits seit etwa 15 Jahren – anfangs unter dem Namen Rechentechnik – durchgeführt wurde, auf eine qualitativ höhere Ebene gestellt. Wir begannen also mit der Lehrerausbildung, da wir recht sorgsam die Probleme, besonders in kapitalistischen Ländern, betrachteten, die entstehen, wenn man als erstes die Schulen voller Technik steckt. Beginnend mit dem Schuljahr 1985/86, setzte an den 10 Spezialschulen und einigen Spezialklassen für math.-technischen Unterricht ein obligatorischer Kurs Informatik für Klasse 9 ein. Dieser Kurs wird in Klasse 10 fortgesetzt, und in den Klassen 11 und 12 wird Gelegenheit zur Vertiefung der Kenntnisse auf fakultativer Basis gegeben. Bei den Schülern dieser Einrichtungen handelt es sich um Begabungen und Talente. Die Streuung der Spezialschulen bzw. Klassen ist gering. Es kommt eine Einrichtung auf etwa 1,5 Mio Einwohner. Aller-

dings haben sich darüber hinaus gerade für Informatik und Computerwissen auf territorialer Basis mit Förderung der Jugendverbände »Junge Pioniere«, FDJ und lokaler Industriebetriebe (Patenbetriebe und Schulen) und Hochschulen Zirkel und Arbeitsgruppen gebildet. Beim Pionierpalast Dresden z.B. existiert ein Schülerrechenzentrum seit 1982 für junge Talente ab Klasse 7. Es wird unterstützt vom Kombinat ROBOTRON, und es unterrichten Lehrkräfte von Dresdener Hochschulen.

Der nächste große Schritt wurde in der Berufsbildung vorgesehen. Wie schon gesagt, kommt ab 1986/87 der größte Teil aller Lehrlinge mit Computern in Berührung. Gleichzeitig setzte ein für unsere Verhältnisse als Großerprobung zu bezeichnendes Experiment ein. An jeweils 20 erweiterten Oberschulen (EOS, Abiturstufe, Klassen 11 und 12) und polytechnischen Oberschulen (POS, Klassen 9 und 10) wurden die in mehreren Beratungsstufen konzipierten Lehrpläne für die fakultativen Kurse »Informatik« und »Mikroelektronik und Automatisierungstechnik« erprobt. Das sind 10% der existierenden etwa 200 EOS (1 EOS pro 90000 Einwohner) und 0,3% der etwa 6000 POS (1 POS pro 3000 Einwohner). Die Lehrer für diese Kurse wurden geschult, und die ausgewählten Schulen haben entweder ein eigenes Computerkabinett erhalten, oder aber für viele wurde der Zugriff zu einem solchen gesichert. *Nach der Erprobung* findet die *etappenweise Verallgemeinerung* statt, mit ihr wird ab 1988 gerechnet. Im polytechnischen Unterricht ESP gibt es ähnlich zur Berufsausbildung einen Abschnitt »Automatisierung« mit Vermittlung von Computergrundkenntnissen für die Klasse 9. Dazu wurden die polytechnischen Zentren in den Betrieben (etwa 2500) mit Computertechnik ausgerüstet. Im Schuljahr 1987/88 wurde an 10 EOS in Klasse 11 ein 60-Stunden-Lehrgang für den obligatorischen Informatikunterricht erprobt. Die Erprobung wird in größerem Umfang, etwa 80 bis 100 EOS, im Schuljahr 1988/89 fortgesetzt werden, und dann werden alle EOS einbezogen. Es entstehen ein Schulbuch »Informatik« für diesen Unterricht, natürlich ein Lehrplan und Lehrermaterialien sowie entsprechende Software. Auch die Frage des obligatorischen verbindlichen Informatikunterrichts in der DDR und die der niedrigstmöglichen Klassenstufe wird in diesem Zeitraum entschieden werden. Momentan denkt man an Klasse 9. Es gibt jedoch schon Experimente mit Klasse 7 (Dresden), Klasse 5 (Potsdam) und im Extrem im Vorschulalter (Dresden). Es ist überhaupt bemerkenswert, daß die Initiative der Bürger in dieser Frage den zentralen Maßnahmen öfter voraneilt.

4. Die Lehreraus- und -weiterbildung/ Methodik und Didaktik eines Informatikunterrichts

Wie bereits gesagt wurde, bilden wir in der DDR seit Anfang der 70er Jahre alle Mathematiklehrer auch in Rechentechnik und seit 1982 in Informatik aus. Das könnte bedeuten, daß wir 20 bis 25000 Lehrer mit Grundkenntnissen der Befähigung zum Erteilen eines Informatikunterrichts zur Verfügung haben. Die Aufrechnung ist aber nicht real. Zunächst waren die Lehrerstudenten während der Ausbildung nicht genügend motiviert, denn sie wußten sehr genau, daß ein Unterricht in Informatik und selbst der Einsatz der Taschenrechner auf Jahre hinaus nicht in Sicht war. Dann war die Technik an den pädagogischen Hochschulen für ein solches Schulfach

nicht überzeugend. Es bestanden auch ideologische Barrieren im Lehrkörper der Mathematik und der Naturwissenschaften, die bis auf die Lernhaltung der Studenten wirkten. Schließlich ging in jahrelanger Schularbeit ohne Informatik viel erworbene Kenntnis einfach wieder verloren, ganz abgesehen davon, daß die heute als nötig erkannten Kenntnisse und Fertigkeiten in einem längeren Prozeß selektiert und compiliert wurden.

Heute erhalten Mathematiklehrer im dritten Studienjahr einen 45stündigen Lehrgang in Informatik und absolvieren ein dreiwöchiges Praktikum am dialogfähigen Bildschirmcomputer. Bei 30 bis 50% der Studenten ist die Motivation stark, und sie leisten mehr und steigen über die gebotenen Möglichkeiten der fakultativen Kurse, des wissenschaftlichen studentischen Wettstreites und der Jugendobjekte eher in das Informatikstudium ein, als es der offizielle gültige Lehrplan zuläßt. Dieser Lehrplan, der seit der Eröffnung des fünfjährigen Lehrerstudiums 1982 gültig ist, wird als bewährt eingeschätzt. Seine Potenzen zur gezielten und intensiven Vorbereitung auf den Ausbildungsteil Informatik werden gegenwärtig geprüft und erschlossen. Es besteht das Ziel, alle künftigen Mathematiklehrer zur Durchführung eines Informatikunterrichts an POS oder EOS zu befähigen. Es gibt Ansätze zu Überlegungen, die Informatikausbildung auf das Fachlehrerstudium anderer Unterrichtsfächer auszudehnen.

Da die Anzahl der im Beruf stehenden Lehrer mit Informatikkenntnissen nicht ausreichend ist und das Bereitstellen über die reguläre Ausbildung zu lange dauert, bleibt nur die Weiterbildung. Das Bereitstellen der Lehrkräfte, d.h. die gezielte Vorbereitung durch Kurse für die Erprobungsmaßnahmen – etwa 80 Lehrer –, erfolgte in Wochenkursen an pädagogischen Hochschulen. Gegenwärtig werden durch die Weiterbildungskabinette für Lehrer in den Bezirken entweder zweiwöchige Ferienkurse für Lehrer oder aber halbjährige Kurse mit einem Tag pro Woche organisiert. Man wählt dafür i. allg. einen Lehrer pro Kreis aus, der dann über das Kreis-Weiterbildungskabinett seinerseits als Unterrichtender und damit als Multiplikator wirken soll.

Inzwischen wurde ein Studienprogramm für die Ausbildung von Informatiklehrern bestätigt. Diese Ausbildung soll vorerst als Zusatz- oder postgraduales halbjähriges Studium durchgeführt werden. Es wird zwei Richtungen geben: Informatik und Mathematik einerseits und Informatik und Automatisierung andererseits. Vom Frühjahrssemester 1989 an wird mit dieser Ausbildung an etwa 10 Pädagogischen Hochschulen und Universitäten für je etwa 25 Teilnehmer begonnen.

So entwickelt sich gegenwärtig einfach aus dem Zwang der Notwendigkeit eine Methodik des Informatikunterrichts. Dazu einige Gedanken zu der Grundlage Programmierung und Auswahl der Sprache. Wir sind in der Anfangsphase allein auf BASIC angewiesen. Für die allein in größerer Zahl verfügbaren Kleincomputer gab es nur diese Sprache. Jetzt ist auch PASCAL verfügbar. Ein späterer Bildungscomputer wird ein Spektrum von Sprachen bieten, aus dem dann gewählt werden kann (COMAL, PASCAL, FORTH, LOGO, PROLOG). Aus der Liste erkennt man, daß ganz unterschiedliche Sprachtypen in Untersuchung sind. Man hat befehls- und funktionalorientierte Sprachen mit und ohne Trennung von Daten und Algorithmus ebenso im Blick wie die deskriptive Problemlösung durch den Übergang von der Daten- zur Wissensverarbeitung. Vorläufig aber ist die Didaktik da-

durch bestimmt, daß das Programmieren in einem guten Stil auch mit BASIC vermittelt wird. Das erste Programm soll schon davon überzeugen, daß man den Computer zur Problemlösung braucht. Nach dem Variablenkonzept mit den drei Aspekten »Name (im Programm) – Wert (im Computer) – Bedeutung (für die Problemlösung)« und dem Modellaspekt der problem- und lösungbeschreibenden Daten einerseits und Lösungsalgorithmus andererseits beginnen wir mit der Zyklusstruktur FOR – TO – NEXT. Für die Verzweigungen IF – THEN und IF – THEN – ELSE sowie die beiden Zyklustypen WHILE – DO (mit vorangestelltem Test) und REPEAT – UNTIL (mit nachgestelltem Test) werden BASIC-Ersatzkonstruktionen gegeben. Das GOTO wird nur als Bestandteil dieser Ersatzkonstruktionen vermittelt. Die Algorithmenkonstruktion soll nur in den angegebenen Programmkonstrukten vollzogen werden. Dabei stellt sich heraus, daß dieser Unterricht zu einem guten Programmierstil nahezu unmöglich ist, wenn schon »wild« erworbene BASIC-Kenntnisse vorhanden sind.

In BASIC müssen die für das Modellieren vieler Probleme so wichtigen linearen Listen und verzweigten Listen (Bäume), also die strukturierten Daten durch geschicktes Arbeiten mit Indizes simuliert werden. Gleiches gilt für die unterschiedlichen Methoden der Parametervermittlung (Wert – Name – Prozedur), da BASIC nur die Wertvermittlung über globale Variablen kennt und das rekursive Aufrufen von Prozeduren bzw. Subroutinen. Alle genannten Dinge werden in einem BASIC-Kurs nicht gelehrt, wenn dieser einfach alle BASIC-Befehle der Reihe nach erklärt. Sie bilden aber Bestandteile eines Kurses im »Strukturierten Programmieren«, wie ihn die Methodik vorsieht und empfiehlt. Die Anwendungsfälle wechseln bald von der Mathematik zur Textverarbeitung und Grafik.

Damit wird gezeigt, daß der Computer nicht einfach ein Rechner, sondern eine informationsverarbeitende Maschine ist.

5. Informatik und Computer als neues pädagogisches Medium

Die Eigenschaft der Informationsverarbeitung führt sehr schnell zu der Erkenntnis, daß *Computer* auch in der pädagogischen Arbeit *als unterstützendes Instrument in der Hand des Lehrenden* eingesetzt werden können. Dieser Prozeß setzt in verschiedenen Stufen und Etappen ein. Zunächst hat man den Informatikunterricht selbst, die Mathematik, die Naturwissenschaften (Physik, Chemie als exakte und Geografie, Biologie als beschreibende Wissenschaften) und die Polytechnik im Blickfeld. Dann greift der Einsatz auf Sprachfächer, Gesellschaftswissenschaften und sogar Kunsterziehung (Musik und Zeichnen) über. Die erste Anwendung ist gewöhnlich die der *Demonstration von Vorgängen*, das *Simulieren* von teuren, gefährlichen, zu langsam oder zu schnell in der realen Welt ablaufenden Experimenten in beliebiger Wiederholbarkeit mit Parametervarianten. Ein typisches Beispiel ist der schräge Wurf in der Physik. In der Hochschulausbildung sind bereits viele Ausbildungsstoffe durch Computerunterstützung im Hörsaal untersetzt, ein anderer schon verbreiteter Einsatz findet im Prozeß der *Leistungskontrolle* und des Examinierens statt. Es gibt Programme für unterschiedliche Verfahren der Antwortkontrolle. Aber noch sind dieser Verwendung Grenzen auferlegt, da die Verfahren der künstlichen Intelligenz (freie Antwortformulierung und semantische

Textanalyse) dies in voller Breite mit den verfügbaren Computerressourcen nicht zulassen. Ein weiterer pädagogischer Prozeß, gut geeignet für Computereinsatz, ist das *Üben und Stofffestigen*. Dieser Vorgang kann mit Computern gut individualisiert werden. Ein Übungs- oder Trainingsprogramm paßt sich in der Arbeitsgeschwindigkeit dem Lernenden an und auch die Steuerung des Lernweges hängt von den individuellen Leistungen ab. Ähnliches gilt für die *Stoffvermittlung*. Aber dabei vertreten wir den Standpunkt, daß der Computer *nur* in *Stoffvermittlungen mit Massen- und Routinecharakter* eingesetzt werden soll. Allgemein soll das Lehrer-Schüler-Verhältnis und der Unterricht in breiter Front mit dem Schüler- oder Studentenkollektiv durch keine Technisierung gestört oder beeinträchtigt werden. Ein Beispiel für richtigen und angepaßten Unterricht mit Computern ist das Selbsterklären des Computers über seinen Aufbau, über das Bedienen der Tastatur, das Korrigieren von Programmteilen usw.

Für den Schulunterricht ist dieser Computereinsatz erst in der Vorbereitungsphase. Das Erzeugen von guten Lehrprogrammen setzt eine diesen Vorgang unterstützende Autorensprache und Schulung von Autoren voraus. Auf der Basis von PASCAL wurde in der DDR (PH, Dresden) eine Lehrprogramm-Formulierungssprache LEFO entwickelt. Richtlinien zur Programmentwicklung werden gegenwärtig von einer interdisziplinären Arbeitsgruppe »Computer in pädagogischen Prozessen (CIPP)« an der PH Dresden unter Anleitung der APW untersucht. Dieser Arbeitsgruppe gehören außer Informatikern, Fachlehrern und Fachdidaktikern auch Pädagogen und Psychologen an. Wir streben an, daß qualifizierte Lehrprogramme erzeugt und verwendet werden. Sie sollen die Möglichkeiten der Computer optimal nutzen und sich von den programmierten Lehrbüchern in Form und Inhalt deutlich abheben.

Autor:

Prof. Dr. sc. nat. Immo Kerner

Pädagogische Hochschule Dresden
Sektion Mathematik

Taschenrechner in der Schule

1. Zur Vorgeschichte

Auf dem VIII. Pädagogischen Kongreß, der im Oktober 1978 in Berlin stattfand, wurde im Referat des Ministers für Volksbildung, Margot Honekker, u.a. die Frage aufgeworfen, welche Rolle künftig Taschenrechnern in der Schule zukommen sollte. Es wurde gefordert zu prüfen, »ab welcher Klassenstufe, zu welchen Stoffgebieten und mit welchem Ziel sich die Nutzung dieser Rechner als sinnvoll erweist oder wo ihr Einsatz nicht richtig ist« ([1], S. 15). Zur Realisierung dieser Aufgabenstellung ist danach durch die Akademie der Pädagogischen Wissenschaften der DDR eine Arbeitsgruppe gebildet worden, die durch theoretische und schulpraktische Untersuchungen Voraussetzungen für künftige schulpolitische Entscheidungen hinsichtlich des Taschenrechnereinsatzes schaffen sollte. Unter Leitung dieser Arbeitsgruppe wurde u.a. von 1979 bis 1983 ein Schulversuch durchgeführt, bei dem zunächst 21 Klassen mit Taschenrechnern des Typs MR 410 und später noch zwei Klassen mit Taschenrechnern der Typen MR 610 bzw. MR 609 ausgestattet worden sind.

Bereits die ersten Zwischenauswertungen der Versuchsergebnisse ließen die Vorteile erkennen, die der Taschenrechnereinsatz mit sich brachte: Die Schüler der Versuchsklassen benötigten im Vergleich zu denen der (leistungsmäßig sonst gleichwertigen) Kontrollklassen bei Berechnungen, die nicht im Kopf ausführbar waren, wesentlich weniger Zeit und machten vor allem viel weniger Fehler! Dies eröffnete Möglichkeiten einer intensiveren Stoffbehandlung nicht nur im Fach Mathematik, sondern auch in den Fächern Physik, Chemie und ESP.

Die sehr positiven Ergebnisse des Schulversuchs und die Produktion von Taschenrechnern in den erforderlichen hohen Stückzahlen ermöglichten die Entscheidung, ab Schuljahr 1984/85 in der Abiturstufe und mit Beginn des Schuljahres 1985/86 ab Klasse 7 Taschenrechner in die Schulen einzuführen. Dabei entschied man sich zu Recht nicht für den relativ einfachen Typ MR 410, sondern mit dem SR 1 für einen wesentlich leistungsfähigeren Rechner, der allen Anforderungen der Schule (bis Klasse 12) und auch den Bedürfnissen vieler Fachrichtungen der Berufs-, Fachschul- und Hochschulausbildung voll gerecht wird. Er entspricht dem im Schulversuch eingesetzten Typ MR 609, ermöglicht also neben den Grundrechenoperationen, dem Quadrieren und Quadratwurzelziehen beispielsweise trigonometrische Berechnungen, besitzt einen wesentlich größeren Zahlenbereich, arbeitet

mit Vorrangautomatik und ähnliches mehr.

Bei der zur Zeit laufenden Erneuerung der Mathematiklehrbücher wird die Verwendung des SR 1 ab Klasse 7 übrigens voll berücksichtigt – sowohl durch entsprechende Lehrabschnitte als auch durch das Aufgabenmaterial.

2. Gründe für die Verwendung von Taschenrechnern in der Schule

Zeiteinsparung

Es wurde bereits erwähnt, daß die Schüler durch die Benutzung des Taschenrechners bei Berechnungen weniger Zeit als bei Verwendung des Rechenstabs, der Zahlentafel oder bei schriftlichem Rechnen benötigen. Diese Erscheinung wird um so deutlicher, je komplizierter die auszuführenden Rechnungen sind. Im Schulversuch wurde z.B. festgestellt, daß Schüler der Klasse 10 die Berechnung einer Länge mit Hilfe des Cosinussatzes (nach der Gleichung $c = \sqrt{a^2+b^2-2\,ab\cdot\cos\gamma}$) bei Verwendung des SR 1 in durchschnittlich 2 Minuten bewältigen, während sie ohne Taschenrechner – bei Benutzung von Tabellen – etwa 12 Minuten benötigen.

Diese an vielen Stellen auftretenden Zeiteinsparungen können genutzt werden, um zum Beispiel komplizierte Begriffe oder Zusammenhänge ausführlicher zu erläutern, für das Suchen von Lösungsansätzen mehr Zeit aufzuwenden, durchgeführte Berechnungen sorgfältig zu überprüfen und ähnliches mehr. Damit entstehen *günstigere Bedingungen für eine vertiefte mathematische (bzw. naturwissenschaftliche) Bildung der Schüler.*

Verschiedentlich wird in diesem Zusammenhang der Einwand erhoben, daß die Verwendung des Taschenrechners aber auch *negative Folgen* habe, daß die Schüler vor allem das *Rechnen verlernen.*

Man befürchtet

- abnehmende Fertigkeiten im Kopfrechnen und auch abnehmende Bereitschaft dazu sowie abnehmende Fertigkeiten im schriftlichen Rechnen – also eine große **Rechnerabhängigkeit**;
- abnehmende Kontrollfähigkeit und auch fehlende Kontrollbereitschaft – also eine blinde **Rechnergläubigkeit**.

Dem sind folgende Feststellungen entgegenzuhalten:

Man darf Rechnenkönnen *nicht ausschließlich* auf den Aspekt mündlicher oder schriftlicher Rechenfertigkeiten reduzieren. Rechnenkönnen ist eine relativ *komplexe* Persönlichkeitsqualität, die zwar Fertigkeiten im mündlichen und schriftlichen Rechnen als *eine* wichtige Komponente einschließt, aber zum Rechnenkönnen gehören auch Fertigkeiten im Arbeiten mit Variablen, im Umgang mit Rechenhilfsmitteln (einschließlich Taschenrechner!), Größenvorstellungsvermögen, Fähigkeiten im Analysieren von Termen, im Auswählen rationeller Lösungswege und im Nutzen von Rechenvorteilen. Ferner gehört dazu, die Lösungswege exakt und übersichtlich darzustellen, die Ergebnisse stets kritisch zu überprüfen und die Lösungen mit sinnvoller Genauigkeit anzugeben (vgl. [2]).

Die Entwicklung eines so umfassend verstandenen Rechnenkönnens ist nach wie vor eine wichtige Aufgabe des Mathematikunterrichts, für deren Bewältigung der Taschenrechner kein Hindernis, sondern – sinnvoll verwendet – sogar ein wirkungsvolles Hilfsmittel darstellt!

Es seien nur die Möglichkeiten erwähnt, mit Hilfe des Taschenrechners Einsichten in funktionale Zusammenhänge zu gewinnen, durch Wert-

	Durchschnittliche Anzahl richtig gelöster Aufgaben im Rechentest				
	Kl. 7 Anfang	Kl. 7 Ende	Kl. 8	Kl. 9	Kl. 10
Klassen mit TR	82	96	105	108	111
Klassen ohne TR	80	98	106	112	112

Tabelle 1

schrankenberechnungen Verständnis für Regeln zu erreichen, die beim Rechnen mit Näherungswerten zu beachten sind, relativ mühelos Kontrollrechnungen durchführen zu können und ähnliches mehr.

Natürlich besteht die Gefahr, daß durch ständige Benutzung des Taschenrechners – auch bei Aufgaben, die man eigentlich leicht im Kopf lösen kann $\left(\text{wie z. B. } 7 + 2 \cdot 4{,}5; \frac{5}{12} + \frac{1}{6}; -19 - 11; 14^2; \sqrt{121}; 190 + 1{,}073\right)$ – die Rechenfertigkeiten verkümmern. Dem kann aber entgegengewirkt werden, wenn die Lehrer (und auch die Eltern) insbesondere *dem Training des Kopfrechnens genügend Aufmerksamkeit widmen.*

Im Schulversuch ist deshalb immer wieder darauf orientiert worden, bestimmte Aufgaben *ohne* Benutzung von Rechenhilfsmitteln zu lösen.

Durch ein *ausgewogenes Verhältnis zwischen hilfsmittelfreiem Rechnen und dem Verwenden des Taschenrechners bzw. der Zahlentafel* wurde erreicht, daß die Schüler aus den Versuchsklassen im Kopfrechnen nicht schlechter waren als die Schüler der Kontrollklassen. Beispielsweise hatten die Schüler in einem speziellen Test 120 sogenannte Grundaufgaben der Addition, Multiplikation und Division (z. B. $8 + 7$; $12 - 3$; $4 \cdot 9$; $64 : 8$) zu lösen, wofür ihnen 6 Minuten zur Verfügung standen. Dieser Rechentest wurde vor Einführung des Taschenrechners (zu Beginn der Klasse 7) und dann am Ende eines jeden Schuljahres durchgeführt.

Die Tabelle 1 gibt eine Übersicht über die Schülerleistungen.

Wie die Ergebnisse zeigen, traten zwischen beiden Gruppen im gesamten Untersuchungszeitraum keine nennenswerten Unterschiede auf. Von Schuljahr zu Schuljahr war in beiden Populationen ein Leistungsanstieg zu verzeichnen, der vor allem auf einen merklichen Rückgang der Anzahl nicht bearbeiteter Aufgaben zurückzuführen war. Das Rechentempo und die Konzentrationsfähigkeit der Schüler haben also im Verlaufe der 4 Schuljahre zugenommen.

Eine zu starke Rechnerabhängigkeit kann demnach bei einem sinnvollen Einsatz von Taschenrechnern in der Schule vermieden werden (vgl. [3]).

Bei einem Vergleich *schriftlicher* Rechenleistungen schnitten die Schüler der Versuchsklassen allerdings etwas schlechter ab als die Schüler der Kontrollklassen. Das war zu erwarten, ist u. E. aber kein Anlaß zur Besorgnis, denn die Verwendung eines Taschenrechners macht schriftliches Rechnen eben weitgehend überflüssig. Das ist ja gerade ein Vorteil des Taschenrechners.

Größere Erfolgssicherheit beim Rechnen

Mindestens genauso bedeutsam wie die Zeiteinsparung ist der starke Rückgang der Fehlerzahl beim Rechnen mit

Tabelle 2

Aufgabe	Anteil richtiger Lösungen	
	in den Versuchsklassen	in den Kontrollklassen[1])
$33 \cdot 0{,}072$	98%	38%
$7{,}9 : 2{,}3$	95%	58%
$\frac{2{,}71 \cdot 1{,}27}{5{,}7}$	84%	24%
$0{,}92^2$	87%	70%
$12{,}4^2 : 3{,}8^2$	80%	47%
$\sqrt{42} \cdot \sqrt{6{,}5}$	84%	57%

[1]) Die Schüler der Kontrollklassen arbeiteten dabei mit dem Rechenstab bzw. mit der Zahlentafel.

dem Taschenrechner. Einige Beispiele aus Kontrollarbeiten, die im Schulversuch durchgeführt worden sind, möge Tabelle 2 belegen.
Dieser Effekt hatte zur Folge, daß die Schüler der Versuchsklassen beim Lösen von Anwendungsaufgaben häufiger zu richtigen Endergebnissen kamen als die Schüler der Kontrollklassen, obwohl die Leistungen beider Schülergruppen im Finden eines geeigneten Lösungsweges etwa gleich waren.

Beispiele:

a) Bei einer Anwendungsaufgabe, die auf die Berechnung eines Zylindervolumens führte (Ausschachten eines Brunnens), fanden in den Versuchsklassen 85% der Schüler, die einen richtigen Ansatz gemacht hatten, auch das richtige Endergebnis. In den Kontrollklassen betrug dieser Anteil nur 58%.

b) Bei einer stöchiometrischen Berechnung im Chemieunterricht gelangten 94% der Schüler aus Versuchsklassen, die einen richtigen Ansatz hatten, auch zum richtigen Ergebnis. In den Kontrollklassen waren es nur 67%.

Die höhere Erfolgswahrscheinlichkeit beim Lösen von Aufgaben wirkte sich positiv auf das Interesse der Schüler am Fach Mathematik aus, und dies wiederum schafft günstigere Bedingungen für das Erreichen guter Bildungsergebnisse in diesem Fach.

Heranführen an informationsverarbeitende Technik

Gegenwärtig vollzieht sich in raschem Tempo die Einführung informationsverarbeitender Technik in sehr vielen Bereichen des wirtschaftlichen und gesellschaftlichen Lebens. Das Volksbildungswesen muß darauf in angemessener Weise reagieren. Die Einführung der Taschenrechner ist dabei ein erster Schritt.

Obwohl der SR 1 die entscheidende Qualität der neuen Technik – nämlich *Programmierbarkeit* – kaum besitzt (lediglich durch die Konstantenautomatik ist eine sehr einfache Form von Programmierung realisierbar), können die Schüler beim Arbeiten mit den Taschenrechnern an *Denk- und Arbeitsweisen* herangeführt werden, die für den Umgang mit jeglichen informationsverarbeitenden Geräten bedeutsam sind. Beispielsweise enthält das *Aufstellen, Prüfen, Abarbeiten und Werten von Rechenablaufplänen solch wichtige Elemente algorithmischen Arbeitens* wie das *Zerlegen eines Rechenvorgangs in elementare Einzelschritte* (deren Darstellung für den Taschenrechner »verständlich« sein muß), die Festlegung einer geeigneten *Reihenfolge der Schritte*, die Durchführung von *»Proberechnungen«* (vor allem bei komplizierteren Termen), Überlegungen zur möglichen bzw. notwendigen *Genauigkeit von Ergebnissen* und ähnliches mehr. Dazu kommt die Schulung des *Konzentrationsvermögens* und die Gewöhnung an *Exaktheit beim Arbeiten* eines Rechenablaufplans.

Schließlich ist noch ein Effekt zu nennen, der nicht unterschätzt werden sollte: die Taschenrechner bedeuten einen »Einstieg« in die Welt der informationsverarbeitenden Technik, sie verhindern das Entstehen zu großer »Hemmschwellen« bei der Begegnung mit komplizierteren Geräten (z.B. Bürocomputer, Heimcomputer usw.), mit denen immer mehr Menschen in zunehmendem Maße in Berührung kommen werden. Gegenwärtig sind derartige »Hemmschwellen« insbesondere bei Erwachsenen (kaum bei Jugendlichen) oft vorhanden und nicht immer leicht zu überwinden.

3. Das Heranführen der Schüler an den SR 1

Wenn die Schüler in Klasse 7 erstmalig mit dem Taschenrechner arbeiten (die Wahl gerade dieser Klassenstufe ist übrigens nicht zufällig, auf die Gründe soll hier aber nicht weiter eingegangen werden), so ist keineswegs beabsichtigt, sie sofort mit *allen* Möglichkeiten dieses Gerätes vertraut zu machen. Ein solches Vorgehen wäre nicht möglich (weil den Schülern notwendige mathematische Vorkenntnisse noch fehlen), und es ist auch nicht erforderlich. So lernen die Schüler am Beginn der Klasse 7 zunächst mit Hilfe des Taschenrechners die vier Grundrechenoperationen mit gebrochenen Zahlen in Dezimalbruchdarstellung sicher auszuführen. Darüber hinaus werden sie befähigt, Terme mit mehreren gleichen bzw. verschiedenen Operationen, wie z.B.

$$a \cdot b \cdot c; \quad a + b + c; \quad \frac{a+b}{c}; \quad \frac{a \cdot b}{c};$$

$$a + b \cdot c; \; (a + b) \cdot c,$$

unter Nutzung des SR 1 zu berechnen (vgl. [4]).

In dieser Anfangsphase werden die Schüler also mit den Tasten [+]; [×]; [÷]; [−]; [=]; [CE−C] des SR 1 sowie mit der Konstantenautomatik bei den Grundrechenoperationen vertraut gemacht. Außerdem erfahren sie, daß der SR 1 über Vorrangautomatik verfügt (er beachtet: »Punktrechnung geht vor Strichrechnung«). Da möglicherweise nicht alle Schüler in der Klasse mit dem SR 1 arbeiten, müssen die anderen prüfen, wie ihr Gerät arbeitet. Der Lehrer kann dabei etwas Hilfestellung geben, aber sicher nicht ständig auf alle unterschiedlichen Details eingehen. Zwangsläufig kommt dann den Eltern dieser Schüler eine gewisse Verantwortung dafür zu, daß ihr Kind *seinen* Taschenrechner auch wirklich beherrscht.

Die **Mathematiklehrbücher** sind vor allem auf den SR 1 abgestimmt. Es ist also am günstigsten, wenn alle Schüler der Klasse diesen Taschenrechner verwenden.

Im weiteren Mathematikunterricht der Klasse 7 werden den Schülern noch folgende Tasten des SR 1 erläutert:

- die Taste [%] in der Prozentrechnung;
- im Stoffgebiet »Rationale Zahlen« die Vorzeichenwechseltaste [+/−];
- beim Thema »Quadratzahl und Quadratwurzel« die Funktionstasten [x^2] und [$\sqrt{}$];
- im Stoffabschnitt »Kreisberechnung« die Taste [π].

Tabelle 3 gibt eine Gesamtübersicht darüber, welche Tasten des SR 1 die Schüler in welcher Klassenstufe kennenlernen. Das schließt nicht aus, daß ein Schüler sich selbst (bzw. mit Hilfe der Bedienungsanleitung) gewisse Tasten »vorzeitig« erschließt und bei bestimmten Rechnungen auch nutzt.

Die Benutzung des Speichers, den die

Tabelle 3

Klasse	Im Mathematikunterricht einzuführende Tasten des SR 1
7	[+], [×], [−], [÷], [%], [+/−], [x^2], [√], [π], [=] [CE—C] (Konstantenautomatik, Vorrangautomatik)
8	[1/x], [MR], [X→M], [M_+] [y^x] (insbesondere für $x = 3$ und $x = \frac{1}{3}$)
9/10	[y^x], [EEX], [lg] [sin], [cos], [tan], [F] (Schalter DEG / RAD / GRD)
11/12	[ln]

Schüler in Klasse 8 kennenlernen (also die Tasten [X→M], [M_+], [MR]), bereitete manchen Schülern im Schulversuch zunächst einige Schwierigkeiten. Das Problem bestand für sie darin, sich zu merken, welche Operation(-en) schon ausgeführt und welche Zahl gespeichert war und wann man das gespeicherte, nicht sichtbare Zwischenergebnis erneut in die Rechnung einbeziehen muß. Eine Hilfe für diese Schüler war die Anfertigung von Rechenablaufplänen, aus denen jeweils hervorgeht, in welcher Reihenfolge die Tasten des Taschenrechners zu bedienen sind. Der Schulversuch zeigte im übrigen, daß die Schüler nach einiger Zeit auch im Umgang mit dem Speicher sicher werden.

Nachdem der Taschenrechner im Unterricht eingeführt wurde, kann er bei geeigneten Aufgabenstellungen im Mathematikunterricht, aber auch in anderen Fächern, wie z. B. Physik, Chemie, ESP, genutzt werden.

4. Konsequenzen des Taschenrechnereinsatzes

Die zunehmende Verwendung von Taschenrechnern in der gesellschaftlichen Praxis und nunmehr auch in der Schule zwingen dazu, sich über einige Konsequenzen dieser Entwicklung klar zu werden. Am nächstliegenden sind dabei wohl vor allem folgende Fragen:

a) Welche Bedeutung hat künftig das sog. »Kopfrechnen«?
b) Wie weit müssen Fertigkeiten im schriftlichen Rechnen entwickelt werden?
c) Welche Rolle spielen *andere* Rechenhilfsmittel als der Taschenrechner?

Zu a): Das »*Kopfrechnen*« behält nach wie vor seine Bedeutung, es *gewinnt eher noch an Wichtigkeit*. Dafür gibt es zwei Gründe:

Erstens wäre es geradezu lästig, so stark vom Taschenrechner abhängig zu sein, daß man ihn auch für einfachste Rechnungen heranziehen müßte. Aufgaben wie $3 \cdot 18$; 4^3; $\sqrt{169}$; $36 : 0{,}5$ und ähnliche mehr sind für den Einsatz

des Taschenrechners eigentlich »zu leicht«!

Zweitens – und das ist besonders wichtig – muß man in der Lage sein, Ergebnisse des Taschenrechners durch *im Kopf* ausgeführte Überschlagsrechnungen oder Abschätzungen beurteilen zu können (im Sinne von »möglich«, »verdächtig« oder «unmöglich«).

Zu b): *Schriftliches Rechnen* (mit ganzen Zahlen bzw. mit Dezimalbrüchen) *verliert an Bedeutung*, da jene Aufgaben, für die es bisher herangezogen wurde, überwiegend mit dem Taschenrechner gelöst werden – und zwar schneller und sicherer. Zwar werden im Unterricht auch künftig den Schülern die schriftlichen Rechenverfahren vermittelt (im wesentlichen in Klasse 4), dabei liegt der Schwerpunkt aber mehr auf dem prinzipiellen Vorgehen und dessen Festigung mit Hilfe einfacher Aufgaben als auf langwierigen Übungen mit vielstelligen, »komplizierten« Zahlen.

Zu c): Der *Rechenstab ist* selbstverständlich *überholt* und seine Behandlung wurde zu Recht aus dem Lehrplan gestrichen. Anders ist die Situation dagegen bei *Tabellen*. Sie *spielen* in der gesellschaftlichen Praxis (sowohl in den Naturwissenschaften und der Technik als auch in ökonomischen Bereichen) *nach wie vor eine bedeutende Rolle*, so daß ein gewisser Grad von Vertrautheit im Umgang mit ihnen durch die Schule erreicht werden muß. Deshalb wird z. B. im Mathematikunterricht der Klasse 7 den Schülern gezeigt, wie man mit Hilfe der Quadrattafel das Quadrat bzw. auch die Quadratwurzel einer Zahl bestimmen kann. Dabei geht es nicht so sehr um die Entwicklung von Fertigkeiten (mit dem Taschenrechner löst man derartige Aufgaben sowieso schneller), sondern mehr um ein Bekanntmachen mit Tabellen überhaupt.
Neben diesen »naheliegenden« Konsequenzen des Taschenrechnereinsatzes gibt es aber auch noch andere Punkte, die bedacht werden müssen. Auf zwei sei hier noch hingewiesen:

d) Es ist notwendig, daß die Schüler bereits *vor* der Verwendung des Taschenrechners (also *vor* Klasse 7) ein elementares Verständnis für den Begriff »Näherungswert« und das Rechnen mit Näherungswerten erreichen, damit sie eine Orientierung haben, wieviel Ziffern sie von einer möglicherweise achtstelligen Ergebnisanzeige eigentlich für das Resultat verwenden sollen bzw. können.
Im neuen Mathematik-Lehrbuch für Klasse 6 wird versucht, diese Forderung zu erfüllen, indem Regeln für das Bestimmen der Anzahl zuverlässiger Stellen bzw. Ziffern entwickelt und formuliert werden (vgl. [5], S. 89).

e) Die scheinbare Leichtigkeit des Rechnens mit dem Taschenrechner kann dazu verführen, mit ihm ermittelte Ergebnisse auf jeden Fall für richtig zu halten. (Hier wirkt die Erfahrung, daß man bei einfachen Tätigkeiten viel seltener Fehler macht als bei komplizierten.) Um so notwendiger ist es, die Schüler zu einer *kritischen Haltung zu erziehen*, zur Gewohnheit, *Rechenergebnisse zu überprüfen*. Das erfordert sowohl das Vertrautmachen der Schüler mit geeigneten Kontrollverfahren als auch eine *Wertung* ihres »Kontrollverhaltens« – und zwar nicht erst ab Klasse 7, sondern natürlich von Klasse 1 an.

5. Abschließende Bemerkungen

Die Einführung und Nutzung von Taschenrechnern in der Schule ist ein bedeutsamer Schritt. Er schafft in

einer ganzen Reihe von Fächern günstigere Bedingungen als bisher, die in den jeweiligen Lehrplänen fixierten Erziehungs- und Bildungsziele zu erreichen, und es kommt darauf an, diese günstigeren Bedingungen auch zielstrebig zu nutzen. Auf das Fach Mathematik bezogen, heißt das, mit Hilfe der neuen Lehrbuchreihe eine generell höhere Qualität des Mathematikunterrichts und damit der mathematischen Bildung der Schüler zu erreichen. Durch die Verwendung der Taschenrechner kann dieser Prozeß *unterstützt* werden, es wäre aber eine gänzlich unangemessene Sichtverengung, in der Einführung der Taschenrechner allein schon die neue Qualität zu erblicken. In unserer Schule zielt mathematische Bildung auf entschieden *mehr* als nur auf Umgehenkönnen mit einem (wenn auch modernen) *Rechenhilfsmittel.*

Literatur

[1] Honecker, M.: Der gesellschaftliche Auftrag unserer Schule. – Bulletin 2 vom VIII. Pädagogischen Kongreß. – Berlin, 1978

[2] Fanghänel, G., Flade, L.: Zur Bedeutung des Rechnen-Könnens für die mathematische Allgemeinbildung. – In: Mathematik in der Schule, H. 10. – S. 524—531

[3] Flade, L.; Walsch, W.: Taschenrechner im Mathematikunterricht – Ergebnisse eines Langzeitversuchs. – Wiss. Z. Univ. Halle XXXIII'84 M. H., H. 5. – S. 105—116

[4] Lehrplan Mathematik – Klasse 7. – Berlin, 1985

[5] Mathematik – Lehrbuch für Klasse 6. – Berlin, 1984

Autoren:

Prof. Dr. Werner Walsch
Doz. Dr. Lothar Flade
Martin-Luther-Universität Halle

Hinweise für Autoren

Herausgeber und Verlag danken den Lesern für das Interesse an den »Kleinstrechner-TIPS«, das sich in zahlreichen Zuschriften und Veröffentlichungsangeboten äußert. Beim Einsenden von Artikeln bitten sie folgende Hinweise zu beachten:

- In den »Kleinstrechner-TIPS« werden Artikel aus den auf der 4. Umschlagseite angegebenen Gebieten veröffentlicht.
- Manuskripte sind zweizeilig mit schwarzem Farbband mit Schreibmaschine zu schreiben, ä, ö und ü dürfen nicht durch ae, oe und ue ersetzt werden.
- Bilder sollten auf getrennten Blättern gezeichnet sein. Eine Bildunterschriftenliste ist beizufügen.
- Rechnerausdrucke (z. B. Programme) sollen tiefschwarz mit guter Ausnutzung des Formats (engzeilig), kein zu breites Kommentarfeld usw.) gedruckt sein.
- Die Autorenangabe soll enthalten: akadem. Grad, Vorname, Name, Tätigkeit, Arbeitsstelle, Privatanschrift.

Manuskripte mit einem Umfang von nicht mehr als 15 Schreibmaschinenseiten (einschließlich Bilder und Programme) sind in zwei Exemplaren an einen der Herausgeber zu senden (Anschriften auf 3. Umschlagseite).

Kann man bereits beim Programmieren die Rechenzeit beeinflussen?

Vielfach wird die Meinung vertreten: »Ein Computer rechnet außerordentlich schnell. Wozu soll ich mir also beim Programmieren den Kopf anstrengen, um ein möglichst elegantes und effektives Programm aufzustellen? Die Zeit, die ich zusetze, wenn ich mein Programm verbessern will, die holt der Rechner doch durch seine hohe Rechengeschwindigkeit wieder ein, auch wenn er ein paar Befehle mehr als nötig ausführen muß.«
Für einfache Programme mit geringem Rechenaufwand mag dies zutreffen. (Und trotzdem sollte man sich bei *jedem* Programm bemühen, die Eigenschaften des Computers so gut wie möglich auszunutzen.) Aber bei umfangreichen Programmen mit einem hohen Rechenaufwand, bei denen es dazu noch darauf ankommt, möglichst schnell zu den Ergebnissen zu kommen, kann man die Laufzeit eines Programmes merklich beeinflussen, wenn man weiß, welche Operationen schnell ablaufen und welche weniger schnell. Die folgenden Betrachtungen sollen Hinweise und Anregungen geben, wie man zu Programmen mit möglichst kurzer Laufzeit kommen kann.
Der Kleinrechner KC 85/1 besitzt eine interne Uhr, die die Zeit angibt, die seit dem Einschalten des Rechners verstrichen ist.
Mit dem Befehl

```
PRINT PEEK(29); PEEK(30);
PEEK(31)
```

kann man die seit dem Einschalten verflossene Zeit in Stunden (PEEK(29)), Minuten (PEEK(30)) und Sekunden (PEEK(31)) über den Bildschirm ausgeben lassen.
Wir wollen diese drei Funktionen PEEK(29) bis PEEK(31) nun dazu verwenden, um die Laufzeiten einfacher BASIC-Befehle festzustellen. Dazu schreiben wir ein kleines Rahmenprogramm, durch das vor dem Abarbeiten des BASIC-Befehles die Uhrzeit festgestellt wird, dann soll der Befehl 10000mal unmittelbar nacheinander ausgeführt werden und danach wird erneut die Uhrzeit festgestellt. Aus der Differenz der Uhrzeiten zu Beginn und am Ende des Programmes läßt sich dann leicht die benötigte Rechenzeit ermitteln. Dieses Rahmenprogramm könnte wie folgt aussehen:

```
10 H1 = PEEK(29):M1 = PEEK(30):S1
= PEEK(31)
20 FOR I = 1 TO 10000
30 REM HIER IST DER AUSZU-
FUEHRENDE BEFEHL
EINZUTRAGEN
40 NEXT I
50 H2 = PEEK(29):M2
= PEEK(30):S2 = PEEK(31)
60 PRINT:PRINT
70 PRINT
" BEGINN :";H1;":";M1;":";S1
```

```
80 PRINT
"  ENDE :  ";H2;":";M2;":";S2
90 PRINT
```

Die Ermittlung der benötigten Rechenzeit für die 10000 gleichartigen Rechenoperationen sollten wir aber auch noch dem Computer überlassen.
Dadurch daß unser Zeitmaß nicht auf dem gebräuchlichen Dezimalsystem beruht, sind hierzu einige Zusatzüberlegungen notwendig. Solange nämlich die Stunden-, Minuten- und Sekundenangaben zu Beginn der Rechnung jeweils kleiner sind als die entsprechenden Angaben am Ende der Rechnung, kann man die benötigten Stunden, Minuten und Sekunden durch einfache Subtraktionen ermitteln.

Wie hilft man sich aber, wenn beispielsweise die Sekundenangabe zu Beginn der Rechnung größer ist als die Sekundenangabe am Ende der Rechnung? Die folgende Programmerweiterung gibt darüber Auskunft:

```
100 IF S1 > S2
THEN S2 = S2 + 60 : M2 = M2 — 1
110 IF M1 > M2
THEN M2 = M2 + 60 : H2 = H2 — 1
120 T = (H2 — H1) *
3600  + (M2 — M1) * 60 + S2 — S1
130 PRINT
"  RECHENZEIT : ";T;" SEKUNDEN"
140 PRINT : PRINT : PRINT
150 END
```

Mit Hilfe dieses Rahmenprogramms führte eine Schülerin der 9. Klasse einer POS eine Reihe von Testrechnungen durch, um die Rechenzeiten für bestimmte, miteinander vergleichbare Rechenoperationen zu ermitteln. Einige Ergebnisse dieser Testrechnungen sind in der Tabelle 1 zusammengefaßt, in deren mittlerer Spalte jeweils die untersuchten Rechenoperationen angegeben sind. (Wenn man diese Rechnungen nachvollziehen möchte, dann brauchen nur die in der mittleren Spalte angegebenen Anweisungen in das obige Rahmenprogramm eingefügt zu werden.)
Wir wollen einige dieser Ergebnisse näher betrachten, um daraus Schlußfolgerungen für das Aufstellen zeitoptimaler Programme ziehen zu können.
In den beiden Programmen Nr. 1 und 2 geschieht eigentlich genau das gleiche: es wird in beiden Fällen 10000mal die bekannte Zahl π gespeichert. Der Unterschied besteht lediglich darin, daß im Programm 1 die Zahl π als numerischer Wert 3.14159 eingegeben wird, während im Programm 2 für π die rechnerinterne Variable PI verwendet wird.
Wie kommt nun der bemerkenswerte Unterschied in den Laufzeiten der beiden Programme zustande?
Dazu muß man wissen, daß die Computer alle numerischen Werte im allgemeinen *nicht* in der uns geläufigen Form *als Dezimalzahlen* speichern und verarbeiten, *sondern in Form von Dualzahlen.* Daraus folgt, daß der Rechner bei der Abarbeitung des Programmes Nr. 1 die Dezimalzahl 3.14159 10000mal in die zugehörige Dualzahl umwandeln muß, bevor er sie speichern kann. Beim Aufruf der rechnerinternen Konstanten PI hingegen wird auf einen Speicherplatz zugegriffen, in dem die gewünschte Zahl bereits in der erforderlichen Form als Dualzahl vorhanden ist. Es entfallen also beim Programm Nr. 2 die 10000 Umwandlungen einer Dezimalzahl in die Dualzahl, wodurch zwangsläufig das zweite Programm wesentlich schneller sein muß als das erste.
Die gleiche Feststellung können wir beim Vergleich der beiden Programme 3 und 4 treffen. (Im Programm 3 wird die Zahl 1.2345E-13 bei jedem Zyklendurchlauf neu in eine Dualzahl verwandelt, während sie im Programm 4

Tabelle 1. RECHENZEIT – VERGLEICHE

Programm-Nummer	Anweisungen	Rechenzeit in Sekunden für 10000 Rechenoperationen
1	30 K = 3.14159	272
2	30 K = PI	40
3	30 K = 1.2345E-13	647
4	5 A = 1.2345E-13	
	30 K = A	43
5	30 K = 2 ∗ 2	60
6	30 K = 2 ∧ 2	375
7	30 K = 2.3702 ∗ 2.3702	422
8	5 A = 2.3702	
	30 K = A ∗ A	65
9	30 K = 2.3702 ∧ 2	567
10	30 K = 2 ∗ 2 ∗ 2	78
11	30 K = 2 ∧ 3	375
12	30 K = 2.3702 ∗ 2.3702 ∗ 2.3702	624
13	5 A = 2.3702	
	30 K = A ∗ A ∗ A	88
14	30 K = 2.3702 ∧ 3	569
15	30 K = 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702	819
16	30 K = 2.3702 ∗ 2.3702	
	35 K = K ∗ K	465
17	30 K = 2.3702 ∧ 4	564
18	30 K = 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702	1617
19	30 K = 2.3702 ∗ 2.3702 ∗ 2.3702 ∗ 2.3702	
	33 K = K ∗ K	864
20	30 K = 2.3702 ∗ 2.3702	
	33 K = K ∗ K	
	36 K = K ∗ K	511
21	30 K = 2.3702 ∧ 8	566
22	30 K = 24.16 ∗ 0.1709 ∗ 3.648 ∗ 8.172 ∗ 0.1413 ∗ 82.67 ∗ 19.08 ∗ 4.0502 ∗ 0.0001746 ∗ 5.17624	1847
23	5 A=24.16 : B=0.1709 : C=3.648	
	6 D=8.172 : E=0.1413 : F=82.67	
	7 G=19.08 : H=4.0502	
	8 I=0.0001746 : J=5.17624	
	30 K=A ∗ B ∗ C ∗ D ∗ E ∗ F ∗ G ∗ H ∗ I ∗ J	271
24	30 K = SQR(58.27)	493
25	5 A = 58.27	
	30 K = SQR(A)	391
26	30 K = 58.27 ∧ 0.5	532
27	5 A = 58.27	
	30 K = A ∧ 0.5	428
28	30 K = 58.27 ∧ (1/2)	542
29	5 A = 58.27 : N = 1/2	
	30 K = A ∧ N	394
30	30 K = 58.27 0.33333	720

Programm-Nummer	Anweisungen	Rechenzeit in Sekunden für 10000 Rechenoperationen
31	5 A = 58.27 : N = 1/3	
	30 K = A∧N	396
32	30 K = EXP(0.32174)	447
33	5 A = 0.32174	
	30 K = EXP(A)	230
34	30 K = LN(673.19)	296
35	30 K = LN(5.7319)	361
36	30 K = LN(0.057319)	440
37	5 A = 673.19	
	30 K = LN(A)	180
38	5 A = 5.7319	
	30 K = LN(A)	180
39	5 A = 0.057319	
	30 K = LN(A)	186
40	5 A = 5.7319	
	30 K = SIN(A)	213
41	5 A = 5.7319	
	30 K = COS(A)	216
42	5 A = 5.7319	
	30 K = TAN(A)	412
43	5 A = 5.7319	
	30 K = ATN(A)	275
44	5 A = 5.7319	
	30 K = ABS(A)	47
45	5 A = —5.7319	
	30 K = ABS(A)	48
46	5 A = 5.7319	
	30 K = INT(A)	53
47	20 I = 0	
	30 I = I + 1	
	40 IF I < 10000 THEN 30	106
48	5 DI = 1 : IMAX = 10000	
	20 I = 0	
	30 I = I + DI	
	40 IF I < IMAX THEN 30	75
49	20 FOR I = 1 TO 10000	
	30 REM	
	40 NEXT I	22

nur einmal zu Beginn der Rechnung (Anweisung Nr. 5) bereitgestellt wurde. – Aus diesem Beispiel können wir noch etwa weiteres erkennen: Die Umwandlung einer dezimalen Gleitkommazahl (1.2345E-13) dauert wesentlich länger als die Umwandlung der Zahl 3.14159. Vergleicht man schließlich noch die Rechenzeiten der Programme 7 und 8, 12 und 13 sowie 22 und 23 miteinander, so liegt folgende **Empfehlung** nahe:

Wird in einem Programm eine numerische Konstante *mehrfach* benötigt, so ist es nicht ratsam, diese Konstante jedesmal in Form ihres Zahlenwertes

in die Anweisungen zu übernehmen. Es ist günstiger, für die Konstante gleich zu Beginn des Programms einen eigenen Speicherplatz zu reservieren und bei jedem Auftreten der Konstanten auf diesen Speicherplatz zurückzugreifen.

Bei den Beispielen 5 bis 23 werden die erforderlichen Rechenzeiten für unterschiedliche Rechenoperationen zweiter und dritter Stufe miteinander verglichen. Der Vergleich der Beispiele 5 mit 6, 7 mit 9, 10 mit 11, 12 mit 14, 15 mit 17 sowie 18 mit 21 zeigt, daß die Anwendung des Potenzoperators ∧ um so günstiger wird, je höher der Exponent der zu berechnenden Potenz ist.

Die zweite und die dritte Potenz einer Zahl erhält man im allgemeinen schneller, wenn man das Produkt von zwei bzw. drei Faktoren bildet. Bei höheren Potenzen dagegen empfiehlt es sich, den Operator ∧ anzuwenden.

Die Beispiele 25 bis 46 demonstrieren, daß zur Ermittlung der Funktionswerte der gebräuchlichsten mathematischen Funktionen doch ein recht hoher Zeitaufwand erforderlich ist. Ausnahmen davon stellen nur die Funktionen INT(A), ABS(A) und SGN(A) dar.

Bemerkenswert ist schließlich das, was in den drei letzten Beispielen demonstriert wird. Es handelt sich bei jedem Beispiel darum, daß eine Zählschleife insgesamt 10000mal durchlaufen wird. Dabei zeigt es sich, daß das Erhöhen des Zählers um den Wert 1 mit nachfolgendem Rücksprung zum Befehl 30 länger dauert, als wenn man die Schrittweite für die Erhöhung des Zählers und den Schlußwert für die Schleife von vornherein als Variable in das Programm eingibt. Noch wesentlich schneller ist aber der Aufbau einer Zählschleife mit Hilfe der FOR – TO – NEXT – Anweisung. Der Grund hierfür liegt darin, daß die FOR – TO – NEXT – Anweisung innerhalb des BASIC-Interpreters durch ein Maschinen-Unterprogramm realisiert worden ist, und daß Maschinen-Unterprogramme wesentlich schneller laufen als BASIC-Programme.

Zusammenfassend kann festgestellt werden, daß es der Programmierer durchaus in der Hand hat, ob ein von ihm erarbeitetes Programm schnell läuft oder nicht. Er muß nur unter den vorhandenen Realisierungsmöglichkeiten für eine Folge von Anweisungen diejenigen heraussuchen, die den geringsten Zeitbedarf haben. Es ist nicht immer leicht, diese optimale Variante für eine Anweisungsfolge zu finden. Dazu braucht man Programmier-Erfahrung, und dazu muß man auch einiges über den Ablauf der verwendeten Anweisungen im Rechner wissen. Das bedeutet aber, daß man sich intensiv mit den Eigenschaften »seines« Rechners und mit den Möglichkeiten der verwendeten Programmiersprache auseinandersetzen muß.

Für das Aufstellen eines Computer-Programms sollte man sich aber stets den folgenden **Grundsatz** zu eigen machen:

Es genügt nicht, irgendeine Anweisungsfolge niederzuschreiben, durch die das in der Aufgabe gestellte Ziel erreicht wird. Man sollte diese Anweisungsfolge immer noch einmal gründlich untersuchen, ob es nicht noch Möglichkeiten gibt, sie zu verbessern, denn jede unnötige oder ungünstige Anweisung bringt für den Computer Mehrarbeit mit sich.

Der Computer ist auch nur ein Mensch! Und welcher Mensch arbeitet schon gerne unnötig oder uneffektiv?

Autor:
Prof. Dr.-Ing. Hans Kreul
Technische Hochschule Zittau
Abteilung EDV und Rechentechnik

Lösungen quadratischer Gleichungen

Die folgenden beiden Programme ermitteln für quadratische Gleichungen mit reellen Koeffizienten alle Lösungen. Bei der ersten Version QUAGL1 wird die bekannte Lösungsformel programmiert. Eine zweite verbesserte Version erhöht für viele Fälle die Genauigkeit der Lösungen. Beide Programme sind für den Kleincomputer KC 85/2 geschrieben. Da keine rechnerspezifischen Befehle verwendet werden, läuft das Programm auch auf anderen BASIC-Computern problemlos.

1. Allgemeine Form der quadratischen Gleichung

Die allgemeine Form einer quadratischen Gleichung ist

$$A \cdot x^2 + B \cdot x + C = 0. \qquad (1)$$

Dabei wird vorausgesetzt, daß die Koeffizienten A, B und C reelle Zahlen sind. In der Literatur setzt man oft noch zusätzlich voraus, daß A von Null verschieden sein soll. Auf diese zusätzliche Voraussetzung wird hier verzichtet. Das hat den Vorteil, daß dieses Programm auch als Unterprogramm oder Programm-Modul verwendet werden kann, ohne daß im Hauptprogramm zusätzliche Fallunterscheidungen erforderlich wären.

Für $A = 0$ wird aus Gl. (1) eine einfache lineare Gleichung mit der Lösung

$$x = -\frac{C}{B} \qquad (2)$$

Der Sonderfall $A = B = 0$ hat keine Lösung, wenn C von Null verschieden ist. Gl. (1) führt dann auf einen Widerspruch.

Ist jedoch C ebenfalls Null, dann existieren unendlich viele Lösungen, weil jede beliebige Zahl x die Gl. (1) bei $A = B = C = 0$ erfüllt.

Es ist nun der häufigste Fall zu untersuchen, bei dem mindestens A von Null verschieden ist. Aus Gl. (1) erhält man nach Division der Gleichung mit A die Normalform der quadratischen Gleichung

$$x^2 + \frac{B}{A}x = -\frac{C}{A} \qquad (3)$$

Aus dieser gemischtquadratischen Gleichung wird durch eine quadratische Ergänzung

$$x^2 + \frac{B}{A}x + \left(\frac{B}{2A}\right)^2 = \left(\frac{B}{2A}\right)^2 - \frac{C}{A} \qquad (4)$$

und Anwendung der binomischen Formel die reinquadratische Gleichung

$$\left(x + \frac{B}{2A}\right)^2 = \left(\frac{B}{2A}\right)^2 - \frac{C}{A} \qquad (5)$$

mit der Lösung

$$\left|x + \frac{B}{2A}\right| = \sqrt{\left(\frac{B}{2A}\right)^2 - \frac{C}{A}} \qquad (6)$$

gefunden.

Für den Fall, daß der Term, dessen Betrag auf der linken Seite steht, positiv ist, ergibt sich die erste Lösung. Ist der Term negativ, erhält man die zweite Lösung. Es ist üblich und einfach, beide Lösungen in einer Lösungsformel zu schreiben:

$$x_{1,2} = -\frac{B}{2A} \pm \sqrt{\left(\frac{B}{2A}\right)^2 - \frac{C}{A}} \qquad (7)$$

bzw.

$$x_{1,2} = \frac{1}{2A}\left(-B \pm \sqrt{B^2 - 4AC}\right). \qquad (8)$$

Aus der Lösungsformel lassen sich drei Fälle ableiten. Über die Art der Lösung entscheidet offensichtlich der Radikand der Wurzel, den man deshalb als Diskriminante (discriminare = trennen, scheiden) bezeichnet.
Mit der Diskriminante

$$DI = B^2 - 4AC \qquad (9)$$

erhält man drei Fälle:

$DI > 0$: Wurzel reell, beide Lösungen reell und voneinander verschieden

$DI = 0$: Wurzel Null, beide Lösungen reell und gleich

$DI < 0$: Wurzel imaginär, beide Lösungen konjugiert komplex

Weitere Fälle, die sich aus Gl. (1) ergeben, wenn A von Null verschieden ist und B oder C oder beide Null sind, müssen hier nicht unterschieden werden. Man erhält für diese Fälle imaginäre oder reelle Lösungen, die auch Null sein können.
Das alles sieht auf den ersten Blick vielleicht komplizierter aus als es ist. Schafft man sich eine Übersicht über die Zusammenhänge, wie sie im oberen Teil des Struktogrammes von Bild 1 zu sehen sind, dann wird deutlich, wie sich die verschiedenen Möglichkeiten auf die Lösungen auswirken.

2. Genauigkeit der Lösungen

Auf den ersten Blick scheint es naheliegend zu sein, die Lösungsformel Gl. (8) zu programmieren und daraus die Lösungen x_1 und x_2 zu berechnen. Für einen großen Teil möglicher Anwendungen mag das auch ausreichend. Da dieses Programm später noch verbessert werden soll, wird nachfolgend ein kleines Programm angegeben, bei dem Gl. (8) verwendet wird. Um dieses Programm abzukürzen, beschränken wir uns hier auf quadratische Gleichungen, die mit Sicherheit reelle Lösungen haben.

Programm QUAGL1

```
10 REM QUAGL1
20 REM QUADRATISCHE
GLEICHUNGEN
30 REM MIT REELLEN LOESUNGEN
40 PRINT
"LOESUNGEN DER GLEICHUNG"
50 PRINT "A*X^2 + B*X + C = 0"
60 PRINT "FUER" : PRINT
70 INPUT "A = ";A
80 INPUT "B = ";B
90 INPUT "C = ";C
100 PRINT : PRINT "SIND"
110 DI = B*B-4*A*C
120 IF DI < 0 THEN PRINT
"KEINE REELLE LOESUNG" :
GOTO 170
130 X1 = (-B - SQR(DI))/(2*A)
140 X2 = (-B + SQR(DI))/(2*A)
150 PRINT : PRINT "X1 = ";X1
160 PRINT "X2 = ";X2
170 END
```

Nach dem Start dieses Programms mit RUN ENTER wird die Überschrift ausgegeben, und anschließend werden die Koeffizienten A, B und C vom Bildschirm angefordert. Nach der letzten Eingabe erscheinen die Lösungen der quadratischen Gleichung.

Beispiele:

Die quadratische Gleichung

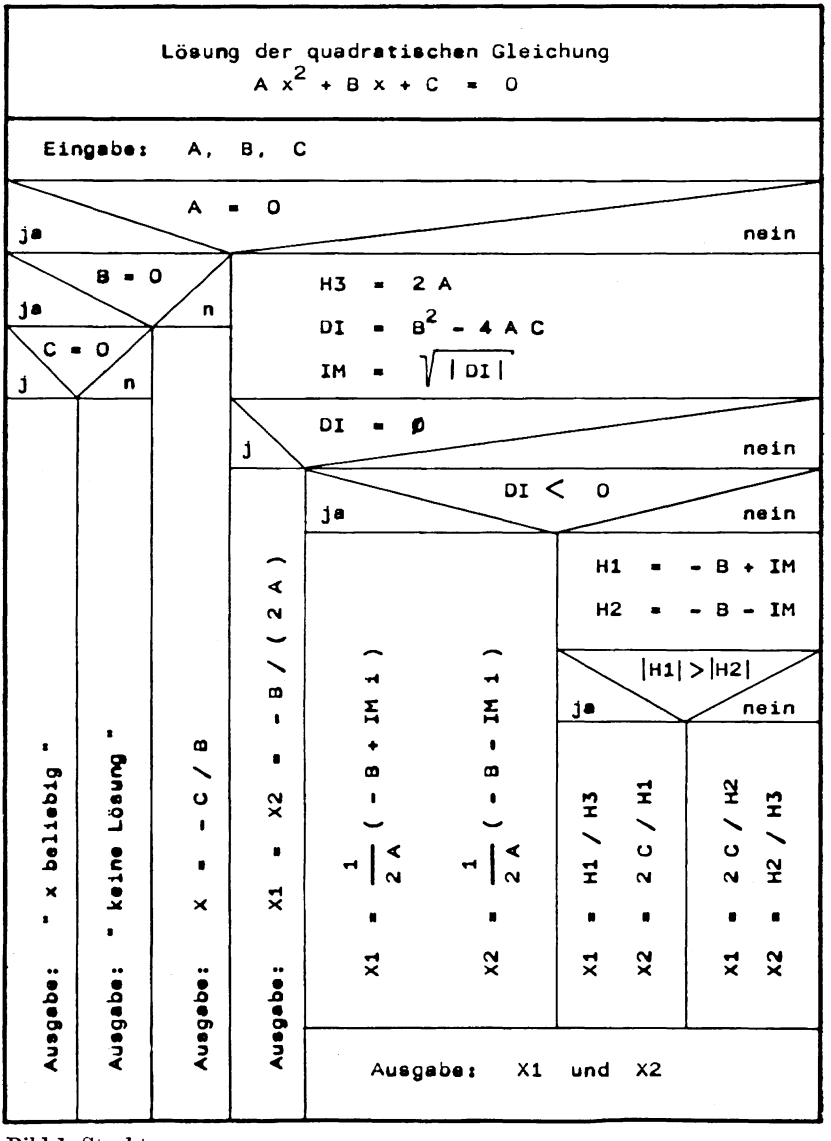


Bild 1. Struktogramm

$5\,x^2 - 20\,x - 105 = 0$

ist zu lösen.
Nach der Abarbeitung des Programms hat der Bildschirm folgenden Inhalt:

LOESUNGEN DER GLEICHUNG
A ∗X∧2 + B ∗X + C = Ø

FUER
A = 5
B = −2Ø
C = −1Ø5
SIND
X1 = 7
X2 = −3

Diese Lösungen sind exakt, und wir könnten zufrieden sein. Bei anderen Beispielen sieht die Lösung nicht so gut aus.
Für die quadratische Gleichung

$$x^2 + 111111\,x + 111110 = 0$$

liefert das Programm die Lösungen

$$x_1 = -1{,}02344 \text{ und } x_2 = -111110.$$

Für die Gleichung

$$-2\,x^2 + 4489{,}96\,x + 89{,}8 = 0$$

erhält man die Lösungen

$$x_1 = -0{,}201416 \text{ und } x_2 = 2245.$$

Setzt man die Lösungen in die Gleichungen ein, so fällt auf, daß die Gleichungen von den Lösungen mit den größeren Beträgen erfüllt werden, bei den kleinen Beträgen von x weicht die linke Seite oft beachtlich von Null ab. Die Lösungen können mit Fehlern von mehr als $10\,\%$ behaftet sein, wie z. B. bei $A = 0{,}1$, $B = 111110{,}9$ und $C = 11111{,}1$.
Ursache für die ungenauen Lösungen ist Gl. (8). Sind nämlich bei bestimmten Gleichungen der Koeffizient B und die Wurzel aus der Diskriminante annähernd gleich groß, dann entsteht die dem Betrage nach kleinere Lösung aus der Subtraktion von zwei nahezu gleichen Zahlen. Eine der beiden Zahlen ist mit der Funktion SQR(X) errechnet worden und dadurch naturgemäß mit Rundungsfehlern behaftet. Die Subtraktion löscht sehr viele, vielleicht sogar alle geltenden Ziffern aus. Die Lösung mit dem kleineren Betrag wird dann ganz oder zum wesentlichen Teil aus den Rundungsfehlern gebildet – sie wird falsch.
Solchen Fehlerquellen soll bei der Programmierung vorgebeugt werden:
Nach dem Wurzelsatz von VIETA kann Gl. (1) auch in der Form

$$A\,(x - x_1)\,(x - x_2) = 0 \qquad (10)$$

geschrieben werden.
Multipliziert man Gl. (10) aus und vergleicht die so entstehenden Koeffizienten mit den Koeffizienten von Gl. (1), dann ergeben sich zwei Bedingungen für die Lösungen:

$$-A\,(x_1 + x_2) = B \qquad (11)$$

$$A\,x_1 \cdot x_2 = C. \qquad (12)$$

Die Genauigkeit der Lösung mit dem kleineren Betrag wird verbessert, wenn man zuerst nach Gl. (8) die Lösung mit dem größeren Betrag ermittelt und anschließend die Lösung mit dem kleineren Betrag aus Gl. (11) oder aus Gl. (12) berechnet. Verwendet man Gl. (12), so gilt:

$$x_{\min} = \frac{C}{A \cdot x_{\max}}. \qquad (13)$$

Eine ähnliche Strategie läßt sich auch für den Imaginärteil komplexer Lösungen angeben. Darauf wird jedoch an dieser Stelle verzichtet, weil die Gefahr, daß die Lösungen durch Rundungen falsch werden, bei komplexen Lösungen bedeutend geringer ist als bei reellen. Interessierten Lesern wird es nicht schwerfallen, das Programm entsprechend zu ergänzen.

3. Programmbeschreibung

Die Logik des Programms ist im Struktogramm leicht zu verfolgen. Nach der Eingabe von A, B und C, die beliebige reelle Zahlen sein dürfen, wird geprüft, ob eine quadratische Gleichung vorliegt und ob sie lösbar ist. Wie die Sonderfälle A, B, $C = 0$ behandelt werden, zeigt der linke Teil des Struktogramms.
Ist A von Null verschieden, dann liegt eine quadratische Gleichung im engeren Sinne vor.
Wird die Diskriminante Null, so gibt es eine doppelte Lösung. Ist die Dis-

kriminante kleiner als Null, dann hat die quadratische Gleichung zwei konjugiert komplexe Lösungen. Bei positiver Diskriminante wird die Lösung mit dem größeren Betrag nach Gl. (8) errechnet und die Lösung mit dem kleineren Betrag nach Gl. (13), wie der rechte Teil des Struktogramms zeigt.
Als Variablen werden verwendet:

A, B, C	Koeffizienten der quadratischen Gleichung
DI	Diskriminante
H1, . . ., H3	Hilfsvariablen; ihre Bedeutung geht aus dem Struktogramm hervor
IM	Wurzel aus der Diskriminante, Imaginärteil bei komplexen Lösungen
X1, X2	Lösungen der quadratischen Gleichung

Das Programm QUAGL wird mit RUN ENTER gestartet. Die Überschrift wird angezeigt, und die Koeffizienten sind einzugeben.
Hat die quadratische Gleichung beliebig viele Lösungen, keine Lösung, nur eine Lösung bei $A = 0$ oder eine doppelte Lösung, so erscheint die entsprechende Anzeige. Konjugiert komplexe Lösungen werden ebenfalls ausgegeben.

Programm QUAGL

```
10 REM     QUAGL
20 REM
QUADRATISCHE GLEICHUNGEN
30 PRINT
"LOESUNGEN DER GLEICHUNG"
40 PRINT "A*X^2 + B*X + C = 0"
50 PRINT "FUER": PRINT
60 INPUT "A = "; A
70 INPUT "B = "; B
80 INPUT "C = "; C: PRINT
90 IF A = 0 THEN 210
100 H3 = 2*A: DI = B*B-4*A*C:
IM = SQR(ABS(DI))
110 IF DI = 0 THEN 200
120 IF DI < 0 THEN 180
130 H1 = -B + IM: H2 = -B - IM
140 IF ABS(H1) > ABS(H2) THEN 160
150 X1 = 2*C/H2: X2 = H2/H3:
GOTO 170
160 X1 = H1/H3: X2 = 2*C/H1
170 PRINT "X1 = ";X1: PRINT
"X2 = ";X2: GOTO 250
180 PRINT "X1 = "; -B/H3;" + ";
IM/H3;"*I"
190 PRINT "X2 = "; -B/H3; " - ";
IM/H3; "*I": GOTO 250
200 PRINT "X1 = X2 = ";
-B/H3: GOTO 250
210 IF B = 0 THEN 230
220 PRINT "X = "; -C/B: GOTO 250
230 IF C = 0 THEN PRINT
"X IST BELIEBIG": GOTO 250
240 PRINT "KEINE LOESUNG"
250 END
```

4. Beispiele

Hier werden nur die Lösungen für solche Beispiele mitgeteilt, bei denen das Programm QUAGL 1 fehlerhafte Ergebnisse bringt.

Für $A = 1$, $B = 111111$ und $C = 111110$

liefert dieses Programm die exakten Lösungen

$X1 = -1$ und $X2 = -111110$.

Bei $A = -2$, $B = 4489{,}96$ und $C = 89{,}8$
erhalten wir ebenfalls die exakten Lösungen

$X1 = -.02$ und $X2 = 2245$.

Selbst bei $A = -2$, $B = -39999{,}9994$ und $C = 12$ ergeben sich mit

$X1 = -20000$ und $X2 = 3E{-}04$

noch exakte Lösungen.
Es ist also durchaus lohnend, vor der Programmierung eine Lösungsformel oder ein Verfahren etwas genauer zu untersuchen. Der größere Aufwand zahlt sich aus. Das Programm wird unempfindlicher gegen Fehler und ist vielseitiger zu verwenden.

Autor:
Doz. Dr.-Ing. Peter Fischer
Hochschule für Verkehrswesen Dresden

Zur Ermittlung der Extremwerte einer Funktion $y=f(x)$ mit Hilfe eines Kleincomputers

Aufbauend auf der geometrischen Interpretation der ersten und der zweiten Ableitung einer Funktion, werden in der Abiturstufe die Extremwerte einer Funktion $y = f(x)$ mit Hilfe der folgenden Lösungsschritte ermittelt:

- Bilden der ersten Ableitung

  $y' = f'(x)$;

- Lösen der Gleichung $f'(x) = 0$, die Lösungen seien

  $x_{E_1}, x_{E_2}, \ldots, x_{E_n}$;

- Bilden der zweiten Ableitung

  $y'' = f''(x)$;

- Untersuchung der zweiten Ableitung an den Stellen x_{E_i}, $(i = 1, 2, \ldots, n)$;

  ist dabei

  $f''(x_{E_i}) > 0$,

  dann besitzt die Funktion $y = f(x)$ an der Stelle $x = x_{E_i}$ ein *Minimum*, ist

  $f''(x_{E_i}) < 0$,

  dann besitzt sie an der Stelle $x = x_{E_i}$ ein *Maximum*, ist jedoch

  $f''(x_{E_i}) = 0$,

  so ist die Entscheidung darüber, ob ein Minimum oder ein Maximum vorliegt nur mit Hilfe der *höheren Ableitungen* von $y = f(x)$ möglich.

Da der Rechenaufwand bei der Lösung einer »Extremwertaufgabe« recht umfangreich werden kann, sind in den Schulbeispielen die Funktionen $y = f(x)$ meist so gewählt, daß die Bestimmungsgleichungen $f'(x) = 0$ zum Auffinden der möglichen Extremstellen mit elementaren Mitteln in geschlossener Form lösbar sind. Bei Aufgaben aus der Praxis, die auf Extremwertprobleme führen, ist dies leider meist nicht der Fall. Was liegt also näher, als den Computer zur Bewältigung der umfangreichen Rechenarbeiten heranzuziehen? Wollte man nun die vorn geschilderte Vorgehensweise in ein Computerprogramm überführen, so würde sich ein Programm ergeben, das eine Reihe von Nachteilen besitzt, von denen einige angeführt werden sollen.

Es müssen mindestens die erste und die zweite Ableitung der gegebenen Funktion $y = f(x)$ gebildet werden. Dies muß, solange nur ein Kleincomputer zur Verfügung steht, der Nutzer des Programms selbst tun. (Für größere Rechner gibt es bereits Spezialprogramme, die gegebene Funktionen formal differenzieren oder integrieren können.) Gegebenenfalls sind sogar noch eine Reihe von höheren Ableitungen zu bilden. Hier treten die ersten Fehlerquellen auf, denn jeder weiß, wie schnell sich *beim Differenzieren* einer Funktion *Fehler* einschleichen,

und je öfter die Ableitung gebildet werden muß, um so komplizierter werden die entstehenden Ausdrücke und um so mehr Fehler können gemacht werden. Dann müssen die Ableitungen der Funktion als BASIC-Anweisungen in das Programm übernommen werden. Je komplizierter die Funktion ist, um so mehr besteht die Gefahr, daß *Programmierfehler* gemacht werden. Dann liegt es nahe, die Funktionsgleichung sowie die Gleichungen der Ableitungen in Form von **DEF FNname-Anweisungen** in das BASIC-Programm zu übernehmen. Bei den Kleincomputern ist jedoch eine solche **DEF FNname-Anweisung** auf 72 Zeichen beschränkt. So kann es passieren, daß schon die zweite Ableitung der Funktion ein Ausdruck ist, der mehr als 72 Zeichen zu seiner Darstellung innerhalb des BASIC-Programms benötigt.

Um die genannten Fehlerquellen auf ein Minimum reduzieren zu können, liegt es nahe, nach einer Vorgehensweise zu suchen, bei der *allein die Gleichung der Funktion* $y = f(x)$ ausreicht, um die Extremstellen dieser Funktion ermitteln zu können.

Im folgenden soll ein BASIC-*Programm zur Bestimmung der relativen Extrempunkte einer Funktion* $y = f(x)$ in einem Intervall $[a; b]$ vorgestellt werden, bei dem keinerlei Kenntnisse aus der Differentialrechnung vorausgesetzt werden müssen. Diesem Programm liegt folgende Überlegung zugrunde:

Man unterteilt das zu untersuchende Gesamtintervall $[a; b]$ in eine hinreichend große Anzahl von Teilintervallen mit gleicher Intervallbreite h, so daß die Stützstellen

$$x_1 = a,\ x_2 = x_1 + h,\ x_3 = x_1 + 2 \cdot h, \ldots$$

$$x_{n+1} = x_1 + n \cdot h = b$$

entstehen. Dabei muß vorausgesetzt werden, daß innerhalb des Gesamtintervalls $[a; b]$ keine Unstetigkeitsstellen der Funktion $y = f(x)$ vorhanden sind.

Nun werden der Reihe nach jeweils zwei benachbarte Teilintervalle mit den Stützstellen x_i, x_{i+1} und x_{i+2} $(i = 1, 2, \ldots, n-1)$ betrachtet und untersucht, ob der mittlere der drei zugehörigen Funktionswerte y_{i+1} größer oder kleiner ist als die beiden Funktionswerte y_i und y_{i+2} am Rande des betrachteten Doppelintervalles, ob also gilt

$$y_{i+1} > y_i \quad \textit{und} \quad y_{i+1} > y_{i+2}$$

oder ob gilt

$$y_{i+1} < y_i \quad \textit{und} \quad y_{i+1} < y_{i+2}.$$

Ist dies der Fall, so liegt zwischen den beiden Stützstellen x_i und x_{i+2} ein Maximum bzw. ein Minimum der Funktion (Bild 1).

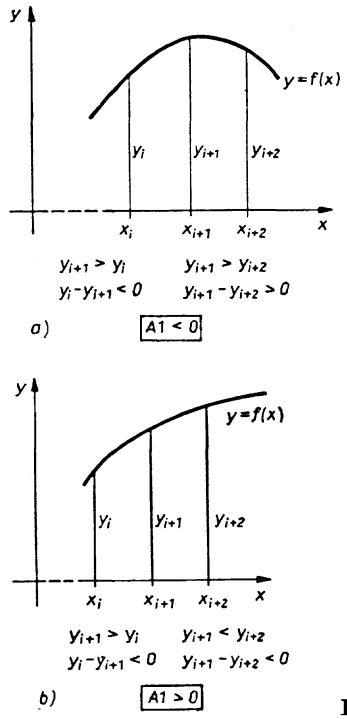
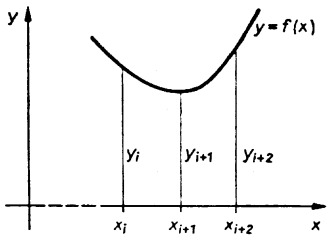
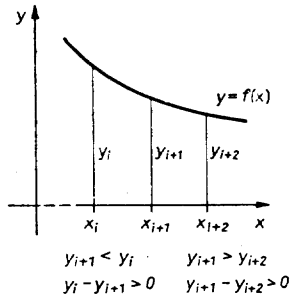


Bild 1





Zu Bild 1

Um die Lage des Extrempunktes genauer fixieren zu können, wird nun jedes der beiden Teilintervalle durch *Halbieren der Schrittweite h* in jeweils zwei neue Teilintervalle zerlegt, und für jedes dieser beiden Teilintervalle wird die oben angedeutete Untersuchung wiederholt. Dieses Eingrenzen in immer engere Teilgebiete wird so lange wiederholt, bis die Extremstelle mit der gewünschten Genauigkeit ermittelt worden ist.

Die Entscheidung darüber, ob der mittlere Funktionswert der jeweils betrachteten beiden Teilintervalle größer oder kleiner ist als die Funktionswerte, die zu den beiden Randpunkten gehören, trifft der Rechner dadurch, daß er das Produkt

$$A_1 = (y_i - y_{i+1}) \cdot (y_{i+1} - y_{i+2})$$

untersucht. Wie man sich nämlich leicht überzeugen kann, wird A_1 *negativ*, wenn

$y_{i+1} > y_i$ und $y_{i+1} > y_{i+2}$ bzw.

$y_{i+1} < y_i$ und $y_{i+1} < y_{i+2}$.

Hingegen wird in den beiden Fällen

$y_i < y_{i+1} < y_{i+2}$ bzw.

$y_i > y_{i+1} > y_{i+2}$

der Ausdruck A_1 *positiv*.

Das bisher geschilderte Verfahren zum Aufsuchen eines Extrempunktes funktioniert, wenn der Extrempunkt nicht zu nahe an der linken Intervallgrenze liegt (Bild 1). Das Verfahren kann jedoch auch positive Werte für A_1 liefern, obwohl im Bereich des Doppelstreifens ein Extremwert liegt. Dies ist dann der Fall, wenn der Extrempunkt sehr nahe an den Rand des Doppelstreifens heranrückt (Bild 2). Aus diesem Grunde wurde das Suchverfahren nach der Verkleinerung der Schrittweite h etwas verändert. Der zugehörige Algorithmus kann den Anweisungen 480 bis 700 des BASIC-Programms entnommen werden.

Schließlich seien noch einige Bemerkungen zu dem Problem angefügt, wie die Gleichung der zu untersuchenden Funktion in das Programm einzufügen ist. Es wurde von der Eigenschaft des BASIC-Interpreters Gebrauch ge-

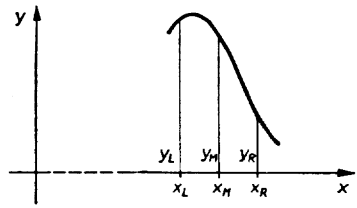


Bild 2

macht, daß Kommandos wie **EDIT xxx, STOP** und dgl. auch als Anweisungen in ein Programm übernommen werden können. Durch die beiden Anweisungen

220 EDIT 230

230 DEF FNZ(X) =

wird der Rechner veranlaßt, in der Anweisung 220 aus dem Programm-Modus in den EDIT-Modus überzugehen. Der Rechner gibt die Anweisung 230 aus, so daß dort die rechte Seite der zu untersuchenden Funktion ohne Schwierigkeiten eingefügt werden kann. Es muß nur noch dem Nutzer des Programms in geeigneter Form mitgeteilt werden, wie er aus dem Editier-Modus nunmehr wieder in den Abarbeitungsmodus zurückkommt. Dies ist in den vorangehenden Anweisungen 190 bis 210 bereits geschehen.
Das vorliegende Programm ist lauffähig auf dem Kleincomputer KC 85/1. Um es auch auf den Kleincomputern KC 85/2 bzw. KC 85/3 abarbeiten zu können, sind nur geringfügige Änderungen erforderlich. Es braucht nur in der Anweisung 204 das Wort STOP durch BREAK ersetzt zu werden, und es müssen in den Anweisungen 290 bis 300 sowie 730 und 800 die dort auftretenden Zeichenfolgen CHR$(160) und CHR$(161) jeweils durch CHR$(45) und CHR$(33) ersetzt werden.

Für die Abarbeitung des Programms müssen vom Nutzer bereitgestellt werden:

- Die explizite Darstellung der Funktionsgleichung in der Form $y = f(x)$,
- die Intervallgrenzen a und b für den Bereich, innerhalb dessen nach Extremwerten gesucht werden soll. (Dieser Bereich darf keine Punkte enthalten, die nicht zum Definitionsbereich der Funktion $y = f(x)$ gehören.)
- die Anfangsschrittweite h zur Unterteilung des Intervalls $[a, b]$ in Teilintervalle sowie
- eine Abbruchschranke ε als Maß für die Genauigkeit, mit der die Extrempunkte bestimmt werden sollen.

Diese Größen werden vom Rechner zu Beginn der Rechnung in einer nutzerfreundlichen Art angefordert.
Neben seinem eigentlichen Bestimmungszweck, der Ermittlung von Extremstellen einer Funktion, kann das Programm (s. S. 34) auch für verwandte Aufgabenstellungen genutzt werden, beispielsweise für die Ermittlung des Wertebereiches einer Funktion.

Autoren:
Oberlehrer Ingrid Kern
Lehrer im Hochschuldienst

Dr. paed. Marianne Kreul
Lektor
Technische Hochschule Zittau
Abteilung Mathematik

PROGRAMM ZUR ERMITTLUNG DER EXTREMSTELLEN EINER FUNKTION Y = F(X)

```
5 CLS : PRINT : PRINT
10 PRINT TAB(4)
"PROGRAMM ZUR
ERMITTLUNG DER"
20 PRINT TAB(4)
"RELATIVEN EXTREMWERTE"
30 PRINT TAB(4)
"UND DER RANDPUNKTE"
40 PRINT TAB(4)
"EINER FUNKTION Y=F(X)"
45 PRINT TAB(4)
"FUER ALLE A< =X< =B"
50 PRINT TAB(4) "————————————"
60 PRINT: PRINT
70 PRINT TAB(6)
"A, B INTERVALLGRENZEN"
80 PRINT TAB(10)
"SIE SIND SO ZU WAEHLEN,"
90 PRINT TAB(10)
"DASZ SIE IM DEFINITIONS-"
100 PRINT TAB(10)
"BEREICH VON Y=F(X) LIEGEN."
110 PRINT
120 PRINT TAB(6)
"H  SCHRITTWEITE"
130 PRINT TAB(10)
"FUER DIE SCHRITTWEITE IST"
140 PRINT TAB(10)
"ZU EMPFEHLEN:" : PRINT
150 PRINT TAB(16)
"H=(B--A)/100" : PRINT
160 PRINT TAB(6)
"EPS ABBRUCHSCHRANKE"
170 PRINT AT(23,25);
"WEITER MIT< CONT >"
180 PAUSE : CLS : PRINT : PRINT
190 PRINT "BRINGEN SIE IHRE
FUNKTIONSGLEICHUNG"
191 PRINT "AUF DIE FORM Y=F(X)."
192 PRINT "ERSETZEN SIE DIE
RECHTE SEITE"
193 PRINT "DER UNTEN STEHENDEN
GLEICHUNG"
194 PRINT "DURCH F(X),"
195 PRINT : PRINT
200 PRINT "DRUEKEN SIE AN-
SCHLIESZEND IN DER"
201 PRINT
"ANGEGEBENEN REIHENFOLGE"
202 PRINT
203 PRINT TAB(6)
"— DIE <ENTER>-TASTE"
204 PRINT TAB(6)
"— DIE <STOP>-TASTE"
205 PRINT TAB(6)
"— DIE ZEICHENFOLGE GOTO 230"
206 PRINT TAB(6)
"— DIE <ENTER>-TASTE!"
210 PRINT: PRINT
220 EDIT 230
230 DEF FNZ(X)=
234 CLS
235 PRINT
236 PRINT "GEBEN SIE DIE UNTERE
INTERVALLGRENZE"
238 PRINT "EIN"
239 PRINT
240 INPUT "      A=";A
241 PRINT
245 PRINT "GEBEN SIE DIE OBERE
INTERVALLGRENZE"
246 PRINT "EIN"
250 INPUT "      B=";B
251 PRINT
255 PRINT "GEBEN SIE DIE
SCHRITTWEITE EIN"
256 PRINT
260 INPUT "      H=";H
261 PRINT
265 PRINT "GEBEN SIE IHRE
ABBRUCHSCHRANKE EIN"
266 PRINT
270 INPUT "      EPS=";EPS
280 CLS:PRINT:PRINT
290 PRINT TAB(4) "X";
TAB(12)CHR$(161); TAB(16) "Y";
TAB(27)CHR$(161);
291 PRINT TAB(30) "EXTREMA"
295 PRINT STRING$(12,CHR$(160));
CHR$(161); STRING$(14,CHR$(160));
297 PRINT CHR$(161);
STRING$(10,CHR$(160))
300 PRINT A; TAB(12)CHR$(161);
TAB(14)FNZ(A); TAB(27)CHR$(161)
310 X1=A
320 X2=X1+H
330 X3=X2+H
340 IF X3>B THEN 790
350 Y1=FNZ(X1)
360 Y2=FNZ(X2)
370 Y3=FNZ(X3)
380 AL=Y1—Y2
```

```
390 AR=Y2—Y3
400 A1=AL*AR
410 IF A1<=0 THEN 450
420 X1=X2
430 Y1=Y2
440 GOTO 320
450 H1=H
460 H1=0.5*H1
470 IF H1 0.5*EPS THEN 710
480 XL=X1
490 XM=X1+H1
500 XR=XM+H1
510 YL=FNZ(XL)
520 YM=FNZ(XM)
530 YR=FNZ(XR)
540 XE=XM
550 AL=YL—YM
560 AR=YM—YR
570 A1=AL*AR
580 XU=XL+2*H1
590 XN=XM+2*H1
600 XO=XR+2*H1
610 YU=FNZ(XU)
620 YN=FNZ(XN)
630 YO=FNZ(XO)
640 XE=XN
650 AU=YU—YN
660 AO=YN—YO
670 A2=AU*AO
680 IF A1<A2 THEN
XE=XM:GOTO 460: ELSE XE=XN
690 X1=XU
700 GOTO 460
710 YE=FNZ(XE)
720 YF=FNZ(XE+H)
730 PRINT XE; TAB(12)CHR$(161);
TAB(14)YE; TAB(27)CHR$(161);
740 IF YE<YF THEN 745: ELSE 760
745 PRINT TAB(30) "MINIMUM"
750 GOTO 770
760 PRINT TAB(30) "MAXIMUM"
770 X1=XO
780 GOTO 320
790 YB=FNZ(B)
800 PRINT B; TAB(12)CHR$(161);
TAB(14)YB; TAB(27)CHR$(161);
810 PRINT:PRINT:PRINT
820 PRINT "MOECHTEN SIE EIN
ANDERES INTERVALL"
830 PRINT "AUF EXTREMWERTE
UNTERSUCHEN? (J/N)"
840 INPUT Z$
850 IF Z$="J" THEN GOTO 234
860 PRINT
870 PRINT "MOECHTEN SIE EINE
ANDERE FUNKTION"
880 PRINT "AUF EXTREMWERTE
UNTERSUCHEN? (J/N)"
890 INPUT Z$
900 IF Z$="J" THEN GOTO 185
910 CLS
920 PRINT AT(15,10);
"ENDE IHRER RECHNUNG"
930 END
```

Zur Ermittlung der exakten Lösungen algebraischer Gleichungen auf einem Rechner

1. Einleitung

Eine Gleichung der Form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0,$$

($a_n \neq 0$; $a_0, \ldots, a_n$ reell) heißt algebraische Gleichung n-ten Grades. Sie besitzt nach dem Hauptsatz der Algebra genau n Lösungen (oder Wurzeln), wenn man komplexe Lösungen mit berücksichtigt und mehrfache Lösungen entsprechend ihrer Vielfachheit zählt. Erstmalig gab Luca Pacioli (1445 bis 1517) für lineare und quadratische Gleichungen Lösungsformeln an. Uns allen ist die Darstellung für die beiden Lösungen x_1 und x_2 der quadratischen Gleichung

$$a_2 x^2 + a_1 x + a_0 = 0,$$

nämlich

$$x_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4 a_2 a_0}}{2 a_2} \qquad (1)$$

bekannt.

Ähnliche Formeln, die auch aus endlich vielen Anwendungen der rationalen Rechenoperationen und des Wurzelziehens bestehen, wurden von Nicolo Tartaglia (1500 bis 1577) und Girolamo Cardano (1501 bis 1577) für Gleichungen 3. Grades sowie von Ludovico Ferrari (1522 bis 1576) für Gleichungen 4. Grades angegeben. Die Suche nach Auflösungsformeln für Gleichungen höheren als vierten Grades beschäftigte Jahrhunderte die Mathematiker. Erst Niels Henrik Abel (1802 bis 1829) zeigte 1826, daß es für allgemeine Gleichungen höheren als vierten Grades solche Formeln nicht geben kann.

Trotz der Kenntnis von Auflösungsformeln werden in der Praxis meist Lösungen von algebraischen Gleichungen dritten oder vierten Grades über Näherungsverfahren (z.B. Newton-Verfahren oder Regula falsi) bestimmt. Diese Vorgehensweise ist aufwendig – es kann jeweils nur eine reelle Lösung iterativ ermittelt werden – und sie widerspricht der modernen Auffassung, so lange wie möglich (evtl. unter Benutzung von Formelmanipulationssprachen) analytisch zu rechnen. Erst wenn das analytische Vorgehen versagt, sollten Näherungsverfahren der Numerik zum Einsatz gebracht werden.

Aus diesem Grunde stellen wir in dieser Arbeit ein BASIC-Programm zur analytischen Bestimmung der Lösungen (einschließlich der komplexen Lösungen) algebraischer Gleichungen bis 4. Grades vor.

2. Analytische Nullstellenberechnung für Polynome 3. und 4. Grades

Es sei die folgende Polynomgleichung gegeben:

$$x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0. \quad (2)$$

Unser Ziel ist es, die linke Seite dieser Gleichung in ein Produkt mit zwei quadratischen Termen zu zerlegen:

$$(x^2 + h_1x + h_2)\,(x^2 + h_3x + h_4) = 0. \quad (3)$$

Von dieser Gleichung sind leicht die Nullstellen zu bestimmen, denn man braucht nur die aus (3) folgenden Gleichungen (4) und (5) einzeln mit Formel (1) zu lösen:

$$x^2 + h_1x + h_2 = 0, \quad (4)$$

$$x^2 + h_3x + h_4 = 0. \quad (5)$$

Da das Ausmultiplizieren der Faktoren der linken Seite von Gleichung (3) die linke Seite der Gleichung (2) liefern muß, gilt:

$$h_1 + h_3 = a_3,$$

$$h_2 + h_4 + h_1h_3 = a_2,$$

$$h_1h_4 + h_3h_2 = a_1,$$

$$h_2h_4 = a_0.$$

Jetzt erweist es sich als günstig, wenn man

$$h_1 = \frac{a_3}{2} + 2f, \qquad h_2 = \frac{a_2}{6} + \frac{x}{2} + g,$$

$$h_3 = \frac{a_3}{2} - 2f \text{ und } h_4 = \frac{a_2}{6} + \frac{x}{2} - g$$

ansetzt, denn dann ist bereits

$$h_1 + h_3 = a_3$$

erfüllt.
Aus der Beziehung

$$h_2 + h_4 + h_1h_3 = \frac{a_2}{3} + x + \frac{a_3^2}{4} - 4f^2 = a_2$$

folgt

$$f^2 = \frac{a_3^2}{16} + \frac{x}{4} - \frac{a_2}{6}, \quad (6)$$

und aus

$$h_1h_4 + h_2h_3 = a_3\left(\frac{a_2}{6} + \frac{x}{2}\right) - 4gf = a_1$$

folgt

$$gf = \frac{a_3}{8}\left(\frac{a_2}{3} + x\right) - \frac{a_1}{4}. \quad (7)$$

Schließlich erhält man aus

$$h_2h_4 = \frac{1}{4}\left(\frac{a_2}{3} + x\right)^2 - g^2 = a_0$$

die Gleichung

$$g^2 = \frac{1}{4}\left(\frac{a_2}{3} + x\right)^2 - a_0. \quad (8)$$

Weil f und g reelle Zahlen sind, gilt für sie die folgende bekannte Beziehung:

$$f^2 \cdot g^2 - (f \cdot g)^2 = 0. \quad (9)$$

Durch Einsetzen von (6), (7) und (8) in (9) erhält man

$$\left[\frac{a_3}{16} + \frac{x}{4} - \frac{a_2}{6}\right]\left[\frac{1}{4}\left(\frac{a_2}{3} + x\right)^2 - a_0\right]$$

$$-\left[\frac{a_3}{8}\left(\frac{a_2}{3} + x\right) - \frac{a_1}{4}\right]^2 = 0.$$

Das Ausmultiplizieren der Faktoren der linken Seite dieser Gleichung liefert eine kubische Gleichung für x, die *reduzierte kubische Resolvente:*

$$x^3 + px + q = 0$$

mit

$$p = a_3a_1 - 4a_0 - \frac{a_2^2}{3}$$

und

$$q = \frac{8}{3}a_2a_0 + \frac{a_3a_2a_1}{3} - a_1^2 - a_3^2a_0 - \frac{2}{27}a_2^3.$$

Diese Resolvente hat mindestens eine reelle Lösung. Hat sie drei, so benutzt man diejenige, für die

$$g^2 = \frac{1}{4}\left(\frac{a_2}{3} + x\right)^2 - a_0$$

und

$$f^2 = \frac{a_3^2}{16} + \frac{x}{4} - \frac{a_2}{6}$$

positiv werden, denn nur dann erhält man für f und g reelle Werte. (Es gibt wenigstens ein solches x.) Den Werten von f und g erteilt man solche Vorzeichen, daß

$$\text{sign } g \cdot \text{sign } f = \text{sign } (f\,g)$$

gilt.
Die Lösung der kubischen Resolvente $x^3 + px + q = 0$ erhält man durch die sogenannten CARDANIschen Formeln. Es wird

$$t = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2$$

gesetzt.
Für $t \geq 0$ lassen sich dann die reellen und komplexen Nullstellen der Resolvente wie folgt berechnen:

$$x_1 = u + v,$$

$$x_2 = -\frac{(u+v)}{2} + \frac{(u-v)}{2}\sqrt{3}\,\mathrm{i},$$

$$x_3 = -\frac{(u+v)}{2} - \frac{(u-v)}{2}\sqrt{3}\,\mathrm{i},$$

wobei

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{t}}$$

und

$$v = \sqrt[3]{-\frac{q}{2} - \sqrt{t}}\,.$$

Für $t < 0$ gilt (*Casus irreducibilis*):

$$x_1 = 2\sqrt[3]{r}\,\cos\left(\frac{\varphi}{3}\right),$$

$$x_2 = 2\sqrt[3]{r}\,\cos\left(\frac{\varphi}{3} + \frac{2\pi}{3}\right),$$

$$x_3 = 2\sqrt[3]{r}\,\cos\left(\frac{\varphi}{3} + \frac{4\pi}{3}\right),$$

wobei

$$\varphi = \arccos\left(-\frac{q}{2r}\right)$$

und

$$r = \sqrt{\frac{-p^3}{27}}$$

gesetzt wurde.
Wie man sieht, hat die kubische Resolvente für $t < 0$ drei reelle Lösungen.
Ist die kubische Gleichung in der Form

$$x^3 + a_2\,x^2 + a_1\,x + a_0 = 0$$

gegeben, so kann man sie durch die Transformation

$$y = x + \frac{a_2}{3}$$

in die Form

$$y^3 + \bar{p}y + \bar{q} = 0$$

überführen. Nach dem Lösen erfolgt die Rücktransformation der Lösungen gemäß:

$$x_k = y_k - \frac{a_2}{3}, \quad k = 1, 2, 3.$$

Es wird nun ein BASIC-Programm zur Berechnung der reellen Polynom-Nullstellen bis höchstens zum 4. Grade (Version KC 85/2/3) angegeben.

```
100 CLS
105 CLEAR
110 PRINT:PRINT:PRINT
115 PRINT "     AUFLOESUNG DER GLEICHUNG"
120 PRINT
125 PRINT "     A*X^4+B*X^3+C*X^2+D*X+E=0"
130 PRINT
135 INPUT "     A=?";E1
140 INPUT "     B=?";E2
145 INPUT "     C=?";E3
150 INPUT "     D=?";E4
155 INPUT "     E=?";E5
160 PRINT
210 IN=1
```

```
215 IF ABS(E1)<1E—6 THEN GOTO 265
220 A=1
225 B=E2/E1
230 C=E3/E1
235 D=E4/E1
240 E=E5/E1
245 Z=0
250 P=B*D—4*E—C*C/3
255 Q=(8*C*E+B*C*D)/3—D*D—B*B*E—2*C*C*C/27
260 GOTO 375
265 IF ABS(E2)<1E—6 THEN GOTO 310
270 A=1
275 B=E3/E2
280 C=E4/E2
285 D=E5/E2
290 Z=1
295 P=C—B*B/3
300 Q=2*B*B*B/27—B*C/3+D
305 GOTO 375
310 IF ABS(E3)<1E—6 THEN GOTO 330
315 B1=E4/E3
320 B2=E5/E3
325 GOTO 690
330 IF ABS(E4)<1E—6 THEN GOTO 350
335 NI=—E5/E4
340 GOSUB 790
345 GOTO 775
350 IF ABS(E5)<1E—6 THEN GOTO 365
360 GOTO 775
365 PRINT"     ALLE KOEFFIZIENTEN SIND NULL!"
370 GOTO 780
375 O5=1/3
380 O1=Q/2
385 O2=P/3
390 DI= O1*O1+O2*O2*O2
395 IF DI<—1E—5 THEN GOTO 455
400 O3=—O1+SQR(ABS(DI))
405 O4=—O1—SQR(ABS(DI))
410 U=SGN(O3)*(ABS(O3))^O5
415 V=SGN(O4)*(ABS(O4))^O5
420 R1=U+V
425 I1=0
430 R2=—(U+V)/2
435 I2=(U—V)*SQR(3)/2
440 R3=R2
445 I3=—I2
450 GOTO 520
455 R=SQR((—P)*(—P)*(—P)/27)
456 HY=—Q/(2*R)
457 IF HY>=1 THEN FI=0: GOTO 465
458 IF HY<=—1 THEN FI=PI: GOTO 465
460 FI=—TAN(HY/SQR(1—HY*HY))+PI/2
465 R1=2*COS(FI/3)*R^O5
```

```
470 I1=0
475 R2=2*COS((FI+2*PI)/3)*R^05
480 I2=0
485 R3=2*COS((FI+4*PI)/3)*R^05
490 I3=0
495 GOTO 520
500 R1=R1—B/3
505 R2=R2—B/3
510 R3=R3—B/3
515 GOTO 730
520 IF Z=1 THEN GOTO 500
525 N1=B*B/16+R1/4—C/6
530 M1=(C/3+R1)*(C/3+R1)/4—E
535 IF ABS(I1)<1E—5 AND ABS(I2)<1E—5 AND ABS(I3)<1E—5 THEN GOTO 537
536 GOTO 540
537 IF N1<—1E—5 OR M1<—1E—5 THEN GOTO 560
540 NQ=N1
545 MQ=M1
550 RW=R1
555 GOTO 610
560 N2=B*B/16+R2/4—C/6
565 M2=(C/3+R2)*(C/3+R2)/4—E
570 IF N2<—1E—5 OR M2<—1E—5 THEN GOTO 595
575 NQ=N2
580 MQ=M2
585 RW=R2
590 GOTO 610
595 NQ=B*B/16+R3/4—C/6
600 MQ=(C/3+R3)*(C/3+R3)/4—E
605 RW=R3
610 MN=B/8*(C/3+RW)—D/4
615 WN=SQR(ABS(NQ))
620 WM=SQR(ABS(MQ))
625 IF MN>—1E—5 THEN GOTO 635
630 WN=—WN
635 A1=B/2+2*WN
640 A2=C/6+RW/2+WM
645 B1=B/2—2*WN
650 B2=C/6+RW/2—WM
655 H1=—A1/2
660 BI=H1*H1—A2
665 IF BI<—1E—2 THEN GOTO 690
670 NI=H1+SQR(ABS(BI))
675 GOSUB 790
680 NI=H1—SQR(ABS(BI))
685 GOSUB 790
690 H1=—B1/2
695 BI=H1*H1—B2
700 IF BI<—1E—2 THEN GOTO 775
705 NI=H1+SQR(ABS(BI))
710 GOSUB 790
715 NI=H1—SQR(ABS(BI))
720 GOSUB 790
```

```
725 GOTO 775
730 IF ABS(I1)>1E—2 THEN GOTO 745
735 NI=R1
740 GOSUB 790
745 IF ABS(I2)>1E—2 THEN GOTO 760
750 NI=R2
755 GOSUB 790
760 IF ABS(I3)>1E—2 THEN GOTO 775
765 NI=R3
770 GOSUB 790
775 IF IN=1 THEN PRINT "     ES EXISTIERT KEINE LOESUNG"
780 STOP
790 PRINT "     X"; IN;"=";NI
795 IN=IN+1
800 RETURN
```

Interessieren auch die komplexen Lösungen, so sind folgende Zeilen zu überschreiben bzw. einzufügen.

```
665 IF BI<—1E—2 THEN GOTO 686
684 GOSUB 790
685 GOTO 690
686 PRINT "X";IN;"=";H1;"+I*";SQR(ABS(BI))
687 IN=IN+1
688 PRINT "X";IN;"=";H1;"+I*";SQR(ABS(BI))
689 IN=IN+1
700 IF BI<—1E—2 THEN GOTO 777
730 IF ABS(I1)>1E—2 THEN GOTO 741
739 GOSUB 790
740 GOTO 745
741 PRINT "X";IN;"=";R1;"+I*";I1
742 IN=IN+1
745 IF ABS(I2)>1E—2 THEN GOTO 756
754 GOSUB 790
755 GOTO 760
756 PRINT "X";IN;"=";R2;"+I*";I2
757 IN=IN+1
760 IF ABS(I3)>1E—2 THEN GOTO 771
769 GOSUB 790
770 GOTO 775
771 PRINT "X";IN;"=";R3;"+I*";I3
772 IN=IN+1
776 GOTO 780
777 PRINT "X";IN;"=";H1;"+I*";SQR(ABS(BI))
778 IN=IN+1
779 PRINT "X";IN;"=";H1;"+I*";—SQR(ABS(BI))
790 PRINT "X";IN;"=";NI
```

Das Programm ist wie üblich mit RUN zu starten. Danach wird ausgedruckt

AUFLOESUNG DER GLEICHUNG

A*X∧4 + B*X∧3 + C*X∧2 +

D*X + E = 0

A = ?

Es muß dann der Koeffizient vor dem Term x^4 eingegeben werden. Dasselbe geschieht mit den Koeffizienten B bis E. Ist der Koeffizient E eingegeben, so erfolgt die Nullstellenberechnung, die 1 bis 2 Sekunden in Anspruch nimmt, und es erscheinen die reellen bzw. komplexen Nullstellen auf dem Bildschirm. Erscheinen keine Nullstellen, so wird eine entsprechende Meldung ausgegeben. Für jeden der Koeffizienten A bis E ist auch der Eingabewert Null zugelassen. Deshalb kann man Polynomnullstellen vom 1. bis zum 4. Grade berechnen.
Die Struktur des Programms wird durch folgende REM-Anweisungen sichtbar gemacht:

```
95 REM EINGABE *
165 REM *
200 REM DIFFERENZIERUNG DER
POLYNOME U. RESOLVENTEN-
BESTIMMUNG **
372 REM **
374 REM CARDANISCHE
FORMELN ***
497 REM ***
517 REM BESTIMMUNG DER
KOEFFIZIENTEN IN GLEICHUNG(3)*
652 REM *
654 REM AUSDRUCK **
805 REM **
```

3. Testergebnisse

Mit diesem Programm wurde eine Zuverlässigkeitsanalyse durchgeführt, bei der 500 Polynome untersucht wurden. Die Koeffizienten A bis E wurden mittels Zufallszahlengenerators zwischen -10^5 und 10^5 ausgewählt. Der maximale absolute Fehler betrug 10^{-3}. Es ist aber trotzdem ratsam, die Ergebnisse wie bei jeder numerischen Berechnung kritisch zu betrachten und durch Einsetzen zu überprüfen. Es gibt sogenannte *schlechtkonditionierte Polynome*, die sehr empfindlich auf kleine Fehler in den Nullstellen reagieren.

Es werden nun einige typische Beispiele behandelt:

Koeffizienten A ··· E	Exakte reelle Nullstellen	Rechnerlösung
1	81	81
—29	—51	—51
—4160	/	/
—4161	/	/
—4131	/	/
1	—372	—371,999
21	352	352,001
—130923	/	/
—130924	/	/
—130944	/	/
1	31	30,9999
53	53	53,0001
—5203	—63	—63,0002
—166517	—74	—73,9998
7659666	/	/
1	0	0
1	0	0
1	/	/
0	/	/
0	/	/
0	4	4
1	6	6
—16	6	6
84	/	/
—144	/	/
1	/	/
0	/	/
2	/	/
0	/	/
1	/	/
0	—1	—1
1	/	/
1	/	/
1	/	/
1	/	/

Koeffizienten A ··· E	reelle und komplexe Nullstellen	Rechnerlösung
1	—0,5+I∗0,866025	—0,5+I∗0,866225
0	—0,5+I∗—0,866025	—0,5+I∗—0,866225
1	0,5+I∗0,866025	0,5+I∗0,865826
0	0,5+I∗—0,866025	0,5+I∗—0,865826
1	/	/
80	—0,16875+I∗ 1,04116	—0,16875+I∗ 1,04116
—3893	—0,16875+I∗—1,04116	—0,16875+I∗—1,04116
3566	47,7433	47,7433
—2741	1,25672	1,25674
5340	/	/
0	0+I∗5,38517	—1,93715E—7+I∗5,38517
—51	0+I∗5,38517	—1,93715E—7+I∗5,38517
—21	—0,411765	—0,411764
—1479	/	/
—609	/	/
14	—0,535714+I∗1,03448	—0,535714+I∗ 1,03448
127	—0,535714+I∗—1,03448	—0,535714+I∗—1,03448
139	0	0
152	—8	—8
0	/	/
64	—0,0625+I∗ 1,05141	—0,0625+I∗ 1,05142
—2104	—0,0625+I∗—1,05141	—0,0625+I∗—1,05142
—5313	35,2683	35,2683
—2983	—2,26833	—2,26833
—5680	/	/
0	—4+I∗ 1	—4+I∗1
12	—4+I∗—1	—4+I∗—1
111	—1,25	—1,25
324	/	/
255	/	/

Koeffizienten A ··· E	Exakte reelle Nullstellen	Rechnerlösung
1	—5	—5
—5	8	8
—34	0	4,76837 E—7
80	2	2
0	/	/
1	28	28
—20	—7	—7
—216	/	/
—217	/	/
—196	/	/

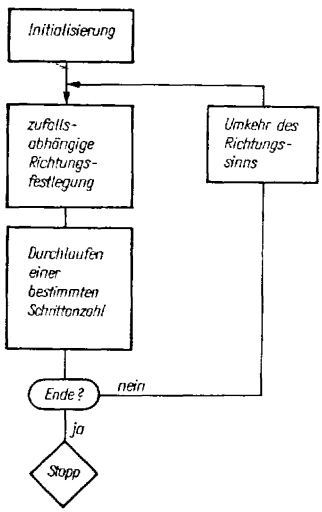
Autoren:
Prof. Dr. rer. nat. habil.
Dieter Oelschlägel

Cand. math. Wolfram Grochow
Technische Hochschule »Carl Schorlemmer«
Leuna-Merseburg
Sektion Mathematik

Zwei Bildschirmspiele mit Zick-Zack-Bewegungen

1. Allgemeines

Mit dem Erscheinen von Heimcomputern werden zunehmend viele Nutzer danach trachten, auch selbst Geschicklichkeits- und Trainingsspiele zu erfinden und zu programmieren. Schon die Anfängersprache BASIC bietet hierfür eine reiche Vielfalt.
Als ein belebendes Element für solche Zwecke soll hier die Nutzung von Zick-Zack-Bewegungen an zwei Beispielen vorgeführt werden. Bereits die Schleifen-Anweisung FOR... NEXT und der Zufallszahlengenerator RND reichen als Skelett für eine solche abwechslungsreiche Bewegung aus, selbst wenn nur bescheidene Programmierungsmöglichkeiten für den Bildschirm vorhanden sind. Das Prinzip läßt sich durch folgendes Schema veranschaulichen (Bild 1).



Zur Programmierung genügen einige BASIC-Zeilen (mit Kleinbuchstaben bezeichnete Variablen sind durch konkrete Zahlen bzw. Zeichen zu ersetzen):

```
1Ø T=t0:B=b0: Z=zb
2Ø FOR I=1 TO za
3Ø V=RND*Z
4Ø FOR J=1 TO z1
5Ø T=T+h:B=B+V
6Ø REM weitere programmzeilen
7Ø PRINT AT(T,B); zei
8Ø NEXT J
9Ø Z==—Z
1ØØ NEXT I
```

Hierbei bestimmen t_0 und b_0 in der Zeile 1Ø die Anfangsposition des bewegten Objektes, z_b im Betrag die maximale Zick-Zack-Weite und im Vorzeichen die Richtung. Mit der Angabe z_a wird die Anzahl der Zick-Zack-Strecken programmiert. Ihre Richtung wird mit V in der Zeile 3Ø zufallsabhängig errechnet, in der Zeile 9Ø dann nach der jeweils anderen Seite orientiert. Die Größe z_l in Zeile 4Ø bestimmt die Länge der einzelnen Zick-Zack-Strecken. In Zeile 5Ø bedeutet die Größe h den Vorwärtsschritt. Sein zeitlicher Vollzug hängt von den unter

Zeile 60 angedeuteten Anweisungen und der Rechengeschwindigkeit des Computers ab. Die Zeile 70 deutet die Ausgabe mit den Möglichkeiten des KC 85/1 an: Das mit *zei* anzugebende Zeichen für das bewegte Objekt wird an der Bildschirmposition dargestellt, die von T für die Zeile und von B für die Spalte bestimmt wird. Allerdings geben die Beschränkungen auf die Zeilen 0 ··· 23 und die Spalten 0 ··· 39 bei diesem Rechner keine besonders großen Bereiche. Es empfiehlt sich auf jeden Fall, eine Randbegrenzung zu bedenken und unter Zeile 60 zu programmieren, wenn man z_a und z_l nicht zu klein wählen will und auch die maximale Zick-Zack-Breite z_b noch für mittlere Zufallszahlen RND (zwischen Null und eins) einen sichtbaren Zick-Zack-Effekt ergeben soll.

In den folgenden beiden konkreten Spielformen wird dieses Grundschema angewendet. Sie wurden beide am KC 85/1 entwickelt und mit mehreren Varianten unter Einbezug zahlreicher Anwender aus verschiedenen Altersgruppen getestet. Während im ersten Beispiel der Spieler die zeitliche Entwicklung selbst bestimmt, ist sie im zweiten Beispiel fest programmiert. Damit sind gleichzeitig die unterschiedlichen Eingabemöglichkeiten INPUT und INKEY$ demonstriert. Das zu steuernde Objekt B in der Darstellung zei ist einmal ein »Wildschwein« W in der Darstellung CHR$(201) und zum zweiten ein »Hase« H in der gleichen Darstellung CHR$(201).

2. Eine harmlose Wildschweinjagd

Der volle BASIC-Text dieses Programms lautet:

```
10 CLS:?:?:?"BILDSCHIRM-
SPIEL":?:?:?
20 ?" 'Die harmlose Wildschweinjagd' ":?
30 ?:?:?:?"Machen Sie mit? (JA)":?:?
40 INPUT J$:CLS
50 ?"Beim Betreten einer Wiese"
60 ?"zwischen Wald und Fluss erblicken"
70 ?"Sie ein Wildschwein.
Es bewegt sich"
80 ?"in einem Dreier-Zick-Zack abwaerts."
90 ?"Sie koennen bei gleicher Abwaerts-"
100 ?"Geschwindigkeit Ihre seitliche"
110 ?"Beschleunigung waehlen durch
Eingabe"
120 ?"von A (in Bildschirmpositionen pro"
130 ?"Abwaertsschritt im Quadrat;
die Wiese"
140 ?"ist 35 Positionen breit).":?
150 ?"Versuchen Sie (Zeichen *), das"
160 ?"Wildschwein
(Zeichen ";CHR$ (201);") zu erjagen"
170 ?"und moeglichst lange waehrend
der 21"
180 ?"Schritte bis zum unteren Rand auf"
190 ?"dem Tier zu reiten!":?
200 ?"Sie sehen die Ausgangs-Situation:"
210 ?"Sind Sie bereit? (JA)":INPUT J$
220 CLS:K=0:E=0:A0=0
230 FOR I=0 TO 21:?AT(I,5);
CHR$ (159):?AT (I,39);
CHR$ (129):NEXT I
240 Z1$="Sie liegen im Fluss!":Z4$="
links!"
250 Z2$="Mit neuem A nach":Z3$="
rechts!"
260 Z5$="Sie haengen im Waldrand!"
270 Z6$="                    "
280 T=0:S=15:W=28:V=0:Z=-4:?"
A"
290 ?AT(T,S);"*":?AT(T,W);
CHR$ (201)
300 FOR I=1 TO 7
310 VW=RND (1)*Z
320 FOR J=1 TO 3
330 INPUT A:S=S+V+(A+2*A0)/
6:E=E+ABS(A):?AT(22,8);
Z6$:?AT(23,8);Z6$
340 IF S<5 THEN ?AT(22,8);
Z1$:?AT(23,8);
Z2$+Z3$:S=5:V=0:A0=0:A=0
350 IF S>39 THEN ?AT(22,8);
Z5$:?AT(23,8);
Z2$+Z4$:S=39:V=0:A0=0:A=0
360 T=T+1:W=W+VW
370 IF W<5 THEN W=5
380 IF W>39 THEN W=39
390 IF INT(S)=INT(W) THEN
```

```
BEEP:K=K+1:?AT(T,S);
CHR$(196):GOTO 410
400 ?AT(T,S);"*":?AT(T,W);
CHR$(201)
410 V=V+0.5*(A+A0):A0=A
420 NEXT J:Z=-Z
430 NEXT I
440 ?AT(22,5);"Sie sassen",K,
"Schritte obenauf."
450 ?AT(23,5);"Ihr Aufwand betrug",E,
"A-Einheiten."
460 END
```

Wer es auf einem KC 85/1 nachvollziehen will, wird dafür die entsprechende BASIC-Version kennen oder die Beschreibung heranziehen. Für andere Leser kann sicherlich zunächst zur Orientierung genügen, daß mit dem Befehl CLS der Bildschirm gelöscht wird, daß das Fragezeichen die Abkürzung für den PRINT-Befehl ist und der Doppelpunkt verschiedene Anweisungen in einer Zeile trennt. Ein INPUT-Befehl löst Warten auf Betätigung entsprechender Eingabetasten durch den Nutzer aus.

Zur Textdarstellung auf dem Bildschirm muß der Programmierer nicht nur an die Erzeugung, sondern auch an das Wieder-Verschwinden denken. Wenn das mit CLS programmiert werden soll, muß vorher genügend Zeit zum Betrachten sein. Im vorliegenden Fall wurde dazu die Frage der Zeile 30 und die INPUT-Anweisung der Zeile 40 geschrieben. Damit bestimmt jeweils der Spieler selbst diese Zeit. Die gleiche Steuerung ist nach der Beschreibung der Spielregeln in Zeile 210 programmiert. Die Zeile 230 besorgt eine sichtbare Begrenzung des Spielfeldes durch »Fluß« und »Wald«. Zwischen den Zeilen 300 und 430 ist die Abwärtsbewegung des »Wildschweins« W in 7 Zügen zu je 3 Schritten ausgeführt. Bei der gewählten Größe $Z = z_b = -4$ könnte W in den beiden denkbaren Extremfällen aus seiner Anfangsposition heraus um $4 \cdot 3 \cdot 4 = 48$ Positionen nach links oder um $3 \cdot 3 \cdot 4 = 36$ Positionen nach rechts laufen. Man muß also eine zufallsabhängige Überschreitung des Bereichs von höchstens 40 Positionen insgesamt gewärtigen. Wenn auch noch etwas Platz für Eingabe-Informationen gebraucht wird und eine Koordinierung mit der Position des Spielers notwendig ist, scheint nach den bisherigen Tests eine Ausgangsposition bei Spalte 28 (Zeile 280) und eine Begrenzung mit den Anweisungen 370 und 380 optimal zu sein.

Der Reiz dieses Spiels besteht darin, daß der Spieler S für seine seitliche Bewegung im Einklang mit den physikalischen Möglichkeiten die Beschleunigung A eingibt und nicht die Geschwindigkeit, die ihm auch auf dem Bildschirm kaum sichtbar wird. Seine Abwärtsbewegung ist über T (Zeile 360) mit W synchron. Die in Zeile 330 programmierte A-Eingabe sorgt mit INPUT dafür, daß der Spieler Zeit zum Betrachten des jeweiligen vorherigen Ergebnisses auf dem Bildschirm hat und den T-Fortschritt selbst steuert. Die Formeln für die Bewegung von S in Zeile 330 und Zeile 410 folgen aus den kinematischen Grundgleichungen bei linearer Verbindung zwischen den eingegebenen A-Werten (ihre Nachprüfung geht über das Niveau der POS hinaus).

Ein längeres »Reiten« gelingt bei diesem Spiel eigentlich nur, wenn man das Wildschwein »am Waldrand« oder »im Fluß« erjagt. Sonst wird man nicht nur beim nächsten Zick-Zack-Wechsel abgeworfen, sondern meist auch vom eigenen Schwung. Aber das steht ja wohl kaum im Widerspruch zur Praxis.

3. Hasen-Schießen

In der zweiten Version wird die Zick-Zack-Bewegung von links nach rechts verwendet und läuft mit einer fest pro-

grammierten Geschwindigkeit ab. Der Text lautet:

```
10 CLS:?AT(8,3);"Wir laden ein zum"
20 ?AT(11,3);
"HASEN — SCHIESSEN"
30 ?AT(14,3);"Das ist Ihr Sitz: ",
CHR$ (204)
40 ?AT(20,9);"Einverstanden? (JA)":
INPUT C$
50 CLS:?:?"Zum Zielen koennen Sie":?
60 ?SPC(6)"zwei EINZIFFRIGE ZAHLEN
eintasten":?
70 ?"  fuer die Richtung nach rechts"
80 ?:?AT(6,21);"bzw. unten.":?
90 ?:?"  Einen Schuss aus ihrer
Schrotflinte"
100 ?"  loesen Sie mit der"
110 ?SPC(22)"TASTE S  aus."
120 ?:?:?:?:?"  Sind Sie bereit? (JA)":
INPUT C$
130 Z$="Weidmanns Heil! Zufrieden bei"
140 Z1$="Schuss? (J/N)  "
150 Z3$="Weitermachen!"
160 Z4$="Haben Sie ueberhaupt gezielt? "
170 Z5$="Etwas besser muessen Sie
schon zielen!"
180 T=—1:Z=3:H=10:C=1:SZ=0:
CLS:?AT(0,0); CHR$(204)
190 FOR I=1 TO 39:?AT(0,I);
CHR$(130):?AT(20,I);
CHR$(131):NEXT I
200 FOR I=1 TO 10:V=RND(1)*Z
210 FOR J=1 TO
4:T=T+1:H=H+V:IF H<0
THEN H=0
220 IF H>20 THEN H=20
230 ?AT(H,T);CHR$(201)
240 FOR K=1 TO 9
250 T$=INKEY$
260 IF T$="S" THEN GOTO 490
270 IF C<0 THEN 380
280 IF T$="1" THEN X=1:GOTO370
290 IF T$="2" THEN X=2:GOTO370
300 IF T$="3" THEN X=3:GOTO370
310 IF T$="4" THEN X=4:GOTO370
320 IF T$="5" THEN X=5:GOTO370
330 IF T$="6" THEN X=6:GOTO370
340 IF T$="7" THEN X=7:GOTO370
350 IF T$="8" THEN X=8:GOTO370
360 IF T$="9" THEN
X=9:GOTO370ELSE GOTO570
370 C=—C:BEEP:GOTO570
380 IF T$="0" THEN Y=0:GOTO480
390 IF T$="1" THEN Y=1:GOTO480
400 IF T$="2" THEN Y=2:GOTO480
410 IF T$="3" THEN Y=3:GOTO480
420 IF T$="4" THEN Y=4:GOTO480
430 IF T$="5" THEN Y=5:GOTO480
440 IF T$="6" THEN Y=6:GOTO480
450 IF T$="7" THEN Y=7:GOTO480
460 IF T$="8" THEN Y=8:GOTO480
470 IF T$="9"
THENY=9:GOTO480ELSE GOTO570
480 BEEP:C=—C:GOTO570
490 BEEP:S=T*Y/X—H:D=SQR
(T*T+H*H)*0.06:SZ=SZ+1
500 IF ABS(S)>D THEN540
510 ?AT(H,T);
CHR$(175):BEEP:?AT(22,0);
Z$,SZ,Z1$:BEEP:BEEP:BEEP
520 BEEP:BEEP:BEEP:BEEP:INPUT
C$:GOTO630
530 ?AT(22,0); Z5$:GOTO550
540 IF ABS(S)>5*D THEN ?AT(22,0);
Z4$:GOTO550ELSE GOTO530
550 IF (H+S)>21 THEN S=21—H
560 ?AT(H+S,T);"*":?AT(23,0); Z3$
570 NEXT K
580 NEXT J:Z=—Z
590 NEXT I
600 ?AT(22,0);"Bei",SZ,"
Schuessen sollten Sie treffen! "
610 ?AT(23,0); "Einverstanden? (JA)"
620 INPUT C$
630 ?AT(22,0); "Sollen wir Ihnen einen  "
640 ?AT(23,0); "neuen Hasen schicken?
(JA)  "
650 INPUT JA$
660 IF JA$="JA" THEN GOTO180
670 END
```

Hier wird bei der »Initialisierung« wieder INPUT (Zeilen 40 und 120) verwendet, um die Zeit zum Lesen der Einführungstexte dem Spieler zu überlassen. Das Skelett der Zick-Zack-Bewegung ist in den Zeilen 200 und 210 begonnen. Es werden 10 Züge mit je 4 Schritten gemacht, um die 40 Bildschirmpositionen zu durchlaufen. Innerhalb der Schleifen bis Zeile 580 bzw. 590 ist kein Befehl enthalten, der vor »Abschuß« des Hasen (Zeile 510) den Programmlauf unterbricht (wie IN-

PUT), so daß die reine Rechenzeit die Geschwindigkeit des Hasens in Vorwärtsrichtung von T = Ø bis T = 39 und damit die Chancen für den Spieler bestimmt. Als einfachste Gestaltungsmöglichkeit hierzu ist die K-Schleife zwischen den Zeilen 24Ø und 57Ø eingerichtet. Der neunmalige Durchlauf ist als Erfahrungswert am KC 85/1 als günstig zu erachten.
Für die Funktion des »Schützen« sind drei Eingabeelemente erforderlich:
X und Y zum »Zielen« in der Ebene und S zum »Abdrücken«. Sie lassen sich ohne Programmunterbrechung mit der Funktion

INKEY$

erfassen, die laut KC 85/1-Unterlagen einen Funktionswert liefert, der »dem der zuletzt betätigten Taste äquivalenten Zeichen« entspricht und als Zeichenkette verarbeitet werden kann. (Bei der Anwendung muß aber zumindest bei der zum hier verwendeten KC 85/1 gelieferten Fassung vor übertriebenen Erwartungen gewarnt werden. Ein mehrfacher Aufruf ohne zwischendurch erfolgte neue Tastenbetätigung war erfolglos.) Zu empfehlen ist bei dieser Funktion eine Rückmeldung an den Spieler, um beispielsweise zu schwachen Tastendruck oder ähnliche Fehler zu offenbaren. Hier wird dazu der BEEP-Piep-Laut in den Zeilen 37Ø, 48Ø und 49Ø verwendet. Die Reihenfolge der drei Eingaben X, Y und S wird mit der Größe C gesteuert. Dabei kann entsprechend dem Spielgedanken auch mehrfach durch S geschossen werden, ohne neu zielen zu müssen. Auch »Dauerfeuer« ist möglich, bei dem dann natürlich der Schußzähler SZ höher stehen wird, als sich durch Auszählen der auf dem Bildschirm wiedergegebenen Schuß-Symbole "*" ergibt.
Die Erfolgschancen des Spielers, das gedehnte BEEP des getroffenen Hasen zu hören, hängen maßgeblich nicht nur von der Geschwindigkeit des Hasen ab, also der Neun in Zeile 24Ø, sondern auch von der Streuung der »Schrotflinte«, die mit D in Zeile 49Ø fixiert ist. Der Faktor Ø,Ø6 hängt mit der Auflösungsmöglichkeit des Bildschirms zusammen und ist offensichtlich ein guter Erfahrungswert.
Bei oberflächlicher Betrachtung scheint es zu diesem Spiel einige leichte sichere Erfolgstaktiken zu geben, z.B. Zielen mit X = Y, also unter 45° nach rechts unten, und mit S warten, bis »der Hase in die Schußlinie läuft«. Sie halten jedoch der Praxis nicht stand, da der Hase schließlich den Zufall nutzt und doch recht schnell von T zu T + 1 »hüpft«.
Bei der Gestaltung der Texte ist zu beachten, daß alte Zeilen überschrieben werden müssen und deshalb Leerzeichen in mehreren Zeilen nicht entbehrlich sind.
Viel Vergnügen beim Nachspielen oder auch eigenem Abändern!

Autor:

Doz. Dr. sc. techn. Werner Gumpert
Technische Universität
Karl-Marx-Stadt

Berechnung der Verfügbarkeit technischer Systeme mit Hilfe des Büro-Computers A 5120

1. Einleitung

Die *Momentanverfügbarkeit* $A(t)$ (*point availability*) eines technischen Systems, das ist die Wahrscheinlichkeit dafür, daß das betrachtete System in einem beliebigen Zeitpunkt t funktionsfähig ist, ist eine wichtige Kenngröße im Rahmen der betrieblichen Zuverlässigkeitsarbeit. Oft läßt sich bei der Untersuchung konkreter technischer Einrichtungen folgendes *Modell* wenigstens näherungsweise zugrunde legen. Die reparierbare Betrachtungseinheit (System, Baugruppe, Baueinheit, Bauelement) kann sich nur in den beiden Zuständen »funktionsfähig« oder »ausgefallen« befinden. Im Zeitpunkt $t = 0$ befindet sich die Betrachtungseinheit im funktionsfähigen Zustand, der die *zufällige* Zeit ξ anhält und durch einen Ausfall beendet wird. In diesem Moment wird sofort mit der Erneuerung der Betrachtungseinheit begonnen. Die Erneuerung nimmt die *zufällige* Zeit η in Anspruch. Die Verteilungsfunktionen und Dichtefunktionen

$$F(t) = Pr(\xi < t), f(t) = F'(t)$$

der Funktionsdauer (Lebensdauer) sowie

$$G(t) = Pr(\eta < t), g(t) = G'(t)$$

der Erneuerungsdauer (Ausfalldauer) werden als bekannt vorausgesetzt [$f(t)$ und $g(t)$ seien stetig für $t > 0$]. Nach jeder Erneuerung sei die Betrachtungseinheit wieder wie neu (*vollständige Erneuerung*), alle Funktions- und Erneuerungsperioden sollen statistisch *unabhängig* sein. Man nimmt also an, daß sämtliche Funktionsperioden die gleiche Verteilungsfunktion $F(t)$ und sämtliche Erneuerungsperioden die Verteilungsfunktion $G(t)$ haben. Diese Folge aus sich abwechselnden Funktions- und Ausfallzuständen der Betrachtungseinheit läßt sich durch einen in der Zuverlässigkeitstheorie betrachteten sogenannten *alternierenden Erneuerungsprozeß* beschreiben.

In der Vergangenheit haben die Praktiker meistens mit der *Dauerverfügbarkeit* A der Betrachtungseinheit gearbeitet, die sich nach der einfachen Formel

$$A = \frac{\mathbf{E}\,\xi}{\mathbf{E}\,\xi + \mathbf{E}\,\eta} \qquad (1)$$

ermitteln läßt. Um A zu berechnen, müssen also lediglich die *mittleren* Funktionsdauern $\mathbf{E}\,\xi$ (*Erwartungswert* von ξ) und die *mittleren* Erneuerungsdauern $\mathbf{E}\,\eta$ (*Erwartungswert* von η) bekannt sein. Wegen der aus der Zuverlässigkeitstheorie bekannten Beziehung $A = \lim_{t \to \infty} A(t)$ liefert die Dauerverfügbarkeit durchaus brauchbare Informationen über die Verfügbarkeit der Betrachtungseinheit nach *hinreichend großer* Betriebszeit t ($t \gg 0$), d.h., sie

gibt Auskunft über das *Langzeitverhalten* der Betrachtungseinheit.
Es kann aber auch Fälle geben, in denen man sich auch auf die Zeit kurz nach Inbetriebnahme einer neu installierten Anlage konzentrieren muß, wenn die tatsächlichen (momentanen) Verfügbarkeitswerte $A(t)$ wesentlich unter die Werte der Dauerverfügbarkeit absinken und die Unkenntnis dieser Werte in der Anfangsphase zu Fehlentscheidungen führen würde (etwa bei der Planung von Reparaturkapazitäten oder Ersatzteilbeständen). Es ist deshalb ratsam, sich Informationen über die Momentanverfügbarkeit $A(t)$ zu beschaffen. Diese läßt sich nach der Formel

$$A(t) = \bar{F}(t) + \int_0^t \bar{F}(t-x)\, h(x)\, \mathrm{d}x \quad (2)$$

($\bar{F} = 1 - F$) im allgemeinen durch *numerische* Integration ermitteln, wenn man den Verlauf der *Erneuerungsdichte* $h(x)$ wenigstens näherungsweise kennt. Diese Erneuerungsdichte muß als Lösung der folgenden VOLTERRAschen Integralgleichung 2. Art (*Erneuerungsgleichung*)

$$h(x) = \varphi(x) + \int_0^x \varphi(x-u)\, h(u)\, \mathrm{d}u$$

$$(x \geqq 0) \quad (3)$$

mit der Funktion

$$\varphi(x) = \int_0^x f(x-u)\, g(u)\, \mathrm{d}u \quad (4)$$

berechnet werden. Übrigens liefert das Ergebnis bei der Auflösung der Gleichung (3) noch eine wichtige Information. Die *Erneuerungsfunktion*

$$H(t) = \int_0^t h(x)\, \mathrm{d}x$$

gibt nämlich die *mittlere Anzahl* der im Zeitintervall $(0, t]$ abgeschlossenen Erneuerungen an, d.h. also den *Erwartungswert* der Anzahl der im Intervall $(0, t]$ anfallenden Erneuerungen. Diese Information ist u.a. für die *Ersatzteilplanung* wichtig.
In den Fällen, in denen man die Erneuerungszeiten η als *fast konstant* gleich c annehmen kann, reduziert sich Formel (4) auf

$$\varphi(x) = f(x-c). \quad (4)^*$$

Sind die Erneuerungsdauern η *exponentiell* verteilt, d.h. ist $G(t) = 1 - \mathrm{e}^{-\mu t}$ für $t \geq 0$ und $G(t) = 0$ für $t < 0$, so kann in (2) $h(x) = \mu(1 - A(x)) = \mu \bar{A}(x)$ gesetzt und die momentane *Nichtverfügbarkeit* $\bar{A}(t)$ sofort aus der Integralgleichung

$$\bar{A}(t) = F(t) - \mu \int_0^t \bar{F}(t-x)\, \bar{A}(x)\, \mathrm{d}x \quad (2)^*$$

berechnet werden (s. [3]). Mit $\bar{A}(t)$ hat man natürlich auch die *Verfügbarkeit* $A(t) = 1 - \bar{A}(t)$.
Während man in den vergangenen Jahrzehnten zur numerischen Lösung einer Integralgleichung schon größere Rechenanlagen benutzen mußte, was sicherlich *ein* Grund dafür war, nur mit der Dauerverfügbarkeit zu arbeiten, kann man sich mit der heute zur Verfügung stehenden leistungsfähigen Kleinrechentechnik durchaus an die Lösung einer derartigen Gleichung heranwagen.
Im folgenden Abschnitt werden sehr einfache Algorithmen und zugehörige BASIC-Programme für den Büro-Computer A 5120 angegeben, mit deren Hilfe der Praktiker für bekannte Verteilungsfunktionen der Funktions- und Erneuerungsdauern die Funktion $A(t)$ näherungsweise berechnen und somit durchaus brauchbare Informationen über den tatsächlichen Verlauf der Momentanverfügbarkeit ermitteln kann. Da keine Fehlerabschätzungen angegeben werden können, muß man

die Rechnung gegebenenfalls mehrmals wiederholen, und zwar mit jeweils halbierter Schrittweite, bis sich im Rahmen der gewünschten Genauigkeit nicht mehr viel an den berechneten Werten ändert. Natürlich ist die Länge der Intervalle $[0, t]$, in denen man die Integralgleichung lösen will, beschränkt, man kann also nicht erwarten, daß sich für beliebig lange Intervalle dieser Art sinnvolle Ergebnisse ermitteln lassen.

Bei allen Rechnungen sollte man beachten, daß sich die Momentanverfügbarkeit $A(t)$ nach hinreichend langer Zeit t dem Wert A (den man vorher nach Formel (1) berechnen sollte) angleichen müßte und daß die Verfügbarkeit als eine Wahrscheinlichkeit nur Werte zwischen 0 und 1 annehmen kann.

2. Algorithmen und BASIC-Programme

Wir stellen uns also die Aufgabe, die Momentanverfügbarkeit $A(t)$ in den Punkten $t = n\sigma$ $(n = 0, 1, \ldots, N;\ \sigma > 0)$ näherungsweise zu berechnen. Die Schrittweite σ und die Anzahl N der Punkte sind vor Beginn einer Rechnung festzulegen.

Vorausgesetzt wird, daß die Dichtefunktionen $f(x)$ der Lebensdauer sowie die zugehörige Verteilungsfunktion $F(x)$ und die Dichtefunktion $g(t)$ der Erneuerungsdauer bekannt sind, wenigstens in Form von Wertetabellen für die Stellen $x = n\sigma$.

Zuerst sind Näherungswerte $\varphi(n\sigma)$ für $n = 0, 1, \ldots, N$ durch numerische Integration aus Gleichung (4) zu ermitteln. (Falls die Fälle (4)* oder (2)* vorliegen, entfällt dieser erste Schritt.) Um das Integral zu berechnen, wenden wir die aus der Grundvorlesung Mathematik bekannte zusammengesetzte *Trapezformel* an:

$$\varphi(0) = 0$$

$$\varphi(\sigma) = \frac{\sigma}{2}(f(\sigma)g(0) + f(0)g(\sigma))$$

$$\varphi(n\sigma) = \frac{\sigma}{2}(f(n\sigma)g(0) + f(0)g(n\sigma)) + \sigma\sum_{k=1}^{n-1} f([n-k]\sigma)g(k\sigma) \quad (5)$$

für $n = 2, 3, \ldots$.

Im Falle (4)* der fast konstanten Erneuerungszeiten $\eta \approx c$ gilt natürlich

$$\varphi(n\sigma) = f(n\sigma - c),\ n = 0, 1, \ldots \quad (5)^*$$

Im nächsten Schritt sind Näherungswerte $h(n\sigma)$ in den Stützstellen $x = n\sigma$ aus der Erneuerungsgleichung (3) zu berechnen. Wir verzichten auf die Verwendung eines komfortablen Algorithmus zur Lösung dieser Integralgleichung (ein solcher ist beispielsweise in [4] zu finden), sondern wenden ebenfalls die Trapezformel auf das in (3) vorkommende Integral an, was zu folgender einfacher *Rekursionsformel* zur sukzessiven Berechnung der Näherungswerte $h(n\sigma)$ führt:

$$h(0) = 0$$

$$h(\sigma) = \varphi(\sigma)$$

$$h(n\sigma) = \varphi(n\sigma) + \sigma\sum_{k=1}^{n-1} \varphi([n-k]\sigma)h(k\sigma) \quad (6)$$

für $n = 2, 3, \ldots$.

Mit den gegebenen Werten $\bar{F}(n\sigma) = 1 - F(n\sigma)$ und den bereits berechneten Werten $h(n\sigma)$ der Erneuerungsdichte lassen sich nun die Näherungswerte $A(n\sigma)$ für die Momentanverfügbarkeit ebenfalls durch *numerische Integration* nach der Trapezformel ermitteln:

$$A(0) = \bar{F}(0) = 1$$

$$A(\sigma) = \bar{F}(\sigma) + \frac{\sigma}{2}h(\sigma)$$

$$= \bar{F}(\sigma) + \frac{\sigma}{2}\varphi(\sigma)$$

$$A(n\sigma) = \bar{F}(n\sigma) + \frac{\sigma}{2} h(n\sigma) + \sigma \sum_{k=1}^{n-1} \bar{F}([n-k]\sigma) h(k\sigma) \quad (7)$$

für $n = 2, 3, \ldots$.

Im Falle exponentiell verteilter Erneuerungsdauern erhält man aus Formel (2)* wieder mit Hilfe der Trapezregel:

$$\bar{A}(0) = F(0) = 0$$

$$\bar{A}(\sigma) = \frac{2}{2+\mu\sigma}(1 - \bar{F}(\sigma))$$

$$\bar{A}(n\sigma) = \frac{2}{2+\mu\sigma}\left(1 - \bar{F}(n\sigma) - \mu\sigma \sum_{k=1}^{n-1} \bar{F}([n-k]\sigma)\, \bar{A}(k\sigma)\right) \quad (8)$$

für $n = 2, 3, \ldots$.

Der durch die Formel (5), (6) und (7) beschriebene Algorithmus zur Bestimmung der Momentanverfügbarkeit läßt sich leicht als ein BASIC-Programm schreiben (vgl. Programm Bild 1).

Im Falle der fast konstanten Erneuerungszeiten ist anstelle von (5) die Formel (5)* zu programmieren. Wird noch die Eingabe der konstanten Erneuerungszeit c vorgesehen, so ergibt sich das Programm in Bild 2.

Bei exponentiell verteilten Erneuerungszeiten kann natürlich auch das Programm »MOMENT 1« zur Berechnung von $A(n\sigma)$ verwendet werden. Andererseits kann in diesem Fall aber auch der Algorithmus (8) programmiert werden. Hierbei entfällt die Berechnung von (5) und (6), und deshalb können mit diesem Algorithmus auch

```
10 PRINT"berechnung der momentanverfuegbarkeit"
20 PRINT"einer reparierbaren betrachtungseinheit"
30 PRINT"(formeln (5),(6),(7))"
40 PRINT
50 PRINT"eingabe numerischer parameter"
60 INPUT"schrittweite=";SIGMA
70 INPUT"anzahl zu berechnender zeitpunkte=";NENDE
80 DIM PHI(NENDE), H(NENDE), A(NENDE)
90 DEF FNF(X)=DICHTEFUNKTION DER FUNKTIONSDAUER
100 DEF FNG(X)=DICHTEFUNKTION DER ERNEUERUNGSDAUER
110 DEF FNZUV(X)=1-VERTEILUNGSFUNKTION DER FUNKTIONSDAUER
120 REM berechnung von phi(0),phi(sigma),h(0),h(sigma),a(0),a(sigma)
130 PHI(0)=0 : PHI(1)=SIGMA/2*(FNF(SIGMA)*FNG(0)+FNF(0)*FNG(SIGMA))
140 H(0)=0 : H(1)=PHI(1)
150 A(0)=FNZUV(0) : A(1)=FNZUV(SIGMA)+SIGMA/2*H(1)
160 REM druck des kopfes der ergebnistabelle
170 LPRINT"momentanverfuegbarkeit"
180 LPRINT
190 LPRINT USING"&######&#.###";"A(";0;")=";A(0)
200 LPRINT USING"&######&#.###";"A(";SIGMA;")=";A(1)
210 REM berechnung von phi(n*sigma),h(n*sigma),a(n*sigma);n=2,3,...,nende
220 FOR N=2 TO NENDE
230 PHI(N)=SIGMA/2*(FNF(N*SIGMA)*FNG(0)+FNF(0)*FNG(N*SIGMA))
240 FOR K=1 TO N-1
250 PHI(N)=PHI(N)+SIGMA*FNF((N-K)*SIGMA)*FNG(K*SIGMA)
260 NEXT K
270 H(N)=PHI(N)
280 FOR K=1 TO N-1
290 H(N)=H(N)+SIGMA*PHI(N-K)*H(K)
300 NEXT K
310 A(N)=FNZUV(N*SIGMA)+SIGMA/2*H(N)
320 FOR K=1 TO N-1
330 A(N)=A(N)+SIGMA*FNZUV((N-K)*SIGMA)*H(K)
340 NEXT K
350 LPRINT USING"&######&#.###";"A(";N*SIGMA;")=";A(N)
360 NEXT N
370 END
```

Bild 1
MOMENT 1

```
10 PRINT"berechnung der momentanverfuegbarkeit"
20 PRINT"einer reparierbaren betrachtungseinheit"
30 PRINT"(formeln (5)*,(6),(7))"
40 PRINT
50 INPUT"konstante erneuerungsdauer=";C
60 PRINT"eingabe numerischer parameter"
70 INPUT"schrittweite=";SIGMA
80 INPUT"anzahl zu berechnender zeitpunkte=";NENDE
90 DIM PHI(NENDE), H(NENDE), A(NENDE)
100 DEF FNF(X)=1.5/1600!*(X/1600!)^.5*EXP(-(X/1600!)^1.5)
110 DEF FNZUV(X)=EXP(-(X/1600!)^1.5)
120 REM berechnung von phi(0),phi(sigma),h(0),h(sigma),a(0),a(sigma)
130 PHI(0)=0
140 IF SIGMA-C<=0 THEN PHI(1)=0 ELSE PHI(1)=FNF(SIGMA-C)
150 H(0)=0 : H(1)=PHI(1)
160 A(0)=FNZUV(0) : A(1)=FNZUV(SIGMA)+SIGMA/2*H(1)
170 REM druck des kopfes der ergebnistabelle
180 LPRINT"momentanverfuegbarkeit"
190 LPRINT
200 LPRINT USING"&#######&#.###";"A(";0;")=";A(0)
210 LPRINT USING"&#######&#.###";"A(";SIGMA;")=";A(1)
220 REM berechnung von phi(n*sigma),h(n*sigma),a(n*sigma);n=2,3,...,nende
230 FOR N=2 TO NENDE
240 IF N*SIGMA-C<=0 THEN PHI(N)=0 ELSE PHI(N)=FNF(N*SIGMA-C)
250 H(N)=PHI(N)
260 FOR K=1 TO N-1
270 H(N)=H(N)+SIGMA*PHI(N-K)*H(K)
280 NEXT K
290 A(N)=FNZUV(N*SIGMA)+SIGMA/2*H(N)
300 FOR K=1 TO N-1
310 A(N)=A(N)+SIGMA*FNZUV((N-K)*SIGMA)*H(K)
320 NEXT K
330 LPRINT USING"&#######&#.###";"A(";N*SIGMA;")=";A(N)
340 NEXT N
350 END
```

Bild 2
MOMENT 2

Lebensdauerverteilungen mit in $t = 0$ unstetigen Dichtefunktionen verarbeitet werden (wie beispielsweise die WEIBULL-Verteilung mit einem Formparameter kleiner als 1, wie im Beispiel 3 demonstriert). Im Hinblick auf die noch zu behandelnden Rechenbeispiele wollen wir uns darauf beschränken, den Algorithmus (8) nur für den Fall der Lebensdauern mit WEIBULL-Verteilung zu programmieren (vgl. Programm »MOMENT 3«, Bild 3).

Bemerkungen:

- Vor Abarbeitung der Programme »MOMENT 1« und »MOMENT 2« sind die Definitionsanweisungen für die Nutzerfunktionen FNF, FNG und FNZUV (soweit vorhanden) zu aktualisieren, d.h., rechts vom Gleichheitszeichen ist der jeweilige mathematische Term einzutragen, der zur gewünschten Funktion gehört.
- Nach der Berechnung können die im Feld A abgespeicherten Ergebniswerte auch graphisch dargestellt werden. Dazu ist *vor* dem Starten des Programms die END-Anweisung zu löschen und das Programm um die *Anweisungsfolge* zu *erweitern* (Bild 4).
- Um eine gute Übersichtlichkeit der Programme zu gewährleisten, wurde auf einige Feinheiten bewußt verzichtet. So kann es beispielsweise passieren, daß bei der Berechnung der Momentanverfügbarkeit $A(t)$ für sehr große Werte $t = n\,\sigma$ der zulässige Argumentbereich der verwendeten Standardfunktionen (z.B. EXP) verlassen und mit Programmfehler abgebrochen wird. Für die meisten praktischen Berechnungen wird dieser Fall aber nicht eintreten, da relativ

```
10 PRINT"berechnung der momentanverfuegbarkeit"
20 PRINT"einer reparierbaren betrachtungseinheit"
30 PRINT"formel (8)"
40 PRINT"funktionsdauer weibullverteilt"
50 PRINT"erneuerungsdauer exponentiell verteilt"
60 PRINT
70 PRINT"eingabe der verteilungsparameter"
80 INPUT"formparameter der weibullverteilung b=";B
90 INPUT"skalenparameter der weibullverteilung a=";A
100 INPUT"parameter der exponentialverteilung my=";MY
110 PRINT"eingabe numerischer parameter"
120 INPUT"schrittweite =";SIGMA
130 INPUT"anzahl zu berechnender zeitpunkte =";NENDE
140 DIM A(NENDE)
150 DEF FNZUV(X)=EXP(-(X/A)^B)
160 REM berechnung von a(0), a(sigma)
170 A(0)=1 : A(1)=1-2*(1-FNZUV(SIGMA))/(2+MY*SIGMA)
180 REM druck des kopfes der ergebnistabelle
190 LPRINT"momentanverfuegbarkeit"
200 LPRINT
210 LPRINT USING"&######&#.###";"A(";0;")=";A(0)
220 LPRINT USING"&######&#.###";"A(";SIGMA;")=";A(1)
230 REM berechnung von a(n*sigma); n=2,3,...,nende
240 FOR N=2 TO NENDE
250 A(N)=0
260 FOR K=1 TO N-1
270 A(N)=A(N)+FNZUV((N-K)*SIGMA)*(1-A(K))
280 NEXT K
290 A(N)=MY*SIGMA*A(N)
300 A(N)=1-FNZUV(N*SIGMA)-A(N)
310 A(N)=2*A(N)/(2+MY*SIGMA)
320 A(N)=1-A(N)
330 LPRINT USING"&######&#.###";"A(";N*SIGMA;")=";A(N)
340 NEXT N
350 END
```

Bild 3
MOMENT 3

schnell der stationäre Wert A erreicht wird (stationärer Zustand) und deshalb sehr große Werte t für die Berechnung nicht interessieren. Andererseits können die beschriebenen Fehler durch vorherige Prüfung der Argumente abgefangen werden.

Es ist angebracht, die berechneten Werte $A(n\sigma)$ mit dem vorher nach Formel (1) berechneten Wert A der Dauerverfügbarkeit zu vergleichen, damit man sich unnötige Rechenarbeit im Zusammenhang mit der Integralgleichung (3) im bereits eingetretenen stationären Zustand $[A(t) \approx A]$ ersparen kann.

Es gibt natürlich auch Fälle, in denen sich die Verfügbarkeit $A(t)$ geschlossen berechnen läßt, etwa dann, wenn Funktions- und Erneuerungszeiten exponentiell verteilt sind. Dieser Spezialfall soll aber nicht gesondert betrachtet werden, da er sich im Rahmen der Berechnung mit Hilfe eines Computers behandeln läßt.

```
400 REM druck des diagramms
410 LPRINT"T  <h>     0.1  0.2  0.3  0.4";
420 LPRINT"  0.5  0.6  0.7  0.8  0.9  1.0"
430 FOR K=1 TO 58
440 LPRINT"_";
450 NEXT K
460 LPRINT" "
470 FOR N=0 TO NENDE
480 LPRINT USING"######&";N*SIGMA;" :";
490 IF A(N)*50>70 THEN 510
500 LPRINT TAB(A(N)*50+8) "*"
510 NEXT N
520 END
```

Bild 4

3. Rechenbeispiele

Die nachfolgenden Beispiele wurden mit Hilfe der angegebenen BASIC-Programme auf einem Büro-Computer A 5120 gerechnet. Die ausgewählten Parameterkombinationen in den Verteilungsfunktionen haben meistens theoretischen Charakter.

Beispiel 1:

Für eine elektronische Überwachungseinrichtung wurde die Lebensdauerverteilung (WEIBULL-Verteilung)

$$F(t) = 1 - e^{-(t/a)^b} \quad (t \geqq 0)$$

mit den Parametern $a = 1{,}6 \cdot 10^3$ h und $b = 1{,}5$ ermittelt. Die Erneuerungsdauer ist nach Angabe des zuständigen Instandhalters exponentiell verteilt mit

$$G(t) = 1 - e^{-\mu t} \; (t \geqq 0),$$

$$\mu = 5 \cdot 10^{-3}/\text{h}.$$

Mit der mittleren Lebensdauer $E\,\xi = a\,\Gamma\left(1 + \frac{1}{b}\right) \approx 1445$ h und der mittleren Erneuerungsdauer $E\,\eta = 1/\mu = 200$ h ergibt sich für die Dauerverfügbarkeit $A \approx 0{,}878$.

Zur Berechnung der Momentanverfügbarkeit wird das Programm »MOMENT 1« verwendet. Die Aktualisierung der Definitionsanweisungen für die Nutzerfunktionen liefert die Anweisungen

```
90 DEF FNF(X)=1.5/1.6E3
*(X/1.6E3)↑0.5*EXP(-(X/1.6E3)↑1.5)
100 DEF FNG(X)=0.5E-2
*EXP(-0.5E-2*X)
110 DEF FNZUV(X)
=EXP(-(X/1.6E3)↑1.5)
```

und die Abarbeitung ergibt die in Bild 5 angegebenen Werte der Momentanverfügbarkeit.

Die Berechnung erfolgte zunächst für $\sigma = 100$ h im Intervall von 0 bis 2000 h und wurde danach mit $\sigma = 50$ h wiederholt. Beim Vergleich einander entsprechender Werte der beiden Berechnungen erkennt man bereits eine gute Übereinstimmung der Ergebnisse. Sollte die erzielte Genauigkeit jedoch nicht ausreichen, so ist die Rechnung mit $\sigma = 25$ h zu wiederholen, usw.

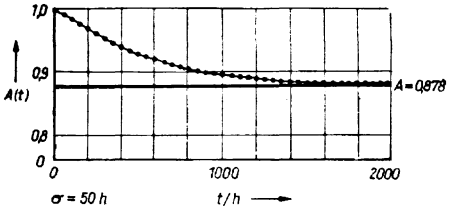


Bild 6

Bild 6 zeigt den zeitlichen Verlauf der Momentanverfügbarkeit. Man erkennt, daß bei $t = 2000$ h bereits annähernd der stationäre Zustand eingetreten ist.

Es ist schwierig, eine allgemeine Empfehlung für die Wahl der Schrittweite σ bei der ersten Berechnung zu geben,

```
momentanverfuegbarkeit  momentanverfuegbarkeit

A(    0)=1.000          A(    0)=1.000    A( 1050)=0.894
A(  100)=0.987          A(   50)=0.995    A( 1100)=0.892
A(  200)=0.970          A(  100)=0.987    A( 1150)=0.891
A(  300)=0.954          A(  150)=0.978    A( 1200)=0.889
A(  400)=0.940          A(  200)=0.970    A( 1250)=0.888
A(  500)=0.929          A(  250)=0.961    A( 1300)=0.887
A(  600)=0.920          A(  300)=0.954    A( 1350)=0.887
A(  700)=0.913          A(  350)=0.946    A( 1400)=0.886
A(  800)=0.908          A(  400)=0.940    A( 1450)=0.885
A(  900)=0.903          A(  450)=0.934    A( 1500)=0.884
A( 1000)=0.900          A(  500)=0.928    A( 1550)=0.884
A( 1100)=0.897          A(  550)=0.923    A( 1600)=0.883
A( 1200)=0.895          A(  600)=0.919    A( 1650)=0.883
A( 1300)=0.894          A(  650)=0.915    A( 1700)=0.883
A( 1400)=0.893          A(  700)=0.911    A( 1750)=0.882
A( 1500)=0.892          A(  750)=0.908    A( 1800)=0.882
A( 1600)=0.892          A(  800)=0.905    A( 1850)=0.882
A( 1700)=0.892          A(  850)=0.902    A( 1900)=0.882
A( 1800)=0.892          A(  900)=0.900    A( 1950)=0.882
A( 1900)=0.892          A(  950)=0.897    A( 2000)=0.882
A( 2000)=0.893          A( 1000)=0.895
```

Bild 5. Druckliste mit den Verfügbarkeitswerten des Beispiels 1 für $\sigma = 100$ h und $\sigma = 50$ h

weil vor der Berechnung nicht bekannt ist, wann der stationäre Wert erreicht wird. Man kann sich nur am Wert der mittleren Lebensdauer $\overline{T} = \mathbf{E}\,\xi$ orientieren und etwa mit $\sigma = \overline{T}/10$ beginnen.

Beispiel 2:

Gegeben sei:

$F(t) = 1 - e^{-\lambda t}\ (t \geqq 0)$

mit $\lambda = 5 \cdot 10^{-3}/\mathrm{h}$;

$G(t) = 1 - e^{-\mu t}\ (t \geqq 0)$

mit $\mu = 2 \cdot 10^{-2}/\mathrm{h}$

Gesucht ist die Momentanverfügbarkeit.

Die Dauerverfügbarkeit ist hier

$$A = \frac{\mathbf{E}\,\xi}{\mathbf{E}\,\xi + \mathbf{E}\,\eta} = \frac{1/\lambda}{1/\lambda + 1/\mu} = 0{,}8.$$

Es wird das Programm »MOMENT 1« benutzt, wobei die Aktualisierung

```
90 DEF FNF(X)=5E-—3
*EXP(—5E—3*X)
100 DEF FNG(X)=2E—2
*EXP(—2E—2*X)
110 DEF FNZUV(X)=EXP(—5E—3*X)
```

vorgenommen werden muß. Bild 7 zeigt die Ergebnisse der Berechnung für die Schrittweiten $\sigma = 10$ h und $\sigma = 20$ h, die eine ausgezeichnete Übereinstimmung aufweisen.

Bei exponentiell verteilten Funktions- und Erneuerungsdauern kann die Momentanverfügbarkeit auch nach der Formel

$$A(t) = \frac{\mu}{\mu + \lambda} + \frac{\lambda}{\mu + \lambda} e^{-(\mu + \lambda)t} \qquad (9)$$

berechnet werden. Der Leser kontrolliere die Ergebnisse mit Hilfe dieser Formel! Rundet man die nach (9) berechneten Werte auf drei Stellen nach dem Komma, so ergibt sich eine Übereinstimmung mit den Näherungswerten für $\sigma = 10$ h bis zum Wert A (230 h). Für A (240 h) bis A (280 h) ergibt sich der gerundete Wert 0,8, der vom Näherungswert in Bild 7 abweicht. Die ausgezeichnete Übereinstimmung der Näherungswerte mit den Werten nach Formel (9) verdeutlicht die gute Genauigkeit des vorgestellten einfachen Verfahrens. Bild 8 zeigt den zeitlichen Verlauf der Momentanverfügbarkeit.

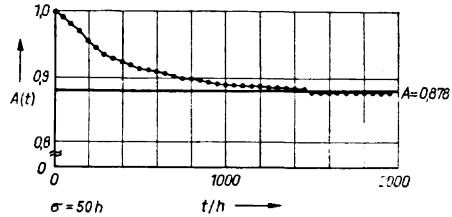


Bild 8

Beispiel 3:

Gegeben:

$F(t) = 1 - e^{-(t/a)^b}\ (t \geqq 0)$

```
momentanverfuegbarkeit

A(    0)=1.000
A(   20)=0.921
A(   40)=0.873
A(   60)=0.844
A(   80)=0.826
A(  100)=0.816
A(  120)=0.809
A(  140)=0.806
A(  160)=0.803
A(  180)=0.802
A(  200)=0.802
A(  220)=0.801
A(  240)=0.801
A(  260)=0.801
A(  280)=0.801
```

```
momentanverfuegbarkeit

A(    0)=1.000
A(   10)=0.956
A(   20)=0.921
A(   30)=0.894
A(   40)=0.873
A(   50)=0.857
A(   60)=0.844
A(   70)=0.834
A(   80)=0.827
A(   90)=0.821
A(  100)=0.816
A(  110)=0.813
A(  120)=0.810
A(  130)=0.808
A(  140)=0.806
A(  150)=0.805
A(  160)=0.804
A(  170)=0.803
A(  180)=0.802
A(  190)=0.802
A(  200)=0.801
A(  210)=0.801
A(  220)=0.801
A(  230)=0.801
A(  240)=0.801
A(  250)=0.801
A(  260)=0.801
A(  270)=0.801
A(  280)=0.801
```

Bild 7. Druckliste für Beispiel 2

momentanverfuegbarkeit

A(0)=1.000	
A(50)=0.594	
A(100)=0.574	
A(150)=0.579	
A(200)=0.583	
A(250)=0.597	
A(300)=0.606	
A(350)=0.613	
A(400)=0.620	
A(450)=0.625	
A(500)=0.630	
A(550)=0.634	
A(600)=0.638	

momentanverfuegbarkeit

A(0)=1.000	A(325)=0.605	
A(25)=0.650	A(350)=0.608	
A(50)=0.597	A(375)=0.612	
A(75)=0.578	A(400)=0.615	
A(100)=0.572	A(425)=0.618	
A(125)=0.572	A(450)=0.620	
A(150)=0.575	A(475)=0.623	
A(175)=0.579	A(500)=0.625	
A(200)=0.583	A(525)=0.627	
A(225)=0.588	A(550)=0.629	
A(250)=0.593	A(575)=0.631	
A(275)=0.597	A(600)=0.633	
A(300)=0.601		

Bild 9. Druckliste für Beispiel 3

mit $a = 100$ h und $b = 0{,}5$;

$G(t) = 1 - e^{-\mu t}$ $(t \geqq 0)$

mit $\mu = 10^{-2}$/h.

Gesucht: $A(t)$

Lösung: Mit der mittleren Lebensdauer $\mathbf{E}\,\xi = a\,\Gamma\left(1+\frac{1}{b}\right) = 200$ h und der mittleren Erneuerungsdauer $\mathbf{E}\,\eta = 1/\mu = 100$ h (das ist praktisch natürlich unsinnig) ergibt sich eine Dauerverfügbarkeit $A = 0{,}6666$.

Die Lebensdauer (WEIBULL-Verteilung) hat einen Formparameter $b < 1$. Die Verteilungsdichte hat in diesem Falle in $t = 0$ eine Unendlichkeitsstelle. Mit dem Programm »MOMENT 1« kann deshalb nicht gearbeitet werden. Da aber die Erneuerungsdauern exponentiell verteilt sind, kann nach Formel (8) und mit dem Programm »MOMENT 3« gerechnet werden. Bei »MOMENT 3« ist keine Aktualisierung von Anweisungen erforderlich. Die *Parametereingabe* erfolgt nach *Bildschirmaufforderung.*

In Bild 9 sind die Ergebnisse der Berechnung für die Schrittweiten $\sigma = 50$ h und $\sigma = 25$ h dargestellt.

Das Bild 10 zeigt das im Anschluß an die Berechnung erstellte *Computer-Diagramm* für den Verfügbarkeitsverlauf.

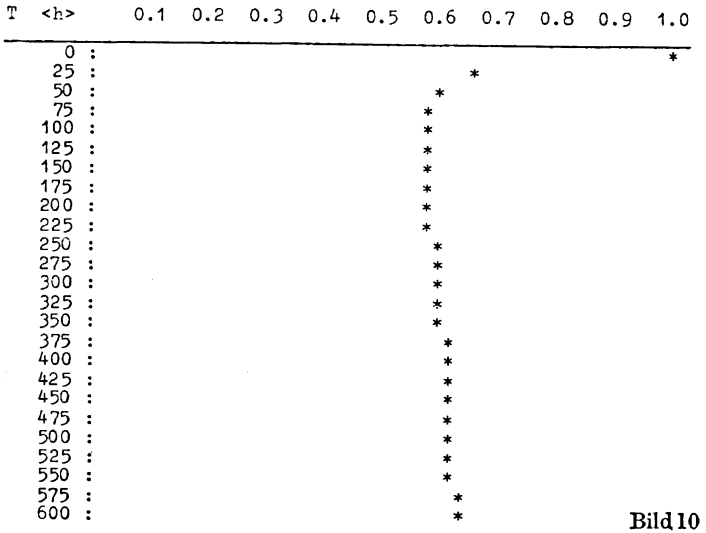


Bild 10

```
momentanverfuegbarkeit
A(    0)=1.000
A(   20)=0.905
A(   40)=0.819
A(   60)=0.788
A(   80)=0.799
A(  100)=0.801
A(  120)=0.800
A(  140)=0.800
A(  160)=0.800
A(  180)=0.800
A(  200)=0.800
A(  220)=0.800
A(  240)=0.800
A(  260)=0.800
A(  280)=0.800
```

Bild 11. Druckliste für Beispiel 4

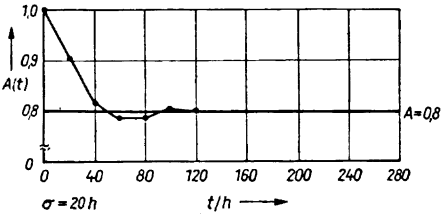


Bild 12

Beispiel 4:

Gegeben:

$F(t) = 1 - e^{-\lambda t}$ $(t \geqq 0)$

mit $\lambda = 5 \cdot 10^{-3}/\text{h}$;

Die Erneuerungsdauer sei fast konstant $c = 50$ h.

Gesucht: $A(t)$

Lösung: $A = \dfrac{1/\lambda}{1/\lambda + c} = 0{,}8$ ist der Wert der Dauerverfügbarkeit.

Anstelle der zufälligen Erneuerungsdauern (und deren Verteilungen) in Beispiel 2 rechnet man hier mit dem Erwartungswert dieser Zufallsgröße. Es ist ein ähnlicher zeitlicher Verlauf der Momentanverfügbarkeit zu erwarten wie in Beispiel 2.

Zur Berechnung wird das Programm »MOMENT 2« verwendet, nachdem die Anweisungen

```
100 DEF FNF(X)=5E-3*EXP
(-5E-3*X)
110 DEF FNZUV(X)=EXP(-5E-3*X)
```

eingegeben wurden.

Bei der Schrittweite $\sigma = 20$ h erhält man die Resultate in Bild 11 sowie die in Bild 12 angegebene Kurve der Momentanverfügbarkeit.

Bei exponentiell verteilten Funktionsdauern und fast konstanten Erneuerungsdauern kann die Momentanverfügbarkeit auch nach der Formel

$$A(t) = \sum_{n=0}^{[t/c]} \frac{\lambda^n}{n!} (t - nc)^n e^{-\lambda(t-nc)} \qquad (10)$$

berechnet werden, worin $[t/c]$ die größte ganze Zahl, die kleiner als t/c ist, bezeichnet (s. [3]). Das Nachrechnen der

```
momentanverfuegbarkeit
A(    0)=1.000
A(  100)=0.984
A(  200)=0.957
A(  300)=0.934
A(  400)=0.921
A(  500)=0.912
A(  600)=0.904
A(  700)=0.898
A(  800)=0.893
A(  900)=0.889
A( 1000)=0.885
A( 1100)=0.883
A( 1200)=0.880
A( 1300)=0.878
A( 1400)=0.877
A( 1500)=0.875
A( 1600)=0.874
A( 1700)=0.873
A( 1800)=0.872
A( 1900)=0.872
A( 2000)=0.871
```

```
momentanverfuegbarkeit
A(    0)=1.000
A(   50)=0.994
A(  100)=0.984
A(  150)=0.972
A(  200)=0.957
A(  250)=0.944
A(  300)=0.936
A(  350)=0.929
A(  400)=0.924
A(  450)=0.919
A(  500)=0.914
A(  550)=0.910
A(  600)=0.907
A(  650)=0.904
A(  700)=0.901
A(  750)=0.898
A(  800)=0.896
A(  850)=0.894
A(  900)=0.892
A(  950)=0.890
A( 1000)=0.889
A( 1050)=0.887
A( 1100)=0.886
A( 1150)=0.885
A( 1200)=0.884
A( 1250)=0.883
A( 1300)=0.882
A( 1350)=0.881
A( 1400)=0.880
A( 1450)=0.880
A( 1500)=0.879
A( 1550)=0.879
A( 1600)=0.878
A( 1650)=0.878
A( 1700)=0.877
A( 1750)=0.877
A( 1800)=0.877
A( 1850)=0.877
A( 1900)=0.876
A( 1950)=0.876
A( 2000)=0.876
```

Bild 13. Druckliste für Beispiel 5

Verfügbarkeitswerte über die Formel (10) bestätigt die genaue Übereinstimmung mit den Werten in Bild 11 in den ersten drei Stellen nach dem Komma.

Beispiel 5:

Gegeben:

$$F(t) = 1 - e^{-(t/a)^b} \quad (t \geqq 0)$$

mit $a = 1{,}6 \cdot 10^3$ h und $b = 1{,}5$;

Die Erneuerungsdauern sind fast konstant $c = 200$ h.

Gesucht: $A(t)$

Lösung: Der Vergleich mit Beispiel 1 läßt einen ähnlichen Verlauf der Momentanverfügbarkeit vermuten. Für die Dauerverfügbarkeit ergibt sich wie dort $A = 0{,}878$.

Es kann mit dem Programm »MOMENT 2« gearbeitet werden. Dazu überschreibt man die Anweisungen 100 und 110 durch

```
100 DEF FNF(X)=1.5/1.6E3
*(X/1.6E3)↑0.5*EXP(—(X/1.6E3)↑1.5)
110 DEF FNZUV(X)
=EXP(—(X/1.6E3)↑1.5)
```

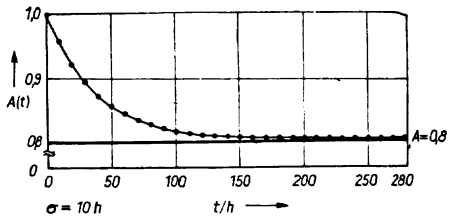


Bild 14

Die Näherungslösungen für $\sigma = 100$ h und $\sigma = 50$ h findet man in Bild 13, die Kurve der Momentanverfügbarkeit in Bild 14.

Der Vergleich mit den Ergebnissen von Beispiel 1 in Bild 5 bestätigt die obige Vermutung.

Literatur

[1] Beichelt, F.; Franken, P.: Zuverlässigkeit und Instandhaltung. – Berlin: Verlag Technik, 1983

[2] Köchel, P.: Zuverlässigkeit technischer Systeme. – Leipzig: Fachbuchverlag, 1982

[3] Preuss, W.; Kossow, A.; Kirchner, H.: Verschiedene Formeln zur Berechnung der instationären Intervallverfügbarkeit bei reparierbaren Betrachtungseinheiten technischer Systeme. – In: ZOR, Vol. 29. – B 17 – B 39

[4] Tabbert, E.: Untersuchungen zur Struktur und Implementation eines Programmiersystems für lineare Volterrasche Integralgleichungen zweiter Art und ihre Realisierung auf ESER-Anlagen. – Diss. A, Pädagogische Hochschule Güstrow, 1977

Autoren:

Dr. rer. nat. Andreas Kossow

Prof. Dr. sc. nat. Wolfgang Preuß

Ingenieurhochschule Wismar
Abteilung Mathematik/Naturwissenschaften

Zeichnen mit schnellen Grafikbefehlen

Mit dem Einzug der Kleincomputer in Hoch- und Fachschulen, Berufsschulen, Stationen junger Naturforscher und Techniker und Pionierhäuser ist auch das Interesse an Lehr- und Lernprogrammen geweckt worden. Die speziellen Möglichkeiten, die ein solches Gerät bietet, werden zunehmend Anlaß für das Entstehen geeigneter Programme. Besonders der Kleincomputer KC 85/2 bzw. KC 85/3 lädt mit seiner Farbtüchtigkeit und hochauflösenden Grafik zu kreativem Arbeiten auf diesem Gebiet ein.

Dieser Beitrag soll Anregung und zugleich *Hilfsmittel zum Darstellen beliebiger Konturen auf dem Bildschirm* sein. Angewendet wurde dieses Hilfsmittel in einem Lehrprogramm für das Fach Geographie der Klasse 8 mit dem Thema: »Ägyptens Topographie und Landwirtschaft«. Das gesamte Programm wurde in der Programmiersprache BASIC geschrieben, deren Vorteile beim Umgang mit Texten ganz beachtlich sind. Allerdings ist eine *schnelle Grafikverarbeitung mit den vorhandenen BASIC-Befehlen nicht möglich. Aus diesem Grund wurde ein kleines Maschinenprogramm entwickelt.* Es fügt sich problemlos in das BASIC-Programm ein und erfordert auch keine speziellen Kenntnisse in dieser Richtung. An Hand der Umrißkarte von Ägypten sollen die Vorteile dieser Ergänzung erläutert werden.

Als erster Schritt sollte eine *maßstabsgerechte Zeichnung* auf Millimeterpapier angefertigt werden. Je sauberer dabei gearbeitet wird, desto leichter fällt die programmtechnische Umsetzung. Günstig hat sich die Darstellung der Linien-

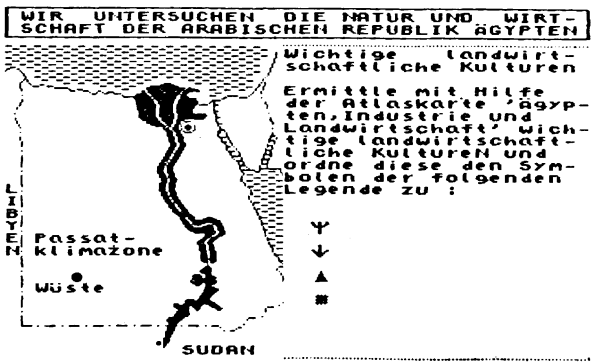


Bild 1
Bildschirmgrafik für ein Lehrprogramm mit dem KC 85/2

züge als Kette von aneinandergereihten Punkten (Pixel) im Millimeterraster erwiesen. Werden mehrere Farben verwendet, so ist darauf zu achten, daß beim KC 85/2 und KC 85/3 jeweils in einem Feld von 4 × 8 Pixel nur eine Vordergrundfarbe möglich ist. Im nächsten Schritt wird die *Reihenfolge der zu zeichnenden Bildelemente* festgelegt. Ein fortlaufender Linienzug macht sich optisch besser als ein unterbrochener. Ebenso sollten zuerst die Umrisse und dann die Details entstehen.

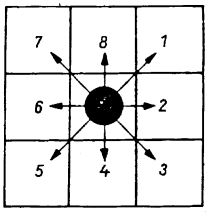


Bild 2. Bewegungsrichtungen der Pixel

Nach diesen Vorbereitungen kann mit der *Einbindung in das Hauptprogramm* begonnen werden. Zur Erhöhung der Übersichtlichkeit ist die Einbindung als Unterprogramm sinnvoll. Die einfachste Variante, um Kurven in BASIC schnell entstehen zu lassen, ist der immer wiederkehrende Aufruf der PSET-Anweisung. Für umfangreiche Darstellungen ist dabei viel Speicherplatz erforderlich. Jeder Punkt benötigt etwa 8 Byte im RAM. Versuche, mit Hilfe anderer Methoden zu sparen, gehen auf Kosten der Geschwindigkeit. Aus dieser mißlichen Situation kann uns eine kleine Maschinenroutine retten. Wir gehen von der Tatsache aus, daß *Linien* aus einer *Aneinanderreihung von Pixeln* bestehen. Von einem Ausgangspixel werden die Koordinaten des jeweils nächsten Pixel nur um 1 erhöht oder verringert. Bild 2 zeigt die 8 möglichen Richtungen, in die sich ein Pixel bewegen kann. Jeder Richtung wird nun eine Ziffer von 1 bis 8 zugeordnet. Um ein Achteck darzustellen, ist also nur noch die Ziffernfolge »1122334455667788« einzugeben und abzuspeichern. Das BASIC-Programm in Bild 3 veranschaulicht die Funktionsweise. Jede Ziffer verkörpert ein Pixel und benötigt im Speicher nur noch 1 Byte, die Einsparung ist deutlich sichtbar. Das Problem besteht nun darin, diese Ziffern dem Maschinenprogramm zu übergeben. Der BASIC-Interpreter des KC 85/2 und KC 85/3 gibt uns dafür eine ganz einfache Lösung. Die Anweisung PRINT#n ermöglicht eine Ausgabe für periphere Geräte.

Wird einem solchen Kanal unsere Maschinenroutine zugeordnet, so erfolgt die Ausgabe der Ziffern dorthin und nicht auf den Bildschirm. Der Kanal muß nun noch einen Namen oder besser eine Nummer bekommen. Kanal 0 ist

```
 10 DIM S(2, 8): CLS
 20 FOR I=1 TO 8:READ S(0, I): READ S(1, I):NEXT
 30 DATA 1, 1, 1, 0, 1, —1, 0, —1, —1, —1, —1, 0, —1, 1, 0, 1
 40 WINDOW 1, 3, 0, 39:INPUT "X, Y:";X, Y:PSET X, Y, 7
 50 CLS:INPUT "ZIFFERNFOLGE:"; A$
 60 IF A$="E" OR A$="END" THEN END
 70 GOSUB 100:GOTO 50
100 ! UNTERPROG. ZUM ZEICHNEN EINER PIXELFOLGE
110 FOR I=1 TO LEN(A$)
120 K=VAL(MIDS(A$, I, 1)):X=X+S(0, K):Y=Y+S(1, K)
130 PSET X, Y:NEXT:RETURN
```

Bild 3. BASIC-Programm zum Zeichnen von Konturen

	Adresse	BASIC-Interpreter	
		Lesen	Schreiben
X-Koordinate	B7D3H- B7D4H	X=VPEEK (14291)+ 256 * VPEEK (14292)	VPOKE 14292, X/256: VPOKE 14291, X—256 * INT (X/256)
Y-Koordinate	B7D5H	Y=VPEEK (14293)	VPOKE 14293, Y
Farbe	B7D6H	F=VPEEK (14294)	VPOKE 14294, F

Bild 4. Systemvariablen und BASIC-Zugriff der Pixelparameter

schon für den Bildschirm reserviert und Kanal 1 für den Kassettenrecorder. Frei sind dagegen noch Kanal 2 und 3. Kanal 2 wollen wir uns für einen Drukker aufheben, so daß noch Kanal 3 übrig bleibt.
Eine weitere Vereinbarung ist noch notwendig. Das Maschinenprogramm braucht zum Setzen eines Pixels die Position und Farbe des vorhergehenden. In Bild 4 sind die Systemvariablen ersichtlich, in denen diese Informationen des jeweils letzten Punktes aufbewahrt werden. Initialisiert werden diese Variablen durch eine PSET- oder PRESET-Anweisung. Mit Hilfe der nachfolgenden Programmzeilen soll das eben Gesagte verdeutlicht werden:

```
10 PSET 90, 100, 7
20 PRINT#3 "1122334455667788"
```

In Zeile 10 wird das erste Pixel gesetzt und die Farbe festgelegt. Mit der Zeile 20 erscheinen 16 weitere Pixel auf dem Bildschirm, Bild 5 zeigt die Figur, die daraus entsteht.
Der aufmerksame Leser wird jetzt berechtigt feststellen, daß das Programm noch gar nicht funktionieren kann, denn das Maschinenprogramm muß erst noch in den Computer eingegeben werden. Dafür gibt es mehrere Möglichkeiten. Wir haben uns für eine Variante entschieden, die folgende Vorteile aufweist:

1. Das Maschinenprogramm kann gleichzeitig mit dem BASIC-Programm von der Kassette geladen werden. Dies ist möglich, da es sich in einer REM-Zeile »versteckt«.
2. Es wird kein zusätzlicher Speicherplatz beansprucht.
3. Das Maschinenprogramm läuft unabhängig vom verwendeten Interpreter und von eventuell gesteckten Zusatzmodulen.

In Bild 6 ist der genaue Ablauf der Eingabe des Maschinenprogrammes angegeben. Alle Punkte müssen in richtiger Reihenfolge abgearbeitet werden, sonst erleiden wir »Schiffbruch«. Nach erfolgreicher Durchführung der Tätigkeiten befindet sich auf der Kassette ein kleines BASIC-Programm mit einer eingebauten Maschinenroutine. Sollen diese Erweiterungen in ein größeres Programm eingebunden werden, so sind lediglich folgende Schritte auszuführen:

- mit NEW altes Programm löschen,
- von Kassette Erweiterung laden,

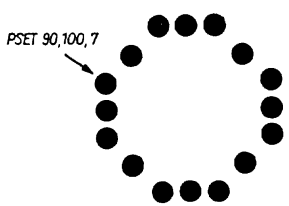


Bild 5
Achteck

– neues Programm anschließen,
– gesamtes Programm auf Kassette abspeichern.

Nach diesem Ablauf ist die Erweiterung in das neue Programm voll integriert. Es empfiehlt sich, nach dem Laden das Programm einmal mit RUN zu starten, ansonsten sieht das Listing etwas merkwürdig aus.

Um das Maschinenprogramm universell zu gestalten, wurden noch einige Sonderfunktionen eingebaut, Bild 7 zeigt die vollständige Aufstellung. An Hand von Beispielen soll in Bild 8 die

```
> NEW
> BYE
% MODIFY 35F[1])
   035F  BD/3D7[2])
   03D7  BF/400[3])
   0400  00 BD 04 00 00 9C 05 01
   0408  0D 06 0F 03 07 02 FE 44
   0410  20 06 3E FF 32 43 01 C9
   0418  FE 53 20 05 AF 32 41 01
   0420  C9 FE 52 20 04 3E FF 18
   0428  F4 FE 4E 20 08 AF 32 43
   0430  01 32 42 01 C9 FE 58 20
   0438  04 3E FF 18 F4 FE 39 D0
   0440  D6 31 D8 E5 C6 06 6F 26
   0448  2B 7E 1F 2A D3 B7 30 09
   0450  1F 38 02 23 23 2B 22 D3
   0458  B7 1F 30 0C 1F 3A D5 B7
   0460  38 02 3C 3C 3D 32 D5 B7
   0468  3A 42 01 E6 01 20 26 3A
   0470  41 01 E6 01 20 19 CD 03
   0478  F0 30 3A 43 01 E6 01 28
   0480  0C 23 22 D3 B7 CD 03 F0
   0488  30 2B 22 D3 B7 E1 C9 CD
   0490  03 F0 2F E1 C9 7D E6 07
   0498  C5 CB 3C CB 1D CB 3D CB
   04A0  3D 47 3A D5 B7 2F 67 D5
   04A8  CD 03 F0 34 D1 38 08 AF
   04B0  37 04 1F 10 FD AE 77 C1
   04B8  E1 C9 0C 12 00 00 00
% REBASIC
> 1  IF PEEK (864)< >4 THEN POKE 10847, 189 :
     VPOKE 14277,43 : POKE 11080,43 : GOTO 3
> 2  POKE 863, 189 : VPOKE 14277, 4 : POKE 1096, 4
> 3  VPOKE 14276, 14
> CSAVE "GRAFIK"
```

Bild 6. Eingabealgorithmus für die Maschinenroutine nach Aufruf des BASIC-Interpreters

Für die Kassettenvariante des BASIC-Interpreters müssen folgende Adressen geändert werden:

1) 2A5F statt 35F
2) 2AD7 statt 3D7
3) 2B00 statt 400

Zeichen	Funktion	Erläuterung
R	Rücksetzen	Alle nachfolgenden Pixel nehmen Hintergrundfarbe an.
S	Setzen	Zusammen mit der Funktion R können unterbrochene Linienzüge gezeichnet werden.
D	Doppelte Breite	Empfiehlt sich, bei Fernsehgeräten mit schlechter Darstellung der senkrechten Konturen.
X	Exklusiv – Oder	Alle nachfolgenden Pixel werden negiert, aus Vordergrundfarbe wird Hintergrundfarbe und umgekehrt.
N	Normal	Funktionen D und X werden außer Kraft gesetzt.

Bild 7. Sonderfunktionen

gesamte Anwendungsbreite verdeutlicht werden. Was nicht aus den Abbildungen hervorgeht, ist die hohe Geschwindigkeit mit der die Figuren auf dem Bildschirm erscheinen.

Befehle	Darstellung
a) PRINT#3 "2222R22S2R22S2222"	
b) PRINT#3 STRING$(4,"2222R2222S")	
c) CLS : PRINT "■ ■ "	
PSET 0,252,7	
PRINT#3 "X";STRING$(40,"2");"N"	
d) A$ = '8888888822222222'	
PRINT#3 A$;"X";A$;"N"	

Bild 8. Anwendungsbeispiele

Autor:
Stefan Wendt
Fernsehgerätewerk Staßfurt
Betriebsteil Halle

ISBN 3-343-00385-9

1. Auflage
Lizenznummer 114-210/1/88
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328
Redaktionsschluß: 15. 3. 1988
Bestellnummer: 5473709
00780

Kleinstrechner-TIPS/Hrsg. von
Hans Kreul u. a. – Leipzig: Fachbuchverl.,
H. 8. – 1. Aufl. – 1988 – 64 S.: 26 Bild.

Anschrift des Verlages:
VEB Fachbuchverlag Leipzig
PSF 67
Leipzig
DDR – 7031

Vorschau auf Heft 9

Girlich: Iterationen und das Apfelmännchen

Fischer: Bestimmte Integrale

Stammler: Geometrie mit dem KC 85/1 »getupft«

Gutzer: Der projizierende Computer

Gutzer: Der Kleincomputer als optischer Täuscher

Kossow/Preuß: Berechnung optimaler Erneuerungsintervalle für die vorbeugende Instandhaltung mit Hilfe des Büro-Computers A 5120

Köhler: Komfortable Z 1013-Tastatur

Herausgeber:

Prof. Dr.-Ing. *Hans Kreul*
Bruno-Schröter-Str. 1
Zittau
DDR – 8800

Doz. Dr.-Ing. *Wilhelm Leupold* und
Doz. Dr. sc. techn. *Thomas Horn*
Informatikzentrum des Hochschulwesens
an der Technischen Universität Dresden
Mommsenstr. 13
Dresden
DDR – 8027

7.80

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00385-9