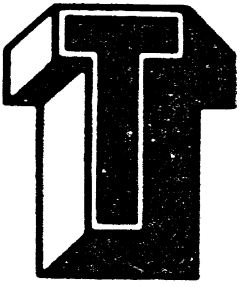
Kleinstrechner

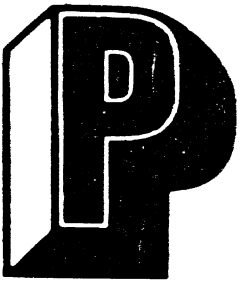


Tendenzen
und Theorien

Z 1013-Tastatur

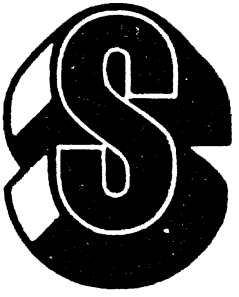
Informationen
und Ideen

Iterationen



Programme
und Projekte

Geometrie



Spaß
und Spiel

Kleinstrechner-TIPS

Heft 9

Mit 60 Bildern

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

VEB Fachbuchverlag Leipzig

Inhalt

Girlich: Iterationen und das Apfelmännchen *4*

Fischer: Bestimmte Integrale *11*

Stammler: Geometrie mit dem KC 85/1 »getupft« *20*

Gutzer: Der projizierende Computer *28*

Beherrschen Sie BASIC? *38*

Gutzer: Der Kleincomputer als optischer Täuscher *39*

Kossow/Preuß: Berechnung optimaler Erneuerungsintervalle für die vorbeugende Instandhaltung mit Hilfe des Bürocomputers A 5120 *44*

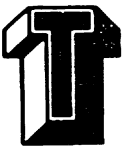
Köhler: Komfortable Z 1013-Tastatur *53*

Kreul: Hut ab vor dem Computer! (?) *62*

Hinweise für Autoren *64*

ISBN 3-343-00386-7

Zum Inhalt



Auch im Heft 9 der Schriftenreihe Kleinstrechner-TIPS sollen wieder allgemein interessierende Probleme, die Laien, Arbeitsgemeinschaften und Computerclubs bei ihrer Beschäftigung mit den Möglichkeiten der Anwendung der Informatik nützlich sein können, behandelt werden.

Zunächst gibt GIRLICH in Fortsetzung seines bereits in Heft 5 erschienenen Artikels Anregungen für eine anspruchsvolle Beschäftigung mit modernen mathematischen Verfahren und deren Darstellung am grafikfähigen Computer.

Wie bestimmte Integrale mit Hilfe eines Computers mit einer vorgegebenen Genauigkeit ermittelt werden können, wird von FISCHER gezeigt. Dabei wird auch auf die Gefahren einer leichtfertigen Nutzung von Programmen eingegangen, die in bestimmten Fällen zu völlig falschen Ergebnissen führen kann.

Daß der KC 87 (KC 85/1) für grafische Darstellungen gegenüber dem KC 85/2 (3) im Nachteil ist, dürfte allgemein bekannt sein. Von STAMMLER wird gezeigt, wie man dennoch mit diesem Rechner recht befriedigende Grafiken erzielen kann.

GUTZER stellt Programme vor, mit deren Hilfe räumliche Gebilde durch Parallel- oder Zentralprojektion am KC 85/2 (3) dargestellt werden können. Wenn man will, kann man diese Überlegungen bereits als Anfänge der CAD-Arbeit an Kleincomputern deuten.

Vom gleichen Autor stammt ein weiterer Artikel über grafische Darstellungen mit dem KC 85/2 (3). Er zeigt, wie bekannte optische Täuschungen am Bildschirm dargestellt werden können.

Von KOSSOW und PREUSS wird ein Programm für den Bürocomputer A 5120 zur Ermittlung optimaler Erneuerungsintervalle für die vorbeugende Instandhaltung vorgestellt. Wie bereits im vorhergehenden Heft wird deutlich, daß nicht immer Großcomputer erforderlich sind, wenn anspruchsvolle technische Probleme zu lösen sind.

Für den Elektronik-Bastler stellt KÖHLER eine komfortable Tastatur für den Computer-Bausatz Z 1013 vor, die gegenüber der mit dem Bausatz mitgelieferten Folientastatur wesentliche Vorteile aufweist.

In der Hoffnung, wiederum einen abwechslungsreichen Einblick in die vielfältigen Anwendungsmöglichkeiten der Kleincomputer gegeben zu haben, fordern wir unsere Leser zur Kritik, zu Verbesserungsvorschlägen sowie zu Manuskriptangeboten für die folgenden Hefte auf.

Hans Kreul

Iterationen und das Apfelmännchen

(Mathematische Experimente mit dem Kleincomputer)

In Heft 5 der Kleinstrechner-TIPS haben wir uns mit iterativ erzeugten Folgen von Zahlen beschäftigt und dabei den »Feigenbaum« kennengelernt (vgl. [1]). Wir wollen nun von den Zahlen zu Punkten in der Ebene übergehen und das Verhalten von iterativ erzeugten Punktfolgen betrachten, die wir kurz *Sequenzen* nennen. Solch eine Sequenz kann zum Beispiel einen Wirbel bilden, wobei die Punkte der Folge zu einem festen Punkt hinstreben. Bei anderen Sequenzen driften dagegen die Punkte über alle Grenzen. Wir interessieren uns hier insbesondere für beschränkte Sequenzen, das sind solche, deren Punkte alle innerhalb eines Kreises liegen. Charakterisieren wir jede Sequenz durch ihren Startpunkt, so bildet die Gesamtheit der Startpunkte beschränkter Sequenzen ein merkwürdiges Gebilde, das zu Ehren seines Entdeckers MANDELBROT-Menge genannt wird. Das Merkwürdige an dieser Menge besteht darin, daß der Betrachter zunächst ein »Apfelmännchen« zu erkennen glaubt. Wenn er allerdings genauer hinsieht, so bemerkt er eine Vielzahl von Apfelmännchen. Auch wir wollen auf Entdeckungsreise gehen und Apfelmännchen suchen. Dabei können wir durch Verändern des Maßstabs neue Räume erschließen; ein »Adventure« auf dem KC 85/2 ohne Text und ohne Ende!

1. Iterationen in der Ebene

Die Iterationen auf der Geraden wurden in [1] durch folgende Vorschrift beschrieben:

$$a_{n+1} = a_n\,(1 - a_n)\,y,\ a_0 = 0{,}5, \qquad (1)$$

wobei y ein Parameter war, der im Bereich $2 \leqq y \leqq 4$ variierte.
Analog zu (1) wählen wir quadratische Ausdrücke, um Iterationen in der Ebene zu erklären:

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + x, \quad x_0 = 0 \\ y_{n+1} &= 2\,x_n \cdot y_n + y, \quad y_0 = 0, \end{aligned} \qquad (2)$$

wobei x, y zwei Parameter sind, die etwa im Bereich $-4 \leqq x, y \leqq 4$ variieren. Geometrisch interpretiert, erzeugt die Vorschrift (2) eine Folge von Punkten:

$P_0 = (0; 0)$, $P_1 = (x_1; y_1) = (x; y)$,

$P_2 = (x_2; y_2) = (x^2 + x - y^2; 2\,xy + y)$
usw.

Die Folge $P_1, P_2, P_3, \ldots, P_N$ bezeichnen wir als Sequenz $S_N\,(x; y)$ der Länge N mit dem Startpunkt
$P_1 = (x; y)$.
Wir betrachten drei typische Beispiele:

(E1): $\boldsymbol{x = -0{,}50; y = 0{,}50.}$

Über (2) erhalten wir auf zwei Stellen genau:

Tabelle 1

$x_2 = -0{,}50;\quad y_2 = 0{,}00$
$x_3 = -0{,}25;\quad y_3 = 0{,}50$
$x_4 = -0{,}69;\quad y_4 = 0{,}25$
$x_5 = -0{,}09;\quad y_5 = 0{,}16$
$x_6 = -0{,}52;\quad y_6 = 0{,}47$
$x_7 = -0{,}46;\quad y_7 = 0{,}01$
$x_8 = -0{,}29;\quad y_8 = 0{,}49$
$x_9 = -0{,}65;\quad y_9 = 0{,}21$
$x_{10} = -0{,}12;\quad y_{10} = 0{,}22$
$x_{11} = -0{,}53;\quad y_{11} = 0{,}45.$

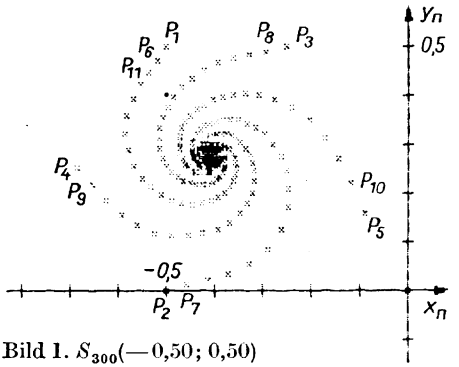


Bild 1. $S_{300}(-0{,}50;\ 0{,}50)$

Die Sequenz S_{300} $(-0{,}50;\ 0{,}50)$ ist auf Bild 1 dargestellt.
Sie bildet einen »Wirbel« mit dem Zentrum $P_{300} = (-0{,}41;\ 0{,}27)$. Die der Tabelle 1 entsprechenden Punkte sind in Bild 1 hervorgehoben, um anzudeuten, daß die Punkte der Sequenz von einem Wirbelarm zum anderen springen und sich dabei spiralförmig auf das Wirbelzentrum hinbewegen, von dem sie gleichsam magisch angezogen werden. Wie groß wir N auch wählen, die Punkte der Sequenz S_N $(-0{,}50;\ 0{,}50)$ bleiben stets innerhalb eines Quadrates der Seitenlänge 0,7.

(E2): $\boldsymbol{x = -0{,}51;\ y = 0{,}51}$

Mittels (2) bekommen wir eine auf zwei Stellen genaue Folge von Werten, die bis zum Wert y_8 sich von denen aus Tabelle 1 um höchstens 0,03 unterscheiden. Dann folgt

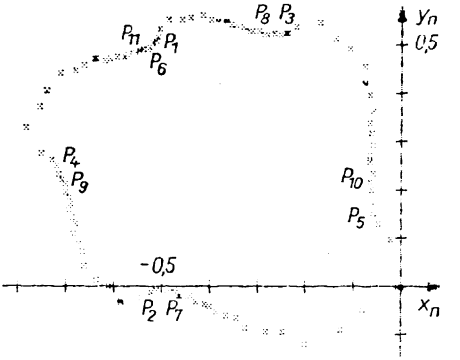


Bild 2. $S_{300}(-0{,}51;\ 0{,}51)$

$x_9 = -0{,}70;\quad y_9 = 0{,}21$
$x_{10} = -0{,}06;\quad y_{10} = 0{,}22$
$x_{11} = -0{,}55;\quad y_{11} = 0{,}48.$

Daraus entsteht die Sequenz S_{300} $(-0{,}51;\ 0{,}51)$ – vgl. Bild 2 – die ein ganz anderes Verhalten als S_{300} $(-0{,}50;\ 0{,}50)$ zeigt.
Selbst wenn wir die Genauigkeit auf 10^{-8} und N entsprechend erhöhen, gilt für $n > 1000$: $P_{n+101} = P_n$.
Anstelle eines einzigen Anziehungspunktes gibt es hier 101, die den Punkten in Bild 2 so nahe liegen, daß sie bei unserer Zeichengenauigkeit mit denen übereinstimmen.

(E3): $\boldsymbol{x = -0{,}52;\ y = 0{,}52}$

Die aus (2) nur auf zwei Stellen genau gewonnenen Werte $x_2, \ldots, y_8$ weichen von denen in Tabelle 1 höchstens um 0,07 ab. Dann folgt

$x_9 = -0{,}75;\quad y_9 = 0{,}21$
$x_{10} = 0{,}00;\quad y_{10} = 0{,}20$
$x_{11} = -0{,}56;\quad y_{11} = 0{,}52.$

Die nächstfolgenden Punkte P_{12}, P_{14}, P_{15}, P_{16}, P_{17} fallen noch in den bei Bild 3 festgelegten Ausschnitt, der dem in Bild 1 und in Bild 2 entspricht. Die übrigen Punkte drängen nach außen. So gilt

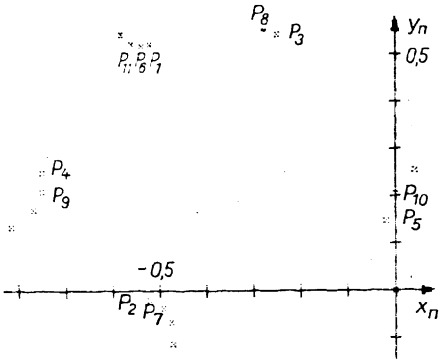


Bild 3. Ausschnitt von $S_{300}(-0{,}52\,;\,0{,}52)$

$x_{32} = 6{,}44;\quad y_{32} = 0{,}13$

$x_{33} = 40{,}88;\quad y_{33} = 2{,}16$

$x_{34} = 1666{,}36;\quad y_{34} = 177{,}27$

$x_{35} = 2\,745\,331{,}80;\quad y_{35} = 590\,788{,}14.$

Bereits nach 36 Iterationen ist die Schranke 10^{10} überschritten. Wir sprechen hierbei von einer unbeschränkten Sequenz.

2. Unbeschränkte Sequenzen

Im folgenden bezeichne $S\,(x;\,y)$ eine durch (2) erzeugte Punktfolge beliebig großer Länge. Nach Beispiel (E3) ist $S\,(-0{,}52\,;\,0{,}52)$ aber auch schon $S(1\,;1)$ eine unbeschränkte Sequenz, da der Abstand r_n zwischen P_n und P_0 mit wachsendem n über alle Grenzen wächst.

Mit einem Computer sind wir natürlich nur in der Lage, Sequenzen von endlicher Länge zu erzeugen. Deshalb wäre das Vorhaben, die Gesamtheit aller unbeschränkten Sequenzen charakterisieren zu wollen, von vornherein zum Scheitern verurteilt, wenn wir nicht über folgende schöne Aussage verfügen würden:

Lemma

Die Sequenz $S\,(x;\,y)$ ist genau dann unbeschränkt, wenn eine natürliche Zahl m existiert, so daß gilt:

$$r_m^2\,(x;\,y) := x_m^2 + y_m^2 > 4. \qquad (3)$$

Wir können unser Lemma einfach beweisen, wenn wir eine Punktfolge in der Ebene gemäß (2) als Folge komplexer Zahlen darstellen:

$$z_{n+1} = z_n^2 + z \text{ mit } z = x + \mathrm{i}y \qquad (4)$$

und den Abstand $r_n = |z_n|$ nach unten abschätzen, wobei die Vergleichsfolge noch bestimmt divergiert.

Mit G_n bezeichnen wir die Gesamtheit der Startpunkte aller unbeschränkten Sequenzen, bei denen (3) erstmalig durch P_n erfüllt wird. Wegen der Monotonie der Folge (r_n) gilt

$$G_n = \{(x;\,y)\,|\,r_{n-1}(x;\,y) \leqq 2 < r_n(x;\,y)\}.$$

Offenbar ist $G_1 = \{(x;\,y)\mid x^2 + y^2 > 4\}$. Die Punkte außerhalb K_2 – der abgeschlossenen Kreisscheibe vom Radius 2 um den Nullpunkt – sind also Startpunkte unbeschränkter Sequenzen.

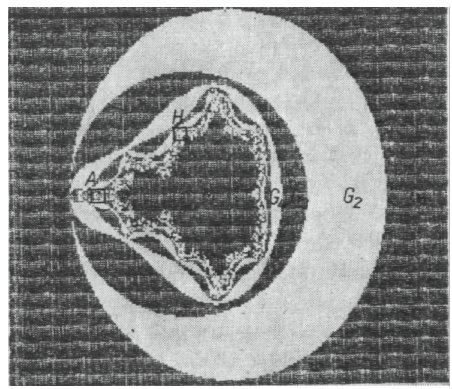


Bild 4. Unbeschränkte Sequenzen

Auf Bild 4 ist G_2 als weiße Mondsichel dargestellt. G_2 enthält insbesondere $P^* = (1\,;\,1)$ – den Startpunkt von $S_N\,(1\,;\,1)$ – denn $r_1 = \sqrt{2} < 2 < \sqrt{10} = r_2$. Danach folgen im Wechsel schwarze und weiße Gebiete. Eine in einem wei-

ßen Gebiet startende Sequenz erreicht das Außengebiet G_1 erstmalig in einem Punkt mit einem geraden Index. Die Sequenz von Beispiel (E3) startet in einem schwarzen Gebiet. Wegen

$1{,}46 < r_{30} < 2 < 2{,}63 < r_{31}$ gilt

$(-1{,}52; 1{,}52) \in G_{31}$.

3. Beschränkte Sequenzen

Eine Sequenz $S(x; y)$ heißt N-beschränkt, wenn keiner ihrer ersten N Punkte in G_1 liegt. Die Menge der Startpunkte N-beschränkter Sequenzen bezeichnen wir mit M_N:

$M_N = \{(x; y) \mid x_n^2 + y_n^2 \leqq 4$ für $n = 1, \ldots, N\}$.

Eine N-beschränkte Sequenz kann nach unserem Lemma auch unbeschränkt sein, denn es könnte ein $m > N$ existieren mit $P_m \in G_1$. Wenn man allerdings N genügend groß wählt, so ändert sich M_N bei vorgegebener Zeichengenauigkeit nicht mehr, und wir erhalten die MANDELBROT-Menge M, die Gesamtheit der Startpunkte von Sequenzen, deren Punkte sämtlich auf K_2 liegen.

Bild 5 zeigt die Menge $M_{50} \approx M$. Offenbar enthält M auch den Startpunkt des Wirbels von Beispiel (E1) und den Startpunkt $(-0{,}51; 0{,}51)$ von Beispiel (E2), denn $S(-0{,}50; 0{,}50)$ und $S(-0{,}51; 0{,}51)$ sind beschränkte Sequenzen.

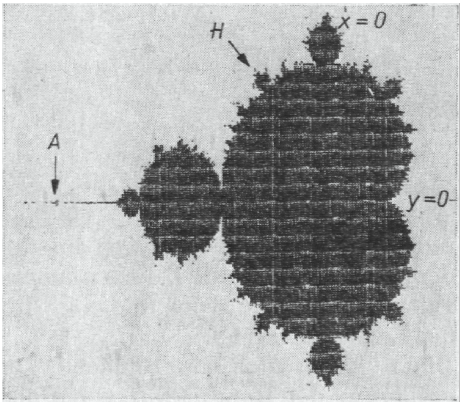


Bild 5. Beschränkte Sequenzen

Dreht man Bild 5 im Uhrzeigersinn um 90°, entsteht eine Figur, die »Apfelmännchen« genannt wird (vgl. [2]). Interessant ist insbesondere die bei mathematischen Figuren selten anzutreffende Kontur des »Randes«, die aus einer Vielzahl von »Wucherungen«, aus vorgelagerten »Punktehaufen« sowie einer »Spitze mit Abschluß« besteht. Wir wollen uns diesen »Abschluß (A)« sowie den »Haufen (H)« etwas näher betrachten. Dazu benötigen wir eine »Lupe«, die wir durch Ausschnittsvergrößerung realisieren.

4. Programmaufbau

Wir stellen uns das Ziel, einen beliebigen Ausschnitt (mit achsenparallelem Rand) von Bild 4 herzustellen. Auf Bild 4 bedeutet ein (schwarzer) Punkt an der Stelle $(x; y)$, daß $(x; y)$ Startpunkt einer Sequenz ist, der in M_{50} oder in G_{2n-1} liegt, wobei $n \in \{1, 2, \ldots, 25\}$ ist. Ist die Stelle $(x; y)$ frei (weiß), so gilt $(x; y) \in G_{2n}$. Wir müssen daher für jeden Punkt $(x; y)$ des zu betrachtenden Ausschnitts prüfen, ob die Sequenz $S(x; y)$ unbeschränkt oder N-beschränkt ist. Das geschieht nach folgendem

Algorithmus:

(A1) Wenn $x^2 + y^2 > 4$ gilt, so ist $(x; y) \in G_1$.

(A2) Berechne x_n, y_n nach (2), beginnend mit $x_1 = x$, $y_1 = y$, so lange bis erstmalig $x_n^2 + y_n^2 > 4$ ausfällt; wenn $n \leqq N$, dann ist $(x; y) \in G_n$, sonst wird mit $n = N$ abgebrochen.

(A3) Wenn $x_N^2 + y_N^2 \leqq 4$ gilt, so ist $(x; y) \in M_N$.

Für die Ausgabe des Ergebnisses schlagen wir zwei Varianten vor:

Variante 1

Wenn $(x; y) \in M_N$ oder $(x; y) \in G_{2m-1}$ gilt, so setze an die Stelle $(x; y)$ einen (schwarzen)Punkt.

Variante 2

Wenn $(x; y) \in M_N$ gilt, so setze an die Stelle $(x; y)$ einen (schwarzen) Punkt.

Auch bei der Variante 2 ist es günstig, mit den Gebieten G_n zu arbeiten. Das Ergebnis $(x; y) \in G_n$ mit $n < N$ bedeutet, die Sequenz $S(x; y)$ ist bereits nach n Iterationen als unbeschränkt erkannt. Die restlichen $N-n$ Iterationen entfallen. Es gilt also $(x; y) \notin M_N$. Die beiden Varianten sollte man deshalb in einem einzigen Programm realisieren (etwa durch Setzen eines Flags).

Offenbar wurde Bild 4 mit Variante 1 und Bild 5 mit Variante 2 gewonnen. Sehen wir von der Einfärbung der Gebiete G_{2m-1} in Bild 4 ab, so ist Bild 5 eine zweifache Vergrößerung eines Ausschnittes von Bild 4.

Wir haben noch den Ausschnitt festzulegen und dessen Abtastung zu organisieren. Dazu führen wir sogenannte Ausschnittsbegrenzer ein: $x_{\min}, x_{\max}, y_{\min}, y_{\max}$.

Da das Verhältnis q der Anzahlen der s ansteuerbaren Bildpunktspalten zu den z Bildpunktzeilen durch den Computer festgelegt ist, können wir den Bildschirm – Verzerrungen vermeidend – nur dann voll auslasten, wenn wir uns auf Ausschnitte beschränken, bei denen gilt:

$$x_{\max} - x_{\min} = (y_{\max} - y_{\min}) \cdot q.$$

Aus drucktechnischen Gründen haben wir bei unseren Bildern ein Verhältnis von 4:3 realisiert. Für den KC85/2 ist $q = 1{,}25$. Den 320×256 Bildpunkten entsprechen hierbei 81920 Punkte des Ausschnitts, die wir als Startpunkte von Sequenzen auffassen und untersuchen können. Beginnend in der linken oberen Ecke des Bildschirms, wird zuerst die Sequenz $S_N(x_{\min}; y_{\max})$ mit unserem Algorithmus untersucht. Danach wird zum Startpunkt $(x_{\min} + \Delta x; y_{\max})$ übergegangen, wobei $(s-1)\,\Delta x = x_{\max} - x_{\min}$ gilt. Jeweils um Δx weiterschreitend, werden bis hin zu $(x_{\max}; y_{\max})$ alle s Bildpunkte der oberen Zeile erfaßt. Dann wird zur zweiten Zeile gesprungen und mit $(x_{\min}; y_{\max} - \Delta y)$ begonnen, diese abzuarbeiten, wobei $(z-1)\,\Delta y = y_{\max} - y_{\min}$ gewählt ist. Entsprechend fortfahrend, werden schließlich alle $s \cdot z$ Punkte abgetastet und die entsprechenden Sequenzen eingeordnet.

Die Maximalzahl der Iterationen N ist mindestens gleich 10 zu wählen. In Abhängigkeit vom Maßstab ist für eine gute Grafik ein N von 50, 100 oder 150 erforderlich. Allerdings kann das bereits zu mehr als einer Million Iterationen führen, so daß lange Rechenzeiten (in den Nachtstunden!) unvermeidlich sind.

5. Verborgene Apfelmännchen

Wir untersuchen nun zwei Stellen des Randes der MANDELBROT-Menge von Bild 5, das mit den Parametern $N = 50/x_{\min} = -2{,}10/x_{\max} = 0{,}7/y_{\min} = -0{,}96/y_{\max} = 0{,}96$ mit der Variante 2 erzeugt wurde. Zunächst betrachten wir die Stelle H von Bild 5 mit 53facher Vergrößerung. Das Programm, mit Variante 1 gefahren, liefert mit den Werten $N = 50/x_{\min} = -0{,}620/x_{\max} = -0{,}567/y_{\min} = 0{,}60/y_{\max} = 64$ gerade Bild 6. Hier sind Figuren entstanden, die an eine Herde Zebras erinnern, von denen nur die Beine und ein Stück der Leiber zu erkennen sind.

Bild 6 kann auch als 106fache Vergrößerung des Randteils H von Bild 1 angesehen werden, das mit den Para-

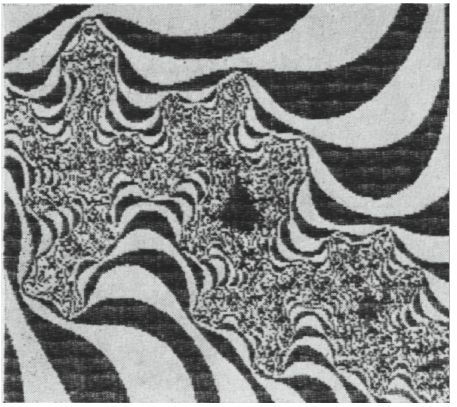


Bild 6. Randausschnitt *II*

meterwerten $N = 50$, $x_{min} = -2{,}8/x_{max} = 2{,}8/y_{min} = -2{,}1/y_{max} = 2{,}1$ aufgebaut wurde. Der ausgeprägte Wechsel zwischen weißen und schwarzen Gebieten läßt auf unbeschränkte Sequenzen schließen. Nur ein schwarzes Dreieck inmitten des Bildes müßte noch näher untersucht werden. Eine weitere 3fache Vergrößerung liefert

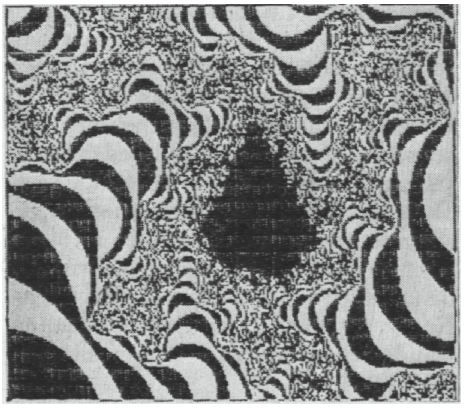


Bild 7. Vergrößerter Randausschnitt *II*

Bild 7 ($N = 50/x_{min} = -0{,}602/x_{max} = -0{,}584/y_{min} = 0{,}613/y_{max} = 0{,}627$). Das schwarze Gebiet in der Bildmitte hat nun die Form eines Sektors mit unscharfem Rand. Behalten wir den Ausschnitt bei, erhöhen aber die Maximalzahl der Iterationen auf $N = 100$, erhalten wir mit Version 2 Bild 8. Das ist wieder ein Apfelmännchen, welches hier völlig getrennt (isoliert) vom »großen Apfelmännchen« liegt.

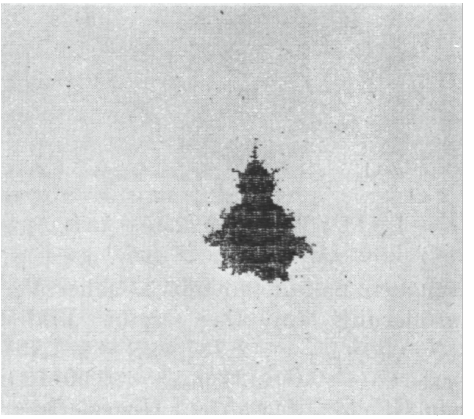


Bild 8. Der Kern von *II*

Die Stelle *A* von Bild 5 soll nun 28fach vergrößert werden. Wir bekommen Bild 9 ($N = 50/x_{min} = -1{,}8/x_{max} = -1{,}7/y_{min} = -0{,}0375/y_{max} = 0{,}0375$) und erkennen ein wohlgeformtes Apfelmännchen, an dessen Spitze zwei Punkte erkennbar sind. Wir bezeichnen den entfernter liegenden mit *Sp* und

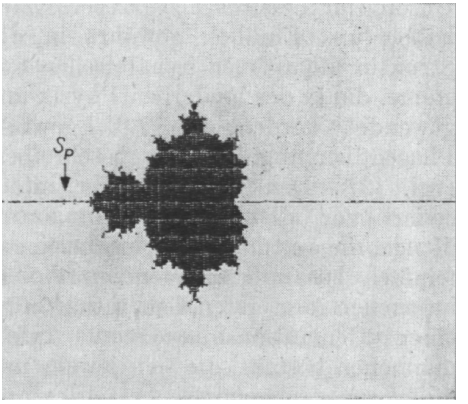


Bild 9. Vergrößerter Randausschnitt *A*

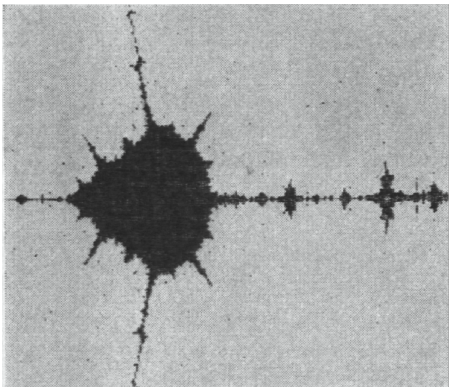


Bild 10. Vergrößerung von *Sp*

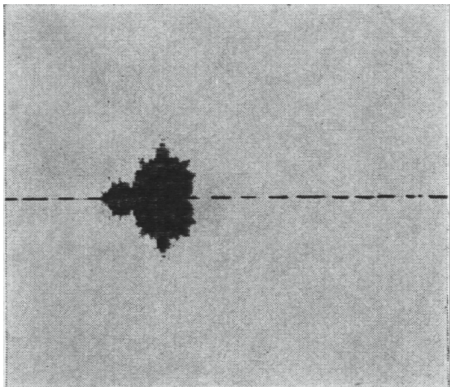


Bild 11. Der Kern von *Sp*

schauen uns diesen mit 33facher Vergrößerung an. Das ergibt Bild 10 ($N = 50/x_{\min} = -1{,}787/x_{\max} = -1{,}784/y_{\min} = -0{,}001125/y_{\max} = 0{,}001125$), ein Gebilde, das an ein Ungeziefer erinnert.

Da hierbei die Vergrößerung des entsprechenden Ausschnitts von Bild 4 bereits den Faktor 1867 erreicht hat, ist es angebracht, die Iterationszahl zu erhöhen. $N = 150$ (bei gleichem Ausschnitt wie in Bild 10) liefert wieder ein Apfelmännchen, dieses in dritter Generation (vgl. Bild 11).

6. Ausblick

Unsere mathematischen Experimente haben uns Einblick gewährt in die Struktur komplexer dynamischer Systeme, die in der modernen Physik angewendet werden, um z. B. gewisse Phasenübergänge zu beschreiben (vgl. [2]). Den Mathematiker interessiert vor allem die MANDELBROT-Menge, die aus unserer Sicht aus einem (einfach zusammenhängenden) Apfelmännchen (der ersten Generation) und einer Generationsfolge von Apfelmännchen besteht, die sich jeweils um die vorangegangene Generation gruppieren und immer kleiner werden.

Der Computerfreak hat hier die einmalige Gelegenheit, seinen Computer hinsichtlich des Auflösungsvermögens bzw. der Vergrößerungsfähigkeit eines speziellen Bildes (unser Bild 4) bis an dessen Leistungsgrenze zu führen. So kann er selbst prüfen, ob eine millionfache Vergrößerung noch zu einem erkennbaren Apfelmännchen (welcher Generation?) führt.

Schließlich kann auch der künstlerisch Interessierte auf seine Kosten kommen und vom Standpunkt der Ästhetik den Rand des Apfelmännchens der ersten Generation unter die Lupe nehmen, um seltsame Gebilde bzw. Landschaften (wie unser Bild 6) zu entdecken.

Derartige Effekte werden enorm gesteigert, wenn man über einen farbigen Monitor verfügt. So brauchen wir nur den Punkten der Menge G_n die Farbe Nr. n zu geben, und wir erhalten wunderschöne farbige Kompositionen (vgl. [2]).

(Literatur s. S. 27)

Autor:

Prof. Dr. sc. nat. Hans-Joachim Girlich

Karl-Marx-Universität Leipzig
Sektion Mathematik

Bestimmte Integrale

Die Berechnung des bestimmten Integrals einer stetigen Funktion kann mit einem Kleincomputer oder mit einem Personalcomputer schnell und sicher durchgeführt werden. In diesem Beitrag werden Programme für den KC 85/2 bzw. KC 85/3 und für den PC 1715 in der Programmiersprache BASIC vorgestellt.

1. Grundlagen

Das bestimmte Integral I einer Funktion $y = f(x)$ in den Grenzen von x_u bis x_o

$$I = \int_{x_u}^{x_o} f(x)\,\mathrm{d}x \qquad (1)$$

läßt sich nach Bild 1 als die Fläche darstellen, die von dem Stück der Abszissenachse, das zwischen der unteren Grenze x_u und der oberen Grenze x_o liegt, den beiden Ordinaten $f(x_u)$ und $f(x_o)$ an den Grenzen und von der Kurve des Integranden $y = f(x)$ begrenzt wird.

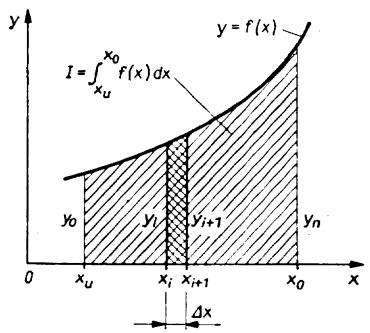


Bild 1. Bestimmtes Integral als Fläche

Hat die Funktion $y = f(x)$ im Integrationsbereich Nullstellen, so ist I die Differenz der Beträge der Teilflächen oberhalb und der Teilflächen unterhalb der x-Achse.

Die Lösung des Integrals I ist eine reelle Zahl, wenn die zu integrierende Funktion $f(x)$ im Integrationsbereich stetig ist und die Grenzen des Integrationsbereichs reelle Zahlen sind. Es bietet sich an, diese reelle Zahl I numerisch zu berechnen und für diese Berechnung einen Klein- oder Personalcomputer zu verwenden.

Teilt man nach Bild 1 den Integrationsbereich in viele schmale Streifen der Breite Δx, so wird der Flächeninhalt eines dieser Streifen näherungsweise

$$\Delta A_i = \Delta I_i = \frac{1}{2}(y_i + y_{i+1})\,\Delta x. \qquad (2)$$

Dabei wird die Fläche des Streifens näherungsweise als die Fläche eines Trapezes angesehen. Die Näherung besteht darin, daß die in Wahrheit gekrümmte obere Begrenzungskurve durch eine Gerade ersetzt wird. Als Summe aller Teilflächen erhält man die Gesamtfläche und damit eine Näherung für den numerischen Wert des

Integrals I:

$$I \approx \sum_{i=1}^{n} \Delta I_i = \sum_{i=1}^{n} \Delta A_i \qquad (3)$$

Wird Gl. (2) in Gl. (3) eingesetzt, dann ergibt sich

$$I \approx$$

$$\left(\frac{1}{2} y_0 + y_1 + y_2 + \cdots + y_{n-1} + \frac{1}{2} y_n\right)$$

$$\times \Delta x. \qquad (4)$$

$$\boxed{I \approx \left[\sum_{i=1}^{n-1} y_i + \frac{1}{2}(y_0 + y_n)\right] \Delta x} \qquad (5)$$

Wegen der Annäherung der Flächenstreifen durch Trapeze wird Gl. (5) als *Trapezformel* bezeichnet.

Von dem englischen Mathematiker Thomas Simpson (1710 bis 1761) stammt die Idee, anstelle der Begrenzung des Flächenstreifens durch eine gerade Linie eine Parabel 2. Grades zu wählen. Es ist einleuchtend, daß sich eine Parabel dem wirklichen, gekrümmten Verlauf der Funktion $y = f(x)$ besser anpassen wird als eine Gerade.

Auf einem ähnlichen Weg wie bei der Trapezformel erhält man

$$I \approx \frac{\Delta x}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + \cdots$$

$$+ 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_0)]. \qquad (6)$$

Etwas umgeordnet wird diese Formel als Simpsonsche Formel bezeichnet:

$$\boxed{\begin{aligned} I \approx{}& \frac{\Delta x}{3} \\ &[y_0 + y_n + 4(y_1 + y_3 + \cdots + y_{n-1}) \\ &\quad + 2(y_2 + y_4 + \cdots + y_{n-2})] \end{aligned}} \qquad (7)$$

Wählen wir für die Summe der Ordinaten mit ungeradzahligem Index die Bezeichnung u_n und für die Summe der Ordinaten mit geradzahligem Index die Bezeichnung g_n, so kann Gl. (7) vereinfacht geschrieben werden:

$$\boxed{I \approx \frac{\Delta x}{3}(y_0 + y_n + 4u_n + 2g_n)} \qquad (8)$$

Hierbei sind

$$u_n = y_1 + y_3 + \cdots + y_{n-1} \qquad (9)$$

und

$$g_n = y_2 + y_4 + \cdots + y_{n-2}. \qquad (10)$$

Die Anzahl n der Flächenstreifen muß bei der Simpsonschen Näherung *geradzahlig* sein.

Wie bei allen numerischen Rechnungen, so entsteht auch hier die Frage, mit welcher Genauigkeit der berechnete Wert von I den wahren Wert von I wiederspiegelt.

Im allgemeinen kann man die Genauigkeit verbessern, wenn die Anzahl n der Flächenstreifen möglichst groß gewählt wird. Zu viele Flächenstreifen können jedoch das Gegenteil bewirken. Wegen der unvermeidlichen Rundungsfehler bei jeder einzelnen Berechnung können sich die Rundungsfehler vieler Berechnungen zu einem so großen Betrag addieren, daß sie nicht mehr zu vernachlässigen sind. Es gilt also, ein vernünftiges Maß für die Anzahl der Flächenstreifen zu finden. Eine Möglichkeit wäre, die Berechnung z.B. mit $n = 4$ zu beginnen und n so lange systematisch um zwei auf $n = 6, 8, 10, \ldots$ zu erhöhen, bis der Unterschied ΔI von zwei aufeinanderfolgenden Näherungswerten für I einen vorgegebenen Wert nicht mehr übersteigt.

Bei diesem Verfahren müßte man jedoch alle Ordinaten y_i nach jeder Erhöhung von n wieder neu berechnen, ohne daß es möglich wäre, die vorher berechneten y_i in der neuen Berechnung wieder zu verwenden. Man könnte folglich auch gleich eine aus der Er-

fahrung gewonnene hinreichend große Zahl für n wählen und nur einmal rechnen. Allerdings fehlte dann jede Information über die Genauigkeit der Lösung.

Für die Programmierung der Berechnung bestimmter Integrale soll deshalb hier eine Methode gewählt werden, bei der die Annäherung an den wahren Wert des Integrals möglichst gut ist, eine einfache Fehlerabschätzung durchgeführt werden kann und die Rechenzeiten dadurch in erträglichen Grenzen gehalten werden, daß alle bei einer vorhergehenden Rechnung bestimmten Funktionswerte bei der nachfolgenden Rechnung wieder verwendet werden.

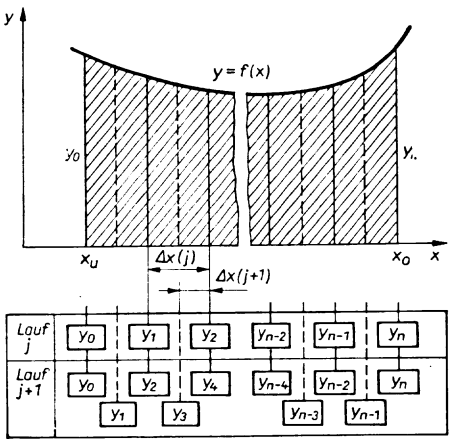


Bild 2. Intervallhalbierung

2. Algorithmus

Die genannten Forderungen lassen sich mit einem Algorithmus erfüllen, dessen Prinzip in Bild 2 dargestellt ist.

Im Durchlauf durch eine Schleife, der hier als der Durchlauf j bezeichnet ist, wurde eine Näherung für das Integral I ermittelt, und die Summen u_n und g_n nach Gl. (9) bzw. Gl. (10) liegen vor. Um eine verbesserte Näherung für I zu erhalten, wird die Intervallbreite Δx halbiert, indem die Anzahl n verdoppelt wird. Wie Bild 2 zeigt, sind für den Schleifendurchlauf $j+1$ nur noch die Ordinatenwerte mit dem neuen ungeradzahligen Index zu berechnen, die anderen wurden bereits im Durchlauf j ermittelt. Sie werden jetzt wieder verwendet. Das Verfahren der Intervallhalbierung wird so lange fortgesetzt, bis eine genügend genaue Näherung gefunden ist.

Dabei erhebt sich die Frage, wie groß die Abweichung des Näherungswertes vom wahren Wert ist. Exakt beantworten kann man diese Frage nur, wenn der wahre Wert bekannt ist. Wäre er bekannt, brauchten wir nicht den Computer für die Lösung zu bemühen. Es ist also notwendig, eine Abschätzung für den Fehler unserer Näherungslösung zu finden. Gleichzeitig ist festzulegen, daß die Berechnung dann abgebrochen werden soll, wenn die Verbesserung einer Näherungslösung durch die folgende Rechnung kleiner als ein vorgegebener relativer Fehler ist. Wir lösen dieses Problem, indem wir annehmen, daß der Fehler, den der zuletzt berechnete Näherungswert von I gegenüber dem wahren Wert von I hat, kleiner ist als der Betrag der Verbesserung, den wir durch den letzten Durchlauf erreichen, und brechen die Rechnung ab, wenn

$$\Delta I = |I_j - I_{j+1}| \qquad (11)$$

kleiner wird als ein gewählter relativer Fehler, den das Ergebnis haben darf.

Um zu vermeiden, daß das Programm in einer unendlichen Schleife läuft, falls das Abbruchkriterium nicht erreicht werden kann, wird die Berechnung auch dann abgebrochen, wenn eine gewählte Zahl n_{Grenz} für die Anzahl der Teilintervalle überschritten wird. Im Ergebnis wird dann mitgeteilt, daß der gewählte relative oder prozentuale Fehler nicht erreicht oder unterschritten werden konnte. In der Regel ist diese Mitteilung ein Hinweis

darauf, daß sich im Integrationsbereich eine Polstelle oder eine Lücke des Integranden befindet.

3. Programmbeschreibung

Das Struktogramm des Bildes 3 zeigt den prinzipiellen Aufbau des Programms. In Bild 4 ist die Anweisungsliste für Kleincomputer und in Bild 5 die Anweisungsliste für den Personalcomputer wiedergegeben.
Für den PC 1715 wurden lediglich die Druckbefehle eingefügt. Auch auf Computern, die andere BASIC-Dialekte ver-

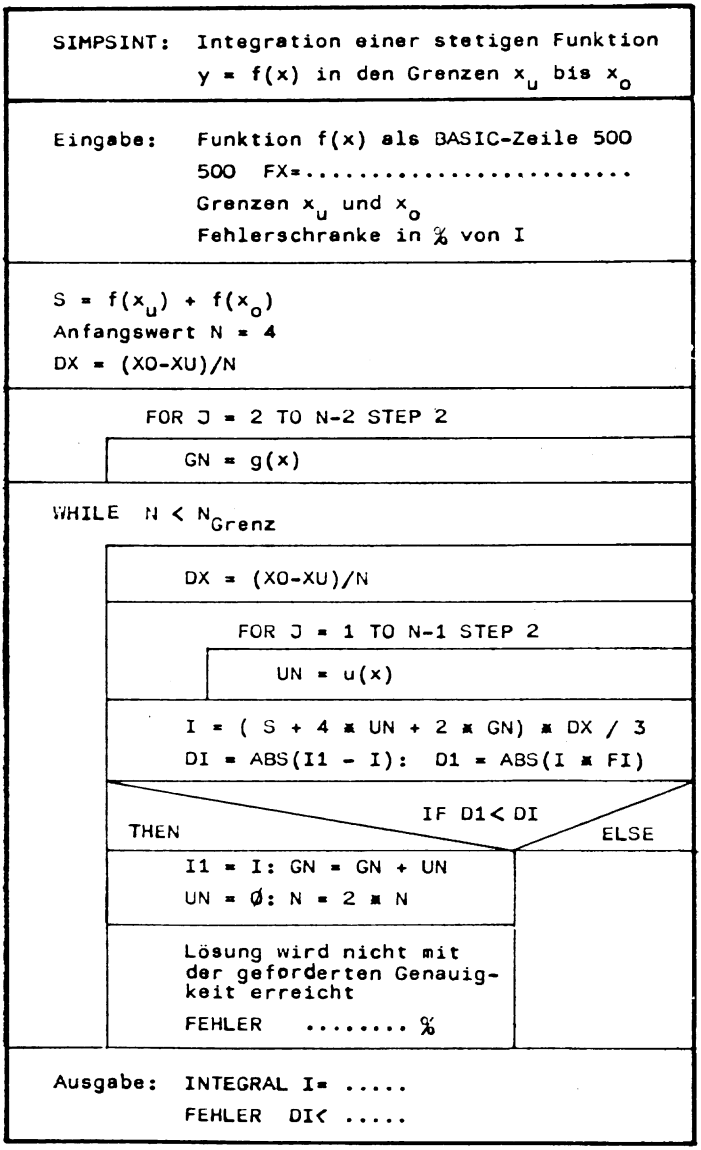


Bild 3
Struktogramm

wenden, läuft dieses Programm ohne wesentliche Änderungen.

Die zu integrierende Funktion muß in Zeile 500 als gültige BASIC-Anweisung 500 FX = ... programmiert sein.

Nach dem Start mit RUN ENTER sind die untere Grenze x_u und die obere

```
10  REM   "SIMPSINT"
20  REM   ---------------------------------------
30  REM   BESTIMMTES INTEGRAL MIT
40  REM   KC 85/2 ODER KC 85/3
50  REM   ---------------------------------------
60  CLS: WINDOW 5, 30, 5, 35
70  PRINT "A C H T U N G !"
80  PRINT: PRINT "DER INTEGRAND MUSS ALS"
90  PRINT: PRINT "ZEILE 500  FX = ....."
100 PRINT: PRINT "PROGRAMMIERT SEIN"
110 PAUSE 30: CLS
120 INPUT "UNTERE GRENZE XU = "; XU
130 INPUT "OBERE  GRENZE XO = "; XO
140 INPUT "ZUL. FEHLER IN % VON I = "; F
150 PRINT: PRINT: FI = F/100
160 X = XO: GOSUB 500: S = FX
170 X = XU: GOSUB 500: S = S + FX
180 N = 4: DX = (XO - XU)/N
190 FOR J=2 TO N-2 STEP 2
200   GOSUB 490: GN = FX + GN
210 NEXT J
220 DX = (XO - XU)/N
230 FOR J=1 TO N-1 STEP 2
240   GOSUB 490: UN = FX + UN
250 NEXT J
260 I = (S + 4*UN + 2*GN)*DX/3
270 DI = ABS(I1 -I): D1 = ABS(FI*I)
280 IF DI< D1 THEN 340
290 IF N >512 THEN 330
300 I1 = I: GN = GN + UN: UN = 0: N = 2*N
310 GOTO 220
320 REM   AUSGABE ------------------------------
330 PRINT "FEHLER      >   "; F; " % ": PRINT
340 PRINT "INTEGRAL I  =   "; I: PRINT
350 PRINT "FEHLER      <   "; ABS(I1 - I)
360 GOTO 520
370 REM    UPRO --------------------------------
490 X = XU + J*DX
500 REM    RESERVIERT FUER FX = ................
510 RETURN
520 END
```

Bild 4
Programm für
KC 85/2 und KC 85/3

Grenze x_o einzugeben. Als letztes wird ein zulässiger Fehler in Prozent von I eingegeben. Das Ergebnis wird gemeinsam mit der Fehlerabschätzung angezeigt beziehungsweise ausgedruckt.
Das Programm bricht ab, wenn die Anzahl der Intervalle größer als 2×512 (Zeile 290 in Bild 4 bzw. 300 in Bild 5) wird. Jeder Anwender kann diese Zahl nach eigenem Ermessen ändern.

4. Möglichkeiten und Grenzen

Mit diesem Programm lassen sich schnell und mit hoher Genauigkeit stetige Funktionen integrieren. Bei komplizierten Stammfunktionen ist die

```
10    REM   "SIMPSINT"
20    REM   ------------------------------------------------------------
30    REM   BESTIMMTES INTEGRAL MIT
40    REM   PC 1715 UND DRUCKER
50    REM   BASIC - VERSION
60    REM   ------------------------------------------------------------
70    PRINT: PRINT: PRINT CHR$(12): PRINT TAB(10) "A c h t u n g  !"
80    PRINT: PRINT TAB(10) "Der Integrand muss als BASIC-Zeile"
90    PRINT: PRINT TAB(10) "500   FX = ......................."
100   PRINT: PRINT TAB(10) "programmiert sein"
110   PRINT: PRINT TAB(30) "> E N T E R <"
120   WZ$=INPUT$(1)
130   INPUT "Untere Grenze Xu = ";XU
140   INPUT "Obere  Grenze Xo = ";XO
150   INPUT "zul. Fehler in % von I = "; F
160   PRINT: PRINT: FI=F/100
170   X=XO: GOSUB 500: S=FX
180   X=XU: GOSUB 500: S=FX+S
190   N=4:  DX=(XO-XU)/N
200   FOR J=2 TO N-2 STEP 2
210     GOSUB 490: GN=FX+GN
220   NEXT J
230   DX=(XO-XU)/N
240   FOR J=1 TO N-1 STEP 2
250     GOSUB 490: UN=FX+UN
260   NEXT J
270   I=(S+4*UN+2*GN)*DX/3
280   DI=ABS(I1-I): D1=ABS(FI*I)
290   IF DI<D1 THEN 350
300   IF N>512 THEN 340
310   I1=I: GN=GN+UN: UN=0: N=2*N
320   GOTO 230
330   REM   Ausgabe auf dem Bildschirm ------------------------------
340   PRINT: PRINT TAB(10) "Fehler     > "; F; " % von I"
350   PRINT: PRINT TAB(10) "Integral I = "; I
360   PRINT: PRINT TAB(10) "Fehler     < "; ABS(I1-I)
370   REM   Ausgabe auf dem Drucker ---------------------------------
380   LPRINT "==================================="
390   LPRINT "Berechnung des bestimmten Integrals"
400   LPRINT "==================================="
410   LPRINT: LPRINT "in den Grenzen"
420   LPRINT "von   Xu = "; XU; "  bis  Xo = "; XO: LPRINT
430   IF N>512 THEN LPRINT "Genauigkeit nicht erreichbar": LPRINT
440   LPRINT "Integral I = "; I
450   LPRINT "Fehler     < "; ABS(I1-I)
460   LPRINT "fuer den programmierten Integranden"
470   LLIST 500
480   REM   Upro ----------------------------------------------------
490   X=XU+J*DX
510   RETURN
520   END
```

Bild 5. Programm für PC 1715 mit Drucker

Rechenzeit meist noch geringer als die Zeit, die man benötigte, um den Wert des bestimmten Integrals mit dem Taschenrechner durch Einsetzen der beiden Grenzen zu berechnen, falls es vorher überhaupt gelungen ist, die gegebene Funktion zu integrieren.
Soll der Flächeninhalt, der zwischen zwei Kurven, die durch die Funktionen $y = f_1(x)$ und $y = f_2(x)$ gegeben sind, liegt, ermittelt werden, dann wird in Zeile 500 einfach die Differenz beider Funktionen als neue Funktion $f(x)$ programmiert:

$$f(x) = f_1(x) - f_2(x) \qquad (12)$$

Das Programm versagt, wenn im Integrationsbereich Unstetigkeitsstellen des Integranden liegen oder wenn die Funktionswerte von $f(x)$ an den Grenzen gegen Unendlich gehen. Obwohl man in manchen Fällen auch für solche Integrale brauchbare Näherungen erhalten kann, sollten nur stetige Funktionen mit diesem Programm integriert werden.
Liegt eine Integrationsgrenze oder die Grenze eines Flächenstreifens genau auf einer Unstetigkeitsstelle, dann wird ein Fehler in Zeile 500 angezeigt. Das geschieht z.B. bei dem Versuch einer Division durch Null.
Erkennt man nicht von vornherein Unstetigkeiten im Integrationsbereich, dann bricht das Programm ab, wenn die vorgegebene Höchstzahl für die Anzahl der Intervalle überschritten wird, und bei der Ausgabe wird als Warnung angegeben, daß die geforderte Genauigkeit nicht zu erreichen war.
Es kann also kaum zu Fehleraktionen des Programms kommen, die der Anwender nicht bemerken würde.
Um Änderungen und Anpassungen dieses Programms zu erleichtern, ist in Bild 6 die Liste der verwendeten Variablen angegeben.

Variable	Bedeutung
D1	Absolutwert des zulässigen Fehlers von I
DI	Betrag der Differenz von I aus zwei aufeinanderfolgenden Durchläufen
DX	Breite Δx eines Teilintervalls
F	zulässiger Fehler in % von I
F1	relativer Fehler von I
FX	Integrand bzw. Funktionswert des Integranden
GN	Summe der Ordinaten y_i mit geradzahligem Index i
I	Näherungswert für I nach dem letzten Durchlauf
I1	Näherungswert für I nach dem vorletzten Durchlauf
J	Schleifenvariable
N	Anzahl der Intervalle
S	Summe $y_0 + y_1$
UN	Summe der Ordinaten y_i mit ungeradzahligem Index i
X	Argument des Integranden
XO	obere Grenze x_o
XU	untere Grenze x_u

Bild 6. Liste der Variablen

5. Anwendungsbeispiele

Aus der Fülle der möglichen Aufgaben werden vier Beispiele ausgewählt. Sie dienen dem, der dieses Programm übernehmen möchte, als Kontrollbeispiele und zeigen gleichzeitig die Effizienz dieses Programms.

1. Beispiel
Das Integral

$$I = \int_{2{,}5}^{15} \frac{\mathrm{d}x}{x}$$

ist mit einem Fehler kleiner als $0{,}0001\,\%$ zu berechnen.
Zunächst wird

500 FX = 1/X

programmiert. Anschließend wird das Programm mit RUN ENTER ge-

startet. Es weist noch einmal darauf hin, daß der Integrand in Zeile 500 programmiert sein muß, und fordert zur Eingabe der Daten auf. Auf dem Bildschirm des Kleincomputers steht nach der Dateneingabe:

UNTERE GRENZE XU = 2.5
OBERE GRENZE XO = 15
ZUL. FEHLER IN % VON I = .0001

Nach 4,8 Sekunden erscheint das Ergebnis

INTEGRAL I = 1.79176
FEHLER < 8.34465E-07

Die exakte Lösung ist

$I = \ln 6 = 1{,}79175947\ldots$

Beim PC 1715 dauert es 9 Sekunden, und das Ergebnis ist so ausgedruckt, wie es Bild 7 zeigt.

```
==================================
Berechnung des bestimmten Integrals
==================================

in den Grenzen
von   Xu =  2.5   bis  Xo =  15

Integral I =  1.79176
Fehler     <  8.34465E-07
fuer den programmierten Integranden
500   FX = 1/X
```

Bild 7. Ergebnis für das 1. Beispiel mit PC 1715

2. Beispiel

Das Integral

$$I = \int \frac{\sin x}{x}\,\mathrm{d}x$$

läßt sich analytisch nicht elementar lösen. Nach einer Reihenentwicklung erhält man als Lösung

$$I = x - \frac{x^3}{3 \cdot 3!} + \frac{x^5}{5 \cdot 5!} - \frac{x^7}{7 \cdot 7!} + \cdots$$

Dieses Integral soll in den Grenzen von 1 bis 2 berechnet werden.
Jetzt ist vor dem Start

500 FX = SIN(X)/X

zu programmieren.

Bei einem zulässigen Fehler von 0,001% erhält man mit dem KC nach 1,3 Sekunden die Lösung

INTEGRAL I = .65933
FEHLER < 1.19209E-06

Der PC liefert das Ergebnis ausgedruckt nach Bild 8 in 7 Sekunden. Würde man die Reihe mit einem Taschenrechner berechnen, brauchte man ungleich länger ohne eine größere Genauigkeit zu erreichen.

```
==================================
Berechnung des bestimmten Integrals
==================================

in den Grenzen
von   Xu =  1   bis  Xo =  2

Integral I =  .65933
Fehler     <  1.2517E-06
fuer den programmierten Integranden
500   FX = SIN(X)/X
```

Bild 8. Ergebnis für das 2. Beispiel mit PC 1715

3. Beispiel

Das Integral

$$I = \int 2 \cdot x \cdot \mathrm{e}^{\cos 2 \cdot x}\,(1 - x \cdot \sin 2 \cdot x)\,\mathrm{d}x$$

ist in den Grenzen von 0 bis π zu berechnen.
Wir programmieren

500 FX = 2 *X *EXP(COS(2 *X))
*(1−X *SIN(2 *X))

und geben $\pi = 3.1415927$ bei einem zulässigen Fehler von 0,1% ein. Nach 4,6 Sekunden erscheint als Ergebnis

INTEGRAL I = 26.8293
FEHLER < .0158501

Hier können die drei letzten angezeigten Ziffern des Ergebnisses noch unsicher sein. Deshalb wird die Rechnung mit einer höheren geforderten Genauigkeit wiederholt. Bei einem zulässigen Fehler von 0,0001% liegt nach 32 Sekunden das Ergebnis

INTEGRAL I = 26.8284
FEHLER < 7.62939E-06

vor. Hieraus erkennt man, daß nun auch die letzte angezeigte Ziffer gesichert ist.
Für die gleiche Rechnung benötigt der PC 1715 29 Sekunden und druckt sogar noch die Lösung nach Bild 9 aus.
Die exakte Lösung dieses Integrals ist

$$I = \pi^2 \, e = 26,828366 \ldots$$

Es hätte schon einiger Mühe bedurft, dieses Integral ohne Computer zu lösen.

```
==================================
Berechnung des bestimmten Integrals
==================================

in den Grenzen
von   Xu =  0   bis  Xo =  3.14159

Integral I =  26.8284
Fehler     <  1.90735E-06
fuer den programmierten Integranden
500  FX = 2*X*EXP(COS(2*X))*
(1-X*SIN(2*X))
```

Bild 9. Ergebnis für das 3. Beispiel mit PC 1715

4. Beispiel

In diesem abschließenden Beispiel wird gezeigt, wie das Programm reagiert, wenn man nicht erkennt, daß im Integrationsbereich eine Polstelle des Integranden liegt.
Bei dem Integral

$$I = \int_0^2 \frac{x}{x^2 - 1} \, dx$$

wird der Nenner Null, wenn $x = 1$ ist.
Wegen des Versuchs, bei $x = 1$ durch Null zu dividieren, wird ein Fehler in Zeile 500 angezeigt.
Verändert man die untere Grenze so, daß kein Rand eines Teilintervalls den Wert 1 annehmen kann, dann tritt auch keine Division durch Null auf. Im allgemeinen gelingt dies mit einer irrationalen Zahl als Grenze. Hier soll als untere Grenze $x_u = 1/2 \sqrt{2} = 0{,}7071068$ gewählt werden. Wie der Rechnerausdruck in Bild 10 zeigt, wird ein Wert

```
==================================
Berechnung des bestimmten Integrals
==================================

in den Grenzen
von   Xu =  .707107   bis  Xo =  2

Genauigkeit nicht erreichbar

Integral I =  15.941
Fehler     <  15.1089
fuer den programmierten Integranden
500  FX = X / ( X*X - 1 )
```

Bild 10. Ergebnis für das 4. Beispiel mit PC 1715

für I berechnet. Es ist jedoch jetzt nicht mehr möglich, die geforderte Grenze für den Fehler einzuhalten. Integral I und Fehler liegen in einer Größenordnung.
Das Ergebnis ist eine deutliche Warnung und weist auf die Unstetigkeit des Integranden hin.

Autor:

Doz. Dr.-Ing. Peter Fischer

Hochschule für Verkehrswesen
Dresden

Geometrie mit dem KC 85/1 »getupft«

Teil 1: Geraden und Kreise

Will man Kleinrechner für Geometrie nutzen, so scheint der KC 85/1 (in seiner Anfangszeit als Z 9001 bezeichnet) im Nachteil zu sein. Im Unterschied zum KC 85/2 (dem früheren HC 900) und zum KC 85/3 bietet er »nur Pseudo-Grafik«. Aber erstens beschränkt sich Geometrie nicht einfach auf Grafik-Ausgabe, und zweitens kann auch der KC 85/1 bei guter rechnerischer Vorbereitung anschauliche geometrische Darstellungen liefern. Die folgenden Überlegungen sollen als Anregung hierzu dienen.

Schrittweise führen sie zum Aufstellen einiger BASIC-Programme für folgende Aufgaben: Ausgabe einer Strecke, Ausgabe anderer Kurven, Einpassung in den Bildschirm. Auch für räumliches Darstellen ergehen einige Programmierhinweise. Man kann die aufgestellten Programme unmittelbar ablaufen lassen, aber – vor allem – in verschiedener Weise abwandeln und in eigene Programme einbauen. Um dies zu erleichtern, werden Begründungen der Programmteile und vorbereitende Betrachtungen ausführlicher dargestellt.

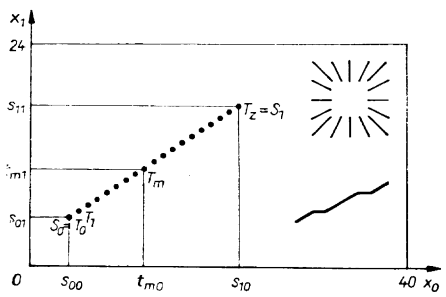


Bild 1

1. Zwei Forderungen zur Streckendarstellung

Zur Erfassung des Bildschirmbereichs (40 Spalten, 24 Zeilen) wollen wir ein x_0, x_1-Koordinatensystem wie in Bild 1 verwenden.

Anfangs- und Endpunkt der darzustellenden Strecke seien S_0, S_1 mit den Koordinaten (s_{00}, s_{01}) bzw. (s_{10}, s_{11}). Als einfachste Variante der folgenden Herleitung betrachten wir den Fall, daß diese Punkte bereits die Bedingungen

$$0 < s_{j0} < 40,\ 0 < s_{j1} < 24\ (j = 0, 1)$$

erfüllen. Wir formulieren eine Information über die Wahl dieser Variante sowie eine einleitende Aufforderung zum Eingeben der Koordinaten etwa in den BASIC-Programmzeilen

```
100 CLS
110 PRINT "  Strecke ausgeben"
120 PRINT "  ---------"
130 PRINT "  Variante 'S0, S1 im Bild-
schirmbereich' ": PRINT
140 INPUT "Anf.-pkt. S0
(Koord. s00, s01) :"; S (0, 0), S (0, 1)
150 INPUT "Endpunkt S1
(Koord. s10, s11) : "; S (1, 0), S (1, 1)
```

Natürlich sind auch andere gewählte Koordinatensysteme möglich. Naheliegend wäre z.B. eine von oben nach unten gerichtete x_1-Achse, wie es der Zeilennumerierung für den PRINT AT-Befehl entspricht. Doch wollen wir schon hier bei Gelegenheit der Streckendarstellung sehen, wie sich ein Koordinatensystem handhaben läßt, das den oft üblichen geometrischen Gewohnheiten angepaßt ist. Weitere Überlegungen erfolgen dann im Abschnitt »Einpassung«, der den – ohnehin häufig zu erwartenden – Fall behandelt, daß die Koordinaten s_{ji} noch nicht die obengenannten Rahmenbedingungen erfüllen.

Die Aufgabe, ein optisch befriedigendes Bild der Strecke S_0S_1 zu erhalten, fassen wir in zwei Forderungen:

Forderung 1: Es sollen »möglichst punktförmige« Grafikzeichen ausgegeben werden, die möglichst nahe bei Teilpunkten

$$T_0 = S_0, T_1, T_2, \ldots, T_{z-1}, T_z = S_1$$

liegen, durch die die Strecke S_0S_1 in eine Anzahl z von gleich langen Teilstrecken zerlegt wird.

Forderung 2: Diese Anzahl z soll so groß wie möglich gewählt werden.

Diese beiden Forderungen einer »getupften Geometrie« sind nicht die einzig denkbare Möglichkeit zur Streckendarstellung. So könnte man daran denken, die Anzahl z zu vergrößern, indem man auf die Forderung gleich langer Teilstrecken $\overline{T_0T_1}$, $\overline{T_1T_2}, \ldots, \overline{T_{z-1}T_z}$ verzichtet. Die dann auftretende unregelmäßige Punktverteilung und vor allem die folglich stärker streuende Richtungsverteilung der Näherungs-Teilstrecken stört aber den optisch »glatten« Eindruck mehr, als die größere Anzahl nützt. Ein anderer Gedanke wäre es, nicht Teilpunkt-, sondern Teilstrecken-Grafiksymbole zu verwenden. Dies aber führt entweder zur Beschränkung auf wenige spezielle Richtungen (Bild 1 rechts oben) oder zu »Zackenlinien«, die zumeist ebenfalls optisch wenig befriedigend ausfallen werden (Bild 1 rechts unten).

2. Berechnung der Teilpunkte

Außer den Koordinaten (s_{j0}, s_{j1}) der Punkte S_j $(j = 0, 1)$ wollen wir zunächst auch die Anzahl z als gegeben betrachten. (Auf ihre Festlegung im Programm gemäß Forderung 2 gehen wir weiter unten ein.) Gesucht sind die Koordinaten (t_{m0}, t_{m1}) der Teilpunkte T_m $(m = 0, 1, \ldots, z)$.

Wie man in Bild 1 durch Projektion auf die x_0-Achse und die x_1-Achse erkennt (Strahlensatz!), lassen sich *getrennt für* $i = 0$ *und für* $i = 1$ die gesuchten Koordinaten t_{mi} aus den gegebenen Zahlen s_{0i} (Anfangswert), s_{1i} (Endwert) und z (Anzahl) ermitteln. Das kann folgendermaßen geschehen: Die Gesamtlänge $s_{1i} - s_{0i}$ wird in z gleiche Teile zerlegt; deren Länge ist somit $(s_{1i} - s_{0i})/z$. Um dann den m-ten Teilpunkt (auf der x_i-Achse) zu finden, geht man von s_{0i} aus um m Schritte dieser Länge weiter. So ergibt sich die Formel

$$t_{mi} = s_{0i} + m \cdot (s_{1i} - s_{0i})/z$$

$$(m = 0, \ldots, z; i = 0, 1).$$

Wir setzen sie in die folgenden BASIC-Programmzeilen um:

```
400 FOR M=0 TO Z: FOR I=0 TO 1
410 T (M, I) = S(0, I)
+M*(S(1, I)—S(0, I))/Z
450 NEXT I
550 NEXT M
```

3. Grafik-Muster für die Teilpunkte

In weiteren Programmzeilen zwischen 410 und 450 soll nun das Aufsuchen der Grafik-Muster ausgedrückt werden, die

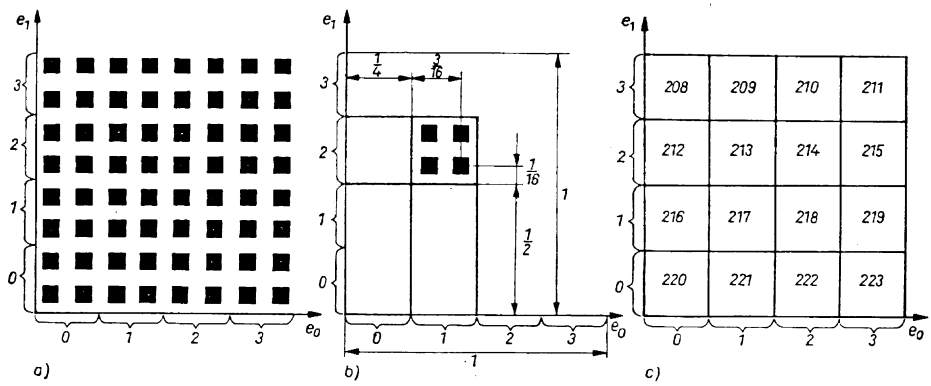


Bild 2

gemäß Forderung 1 möglichst nahe bei den Teilpunkten T_m liegen. Die Pseudo-Grafik des KC 85/1 bietet hierzu an jeder der 40×24 Stellen des Bildschirmbereichs folgende Möglichkeit: An diese Stelle kann ein Grafiksymbol gesetzt werden, d.i. ein »Muster«, das in einer **Auswahl** aus einer quadratischen Anordnung von 8×8 Pixel besteht (s. das stark vergrößerte Bild 2a). Wir erfassen die uns interessierenden Muster mit Hilfe eines »lokalen Koordinatensystems«, dessen Koordinaten e_0, e_1 nur die Werte 0, 1, 2, 3 annehmen. Das genügt aus folgendem Grund: Die kleinsten »möglichst punktförmigen« Grafikmuster, die der KC 85/1 zur Verfügung stellt, sind aus je 2×2 Pixel gebildete Muster, wie z.B. Bild 2b eines zeigt. Zur Beschreibung dieser Muster reicht gerade das eingeführte e_0, e_1-Koordinatensystem aus. Beispielsweise hat das Muster in Bild 2b die Koordinaten $(e_0; e_1) = (1; 2)$.

Der Grafik-Tabelle des KC 85/1 entnimmt man nun, daß diese Muster durch die in Bild 2c angegebenen CHR$-Werte abrufbar sind. Dabei kann man für alle 16 Werte leicht nachrechnen:

Es ist jeweils das
Muster mit den Koordinaten (e_0, e_1),
abrufbar durch CHR$ $(220 + e_0 - 4 \cdot e_1)$.
Beispielsweise erhält man für $(e_0; e_1) = (1; 2)$ den CHR$-Wert
$220 + 1 - 4 \cdot 2 = 213$.

Die so abrufbaren 2×2-Muster werden folgendermaßen an die richtige Stelle des Bildschirms gesetzt:
Man ermittelt diejenigen ganzen Zahlen g_{m0}, g_{m1}, für die

$$g_{mi} \leqq t_{mi} < g_{mi} + 1$$

gilt. Diese Zahlen (g_{m0}, g_{m1}) geben in dem (»globalen«) x_0, x_1-Koordinatensystem aus Bild 1 die **linke untere Ecke** des 8×8-Pixel-Quadrates an, aus dem wir das 2×2-Muster für den Punkt T_m herausgreifen wollen, also gerade den Koordinaten-Anfangspunkt des anzusteuernden »lokalen« e_0, e_1-Koordinatensystems. In BASIC werden die Zahlen g_{mi} gefunden durch

```
420 G (M, I) = INT(T(M, I))
```

Die genaue Lage des Punktes T_m im lokalen Koordinatensystem erhält man (wieder getrennt für $i = 0$ und für $i = 1$ jeweils in Richtung der e_i-Achse), indem man $t_{mi} - g_{mi}$ bildet. Das ist jeweils eine Zahl, für die $0 \leqq t_{mi} - g_{mi} < 1$ gilt. Wenn beispielsweise in Bild 2b der Punkt T_m genau im Mittelpunkt des rechten unteren der vier Pixel lag,

so ist $t_{m0} - g_{m0} = \frac{1}{4} + \frac{3}{16} = 0{,}4375$ und $t_{m1} - g_{m1} = \frac{1}{2} + \frac{1}{16} = 0{,}5625$. Wir multiplizieren diese Zahlen mit 4, um die Maßstabvergrößerung des e_0, e_1-Systems gegenüber dem x_0, x_1-System zu berücksichtigen. Im Beispiel erhält man $4 \cdot (t_{m0} - g_{m0}) = 1{,}75$ und $4 \cdot (t_{m1} - g_{m1}) = 2{,}25$. Nochmaliges Anwenden des INT-Befehls führt somit auf die gesuchten ganzzahligen Werte (e_{m0}, e_{m1}):

```
43Ø E(M, I)
= INT(4 ∗(T(M, I)−G(M, I)))
```

Die Bildschirm-Ausgabe des Musters bei T_m, die damit fertig vorbereitet ist, kann mit einer Anweisung PRINT AT(. ., . .); . . ausgeführt werden. Wir planen den Programmablauf (für unsere jetzige Variante) so, daß die PRINT AT-Ausgabe erst dann erfolgen soll, wenn alles fertig berechnet ist (andere Varianten betrachten wir später). Daher bauen wir für die Ausgabe eine neue FOR-NEXT-Schleife auf:

```
6ØØ FOR M = Ø TO Z
```

Die erste Angabe in der Klammer nach PRINT AT, die Bildschirmzeilennummer, entspricht nun **gegenläufig** den g_{m1}-Werten: Punkte mit $0 \leqq t_{m1} < 1$, also $g_{m1} = 0$ (linke untere Ecke des 8×8-Musters »ganz unten«, auf der x_0-Achse) kommen in Bildschirmzeile 23; Punkte mit $23 \leqq t_{m1} < 24$, also $g_{m1} = 23$ (»ganz oben«) in Bildschirmzeile 0. Allgemein ist stets $23 - g_{m1}$ die gesuchte Zeilennummer; damit haben wir, wie in Abschnitt 1. angekündigt, die Wahl des Koordinatensystems berücksichtigt. Die zweite Angabe nach PRINT AT, die Bildschirmspaltennummer entspricht *gleichsinnig* den g_{m0}-Werten selbst. Die dritte Angabe, der zu druckende »Text«, ist hier das Grafik-Symbol mit dem oben genannten CHR$-Wert. Die Anweisung lautet also

```
61Ø PRINT AT(23−G(M,1),G(M,Ø));
CHR$(22Ø+E(M,Ø)−4 ∗ E(M,1))
62Ø NEXT M
```

4. Zwischenbilanz, Programmübersicht

Bisher haben wir überwiegend solche BASIC-Programmzeilen formuliert, die Details der angestrebten Darstellung betreffen. Natürlich ist es aber auch notwendig, und zwar insbesondere zur Erfüllung von Forderung 2, eine Übersicht über den Gesamtablauf des Programms vorzunehmen.

An den auseinanderliegenden Zeilennummern, die bisher vorkamen, ist schon zu ersehen, daß sie sich in einen solchen Gesamtplan einpassen sollen, bei dem wir zusätzlich die inhaltliche Gliederung dadurch verdeutlichen wollen, daß wir bedeutungsmäßig getrennte Programmabschnitte jeweils mit einer neuen Hunderternummer beginnen lassen:

```
Ab Zeile 1ØØ:
Überschrift, Eingabe
(s. Abschn. 1.)
Ab Zeile 2ØØ:
Obere Schranke für z
(s. den folgenden Abschn. 5.)
Ab Zeile 3ØØ:
Beginn der absteigenden Ermittlung von z
(s. Abschn. 6.);
Sonderfall z=0 (s. Abschn. 7.)
Ab Zeile 4ØØ:
Grafiksymbole (Abschn. 2. und 3.)
Ab Zeile 5ØØ:
Fortsetzung der absteigenden Ermittlung
von z: Test von z, bei Nichteignung Rück-
kehr zu 3ØØ
Ab Zeile 6ØØ:
Grafik-Ausgabe (Abschn. 3.)
Ab Zeile 7ØØ:
Beenden des Ablaufs (s. Abschn. 7.)
```

5. Eine obere Schranke für z

Zur Erfüllung von Forderung 2 ist zu berücksichtigen, daß in jedem 8×8-

Pixel-Quadrat (im folgenden kurz »Quadrat« genannt) nur ein Grafik-Symbol möglich ist. Die Anzahl $1 + z$ der Teilpunkte $T_0, T_1, \ldots, T_z$ kann folglich nicht größer sein als die Anzahl aller derjenigen Quadrate, die von der Strecke S_0S_1 durchquert werden. Wir wollen für diese Anzahl eine obere Schranke ermitteln, die nur von den Koordinaten (s_{00}, s_{01}), (s_{10}, s_{11}) der Punkte S_0, S_1 abhängt. Dazu benutzen wir die – oben schon genannten und mit der INT-Anweisung zu erhaltenden – Zahlen g_{00}, g_{01} und g_{z0}, g_{z1}. Diese Zahlen hängen ja (trotz ihrer aus dem obigen Zusammenhang erklärlichen Bezeichnung) nicht von z ab, sondern sie sind einfach die größten ganzen Zahlen (INT-Zahlen) unter s_{00}, s_{01} und s_{10}, s_{11}. Sie können folglich als solche sogleich nach den INPUT-Zeilen (und einem CLS-Befehl, der den Bildschirm für die Streckenausgabe freisetzt) im BASIC-Programm auftreten.

Denkt man sich alle von einer Strecke S_0S_1 durchquerten Quadrate markiert (s. Bild 3), so entsteht eine »Schlange«,

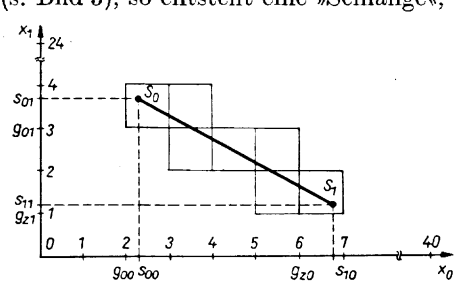


Bild 3

die sich in ihrem gesamten Verlauf in genau eine der acht Haupt- und Nebenrichtungen (rechts, rechts oben, oben, links oben, links, links unten, unten, rechts unten) einordnen läßt. Eine solche »Schlange« enthält stets dann eine nicht mehr vergrößerungsfähige Anzahl von Quadraten, wenn Nachbarquadrate in ihr immer längs einer ganzen Kante und nicht nur über eine einzelne Ecke verbunden sind. Dann aber lassen sich die Quadrate einfach abzählen:

Vom Anfangsquadrat aus geht man zum folgenden Nachbarquadrat stets einen Schritt in einer der vier Hauptrichtungen (rechts, oben, links, unten), wobei natürlich in jeder »Schlange« höchstens zwei dieser Hauptrichtungen vorkommen. In welcher Reihenfolge diese Hauptrichtungen auftreten, ist für die Gesamtzahl unwichtig. In waagerechter Richtung (rechts oder links) sind es genau $|g_{z0} - g_{00}|$ Schritte und in darauf senkrechter Richtung (oben oder unten) genau $|g_{z1} - g_{01}|$ Schritte, die man zu tun hat. Im Beispiel von Bild 3 mit den Koordinaten $(s_{00}; s_{01}) = (2, 3; 3, 7)$ und $(s_{10}; s_{11}) = (6{,}8; 1{,}2)$ der Punkte S_0 bzw. S_1 liegen diese Punkte in den Quadraten mit den linken unteren Eckpunkten $(g_{00}; g_{01}) = (2; 3)$ bzw. $(g_{z0}; g_{z1}) = (6; 1)$, und man hat genau $|6-2| = 4$ waagerechte Schritte ($2 \to 3$, $3 \to 4$, $4 \to 5$, $5 \to 6$) und $|1-3| = 2$ senkrechte Schritte ($3 \to 2$, $2 \to 1$) zu tun.

Als obere Schranke für die genannte Anzahl $1 + z$ ist damit die Zahl

$$1 + |g_{z0} - g_{00}| + |g_{z1} - g_{01}|$$

gefunden. Wir nehmen die Berechnung dieser Zahl in unser BASIC-Programm auf. Allerdings tritt sie dort (zunächst, vorübergehend) als Z und nicht als $1 + Z$ auf. Damit ist das nächste Vorhaben vorbereitet, von dieser oberen Schranke so lange abzusteigen, bis der in Forderung 2 verlangte Wert erreicht ist.

```
200 CLS: Z = 1 : FOR I=0 TO 1
210 Z = Z+ABS(INT (S(1, I))
—INT(S(0, I))): NEXT I
```

6. Absteigendes Ermitteln von z

Wir wollen jetzt Programmzeilen bilden, die das eben genannte schrittweise Absteigen steuern.

Der in Zeile 210 berechnete Wert ist jedenfalls noch (mindestens um 1) zu groß. Das Absteigen beginnt mit

300 Z = Z−1

Was hat nun zu geschehen, nachdem mit diesem Z-Wert in den schon bekannten Zeilen 400 bis 450 die Zahlen g_{mi} und e_{mi} berechnet sind? Wir müssen testen, ob je zwei benachbarte Teilpunkte in verschiedene 8×8-Pixel-Quadrate fallen. Trifft dies nicht zu, so war die Anzahl Z immer noch zu groß. In diesem Fall wird aus der mit M ablaufenden FOR-NEXT-Schleife »ausgestiegen« und in die Zeile 300 zum nächsten Absteige-Schritt zurückgekehrt. Allerdings darf dieser Test (Zeile 510) erst ab M=1 stattfinden; der Wert M=0 muß »übersprungen« werden (Zeile 500):

```
500 IF M=0 THEN 550
510 IF G(M, 0)=G(M—1, 0)
AND G(M, 1)=G(M—1, 1) THEN 300
```

7. Vervollständigung des Programms

Ein paar »Kleinigkeiten« des Programmablaufs sind (auch für die jetzt vorgestellte Version) noch zu bereinigen.
a) Wenn eine Strecke S_0S_1 so kurz ist, daß S_0 und S_1 in demselben 8×8-Pixel-Quadrat liegen, so wird in den Zeilen 200, 210 der Wert Z=1 ermittelt, und Zeile 300 führt auf Z=0.
Würden wir jetzt nicht eingreifen, so ergäbe sich in Zeile 410 die Fehlermeldung »Division durch Null«. Was aber soll stattdessen sinnvollerweise geschehen?
Da die Strecke nur als zu einem Punkt »entartete Strecke« wiedergegeben werden kann, entscheiden wir uns dafür, ihren Mittelpunkt auszugeben. Im Fall Z>0 ist diese Regelung natürlich zu überspringen (Zeile 310), während wir im Fall Z=0 durch eine eingefügte Zeile 405 dafür sorgen, daß Zeile 410 übersprungen wird:

```
310 IF Z>0 THEN 400
320 FOR I=0 TO 1
330 T(0, I) = (S(0, I)+S(1, I))/2: NEXT I
405 IF Z=0 THEN 420
```

b) Da die Laufvariable M möglicherweise Werte über 10 annehmen soll, ist entsprechender Speicherplatz bereitzustellen:

```
220 DIM T(Z,1),G(Z,1),E(Z,1)
```

c) Am Ende möchte man vielleicht das »stolze Ergebnis« so lange sehen, bis eine (beliebige) Taste gedrückt, d.h. die Textvariable INKEY$ mit einem nichtleeren Wort belegt wird:

```
700 IF INKEY$ ="" THEN 700
710 END
```

Dem Leser sei empfohlen, sich das jetzt aufgestellte Programm in der richtigen Reihenfolge der Zeilen zusammenhängend aufzuschreiben. (Zur Kontrolle: Es enthält 27 Zeilen.)

8. Ausgabe von Kreisen

Als nächstes Vorhaben gehen wir daran, das Programm so abzuändern, daß es die Ausgabe von Kreisen bewirkt. Zugleich wollen wir Programmteile zum Weglassen von Punkten vorsehen, die nicht im Bildschirmbereich liegen. (Solche Programmteile kann man auch zur Variantenbildung in anderen Fällen benutzen, z.B. im eben fertiggestellten Streckenausgabe-Programm.) Diejenigen Zeilen aus Abschnitt 1. bis 7., die jetzt nicht wieder genannt werden, sind unverändert zu übernehmen!
Die Eingabe-Zeilen ersetzen wir etwa durch

```
110 PRINT "   Kreis ausgeben"
120 PRINT "   ———————"
130 PRINT
"Variante 'Einpassen durch Abschneiden' ":
PRINT
140 INPUT "Mittelpunkt C
(Koord. c0, c1) :" ;C(0), C(1)
150 INPUT "Radius r:" ;R
```

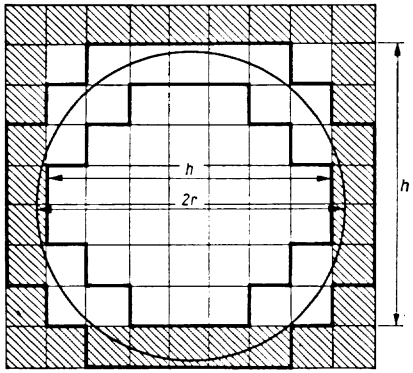


Bild 4

Als eine obere Schranke für die Anzahl der Teilpunkte findet man die um 1 vergrößerte Anzahl von 8×8-Pixel-Quadraten, die so gelegen sind, daß sie ihrerseits eine quadratische Fläche der Seitenlänge h einschließen, wo h die ganze Zahl mit $h \leqq 2r < h + 1$ ist (das sind $4 + 4 \cdot h$ Quadrate; s. Bild 4):

```
200 CLS
210 Z = 5+4*INT(2*R)
```

Hinweis: Falls – vielleicht im Zusammenhang eines größeren Rahmenprogramms – auch negative R-Werte denkbar wären, müßte hier R durch ABS(R) ersetzt werden. (Andernfalls reagiert, wie wir gleich sehen werden, unser Programm auf negative R mit der

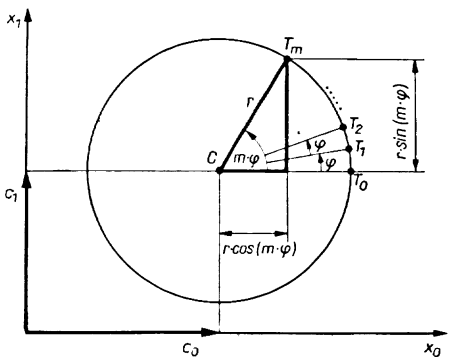


Bild 5

Ausgabe nur des Punktes C; auch das kann ja gelegentlich erwünscht sein.) Den Sonderfall eines für den KC 85/1 zu kleinen Kreises erfassen wir durch Ausgabe des Kreismittelpunktes C:

```
310 IF Z>2 THEN 400
320 Z = 0: FOR I=0 TO 1
330 T(0, I) = C(I): NEXT I
```

Kernstück der Änderungen ist es natürlich, die Zeile 410 (Strecken-Teilpunkte) zu ersetzen durch eine Berechnung von Teilpunkten T_m auf dem Kreis. Bild 5 zeigt, wie sich die Koordinaten (t_{m0}, t_{m1}) von T_m durch die trigonometrischen Funktionen Cosinus und Sinus des m-fachen eines Schrittwinkels φ gewinnen lassen. Das bietet den Vorteil, durch Zerlegung des Vollwinkels 360° in gleich große Schrittwinkel $\varphi = 360°/z$ sofort eine Forderung zu erfüllen, die sinngemäß der Forderung 1 entspricht; denn dabei wird auch der Kreis **in gleich lange Teilbögen zerlegt.** Allerdings muß man (für das BASIC des KC 85/1) die Winkel ins Bogenmaß umrechnen, d.h. den Vollwinkel mit 2π ansetzen, wobei π durch 4 *ATN(1) gefunden werden kann:

```
400 FOR M=0 TO Z
410 F = M*8*ATN(1)/Z
411 T(M, 0) = C(0)+R*COS(F):
T(M, 1) = C(1)+R*SIN(F)
```

Erst danach eröffnen wir die FOR-NEXT-Schleife für den Koordinatenindex I in $G(M, I)$ und $E(M, I)$:

```
419 FOR I=0 TO 1
```

9. Abschneiden auf Bildschirmformat

Die Information, ob der Punkt T_m im Bildschirmbereich liegt oder nicht, kann man z. B. in einer Variablen $B(M)$ speichern, die je nachdem den Wert 1 oder 0 erhält. Ist sie gleich 0, so lassen wir die weiteren Berechnungen und die Ausgabe überspringen:

```
415 IF T (M, Ø)>Ø
AND T (M, Ø)<4Ø AND T (M, 1)>Ø
AND T (M, 1)<24
THEN B (M)=1 : GOTO 419
416 B(M) = Ø: GOTO 55Ø
6Ø5 IF B(M)=Ø THEN 62Ø
```

Auch im Fall $Z=Ø$ muß diese Verzweigung angesteuert werden:

```
4Ø5 IF Z=Ø THEN 415
```

(Die Anweisung B(M) = Ø in Zeile 416 ist hier entbehrlich. Erst bei anderweitiger Unterprogramm-Nutzung könnte sie notwendig werden. Oder man ersetzt sie durch E(M, Ø)= −1, in Zeile 6Ø5 durch E(M, Ø) < Ø und kommt in 415 ohne B(M) mit THEN 419 aus.)

Der Test, ob T_m in dasselbe Quadrat wie T_{m-1} fällt, ist nun nicht nur im Fall $m=0$ zu überspringen, sondern auch in allen denjenigen Fällen, in denen zwar T_m im Bildschirm liegt, aber T_{m-1} nicht. Daher fügen wir vor Zeile 51Ø ein:

```
5Ø5 IF B(M—1)=Ø THEN 55Ø
```

(Falls B(M) »elegant eingespart« wurde, natürlich: IF E(M−1, Ø)<Ø.) Schließlich fehlt noch (wieder: falls nicht »eingespart«):

```
23Ø DIM B(Z)
```

Damit sind alle geplanten Änderungen erfolgt. Wie beim Streckenprogramm empfiehlt es sich, das geänderte Programm zusammenhängend aufzuschreiben. (Es enthält 34 Zeilen.)

Der Ablauf der Programme – sowohl bei längeren Strecken als auch bei Kreisen – wird wegen der langen Wartezeiten noch nicht recht befriedigen. Das liegt an der oft durchlaufenen Such-Routine, die im Kreis-Programm zudem die zeitaufwendigen Anweisungen ATN, COS und SIN enthält. Wir haben den Programmaufbau bisher eben nur an theoretisch einfachen Gedanken orientiert. In einer Fortsetzung (mit den weiteren eingangs angekündigten Themen) sollen auch Hinweise zur Verkürzung und Überbrückung von Wartezeit gegeben werden.

Autor:
Dozent Dr. sc. nat. Ludwig Stammler
Martin-Luther-Universität Halle

(Fortsetzung von S. 10)

Literatur

[1] Girlich, H.-J.: Iterationen und der Feigenbaum (Mathematische Experimente mit dem Kleincomputer) – In: Kleinstrechner-TIPS, Heft 5. – Leipzig, 1986. – S. 4—10

[2] Peitgen, H.-O.; Richter, P. H.: The Beauty of Fractals (Images of Complex Dynamical Systems). – New York: Springer-Verlag, 1986

Der projizierende Computer

Computer sind außerordentlich vielseitige Gesellen. Sie steuern Prozesse, rechnen schnell und genau, speichern und sortieren Buchstaben und Buchstabenketten und suchen darin, setzen Bildpunkte (Pixel) und Farben auf den Bildschirm, erzeugen Zufallszahlen, um damit Zahlen, Texte, Pixel oder Farben zufällig zu manipulieren. Die kombinierte Nutzung all dieser Merkmale ergibt das komplexe Werkzeug Computer. Es wird zur Kontrolle und Steuerung von Fertigungsprozessen, für Berechnungen, zur Textverarbeitung, zum Lehren, Lernen und Spielen, als kreatives Werkzeug für den Konstrukteur und den Künstler und für vieles andere mehr eingesetzt. Natürlich setzt die effektive Nutzung dieses Werkzeugs das entsprechende Informatikwissen voraus. Und hier ist die Computergrafik ein attraktives »Fenster«, das den Blick zur Informatik ausgezeichnet zu weiten vermag.

Wir beginnen damit, uns etwas vorzuwerfen; es sei denn, Sie gehören zu denen, die sich nie etwas vorzuwerfen brauchen. Allerdings werfen wir uns lediglich Schatten vor, mit deren Hilfe räumliche Gebilde auf eine Fläche, in unserem Fall einen Bildschirm, projiziert werden, denn das Wort Projizieren kommt aus dem Lateinischen und bedeutet »vorwerfen«. Unser Projektionsprogramm wird erste, »zarte« Anfänge der CAD-Technologie offenbaren, die Möglichkeiten, aber auch Grenzen eines Kleincomputers mit BASIC-Programm zeigen und zu Programmänderungen und -erweiterungen auffordern. Einen unmittelbaren Bezug zu ihrer Tätigkeit werden Schüler der 7. und 10. Klassen spüren, da dieses Programm eine enge Bindung zum Stoffgebiet *Darstellende Geometrie* im Mathematikunterricht und zum Fach *Technisches Zeichnen* hat. Aber auch den anderen Lesern wird die Möglichkeit geboten, ihren »Traumbungalow« selbst auf dem Papier zu entwerfen und ihn vergrößert, verkleinert, in Vogel- oder Froschperspektive, von vorn, von hinten oder von der Seite auf dem Bildschirm zu betrachten.

Der Weg dahin beginnt bei der Bereitstellung der Koordinaten der Anfangs- und Endpunkte, die jeweils eine Gerade im Raum bilden. Mit Hilfe der entsprechenden Projektionsgleichungen werden aus den Punkten im Raum die Punkte auf der Bildschirmfläche bestimmt, die dann wieder durch eine Gerade verbunden werden. Das »projizierte Ergebnis« wird also stets ein Drahtmodell des Gegenstandes sein, bei dem sich leider die verdeckten Kanten nicht ohne weiteres beseitigen lassen. Das von uns benutzte räumliche Koordinatensystem zeigt Bild 1. Dabei tritt, wie in mathematischen Dar-

stellungen üblich, die x-Achse aus der Papierebene heraus. Im flächigen Koordinatensystem des Bildschirms wird die Abszisse als x-Achse und die Ordinate als y-Achse bezeichnet. Bild 1 macht deutlich, daß die y-Achse im Raum zur x-Achse auf der Bildschirmfläche und die z-Achse im Raum zur y-Achse auf der Bildschirmfläche wird. Dies gilt es zu berücksichtigen, wenn es im folgenden um die einzelnen Projektionsarten mit ihren Projektionsgleichungen geht.

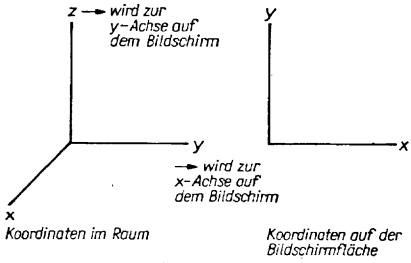


Bild 1. Koordinatensystem im Raum und auf der Bildschirmfläche

Senkrechte Parallelprojektion

Fast jedes Lehrbuch über Darstellende Geometrie beginnt mit der Beschreibung der senkrechten Parallelprojektion als Ein-, Zwei- und Dreitafelprojektion. In unserem Programm verwenden wir nur die *Zweitafelprojektion*, also die Darstellung von Grund- und Aufriß. Die Anschaulichkeit der Bilder, die diese Projektionsart liefert, ist sehr schlecht, dafür können der Zeichnung alle Maße exakt entnommen werden. Letzteres ist natürlich auf dem gewölbten, eventuell zum Rand hin verzerrenden Bildschirm nur schwer möglich. Aber schon ein am Kleincomputer angeschlossener Plotter oder Drucker kann da Erstaunliches leisten, sofern er die Bilder im gleichen Verhältnis von Höhe zu Breite wiedergibt (hier gibt es auch bei sogenannten grafikfähigen Druckern immer wieder böse Überraschungen).

Gemäß dem verwendeten räumlichen Koordinatensystem (Bild 1) gilt für den Grundriß bei der senkrechten Zweitafelprojektion:

$$X' = X \cdot MA \quad Y' = Y \cdot MA \quad Z' = 0.$$

Die Variable mit dem Namen MA enthält einen Maßstabsfaktor, der vom Nutzer eingegeben werden kann. Damit sind, und das bei allen Projektionsarten, beliebige Vergrößerungen oder Verkleinerungen möglich.

Für die Punkte auf dem Bildschirm (ebenes Koordinatensystem gemäß Bild 1) werden durchgängig folgende Bezeichnungen verwendet:

XA = x-Koordinate des Anfangspunktes einer Geraden auf dem Bildschirm

XE = x-Koordinate des Endpunktes einer Geraden auf dem Bildschirm

YA = y-Koordinate des Anfangspunktes einer Geraden auf dem Bildschirm

YE = y-Koordinate des Endpunktes einer Geraden auf dem Bildschirm

Diese Werte müssen dann der Zeichenroutine im Programm, z.B. der LINE-Funktion beim KC 85/3, übergeben werden. Daraus folgt für die Darstellung des Grundrisses:

XA = Y' (entsprechend für XE)

YA = X' (entsprechend für YE)

Für den Aufriß gilt dann

$X' = 0$ $Y' = Y \cdot MA$ $Z' = Z \cdot MA$, woraus für die Darstellung auf dem Bildschirm folgt:

XA = Y' (entsprechend für XE)

YA = Z' (entsprechend für YE).

Schräge Parallelprojektion

Durch die schräge Parallelprojektion wird das »Zerhacken« in zwei oder drei getrennte Bilder verhindert. Da nur noch ein Bild entsteht, steigt die Anschaulichkeit der Darstellung. Die Entnahme wahrer Maße ist aber eingeschränkt, denn nur noch die Maße, die in der Bildschirmebene liegen, sind exakt (deshalb auch der Begriff Frontaldimetrie). Alle Strecken, die »in den Raum gehen«, sind in ihrer Darstellung von folgenden Parametern abhängig:

Verzerrungswinkel WI
Verzerrungsverhältnis VE.

Der Sonderfall WI = 45 Grad und VE = 0,5 ist auch als Kavalierperspektive bekannt. Im Programm sind beliebige Eingaben möglich, so daß z.B. bei stumpfen Winkeln das Objekt auch von der »Rückseite« betrachtet werden kann. Für die schräge Parallelprojektion gelten die folgenden Projektionsgleichungen:

$$Y' = (Y - X \cdot VE \cdot \cos(WI)) \cdot MA$$

$$Z' = (Z - X \cdot VE \cdot \sin(WI)) \cdot MA$$

Daraus folgt für die Bildschirmdarstellung:
XA = Y' (entsprechend für XE) und
YA = Z' (entsprechend für YE).

Dimetrische Projektion

Die dimetrische Projektion gehört zur Gruppe der axonometrischen Darstellungen. Auch bei axonometrischen Darstellungen entsteht durch Parallelprojektion nur ein einziges Bild. Durch die geschickte Wahl von *Verzerrungswinkel* und *Verzerrungsverhältnis* versucht man aber, gute Kompromisse zwischen Anschaulichkeit und Maßtreue zu erreichen. Am häufigsten werden die isometrische, die dimetrische und die trimetrische Projektion verwendet. Wir zeigen im Programm nur die dimetrische Projektion, der kundige Leser kann aber ohne Schwierigkeiten die anderen Projektionsarten nachträglich einbauen. Die dimetrische Projektion benutzt ein Koordinatensystem, dessen y-Achse einen Winkel von 7,2 Grad und dessen x-Achse einen Winkel von 41,4 Grad zur Horizontalen bilden (Bild 2). Für das Verzerrungsverhältnis gilt $x:y:z = 0{,}5:1:1$. Damit ist ebenfalls für frontale Flächen eine exakte Maßabnahme möglich, die Darstellung ist aber anschaulicher als bei der schrägen Parallelprojektion.

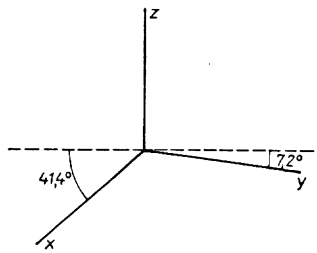


Bild 2. Koordinatensystem bei der dimetrischen Projektion

Da Verzerrungswinkel und Verzerrungsverhältnis festliegen, lassen sich Sinus- und Cosinuswerte sofort ausrechnen, so daß sich folgende *Projektionsgleichungen* ergeben:

$$Y' = (Y \cdot \sqrt{63}/8 - X \cdot 3/8) \cdot MA$$

$$Z' = (Z - Y/8 - X \cdot \sqrt{7}/8) \cdot MA$$

Für die Bildschirmdarstellung folgt wieder XA=Y' und YA=Z'. Der Programmnutzer kann bei der dimetrischen Projektion folglich nur den Maßstab variieren.

Zentralprojektion

Schier unvorstellbare Variationsmöglichkeiten (und damit auch einigen Computerunsinn bei unsinnigen Ein-

gaben) bietet hingegen die *Zentralprojektion*. Der Trumpf der Zentralprojektion ist ihre Anschaulichkeit. Dieser Anschaulichkeit wird ganz rigoros die Maßtreue geopfert. Während die alten Ägypter ihre Bilder noch vorwiegend isometrisch malten, kam erst im 15. Jahrhundert das perspektivische Zeichnen auf. Dies ist unter anderem FILIPPO BRUNELLESCHI (1377 bis 1446) zu verdanken, der vermutlich als erster den Fluchtpunktsatz formuliert hat. Das perspektivische Zeichnen unter Zuhilfenahme eines Fluchtpunktes bewirkte damals eine Wende in der Malerei. Viele Bilder dieser Zeit zeigen, gewissermaßen als fotografischer Ersatz, das Bemühen, die wahren Raumverhältnisse in Zentralperspektive darzustellen. Die Verbindungen zwischen Fotografie und diesem Programmteil der Zentralprojektion sind wahrhaftig sehr eng, wie wir noch sehen werden.

Das Programm bietet die Möglichkeit, das Objekt vor der Darstellung in Zentralprojektion um jede Achse in einem beliebigen Winkel zu drehen. Die Reihenfolge der Drehung erfolgt zunächst in der y-, dann in der z- und dann in der x-Achse gemäß Bild 1. Eine Drehung um die y-Achse bewirkt z. B. die vom Fotografen so gefürchteten stürzenden Linien, was am Bildschirm problemlos ausprobiert werden kann. Die Drehung um die y-Achse geschieht auf folgende Weise:

$$Y_{gedreht} = Y$$
$$Z_{gedreht} = Z \cdot \cos(W_y) + X \cdot \sin(W_y)$$
$$X_{gedreht} = X \cdot \cos(W_y) - Z \cdot \sin(W_y),$$

wobei W_y der Drehwinkel auf der y-Achse ist. Bei der Drehung um die z- und x-Achse ist das Vorgehen ähnlich. Auf diese »gedrehten Koordinaten« wird in den Projektionsgleichungen zur Zentralprojektion zurückgegriffen. Der Nutzer kann hier folgende Parameter eingeben:

MA. Maßstab, eine Änderung der Perspektive wird nicht bewirkt, ein größerer Maßstab entspricht einer stärkeren Vergrößerung, die z. B. am Vergrößerungsapparat in der Dunkelkammer eingestellt werden kann.

EN. Entfernung des Betrachters oder Fotografen, eine Standortveränderung bewirkt eine Perspektivänderung, wird der Standort aus Versehen in das Objekt hineinverlegt, dann entsteht einiger Unsinn, die Perspektivänderung entspricht der Wirkung der einzelnen Objektivarten (Weitwinkel-, Normal-, Teleobjektiv), allerdings muß mit dem Maßstab MA die Bildgröße entsprechend korrigiert werden.

AU. Augenhöhe des Betrachters oder Fotografen, der damit auch Vogel- und Froschperspektive realisieren kann.

SE. Seitenabstand des Betrachters, den man bei Orientierungsschwierigkeiten zunächst so groß wie die halbe Objektbreite machen sollte.

Für die Zentralprojektion gelten folgende Gleichungen:

$$Y' = ((Y_{gedreht} - SE)/(EN - X_{gedreht})) \cdot MA$$

$$Z' = ((Z_{gedreht} - AU)/(EN - X_{gedreht})) \cdot MA$$

Um das Projektionsbild wieder ins Zentrum zurückzuholen, werden noch folgende Korrekturgleichungen angeschlossen:

$$Y' = Y' + SE/EN \cdot MA$$

$$Z' = Z' + AU/EN \cdot MA$$

Für die Bildschirmdarstellung folgt in bekannter Weise: XA=Y′ und YA=Z′ (ebenso für die Endpunktkoordinaten XE und YE).

Das BASIC-Programm

Das Gesamtprogramm, so wie es Bild 3 zeigt, umfaßt etwa 7,8 Kbyte. Gemeinsam mit dem für Zahlenfelder reservierten Platz ergibt sich ein Speicherplatzbedarf von 11,1 Kbyte, der den meisten Kleincomputern keine Schwierigkeiten bereiten wird. Das Programm ist in der hier vorgestellten Fassung sofort auf den Kleincomputern KC 85/2 mit BASIC-Modul und KC 85/3 lauffähig. Der KC 85/2 ohne BASIC-Modul besitzt leider nicht die LINE-Funktion zum Zeichnen von Geraden und bietet in der Grundversion nach Laden des BASIC-Interpreters von der Kassette in den Schreib-Lese-Speicher (RAM) auch nicht mehr genügend Speicherplatz. In diesem Fall müßte ein RAM-Erweiterungsmodul gesteckt und die LINE-Funktion ersetzt werden. Das kann z. B. mit der Zweipunktegleichung erfolgen, mit deren Hilfe dann Punkt für Punkt (mit der PSET-Anweisung) eine Gerade entsteht. Allerdings muß der Nutzer hier einige Geduld mitbringen. Die LINE-Funktion wäre in Zeile 370, in der Zeichenroutine (Zeilen 4000 bis 4040) und in den Zeilen 2570 bis 2590 zu ändern. Hier werden auch diejenigen Leser Veränderungen vornehmen müssen, die das Programm für einen anderen Vollgrafikcomputer umschreiben wollen. Dazu seien folgende Spezifika des KC 85/3 kurz genannt:

- Farbvereinbarungen mit den Anweisungen COLOR, INK und PAPER
- Bildschirm mit 32 Zeilen und 40 Spalten, PRINT AT und LCCATE anwendbar
- Vollgrafikbildschirm mit 320 Pixel in x- und 256 Pixel in y-Richtung
- Koordinatenursprung liegt in linker unterer Bildschirmecke.

Bild 3. Listing des BASIC-Programms ▶

```
10 REM PROJEKTIONEN
20 REM VEREINBARUNGEN
30 WINDOW 0,31,0,39:COLOR 7,0:CL
S
40 DIM AP(50,3):DIM EP(50,3):DIM
 AD(50,3):DIM ED(50,3)
50 REM AUSWAHL DER OBJEKTE
60 PRINT AT(5,1);"WELCHES OBJEKT
 SOLL DARGESTELLT WERDEN?"
70 PRINT AT(7,5);"QUADER = 1"
80 PRINT AT(9,5);"PYRAMIDE = 2"
90 PRINT AT(11,5);"BUNGALOW = 3"

100 PRINT AT(13,5);"FREI WAEHLBA
RES OBJEKT = 4"
110 LOCATE 16,1:INPUT"BITTE KENN
ZAHL EINGEBEN!";KE
120 IF KE<>1 AND KE<>2 AND KE<>3
 AND KE<>4 THEN GOTO 110
130 IF KE=1 THEN RESTORE 5000:GO
SUB 3000:GOSUB 3100
140 IF KE=2 THEN RESTORE 6000:GO
SUB 3000:GOSUB 3300
150 IF KE=3 THEN RESTORE 7000:GO
SUB 3000
160 IF KE=4 THEN GOSUB 2500
170 CLS
180 PRINT AT(5,1);"WELCHE DARSTE
LLUNGSART WIRD GEWUENSCHT?"
190 PRINT AT(7,5);"SENKRECHTE ZW
EITAFELPROJEKTION = 1"
200 PRINT AT(9,5);"SCHRAEGE PARA
LLELPROJEKTION = 2"
210 PRINT AT(11,5);"DIMETRISCHE
PROJEKTION = 3"
220 PRINT AT(13,5);"ZENTRALPROJE
KTION = 4"
230 LOCATE 16,1:INPUT"BITTE KENN
ZAHL EINGEBEN!";KE
240 IF KE<>1 AND KE<>2 AND KE<>3
 AND KE<>4 THEN GOTO 230
250 CLS:ON KE GOTO 300,600,800,1
000
300 REM SENKR. ZWEITAFELPROJEKTI
ON
310 LET PR$="SENKRECHTE ZWEITAFE
LPROJEKTION"
320 PRINT:PRINT PR$
330 INPUT"MASSSTAB (Z.B.=1)=";MA
:CLS
340 PRINT TAB(4);PR$
350 PRINT"MASSSTAB=";MA
360 REM RISSACHSE
370 LINE 0,120,319,120,7
380 REM GRUNDRISS
390 FOR I=1 TO AN
400 LET XA=AP(I,2)*MA+160:LET YA
=110-AP(I,1)*MA
410 LET XE=EP(I,2)*MA+160:LET YE
=110-EP(I,1)*MA
420 GOSUB 4000
430 NEXT I
440 REM AUFRISS
450 FOR I=1 TO AN
460 LET XA=AP(I,2)*MA+160:LET YA
=AP(I,3)*MA+130
470 LET XE=EP(I,2)*MA+160:LET YE
=EP(I,3)*MA+130
480 GOSUB 4000
490 NEXT I
500 GOTO2000
600 REM SCHRAEGE PARALLELPROJEKT
ION
610 LET PR$="SCHRAEGE PARALLELPR
OJEKTION"
620 PRINT:PRINT PR$
630 INPUT"MASSSTAB (Z.B.=1)=";MA

640 INPUT"VERZERRUNGSWINKEL (Z.B
.=45 GRAD)=";WI
650 INPUT"VERZERRUNGSVERHAELTNIS
 (Z.B.=0.5)=";VE:CLS
660 PRINT TAB(6);PR$
670 PRINT"MASSSTAB=";MA;"V.WINKE
L=";WI;"V.VERH.=";VE
```

```
680 FOR I=1 TO AN
690 LET XA=(AP(I,2)-AP(I,1)*VE*C
OS(WI*PI/180))*MA+160
700 LET YA=(AP(I,3)-AP(I,1)*VE*S
IN(WI*PI/180))*MA+120
710 LET XE=(EP(I,2)-EP(I,1)*VE*C
OS(WI*PI/180))*MA+160
720 LET YE=(EP(I,3)-EP(I,1)*VE*S
IN(WI*PI/180))*MA+120
730 GOSUB 4000
740 NEXT I
750 GOTO2000
800 REM DIMETRISCHE PROJEKTION
810 LET PR$="DIMETRISCHE PROJEKT
ION"
820 PRINT:PRINT PR$
830 PRINT"VERZERRUNGSWINKEL UND
VERZERRUNGS-"
840 PRINT"VERHAELTNIS LIEGEN FES
T"
850 INPUT"MASSSTAB (Z.B.=1)=";MA
:CLS
860 PRINT TAB(8);PR$
870 PRINT"MASSSTAB=";MA
880 FOR I=1 TO AN
890 LET XA=(AP(I,2)*SQR(63)/8-AP
(I,1)*3/8)*MA+160
900 LET YA=(AP(I,3)-AP(I,2)/8-AP
(I,1)*SQR(7)/8)*MA+120
910 LET XE=(EP(I,2)*SQR(63)/8-EP
(I,1)*3/8)*MA+160
920 LET YE=(EP(I,3)-EP(I,2)/8-EP
(I,1)*SQR(7)/8)*MA+120
930 GOSUB 4000
940 NEXT I
950 GOTO2000
1000 REM ZENTRALPROJEKTION
1010 LET PR$="ZENTRALPROJEKTION"

1020 PRINT:PRINT PR$
1030 INPUT"DREHWINKEL Y-ACHSE(Z.
B.=0 GRAD)=";WY:LET BY=WY*PI/180

1040 INPUT"DREHWINKEL Z-ACHSE(Z.
B.=15 GRAD)=";WZ:LET BZ=WZ*PI/18
0
1050 INPUT"DREHWINKEL X-ACHSE(Z.
B.=0 GRAD)=";WX:LET BX=WX*PI/180

1060 PRINT:PRINT INK 23;"BITTE W
ARTEN"
1070 PRINT:PRINT"GESAMTDREHUNG W
IRD BERECHNET"
1080 FOR I=1 TO AN
1090 REM DREHUNG Y-ACHSE
1100 LET Y5=AP(I,2):LET Y7=EP(I,
2)
1110 LET Z5=AP(I,3)*COS(BY)+AP(I
,1)*SIN(BY)
1120 LET Z7=EP(I,3)*COS(BY)+EP(I
,1)*SIN(BY)
1130 LET X5=AP(I,1)*COS(BY)-AP(I
,3)*SIN(BY)
1140 LET X7=EP(I,1)*COS(BY)-EP(I
,3)*SIN(BY)
1150 REM DREHUNG Z-ACHSE
1160 LET Y6=Y5*COS(BZ)-X5*SIN(BZ
)
1170 LET Y8=Y7*COS(BZ)-X7*SIN(BZ
)
1180 LET Z6=Z5:LET Z8=Z7
1190 LET X6=X5*COS(BZ)+Y5*SIN(BZ
)
1200 LET X8=X7*COS(BZ)+Y7*SIN(BZ
)
1210 REM DREHUNG X-ACHSE
1220 LET AD(I,2)=Y6*COS(BX)+Z6*S
IN(BX)
1230 LET ED(I,2)=Y8*COS(BX)+Z8*S
IN(BX)
1240 LET AD(I,3)=Z6*COS(BX)-Y6*S
IN(BX)
1250 LET ED(I,3)=Z8*COS(BX)-Y8*S
IN(BX)
1260 LET AD(I,1)=X6:LET ED(I,1)=
X8
1270 NEXT I
1280 CLS
1290 PRINT TAB(10);PR$
1300 PRINT"DREHWINKEL Y-ACHSE=";
WY;"GRAD"
1310 PRINT"DREHWINKEL Z-ACHSE=";
WZ;"GRAD"
1320 PRINT"DREHWINKEL X-ACHSE=";
WX;"GRAD":PRINT
1330 PRINT"FUER DIE DARSTELLUNG
SIND NOCH"
1340 PRINT"FOLGENDE ANGABEN ERFO
RDERLICH:"
1350 INPUT"MASSSTAB(Z.B.=250)=";
MA
1360 INPUT"ENTFERNUNG(Z.B.=300)=
";EN
1370 INPUT"SEITENABSTAND(Z.B.=50
)=";SE
1380 INPUT"AUGENHOEHE(Z.B.=120)=
";AU
1390 CLS
1400 PRINT TAB(10);PR$
1410 PRINT"MASSSTAB=";MA;"ENTFER
NUNG=";EN
1420 PRINT"SEITENA.=";SE;"AUGENH
.=";AU
1430 FOR I=1 TO AN
1440 LET XA=((AD(I,2)-SE)/(EN-AD
(I,1)))*MA
1450 LET XA=XA+SE/EN*MA+160
1460 LET YA=((AD(I,3)-AU)/(EN-AD
(I,1)))*MA
1470 LET YA=YA+AU/EN*MA+120
1480 LET XE=((ED(I,2)-SE)/(EN-ED
(I,1)))*MA
1490 LET XE=XE+SE/EN*MA+160
1500 LET YE=((ED(I,3)-AU)/(EN-ED
(I,1)))*MA
1510 LET YE=YE+AU/EN*MA+120
1520 GOSUB 4000
1530 NEXT I
1540 INPUT"WEITERE VAR. MIT GLEI
CHER DREHUNG(J/N)?";J$
1550 IF J$="J" THEN GOTO1280:ELS
E GOTO2000
2000 REM PROGRAMMWIEDERHOLUNGEN
2010 LOCATE 31,0:INPUT"FORTSETZU
NG(J/N)?";J$
2020 CLS
2030 IF J$="N" THEN END
2040 PRINT"SOLL AM GLEICHEN OBJE
KT"
2050 INPUT"WEITERGEARBEITET WERD
EN(J/N)?";J$
2060 IF J$="J" THEN GOTO 170:ELS
E RUN
2500 REM UP FREI WAEHLBARES OBJE
KT
2510 CLS:PRINT
2520 PRINT"VOM GEWAEHLTEN OBJEKT
 MUESSEN"
2530 PRINT"DIE KOORDINATEN X,Y,Z
 DER AN-"
2540 PRINT"FANGS- UND ENDPUNKTE
DER DAR-"
2550 PRINT"ZUSTELLENDEN LINIEN E
INGEGE-"
2560 PRINT"BEN WERDEN (MAX.50 PA
ARE)!":PRINT
2570 LINE 272,208,272,239,7
2580 LINE 248,184,272,208
2590 LINE 272,208,295,184
2600 PRINT AT(1,34);"Z":PRINT AT
(8,30);"X";PRINT AT(8,37);"Y"
2610 PRINT"EMPFEHLUNGEN FUER DEN
 WERTE-"
2620 PRINT"BEREICH":PRINT"X= 0 B
IS 100"
2630 PRINT"Y= 0 BIS 100":PRINT"Z
= 0 BIS 100":PRINT
2640 PRINT"ABSCHLUSS DER EINGABE
N DURCH"
```

```
2650 PRINT"DIE ZAHL 999":PRINT
2660 PRINT"GEBEN SIE NUN DIE ANF
ANGS-"
2670 PRINT"(AP) UND ENDPUNKTE(EP
) EIN:"
2680 WINDOW 18,31,0,39
2690 LET I=1
2700 INPUT"X-AP=";AP(I,1)
2710 IF AP(I,1)=999 THEN GOTO278
0
2720 INPUT"Y-AP=";AP(I,2)
2730 INPUT"Z-AP=";AP(I,3)
2740 INPUT"X-EP=";EP(I,1)
2750 INPUT"Y-EP=";EP(I,2)
2760 INPUT"Z-EP=";EP(I,3)
2770 LET I=I+1:GOTO2700
2780 LET I=I-1:LET AN=I:WINDOW 0
,31,0,39
2790 RETURN
3000 REM UP DATA-ZEILEN LADEN (M
AX.50)
3010 READ AN
3020 FOR I=1 TO AN
3030 READ AP(I,1),AP(I,2),AP(I,3
),EP(I,1),EP(I,2),EP(I,3)
3040 NEXT I
3050 RETURN
3100 REM UP KANTENLAENGE QUADER
3110 CLS
3120 PRINT"KANTENLAENGEN DES QUA
DERS EINGEBEN":PRINT
3130 INPUT"TIEFE (X-RICHTUNG,10.
..100)=";XL
3140 INPUT"BREITE (Y-RICHTUNG,10
...100)=";YL
3150 INPUT"HOEHE (Z-RICHTUNG,10.
..100)=";ZL
3160 FOR I=1 TO 12
3170 LET AP(I,1)=AP(I,1)*XL:LET
EP(I,1)=EP(I,1)*XL
3180 LET AP(I,2)=AP(I,2)*YL:LET
EP(I,2)=EP(I,2)*YL
3190 LET AP(I,3)=AP(I,3)*ZL:LET
EP(I,3)=EP(I,3)*ZL
3200 NEXT I
3210 RETURN
3300 REM UP KANTENLAENGE PYRAMID
E
3310 CLS
3320 PRINT"KANTENLAENGEN UND HOE
HE DER PYRAMIDE"
3330 PRINT"EINGEBEN":PRINT
3340 INPUT"TIEFE (X-RICHTUNG,10.
..100)=";XL
3350 INPUT"BREITE (Y-RICHTUNG,10
...100)=";YL
3360 INPUT"HOEHE (Z-RICHTUNG,10.
..100)=";ZL
3370 FOR I=1 TO 8
3380 LET AP(I,1)=AP(I,1)*XL:LET
EP(I,1)=EP(I,1)*XL
3390 LET AP(I,2)=AP(I,2)*YL:LET
EP(I,2)=EP(I,2)*YL
3400 LET AP(I,3)=AP(I,3)*ZL:LET
EP(I,3)=EP(I,3)*ZL
3410 NEXT I
3420 RETURN
4000 REM UP ZEICHENROUTINE
4010 IF XA<0 OR XA>319 OR YA<0 O
R YA>255 THEN GOTO 4040
4020 IF XE<0 OR XE>319 OR YE<0 O
R YE>255 THEN GOTO 4040
4030 LINE XA,YA,XE,YE,7
4040 RETURN
5000 REM EINHEITSQUADER
5002 DATA 12:REM ANZAHL DER LINI
EN
5004 DATA 1,0,0,1,1,0
5006 DATA 1,0,0,0,0,0
5008 DATA 1,1,0,0,1,0
5010 DATA 0,0,0,0,1,0
5012 DATA 1,0,0,1,0,1
5014 DATA 1,1,0,1,1,1
5016 DATA 0,1,0,0,1,1
5018 DATA 0,0,0,0,0,1
5020 DATA 1,0,1,1,1,1
5022 DATA 1,1,1,0,1,1
5024 DATA 0,0,1,0,1,1
5026 DATA 1,0,1,0,0,1
6000 REM EINHEITSPYRAMIDE
6002 DATA 8:REM ANZAHL DER LINIE
N
6004 DATA 1,0,0,1,1,0
6006 DATA 1,0,0,0,0,0
6008 DATA 1,1,0,0,1,0
6010 DATA 0,0,0,0,1,0
6012 DATA 1,0,0,.5,.5,1
6014 DATA 1,1,0,.5,.5,1
6016 DATA 0,1,0,.5,.5,1
6018 DATA 0,0,0,.5,.5,1
7000 REM BUNGALOW
7002 DATA 49:REM ANZAHL DER LINI
EN
7004 DATA 0,0,0,103,0,0
7006 DATA 103,0,0,103,63,0
7008 DATA 103,63,0,0,63,0
7010 DATA 0,63,0,0,0,0
7012 DATA 0,0,0,0,0,43
7014 DATA 103,0,0,103,0,43
7016 DATA 103,63,0,103,63,40
7018 DATA 0,63,0,0,63,40
7020 DATA 0,0,40,103,0,40
7022 DATA 103,0,40,103,75,40
7024 DATA 0,75,40,0,0,40
7026 DATA 103,75,40,0,75,40
7028 DATA 0,0,43,103,0,43
7030 DATA 103,0,43,103,75,47
7032 DATA 103,75,47,0,75,47
7034 DATA 0,75,47,0,0,43
7036 DATA 103,75,40,103,75,47
7038 DATA 0,75,40,0,75,47
7040 DATA 103,40,33,103,23,33
7042 DATA 103,23,33,103,23,13
7044 DATA 103,23,13,103,40,13
7046 DATA 103,40,13,103,40,33
7048 DATA 82,63,33,98,63,33
7050 DATA 98,63,33,98,63,13
7052 DATA 98,63,13,82,63,13
7054 DATA 82,63,13,82,63,33
7056 DATA 62,63,33,78,63,33
7058 DATA 78,63,33,78,63,13
7060 DATA 78,63,13,62,63,13
7062 DATA 62,63,13,62,63,33
7064 DATA 43,63,33,57,63,33
7066 DATA 57,63,33,57,63,0
7068 DATA 43,63,0,43,63,33
7070 DATA 22,63,33,38,63,33
7072 DATA 38,63,33,38,63,13
7074 DATA 38,63,13,22,63,13
7076 DATA 22,63,13,22,63,33
7078 DATA 0,40,33,0,23,33
7080 DATA 0,23,33,0,23,13
7082 DATA 0,23,13,0,40,13
7084 DATA 0,40,13,0,40,33
7086 DATA 10,0,33,18,0,33
7088 DATA 18,0,33,18,0,29
7090 DATA 18,0,29,10,0,29
7092 DATA 10,0,29,10,0,33
7094 DATA 23,0,33,31,0,33
7096 DATA 31,0,33,31,0,29
7098 DATA 31,0,29,23,0,29
7100 DATA 23,0,29,23,0,33
```

In Zeile 40 des Programms werden die Felder AP, EP, AD und ED dimensioniert. Das Feld AP nimmt die Anfangs- und das Feld EP die Endpunkte der einzelnen Geraden (max. 50) auf. Diese Anfangs- und Endpunkte werden

entweder aus den DATA-Zeilen der angebotenen Objekte entnommen, oder in einem besonderen Programmteil selbst erzeugt. Die Felder AD und ED werden nur bei der Zentralprojektion benötigt. Sie nehmen die »gedrehten« Anfangs- und Endpunkte auf. Die erste Dimension in den genannten Feldern ist die Nummer der entsprechenden Geraden und die zweite Dimension steht für die Koordinaten x (entspricht 1), y und z. So enthält z. B. das Feldelement AP(8,1) die x-Koordinate des Anfangspunktes der achten Geraden.

Im Menü werden als gespeicherte Objekte ein Quader, eine Pyramide und ein Bungalow angeboten. Für Quader und Pyramide können beliebige Maße (sinnvoll zwischen 10 und 100 Längeneinheiten) eingegeben werden. Diese Eingaben werden dann mit dem »Einheitsquader« (DATA-Zeilen 5000 bis 5026) bzw. der »Einheitspyramide« (DATA-Zeilen 6000 bis 6018) multipliziert. Es ist auch möglich, eigene Objekte durch Eingabe entsprechender Raumkoordinaten darzustellen. Dazu wird ein Unterprogramm genutzt, das in den Zeilen 2500 bis 2790 enthalten ist. Allerdings ist im vorliegenden Programm noch keine Abspeicherung der eingegebenen Raumkoordinaten vorgesehen. Hierzu müßten die Datenfelder AP und EP mit Hilfe des Kommandos CSAVE * beim KC 85/3 als gesonderte Datei abgespeichert und mit CLOAD * wieder eingelesen werden können. Wer das benötigt, sollte sich daran versuchen. Zu beachten ist in diesem Zusammenhang auch, daß der Aufruf eines neuen Objektes im Menü stets die Felder AP und EP löscht und damit auch ein evtl. mühsam eingegebenes eigenes Objekt »vollständig vernichtet«.

Das Programm ist in einzelne Blöcke gegliedert, deren Bedeutung Tabelle 1 zeigt. In den Programmblöcken, die die einzelnen Projektionsarten realisieren, begegnen wir den schon erklärten Projektionsgleichungen. So enthalten z. B. die Programmzeilen 690 bis 720 die Projektionsgleichungen für die schräge Parallelprojektion. Generell wird dabei in x-Richtung (also für XA und XE) die Zahl 160 und in y-Richtung die Zahl 120 addiert. Damit wird beim Zeichnen der Koordinatenursprung annähernd in die Bildschirmmitte (beim KC 85/3) gelegt. Falls Sie diese Parallelverschiebung direkt in die Zeichenroutine einbauen wollen, dann wird dies bei der senkrechten Zweitafelprojektion hinderlich; aber es führen ja viele Wege nach Rom. Die gesamte Berech-

Tabelle 1. Bedeutung der einzelnen Programmblöcke

Programmblock	Zeilennummern
Menü	10 bis 250
Senkr. Zweitafelp.	300 bis 500
Schräge Parallelp.	600 bis 750
Dimetrische Proj.	800 bis 950
Zentralprojektion	1000 bis 1550
Frage nach Programmwiederholungen	2000 bis 2060
Unterprogramme	**Zeilennummern**
frei wählbares Objekt	2500 bis 2790
DATA-Zeilen laden	3000 bis 3050
Kantenlängen des Quaders ermitteln	3100 bis 3210
Kantenlängen der Pyramide ermitteln	3300 bis 3420
Zeichenroutine	4000 bis 4040
DATA-Blöcke	**Zeilennummern**
Einheitsquader	5000 bis 5026
Einheitspyramide	6000 bis 6018
Bungalow	7000 bis 7100

nung (Zeilen 690 bis 720) ist Bestandteil einer Laufanweisung, die AN-mal durchlaufen wird. Die Variable mit dem Namen AN enthält stets die Anzahl der Geraden, aus denen das zu betrachtende Objekt zusammengesetzt ist. Nachdem für jeweils eine Gerade die x- und y-Koordinate des Anfangs- und Endpunktes berechnet worden sind (also XA, YA, XE und YE), erfolgt durch Aufruf der Zeichenroutine (GOSUB 4000) ein sofortiges Zeichnen dieser Geraden.

Ein gesamter Zeichnungsvorgang wird abgeschlossen, indem ab Programmzeile 2000 nach der gewünschten Programmfortsetzung gefragt wird. Soll am gleichen Objekt weitergearbeitet werden, dann bleiben die Inhalte der Datenfelder AP und EP erhalten. Ein Sonderfall der Programmfortsetzung liegt bei der Zentralprojektion vor. In den Zeilen 1540 und 1550 wird gefragt, ob am gleichen Objekt mit gleicher Drehung um die y-, z- und x-Achse weitergearbeitet werden soll. Wird diese Frage bejaht, dann bleiben auch die Inhalte der Felder AD und ED erhalten, denn die Berechnung der Gesamtdrehung nimmt stets einige Sekunden in Anspruch (besonders bei Objekten mit vielen Geraden).

Zum Umgang mit dem Programm

Einfache Objekte, wie Quader oder Pyramide, sind sehr gut geeignet, um die Wirkungen der einzelnen Projektionsarten kennenzulernen. Mehr Spaß bereiten natürlich kompliziertere Objekte (z.B. der Bungalow), und dies vor allem in Zentralprojektion. Die senkrechte Zweitafelprojektion offenbarte uns bei der Darstellung des Bungalows eine betrübliche Ungenauigkeit der LINE-Funktion. In dieser Projektionsart liegen ja z.B. im Aufriß einige Vorder- und Hinterkanten genau aufeinander. Die LINE-Funktion aber zeichnete die Vorder- und Hinterkante des schrägen Bungalowdaches um ein Pixel versetzt (siehe Bild 4). Die Vorderkante wird von links nach rechts und die Hinterkante von rechts nach links gezeichnet, und schon kam die Ungenauigkeit der LINE-Funktion zum Zuge.

Wer sich etwas intensiver mit Projektionen beschäftigen möchte, der sollte mit einem Quader mit 100 mal 100 mal 100 Längeneinheiten beginnen. Bei einem guten Bildschirm kann man sogar ein biegsames Lineal zur Hand nehmen, um die ungefähren Längen einiger Strecken zu ermitteln. Das

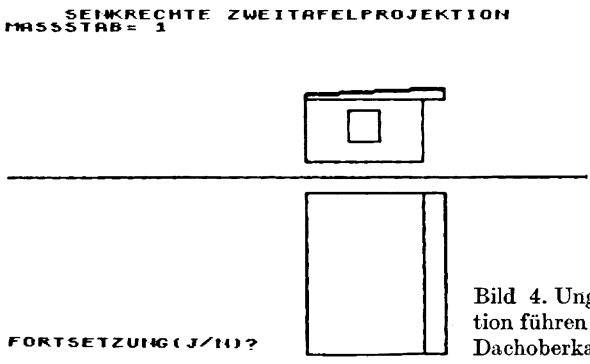
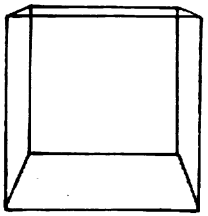


Bild 4. Ungenauigkeiten der LINE-Funktion führen zu doppelter Strichdicke an der Dachoberkante (Aufriß)

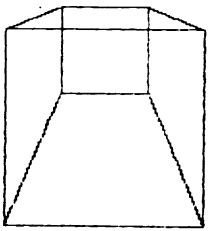
```
              ZENTRALPROJEKTION
MASSSTAB=  400 ENTFERNUNG=  420
SEITENA.=  50 AUGENH.=  120
WEITERE VAR. MIT GLEICHER DREHUNG(J/N)?
N
```



```
FORTSETZUNG(J/N)?
```

Bild 5. Quader mit geringer perspektivischer Wirkung (Teleobjektiv)

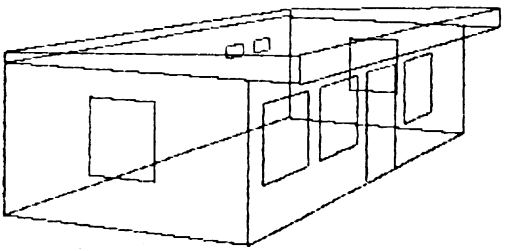
```
              ZENTRALPROJEKTION
MASSSTAB=  100 ENTFERNUNG=  160
SEITENA.=  50 AUGENH.=  120
WEITERE VAR. MIT GLEICHER DREHUNG(J/N)?
N
```



```
FORTSETZUNG(J/N)?
```

Bild 6. Quader mit starker perspektivischer Wirkung (Weitwinkelobjektiv)

```
              ZENTRALPROJEKTION
MASSSTAB=  480 ENTFERNUNG=  300
SEITENA.=  110 AUGENH.=  80
WEITERE VAR. MIT GLEICHER DREHUNG(J/N)?
N
```



```
FORTSETZUNG(J/N)?
```

Bild 7. Bungalow in Zentralprojektion

kann z. B. bei der Überprüfung des Verzerrungsverhältnisses bei der schrägen Parallelprojektion geschehen. Dieser Quader kann auch als Grundfigur dienen, um die etwas komplizierteren Verhältnisse der Zentralprojektion kennenzulernen. Dabei wird der Quader zunächst um keine der drei Achsen gedreht (jeweils Winkel 0 Grad eingeben). Für den Seitenabstand wählt

man 50 Längeneinheiten, womit wegen der Gesamtlänge von 100 Einheiten symmetrische Verhältnisse entstehen. Die Augenhöhe liefert mit 120 Einheiten aussagefähige Bilder. Die Wirkung eines Teleobjektivs entsteht z. B. mit folgenden Parametern: Maßstab = 400; Entfernung = 420; Seitenabstand = 50; Augenhöhe = 120 (Bild 5). Im Gegensatz dazu liefern folgende Parameter die Verhältnisse beim Weitwinkelobjektiv: Maßstab = 100; Entfernung = 180; Seitenabstand = 50; Augenhöhe = 120 (Bild 6).
Wer sich nach einigen Übungen in der Zentralprojektion »zu Hause fühlt«, der wird viel Spaß an der Darstellung eigener Objekte oder des angebotenen Bungalows haben. Letzterer kann nun problemlos in Augenhöhe des unerwünschten Kaffeegastes, aber ebenso in Frosch- oder Vogelperspektive bewundert werden (Bild 7).

Autor:
Dr. Hannes Gutzer
Wissenschaftlicher Mitarbeiter an der Akademie der Pädagogischen Wissenschaften, Forschungszentrum Halle-Neustadt

Beherrschen Sie BASIC?

Zur Darstellungsgenauigkeit von Gleitkommazahlen

Wer glaubt, die Bedeutung der einzelnen BASIC-Anweisungen völlig verstanden zu haben, der wundert sich von Zeit zu Zeit darüber, was der Computer mit dem eingegebenen Programm macht.
Nehmen wir ein Beispiel:

```
10 CLS
20 X = 0
30 FOR I = 1 TO 10000
40 X = X + 0.1
50 PRINT X
60 NEXT I
```

Man müßte doch erwarten, daß auf dem Bildschirm der Reihe nach die Zahlen 0.1, 0.2, 0.3, 0.4 usw. erscheinen und daß diese Zahlenfolge in der begonnenen Weise fortgesetzt wird, um dann schließlich abzubrechen, wenn der Wert 1000 erreicht ist.
Was aber tun die Kleincomputer der KC 85-Reihe?
Bis 7.7 reagieren die Rechner vollkommen richtig. Dann aber erscheinen die Zahlen

```
7.79999
7.89999
7.99999
8.09999
```

um danach wieder in der »ordentlichen« Reihenfolge 8.2, 8.3, ... fortzufahren. Etwas ähnliches passiert später wieder, und zwar werden von 22.5 bis 34.4 anstelle der richtigen Werte die Zahlen

```
22.5001
22.6001
22.7001
   :
34.4001
```

ausgegeben. Dann geht es wieder richtig weiter mit 34.5, 34.6 ... bis 40.9. Und von da an stimmt überhaupt nichts mehr. Die Zahlenwerte, die der Rechner ausgibt, werden von nun an alle zu klein, und zwar werden die Abweichungen vom richtigen Wert immer größer, je länger die Rechnung andauert. Statt 100 wird nur 99.9991 ausgegeben, statt 500 nur 499.921 und für 1000 erscheint als letzter Wert gar nur 999.903.

Wie kommt das? (s. S. 43)

Der Kleincomputer als optischer Täuscher

Nicht nur beim Auftritt eines Magiers müssen wir mit Täuschungen rechnen. Über 80 Prozent seiner Informationen nimmt der Mensch optisch auf; und das mit einer »Sehqualität« des Auges, über die HELMHOLTZ (1821 bis 1894) folgendes berichtet: »Wollte mir ein Optiker ein Instrument verkaufen, das derartige Mängel aufweist, so würde ich in sehr deutlichen Worten meine Mißbilligung seiner Arbeit kundtun und es ihm zurückgeben.« Die kritische Haltung von HELMHOLTZ gegenüber dem Handwerk ist bewundernswert, seine festgestellten Mängel am Auge sind richtig, seine idealistische Auffassung vom Sehen als Erkenntnisprozeß wurde aber von materialistischen Denkern kritisiert. So stellte F. ENGELS fest, daß die spezielle Konstruktion des menschlichen Auges keine absolute Schranke des menschlichen Erkenntnisprozesses ist, sondern daß zu diesem Prozeß unbedingt auch die Denktätigkeit dazugehört.

Der Informatiker bezeichnet das Auge als biologischen Sensor, der Eingangssignale an die Informationszentrale (Gehirn) weitergibt, in der diese Signale mit Hilfe von Interpretationsroutinen verarbeitet werden und damit den Organismus zur Reaktion veranlassen. Diese Interpretationsroutinen bezeichnen Computerfreunde als Programm, womit wir langsam zum Thema kommen.

Optische Täuschungen können ihre Ursache in der Unvollkommenheit des Auges (Mängel am Sensor) oder in der falschen Beurteilung des Gesehenen (Mängel im Programm) oder auch in beidem haben. So unterscheidet z.B. ARTAMONOW optische Täuschungen, die ihren Ursprung ausschließlich im Bau des Auges haben, von solchen, bei denen die Interpretationsroutine Unkorrektheiten liefert. Solche, von den Psychologen als geometrisch-optische Täuschungen bezeichnete Effekte, basieren z.B. auf der Überbewertung vertikaler Linien, auf Figur-Grund-Differenzierungen, auf perspektivischem Betrachten, auf der Beobachtung bewegter Objekte, auf dem Umgang mit Farben, auf der Überschätzung spitzer Winkel und auf der Betrachtung von Teil und Ganzem.

In unserem kleinen Computerprogramm (Bild 1) haben wir nur drei von vielen optischen Täuschungen dargestellt. Dies soll zugleich eine Aufforderung an die Leser sein, das Programm entsprechend zu erweitern (Literatur ist am Schluß des Beitrages angegeben). Für die eigentliche Darstellung der geometrisch-optischen Täuschungen brauchten wir natürlich keinen Computer, da wären Papier und Bleistift billiger. Der Computer kann uns aber auf Wunsch die Aufklärung der Täuschung liefern, indem z.B. eine unterbrochene Linie

```
10 REM OPTISCHE TAEUSCHUNGEN
20 WINDOW 0,3,0,39:COLOR 7,0:CLS

30 WINDOW 4,31,0,39:COLOR 0,6:CL
S
40 LOCATE 5,5:PRINT"OPTISCHE TAE
USCHUNGEN"
50 PRINT TAB(5);STRING$(21,"-")
60 PRINT:PRINT TAB(5);"1 = EFFEK
T VON HERING"
70 PRINT:PRINT TAB(5);"2 = EFFEK
T VON PUGGENDORF"
80 PRINT:PRINT TAB(5);"3 = EFFEK
T VON MUELLER-LYER"
90 LOCATE 20,5:INPUT"BITTE KENNZ
AHL EINGEBEN:";KZ
100 IF KZ<1 OR KZ>3 THEN GOTO30
110 CLS:ON KZ GOTO1000,2000,3000

200 REM ABSCHLUSS
210 PRINT"WEITERE OPTISCHE TAEUS
CHUNGEN"
220 INPUT"GEWUENSCHT (J/N)?";J$
230 IF J$="J" THEN CLS:GOTO30
240 WINDOW 0,31,0,39:COLOR 7,1:C
LS:END
1000 REM HERING
1010 GOSUB1120:GOSUB1200
1020 WINDOW 0,3,0,39:COLOR 7,0
1030 PRINT"SIND DIE HORIZONTALEN
 LINIEN"
1040 PRINT"GERADE ODER KRUMM?"
1050 INPUT"ZUR AUFKLAERUNG: <ENT
ER>";J$:CLS
1060 WINDOW 4,31,0,39:COLOR 0,6:
CLS
1070 GOSUB1200
1080 WINDOW0,3,0,39:COLOR 7,0
1090 INPUT"<ENTER>";J$:CLS
1100 GOSUB1120
1110 GOTO 200
1120 REM UP STRAHLEN
1130 FOR X=10 TO 319 STEP 15
1140 LINE X,0,315+X-2*X,223,0
1150 NEXT X
1160 FOR Y=5 TO 223 STEP 15
1170 LINE 0,Y,319,223+Y-2*Y,0
1180 NEXT Y
1190 RETURN
1200 REM UP PARALLELE GERADE
1210 LINE 0,112-30,319,112-30,0
1220 LINE 0,112+30,319,112+30,0
1230 RETURN
2000 REM PUGGENDORF
2010 FOR I=5 TO 34
2020 PRINT AT(15,I);PAPER 0;" "
2030 PRINT AT(16,I);PAPER 0;" "
2040 PRINT AT(17,I);PAPER 0;" "
2050 PRINT AT(18,I);PAPER 0;" "
2060 PRINT AT(19,I);PAPER 0;" "
2070 NEXT I
2080 LINE 40,25,280,200,0
2090 WINDOW 0,3,0,39:COLOR 7,0
2100 PRINT"SIND DIE BEIDEN GERAD
EN"
2110 PRINT"GEGENEINANDER VERSETZ
T?"
2120 INPUT"ZUR AUFKLAERUNG: <ENT
ER>";J$:CLS
2130 LINE 40,25,280,200,7
2140 GOTO 200
3000 REM MUELLER-LYER
3010 LET XL=56:LET XR=264:LET YU
=60:LET YO=170
3020 LINE XL,YO,XR,YO,0
3030 FOR I=XL TO XL+30:PSET I,YO
+I-XL:PSET I,YO-(I-XL):NEXT I
3040 FOR I=XR TO XR-30 STEP -1:P
SET I,YO-(I-XR):PSET I,YO+I-XR:N
EXT I
3050 LINE XL,YU,XR,YU
3060 FOR I=XL TO XL-30 STEP -1:P
SET I,YU-(I-XL):PSET I,YU+I-XL:N
EXT I
3070 FOR I=XR TO XR+30:PSET I,YU
+I-XR:PSET I,YU-(I-XR):NEXT I
3080 WINDOW 0,3,0,39:COLOR 7,0
3090 PRINT"SIND DIE BEIDEN GERAD
EN GLEICH LANG?":PRINT
3100 INPUT"ZUR AUFKLAERUNG: <ENT
ER>";J$:CLS
3110 FOR I=YU TO YO-5 STEP 5
3120 PSET XL,I:PSET XL,I+1:PSET
XL,I+2
3130 PSET XR,I:PSET XR,I+1:PSET
XR,I+2
3140 NEXT I
3150 GOTO 200
```

Bild 1

in anderer Farbe durchgezeichnet wird oder gestrichelte Linien die Verbindung zwischen zwei Punkten herstellen. Für die Realisierung unseres Vorhabens benötigen wir einen Vollgrafik-Computer. Das Programm in Bild 1 wurde für den Kleincomputer KC 85/3 und KC 85/2 mit BASIC-Modul geschrieben, es ist aber auf andere Vollgrafik-Computer übertragbar. Dabei müßte besonders auf folgendes geachtet werden:

- Der KC 85/3 hat 32 Bildschirmzeilen (die meisten Kleincomputer haben nur 24) und 40 Bildschirmspalten.
- Die Vollgrafik des KC 85/3 bietet 320 Bildpunkte in x- und 256 Bildpunkte in y-Richtung, wobei der Koordinatenursprung links unten liegt.
- Die Farbanweisungen COLOR, INK und PAPER müssen unter Umständen entsprechend geändert werden.
- Die LINE-Funktion zum Zeichnen von Geraden hat folgenden Aufbau: LINE X-Anfang, Y-Anfang, X-Ende, Y-Ende, Vordergrundfarbe.

Zur Erleichterung der Programmübertragung auf andere Grafikbildschirme sind im Programm einige Ausdrücke bewußt nicht zusammengefaßt worden (z. B. Programmzeilen 1140, 1170, 1210 und 1220 in Bild 1). Die Programmzeilen 20 und 30 zeigen, daß die ersten vier Bildschirmzeilen als Anzeigefenster für Text (weiße Schrift auf schwarzem Grund) bei der Betrachtung der einzelnen optischen Täuschungen fungieren werden. Die Täuschungen selbst erscheinen ab Bildschirmzeile 5 als schwarze Zeichen auf gelbem Grund. In einem Menü werden in den Zeilen 40 bis 100 die drei optischen Täuschungen zur Auswahl angeboten. Dieses Menü kann beliebig erweitert werden. Das müßte dann auch bei der ON ... GOTO-Anweisung in Zeile 110 entsprechend

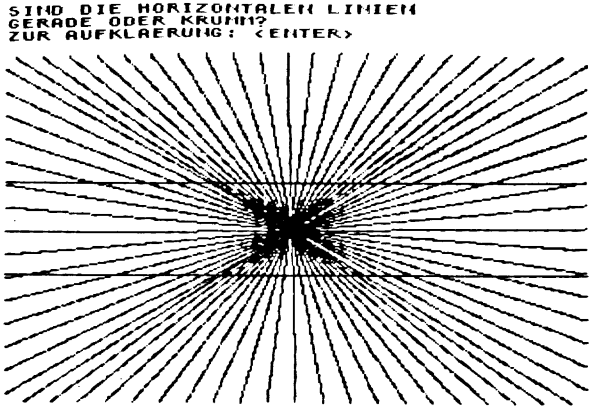


Bild 2

berücksichtigt werden. So könnte die vierte optische Täuschung z. B. ab Programmzeile 4000 beginnen.
Aus jedem Programmteil mit einer optischen Täuschung wird zur Programmzeile 200 zurückgesprungen, um über Programmabbruch oder -fortsetzung zu entscheiden. Bei gewünschtem Abbruch wird das Programm in Zeile 240 mit der END-Anweisung gestoppt.
Die Zeilen 1000 bis 1230 enthalten eine optische Täuschung, die von dem deutschen Physiologen E. HERING (1834 bis 1918) stammt. Bild 2 zeigt die Fragestellung und die Darstellung auf dem Bildschirm. Ursache für die optische Täuschung ist die vorhin schon genannte Überschätzung spitzer Winkel. Sowohl die Strahlen als auch die beiden parallel verlaufenden Geraden werden im Programm durch Unterprogramme realisiert. Der Vorteil zeigt sich erst, wenn ab Programmzeile 1050 zur Aufklärung geschritten wird (die INPUT-Anweisung hat hier nur eine Anhaltefunktion). Zur Aufklärung wird zunächst der Grafikbildschirm gelöscht, dann erscheinen nur die beiden Geraden. Bei erneuter Programmfortsetzung (ab Zeile 1090) wird das Gitter gezeichnet. Hier kann man sehr schön den Umschlag von geraden Linien in scheinbar krumme Linien verfolgen.
Die optische Täuschung, die der deutsche Physiker J. C. POGGENDORFF (1796 bis 1877) aufgestellt hat, basiert ebenfalls auf der Überschätzung spitzer Winkel. Die Poggendorffsche Täuschung wird in den Programmzeilen 2000 bis 2140 realisiert. Bild 3 zeigt das Problem und die Bildschirmdarstellung. Die Aufklärung erfolgt in der Programmzeile 2130, in der die vollständige Gerade in weißer Farbe (Farbzahl 7 beim KC85/3) auf gelbem Grund gut zu erkennen ist. Der große schwarze Querbalken wird übrigens aus schwarzen Leerzeichen (Zeilen 2020 bis 2060) hergestellt, schwarz liegt also als Hintergrundfarbe (PAPER-Anweisung) vor. Das ist dann von Bedeutung, wenn die Gerade mit weißer Vordergrundfarbe durch den schwarzen Querbalken hindurchgeht, denn in einen (4 mal 8)-Bildpunktfeld des KC 85/3 kann es stets nur genau eine Vorder- und Hintergrundfarbe geben. Andere Kleincomputer haben noch schlechtere Farbauflösungen (meist 8 mal 8 Bildpunkte, dies entspricht dann der Größe eines Zeichens auf dem Bildschirm).
Die in Bild 4 dargestellte optische Täuschung stammt von dem deutschen

SIND DIE BEIDEN GERADEN
GEGENEINANDER VERSETZT?
ZUR AUFKLAERUNG: <ENTER>

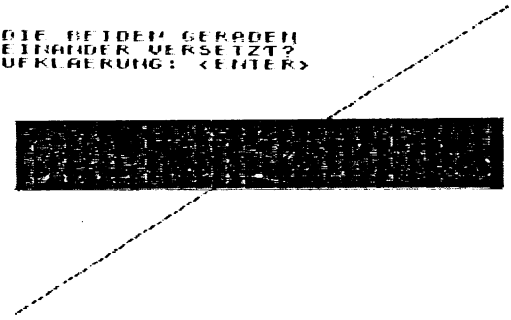


Bild 3

SIND DIE BEIDEN GERADEN GLEICH LANG?
ZUR AUFKLAERUNG: <ENTER>

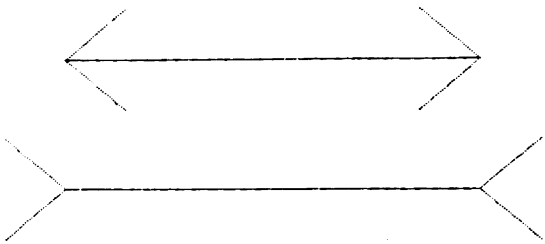


Bild 4

WEITERE OPTISCHE TAEUSCHUNGEN
GEWUENSCHT (J/N)?

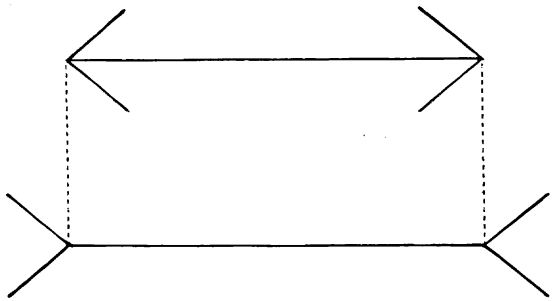


Bild 5

Arzt F. Müller-Lyer (1857 bis 1916). Sie basiert auf einem falschen Schluß vom Ganzen zum Teil in der Weise, daß unsere Interpretationsroutine z.B. Größenvergleiche nicht zwischen den entscheidenden Begrenzungen der Figuren (also Teil der Figur), sondern zwischen den Figuren selbst (also dem Ganzen) durchführt. Bei scharfsichtigem Auge und Inline-Farbbildröhre können deren senkrecht angeordneten Schlitze den Effekt etwas verwischen. Der Betrachter muß dann den Abstand zur Bildröhre vergrößern. Die optische Täuschung von Müller-Lyer wird in den Programmzeilen 3000 bis 3150 realisiert. Zur Erleichterung des Umschreibens dieses Programmteils auf

andere Grafikbildschirme sind in Zeile 3010 die Anfangs- und Endpunkte der beiden Geraden den Variablen XL (x-Koordinate links), XR, YU (y-Koordinate unten) und YO zugeordnet.
Bei der Darstellung der um 45 Grad geneigten »Endstücke« der Geraden hat uns die LINE-Funktion mit ihren Ungenauigkeiten so enttäuscht, daß wir diese Geradenstücke über Laufanweisungen in den Programmzeilen 3030, 3040, 3060 und 3070 realisiert haben. Die Aufklärung der Täuschung erfolgt ab Zeile 3110 mit Hilfe von zwei gestrichelten Geraden, so wie es Bild 5 zeigt.
Ab Programmzeile 4000 können Sie nun weitere geometrisch-optische Täuschungen mit der entsprechenden Aufklärung programmieren. Wie wäre es mit dem SANDERschen Parallelogramm, der ZÖLLNERschen Täuschungsfigur, der EBBINGHAUSschen Kreistäuschung oder dem Effekt von EHRENSTEIN? Diese und viele andere optischen Täuschungen finden Sie in den folgenden Büchern.

Literatur

[1] ARTAMONOW, I. D.: Optische Täuschungen. – Leipzig: B. G. Teubner Verlagsgesellschaft, 1985
[2] KLEBE, J.; KLEBE, I.: Durch die Augen in den Sinn. – Berlin: Deutscher Verlag der Wissenschaften, 1984
[3] PLATONOW, K.: Unterhaltsame Psychologie. – Leipzig: Urania-Verlag, 1976
[4] Form und Dekor. – Leipzig: Deutscher Verlag für Grundstoffindustrie, 1986
[5] Wörterbuch der Psychologie. – Leipzig: Bibliographisches Institut, 1985

Autor:
Dr. Hannes Gutzer
Wissenschaftlicher Mitarbeiter
an der Akademie
der Pädagogischen Wissenschaften
Forschungszentrum Halle-Neustadt

Lösung (von S. 38):

Wenn man mit einem Computer arbeitet, so muß man wissen, daß unser gebräuchliches Dezimalzahlsystem für den Rechner äußerst unbequem ist. Er rechnet viel lieber in dem für ihn geeigneteren Dualzahlsystem, in dem nur die beiden Ziffern Ø und 1 vorhanden sind. Das bedeutet, daß jede Zahl, die wir dem Computer im Programm oder durch eine Eingabe mitteilen, in die sogenannte interne Zahlendarstellung umgewandelt wird. Bei den Rechnern der KC 85-Reihe handelt es sich dabei um eine Gleitkommadarstellung, bei der für jede Zahl 4 Byte zur Verfügung stehen; drei davon für die Mantisse und ein Byte für den Exponenten. Die Mantisse einer jeden Zahl wird also durch insgesamt $3 \cdot 8 - 1 = 23$ Dualstellen dargestellt (ein Bit wird für die Kennzeichnung des Vorzeichens der Zahl benötigt).
Nun hat aber das Dualzahlsystem die unangenehme Eigenschaft, daß ein *endlicher Dezimalbruch nicht unbedingt auch ein endlicher Dualbruch* sein muß. Und dies ist bei Ø.1 der Fall. Wandelt man nämlich die Dezimalzahl $Ø.1_{dez}$ durch die im Dualzahlsystem durchgeführte Divisionsaufgabe

1:1Ø1Ø

in den zugehörigen Dualbruch um, so erhält man

$$Ø.1_{dez} = Ø.ØØØ1ØØ1ØØ1ØØ1ØØ1ØØ1ØØ1ØØ1ØØ\ldots{}_{dual}$$
$$= Ø.\overline{ØØØ1ØØ}_{dual}.$$

(Fortsetzung auf S. 52)

Berechnung optimaler Erneuerungsintervalle für die vorbeugende Instandhaltung mit Hilfe des Bürocomputers A 5120

1. Einleitung

Einen wichtigen Platz in der betrieblichen Zuverlässigkeitsarbeit nehmen die Probleme der *Instandhaltung* vorhandener Anlagen ein. Man unterscheidet generell zwischen Havarieinstandsetzungen (*Havarieerneuerungen*), die nach einem *Ausfall* der Anlage durchgeführt werden, und prophylaktischen Instandsetzungen (*prophylaktischen Erneuerungen*), die an der noch *nicht ausgefallenen* Anlage vollzogen werden. Eine Kenngröße der Zuverlässigkeitsarbeit, die oft als Maß für die Wirksamkeit von Instandhaltungsmaßnahmen dient, ist die *Dauerverfügbarkeit* A, das ist die Wahrscheinlichkeit dafür, daß die betrachtete Anlage in einem beliebigen Zeitpunkt t des bereits annähernd *stationären* Betriebsablaufes (d.h., $t \gg 0$) arbeitet oder *einsatzbereit* ist.

Es sind verschiedene Instandhaltungsstrategien bekannt.

Werden beispielsweise keine prophylaktischen Erneuerungen, sondern *nur* Havarieinstandsetzungen vorgesehen, so kann die Dauerverfügbarkeit nach der Formel

$$A = \frac{E\,\xi}{E\,\xi + E\,\eta_{\mathrm{H}}} \qquad (1)$$

berechnet werden. Darin bezeichnen ξ die *zufällige* Lebensdauer der Anlage, η_{H} die *zufällige* Dauer einer Havarieinstandsetzung und $\mathbf{E}\,\xi$ sowie $\mathbf{E}\,\eta_{\mathrm{H}}$ die zugehörigen Erwartungswerte (*mittlere* Lebens- und Instandsetzungsdauer) Bei der Modellbildung, die sich *näherungsweise* auf viele praktische Fälle anwenden läßt, wird angenommen, daß nach *jedem* Ausfall der Anlage *sofort* mit der Instandhaltung begonnen wird (andernfalls muß man eben Wartezeiten mit zur Instandsetzungsdauer rechnen) und daß die Anlage nach *jeder* Instandsetzung wieder »wie neu« ist, d.h., neben der stets gleichen mittleren Instandsetzungsdauer $\mathbf{E}\,\eta_{\mathrm{H}}$ hat auch die mittlere Lebensdauer $\mathbf{E}\,\xi$ in jeder Funktionsperiode den gleichen Wert.

Die Formel (1) läßt erkennen, daß die Dauerverfügbarkeit A dadurch erhöht werden kann, daß die mittlere Lebensdauer $\mathbf{E}\,\xi$ vergrößert oder/und die mittlere Instandsetzungsdauer $\mathbf{E}\,\eta_{\mathrm{H}}$ verkürzt werden. Da dies aber nur in den wenigsten Fällen möglich ist, werden neben Havarieinstandsetzungen noch vorbeugende Erneuerungen nach bestimmten Strategien durchgeführt.

Eine dieser Strategien ist die *altersabhängige Erneuerung* (*age replacement policy*). Hierbei wird die Anlage nach Ausfällen durch Havarieerneuerungen der zufälligen Dauer η_{H} erneuert. Hat sie, von der Beendigung der letzten Instandhaltungsmaßnahme (ganz gleich welcher Art) an gerechnet, eine vorge-

gebene konstante Zeitspanne τ ausfallfrei gearbeitet, so wird die Anlage stillgelegt und in der zufälligen Zeit η_P prophylaktisch erneuert. Auch hier seien alle Erneuerungen *vollständig*, d.h., nach der Erneuerung ist die Anlage »wie neu«.

Offenbar ist bei dieser Strategie die prophylaktische Instandsetzung nur dann sinnvoll, wenn deren mittlere Dauer $\mathbf{E}\,\eta_P$ kleiner als die mittlere Dauer $\mathbf{E}\,\eta_H$ einer Havarieerneuerung ist. Die Dauerverfügbarkeit der Anlage kann bei der Strategie der altersabhängigen Erneuerung bei gegebenem τ nach der etwas komplizierteren Formel

$$A(\tau) = \frac{\int_0^\tau \bar{F}(x)\,\mathrm{d}x}{\int_0^\tau \bar{F}(x)\,\mathrm{d}x + F(\tau)\,\mathbf{E}\,\eta_H + [1-F(\tau)]\,\mathbf{E}\,\eta_P} \tag{2}$$

$[\bar{F}(x) = 1 - F(x)]$ berechnet werden. Während man in Formel (1) nur die mittleren Dauern $\mathbf{E}\,\xi$, $\mathbf{E}\,\eta_H$ kennen muß, ist in Formel (2) neben den mittleren Instandsetzungsdauern $\mathbf{E}\,\eta_H$, $\mathbf{E}\,\eta_P$ die Kenntnis der Verteilungsfunktion

$$F(x) = Pr(\xi < x) \tag{3}$$

der Lebensdauer ξ gefordert. Bei Formel (1) kann man die Dauerverfügbarkeit nicht erhöhen, wenn die Mittelwerte $\mathbf{E}\,\xi$ und $\mathbf{E}\,\eta_H$ unverändert bleiben. Dagegen läßt sich die Dauerverfügbarkeit $A(\tau)$ in Formel (2) selbst bei unveränderten Werten für $\mathbf{E}\,\eta_H$ und $\mathbf{E}\,\eta_P$ sowie unveränderter Verteilungsfunktion $F(x)$ durch geeignete Wahl des *Parameters* τ erhöhen.

Wir können also folgende *Optimierungsaufgabe* betrachten:

Gesucht ist der *optimale* Wert $\tau = \tau^*$, der die Dauerverfügbarkeit $A(\tau)$ der Anlage im Falle der altersabhängigen Erneuerung *maximiert*!

Man kann mit den Mitteln der Differentialrechnung zeigen (vgl. [1], [4]), daß sich *ein* optimaler Wert τ^* für τ als Lösung der Gleichung

$$q(\tau)\int_0^\tau \bar{F}(x)\,\mathrm{d}x - F(\tau) - \frac{\mathbf{E}\,\eta_P}{\mathbf{E}\,\eta_H - \mathbf{E}\,\eta_P} = 0 \tag{4}$$

ermitteln läßt, wenn zusätzlich zur Forderung $\mathbf{E}\,\eta_P < \mathbf{E}\,\eta_H$ noch folgende *Voraussetzungen* erfüllt sind:

$q(x) := f(x)/\bar{F}(x)$ (die *Ausfallrate*) ist streng monoton wachsend in x $(x \geqq 0), f(x) = F'(x)$ (5)

$$q(\infty)\,E\,\xi > \frac{\mathbf{E}\,\eta_H}{\mathbf{E}\,\eta_H - \mathbf{E}\,\eta_P} \tag{6}$$

Die zu τ^* gehörende *maximale* Dauerverfügbarkeit kann dann auch nach der im Vergleich zu (2) einfacheren Formel

$$A(\tau^*) = \frac{1}{1 + [\mathbf{E}\,\eta_H - \mathbf{E}\,\eta_P]\,q(\tau^*)} \tag{7}$$

berechnet werden.

2. Ein Algorithmus zur Berechnung optimaler Erneuerungsintervalle τ^*

Unter der Annahme, daß alle genannten Voraussetzungen erfüllt sind, besteht die wesentliche Aufgabe bei der Ermittlung optimaler Erneuerungsintervalle τ^* in der numerischen Auflösung einer nichtlinearen Gleichung der Form

$$\varphi(\tau) = 0 \tag{8}$$

nach τ, worin $\varphi(\tau)$ durch

$$\varphi(\tau) := q(\tau)\int_0^\tau \bar{F}(x)\,\mathrm{d}x - F(\tau) - \frac{\mathbf{E}\,\eta_P}{\mathbf{E}\,\eta_H - \mathbf{E}\,\eta_P} \tag{9}$$

gegeben ist.

Verwendet man beispielsweise das NEWTON-*Verfahren* (vgl. [2]), so geht man wie folgt vor:

(1) Man legt sich eine *Wertetabelle* für $\varphi(\tau)$ an und ermittelt ein Intervall $u < \tau < o$, das mit Sicherheit die Nullstelle τ^* von $\varphi(\tau)$ enthält (erkennbar an einem *Vorzeichenwechsel* von $\varphi(\tau)$!). Hierbei müssen i. allg. numerische Integrationen ausgeführt werden, da das Integral $\int_0^\tau \bar{F}(x)\,dx$ in (9) nur in den wenigsten Fällen in geschlossener Form angegeben werden kann. Der Algorithmus zur Berechnung von $\varphi(\tau)$ macht von der *Trapezregel* Gebrauch: Für ein vorher gewähltes hinreichend großes N gilt dann

$$\varphi(\tau) \approx q(\tau)\frac{\tau}{2N}\left[1 + \bar{F}(\tau) + 2\sum_{k=1}^{N-1}\bar{F}\left(\frac{k\tau}{N}\right)\right] - F(\tau) - \frac{\mathbf{E}\,\eta_P}{\mathbf{E}\,\eta_H - \mathbf{E}\,\eta_P}. \qquad (10)$$

Die Wertetabelle kann relativ grob angelegt werden und sollte mit $\varphi(0)$ beginnen. Das *Einschließungsintervall* $u < \tau^* < o$ sollte so klein wie nur möglich gewählt werden.

(2) Anschließend werden *Näherungswerte* τ_n für τ^* schrittweise nach dem NEWTON-*Verfahren* berechnet:

$$\tau_0 = u \text{ (gegebener } \textit{Startwert}) \qquad \tau_{n+1} = \tau_n - \frac{\varphi(\tau_n)}{\varphi'(\tau_n)} \quad (n = 0, 1, 2, \ldots) \qquad (11)$$

In der Iterationsformel des Verfahrens (11) sind $\varphi(\tau)$ nach (10) und $\varphi'(\tau)$ analog nach

$$\varphi'(\tau) = q'(\tau)\frac{\tau}{2N}\left[1 + \bar{F}(\tau) + 2\sum_{k=1}^{N-1}\bar{F}\left(\frac{k\tau}{N}\right)\right] \qquad (12)$$

(N wird auch hier hinreichend groß gewählt) für die jeweiligen Werte $\tau = \tau_n$ zu berechnen.

Nach *jedem* Iterationsschritt überprüft man, ob $u < \tau_{n+1} < o$ gilt. Liegt ein Wert τ_{n+1} außerhalb des Intervalls (u, o), so wird dieses Einschließungsintervall verkleinert und die Iteration mit einem neuen Startwert τ_0 von neuem begonnen. Dabei ist wie folgt vorzugehen: Zunächst stellt man fest, welches der beiden Intervalle $\left(u, \frac{u+o}{2}\right)$ oder $\left(\frac{u+o}{2}, o\right)$ ein neues Einschließungsintervall der gesuchten Nullstelle τ^* ist. Der neue Startwert τ_0 ist dann $\tau_0 = \frac{u+o}{2}$.

Ändern sich schließlich aufeinanderfolgende Näherungswerte τ_{n+1} von τ^* im Rahmen einer gewünschten Genauigkeit nicht mehr, so bricht man das Verfahren ab und verwendet für das gesuchte Optimum τ^* den Näherungswert $\tau^* \approx \tau_{n+1}$.

(3) Es ist angebracht, den Wert $A(\tau^*) \approx A(\tau_{n+1})$ nach Formel (2) zu berechnen und mit den Werten $A(\hat{\tau})$ und $A(\hat{\hat{\tau}})$ für zwei dicht bei τ_{n+1} liegende Nachbarstellen $\hat{\tau}$, $\hat{\hat{\tau}}$ ($\hat{\tau} < \tau_{n+1} < \hat{\hat{\tau}}$) zu vergleichen. Wenn $A(\tau_{n+1})$ der größte dieser drei Werte ist, so kann man sich auf den praktischen Standpunkt stellen, daß τ_{n+1} das gesuchte optimale Erneuerungsintervall ist.

3. BASIC-Programme für den Büro-Computer A 5120

Wir beschränken uns im folgenden auf die Berechnung optimaler Instandhaltungsintervalle im Falle weibullverteilter Lebensdauern, d.h., $F(x) = 1 - e^{-(x/a)^b}$, $x \geqq 0$. Für einen Formparameter $b > 1$ ist die Ausfallrate $q(x)$ streng monoton wachsend, d.h., die Voraussetzung (5) ist erfüllt. Entsprechend der im vorigen Abschnitt vorgeschlagenen Methode wurden drei BASIC-Programme geschrieben. Das

```
10 PRINT"wertetabelle der funktion phi(tau) nach formel (9)"
20 LPRINT"wertetabelle der funktion phi(tau) nach formel (9)'
30 PRINT"funktionsdauer weibullverteilt"
40 PRINT
50 PRINT"eingabe der parameter"
60 INPUT"formparameter der weibullverteilung b=";B
70 INPUT"skalenparameter der weibullverteilung a=";A
80 INPUT"mittlere prophylaktische erneuerungsdauer=";EETAP
90 INPUT"mittlere havarieerneuerungsdauer=";EETAH
100 INPUT"tabellenschrittweite h=";H
110 DEF FNF(X)=1-EXP(-(X/A)^B)
120 DEF FNQ(X)=B/A*(X/A)^(B-1)
130 TAU=H
140 REM berechnung des integrals in formel (2)
150 S1=(2-FNF(TAU))*TAU/2
160 N=2
170 S2=2-FNF(TAU)
180 FOR K=1 TO N-1
190 S2=S2+2*(1-FNF(K*TAU/N))
200 NEXT K
210 S2=S2*TAU/2/N
220 IF ABS(S1-S2)>ABS(S2)/100 THEN S1=S2 : N=N*2 : GOTO 170
230 REM berechnung des funktionswertes phi(tau)
240 PHI=FNQ(TAU)*S2-FNF(TAU)-EETAP/(EETAH-EETAP)
250 LPRINT"phi(";
260 LPRINT USING"#####";TAU;
270 LPRINT")=";PHI
280 IF PHI<0 THEN TAU=TAU+H : GOTO 140
290 END
```

Bild 1. Programm »TABELLE«

```
10 PRINT"berechnung des optimalen instandhaltungsintervalls"
20 PRINT"einer reparierbaren betrachtungseinheit"
30 PRINT"bei altersabhaengiger vorbeugender erneuerung"
40 PRINT"funktionsdauer weibullverteilt"
50 PRINT
60 PRINT"eingabe der parameter"
70 INPUT"formparameter der weibullverteilung b=";B
80 INPUT"skalenparameter der weibullverteilung a=";A
90 INPUT"mittlere prophylaktische erneuerungsdauer=";EETAP
100 INPUT"mittlere havarieerneuerungsdauer=";EETAH
110 PRINT"(u,o) = einschliessungsintervall fuer die nullstelle"
120 INPUT"u=";U
130 INPUT"o=";O
140 DEF FNF(X)=1-EXP(-(X/A)^B)
150 DEF FNQ(X)=B/A*(X/A)^(B-1)
160 GOTO 280
170 REM up zur berechnung von phi(tau)
180 S1=(2-FNF(TAU))*TAU/2
190 N=2
200 S2=2-FNF(TAU)
210 FOR K=1 TO N-1
220 S2=S2+2*(1-FNF(K*TAU/N))
230 NEXT K
240 S2=S2*TAU/2/N
250 IF ABS(S1-S2)>ABS(S2)/100 THEN S1=S2 : N=N*2 : GOTO 200
260 PHI=FNQ(TAU)*S2-FNF(TAU)-EETAP/(EETAH-EETAP)
270 RETURN
280 TAU=U
290 GOSUB 170
```

```
300 REM berechnung von phistrich(tau)
310 PHISTRICH=FNQ(TAU)/TAU*(B-1)*S2
320 REM newton-iteration
330 TAUNEU=TAU-PHI/PHISTRICH
340 PRINT"tau=";TAUNEU
350 IF ABS(TAUNEU-TAU)<=ABS(TAUNEU)/1000 THEN 410
360 IF U<TAUNEU AND TAUNEU<0 THEN TAU=TAUNEU : GOTO 290
370 TAU=(O+U)/2
380 GOSUB 170
390 IF PHI<0 THEN U=(O+U)/2 : GOTO 300
400 O=(O+U)/2 : GOTO 300
410 LPRINT"optimales instandhaltungsintervall tau="; INT(TAU)
420 END
```

Bild 2. Programm »NEWTON«

```
 10 PRINT"berechnung der dauerverfuegbarkeit"
 20 PRINT"einer reparierbaren betrachtungseinheit"
 30 PRINT"bei altersabhaengiger vorbeugender erneuerung"
 40 PRINT"funktionsdauer weibullverteilt"
 50 PRINT
 60 PRINT"eingabe der parameter"
 70 INPUT"formparameter der weibullverteilung b=";B
 80 INPUT"skalenparameter der  weibullverteilung a=";A
 90 INPUT"mittlere prophylaktische erneuerungsdauer=";EETAP
100 INPUT"mittlere havarieerneuerungsdauer=";EETAH
110 INPUT"instandhaltungsintervall=";TAU
120 DEF FNF(X)=1-EXP(-(X/A)^B)
130 REM berechnung des integrals in formel (2)
140 S1=(2-FNF(TAU))*TAU/2
150 N=2
160 S2=2-FNF(TAU)
170 FOR K=1 TO N-1
180 S2=S2+2*(1-FNF(K*TAU/N))
190 NEXT K
200 S2=S2*TAU/2/N
210 IF ABS(S1-S2)>ABS(S2)/100 THEN S1=S2 : N=N*2 : GOTO 160
220 REM berechnung der dauerverfuegbarkeit nach formel (2)
230 AD=S2/(S2+FNF(TAU)*EETAH+(1-FNF(TAU))*EETAP)
240 LPRINT"dauerverfuegbarkeit  a(";
250 LPRINT USING"#####";TAU;
260 LPRINT")=";
270 LPRINT USING"#.###";AD
280 END
```

Bild 3. Programm »DAUERV«

Programm »TABELLE« dient der Berechnung einer Wertetabelle der Funktion $\varphi(\tau)$ nach Formel (10). Mit dem Programm »NEWTON« wird dann das optimale Instandhaltungsintervall τ^* nach der NEWTON-Iteration (11) berechnet. Schließlich kann das Programm »DAUERV« zur Berechnung der Dauerverfügbarkeit nach Formel (2) benutzt werden. Diese Berechnung sollte man für das ermittelte optimale Erneuerungsintervall τ^* und (zur Kontrolle) für Werte τ in der Nähe von τ^* durchführen.

Die *Parametereingabe* erfolgt bei allen Programmen nach *Bildschirmaufforderung*.

4. Rechenbeispiele

Beispiel 1. Die Lebensdauer der reparierbaren Betrachtungseinheit sei verteilt nach $F(x) = 1 - e^{-(x/a)^b}$, $x \geq 0$, mit $a = 1{,}6 \cdot 10^3$ h und $b = 1{,}5$. Eine

Havarieerneuerung dauert im Mittel $\mathbf{E}\,\eta_H = 72$ h. Der Erwartungswert für die prophylaktische Erneuerungsdauer ist $\mathbf{E}\,\eta_P = 6\,h$. Gesucht ist das optimale Erneuerungsintervall τ^*. Mit Hilfe des Programms »TABELLE« bestimmen wir ein Einschließungsintervall für τ^*. Neben den Verteilungsparametern ist die Schrittweite h für die Wertetabelle einzugeben. Es werden dann die Funktionswerte $\varphi(h)$, $\varphi(2h)$, $\varphi(3h)$, ... ausgedruckt. Berücksichtigt man $F(0) = 0$ und $\mathbf{E}\,\eta_H > \mathbf{E}\,\eta_P$, so erkennt man in (9), daß $\varphi(0)$ kleiner als Null ist. Die Wertetabelle wird deshalb abgeschlossen, sobald der erste Funktionswert berechnet wurde, der größer als Null ist.
Für $h = 50$ wurde die Wertetabelle (Bild 4) berechnet.

```
wertetabelle der funktion phi(tau)
nach formel (9)
phi(   50)=-.0881512
phi(  100)=-.0831305
phi(  150)=-.0766705
phi(  200)=-.06903
phi(  250)=-.0605465
phi(  300)=-.0512029
phi(  350)=-.0411511
phi(  400)=-.0304731
phi(  450)=-.0192375
phi(  500)=-6.77063E-03
phi(  550)= 5.62355E-03
```

Bild 4. Wertetabelle für $\varphi(\tau)$ in Beispiel 1

Man erkennt, daß das Optimum τ^* zwischen $u = 500$ h und $o = 550$ h liegt, da hier das Vorzeichen wechselt. Mit diesem Einschließungsintervall arbeiten wir im Programm »NEWTON«. Die Zwischenergebnisse der Iteration werden über Bildschirm angezeigt, und das auf ganze Stunden abgerundete Endergebnis wird ausgedruckt:

optimales instandhaltungsintervall
tau = 527

Mit Hilfe des Programms »DAUERV« berechnen wir nun die Dauerverfügbarkeit für das Optimum $\tau^* = 527$ h und die Werte $\tau = 0{,}8\,\tau^*$, $\tau = 0{,}85\,\tau^*$,

```
dauerverfuegbarkeit   a(  422)=0.965
dauerverfuegbarkeit   a(  448)=0.965
dauerverfuegbarkeit   a(  474)=0.966

dauerverfuegbarkeit   a(  527)=0.966

dauerverfuegbarkeit   a(  580)=0.966
dauerverfuegbarkeit   a(  606)=0.966
dauerverfuegbarkeit   a(  632)=0.965
```

Bild 5. Wertetabelle der Dauerverfügbarkeit in Beispiel 1

$\tau = 0{,}9\,\tau^*$, $\tau = 1{,}1\,\tau^*$, $\tau = 1{,}15\,\tau^*$, $\tau = 1{,}2\,\tau^*$. Bild 5 zeigt die ausgedruckten Verfügbarkeitswerte. Man erkennt, daß die Verfügbarkeit tatsächlich vor und nach τ^* abnimmt, wenn auch in unmittelbarer Nähe von τ^* annähernd dieselben Verfügbarkeitswerte vorliegen (sie unterscheiden sich auf den ersten drei Stellen nach dem Komma – und diese Genauigkeit ist für praktische Zwecke ausreichend – nicht vom Wert für das Optimum τ^*).

Beispiel 2. Die Lebensdauer sei wieder weibullverteilt mit $a = 1{,}6 \cdot 10^3$ h und $b = 3$. $\mathbf{E}\,\eta_H$ und $\mathbf{E}\,\eta_P$ seien wie in Beispiel 1 gewählt.
Das Programm »TABELLE« liefert die in Bild 6 für eine Schrittweite $h = 50$ angegebenen Werte für $\varphi(\tau)$. Mit der Einschließung $u = 550$ h und $o = 600$ h beginnen wir die NEWTON-Iteration. Das Programm »NEWTON« ermittelt:

optimales instandhaltungsintervall
tau = 572

```
wertetabelle der funktion phi(tau)
nach formel (9)
phi(   50)=-.0908481
phi(  100)=-.0904203
phi(  150)=-.0892614
phi(  200)=-.0870045
phi(  250)=-.083206
phi(  300)=-.0777444
phi(  350)=-.0700217
phi(  400)=-.059765
phi(  450)=-.0466286
phi(  500)=-.0302752
phi(  550)=-.0103786
phi(  600)= .0133747
```

Bild 6. Wertetabelle für $\varphi(\tau)$ in Beispiel 2

Das optimale Erneuerungsintervall ist größer als in Beispiel 1. Dies war auch zu erwarten, da wegen $b = 3$ (bei Beibehaltung der übrigen Parameter) die mittlere Lebensdauer der Betrachtungseinheit gegenüber der in Beispiel 1 größer ist.

Für $\tau^* = 572$ h und die Werte $\tau = 0{,}7\,\tau^*$, $\tau = 0{,}8\,\tau^*$, $\tau = 0{,}9\,\tau^*$, $\tau = 1{,}1\,\tau^*$, $\tau = 1{,}2\,\tau^*$, $\tau = 1{,}3\,\tau^*$ bestimmen wir die Dauerverfügbarkeit mit dem Programm »DAUERV«. Bild 7 zeigt die

```
dauerverfuegbarkeit  a(  400)=0.983
dauerverfuegbarkeit  a(  458)=0.984
dauerverfuegbarkeit  a(  515)=0.984

dauerverfuegbarkeit  a(  572)=0.984

dauerverfuegbarkeit  a(  629)=0.984
dauerverfuegbarkeit  a(  686)=0.984
dauerverfuegbarkeit  a(  744)=0.983
```

Bild 7. Wertetabelle der Dauerverfügbarkeit in Beispiel 2

ausgedruckten Werte. Auch hier ist die Dauerverfügbarkeit für Werte vor und nach τ^* kleiner. Der Bereich um τ^* herum, in dem die Dauerverfügbarkeit annähernd gleich ist, ist breiter als im Beispiel 1.

Beispiel 3. Die Lebensdauer sei wie in Beispiel 1 gewählt. Als mittlere Erneuerungsdauern geben wir folgende Werte vor:

a) $\mathbf{E}\,\eta_H = 72$ h, $\mathbf{E}\,\eta_P = 12$ h;
b) $\mathbf{E}\,\eta_H = 72$ h, $\mathbf{E}\,\eta_P = 24$ h.

Gesucht ist das jeweilige optimale Erneuerungsintervall τ^*!

a) »TABELLE« gibt für die Tabellenschrittweite $h = 50$ die in Bild 8 gezeigte Ergebnisliste aus. Als Einschließung für die Nullstelle von $\varphi(\tau^*)$ können wir ablesen: $u = 900$ h und $o = 950$ h. Die Berechnung mit »NEWTON« ergibt:

optimales instandhaltungsintervall
tau = 921

```
wertetabelle der funktion phi(tau)
nach formel (9)
phi(   50)=-.197242
phi(  100)=-.192221
phi(  150)=-.185761
phi(  200)=-.178171
phi(  250)=-.169637
phi(  300)=-.160294
phi(  350)=-.150242
phi(  400)=-.139564
phi(  450)=-.128328
phi(  500)=-.115862
phi(  550)=-.103467
phi(  600)=-.0906396
phi(  650)=-.0774184
phi(  700)=-.0638397
phi(  750)=-.0499361
phi(  800)=-.0357377
phi(  850)=-.0212715
phi(  900)=-6.56302E-03
phi(  950)= 8.36472E-03
```

Bild 8. Wertetabelle für $\varphi(\tau)$ in Beispiel 3a)

Das optimale Instandhaltungsintervall τ^* ist größer als im Beispiel 1. Auch dieses Ergebnis ist einleuchtend, denn das Verhältnis $\mathbf{E}\,\eta_H/\mathbf{E}\,\eta_P$ ist kleiner als im Beispiel 1, d.h., eine prophylaktische Erneuerung ist nicht mehr so »günstig« wie im ersten Beispiel.

Aus gleichem Grund ist für die Teilaufgabe b) ein weiteres Anwachsen des optimalen Intervalls τ^* zu erwarten.

Die Dauerverfügbarkeit für $\tau^* = 921$ h und die Werte $\tau = 0{,}7\,\tau^*$, $\tau = 0{,}8\,\tau^*$, $\tau = 0{,}9\,\tau^*$, $\tau = 1{,}1\,\tau^*$, $\tau = 1{,}2\,\tau^*$, $\tau = 1{,}3\,\tau^*$, $\tau = 1{,}4\,\tau^*$ wird wieder nach »DAUERV« berechnet. Bild 9 enthält den Computer-Ausdruck der berechneten Werte.

Das Ergebnis bestätigt, daß bei $\tau^* = 921$ h das Optimum liegt. Auch

```
dauerverfuegbarkeit  a(  645)=0.958
dauerverfuegbarkeit  a(  737)=0.959
dauerverfuegbarkeit  a(  829)=0.959

dauerverfuegbarkeit  a(  921)=0.959

dauerverfuegbarkeit  a( 1013)=0.959
dauerverfuegbarkeit  a( 1105)=0.959
dauerverfuegbarkeit  a( 1197)=0.959
dauerverfuegbarkeit  a( 1289)=0.958
```

Bild 9. Wertetabelle der Dauerverfügbarkeit in Beispiel 3a)

hier liefern $\tau = 0{,}8\,\tau^*$ und $\tau = 1{,}3\,\tau^*$ noch Verfügbarkeitswerte, die auf den ersten drei Stellen nach dem Komma mit dem für τ^* berechneten Wert übereinstimmen.

b) Bild 10 zeigt die mit Hilfe des Programms »TABELLE« ermittelten Werte für $\varphi(\tau)$. Wir erkennen, daß die Nullstelle von $\varphi(\tau)$ zwischen $u = 1800$ h und $o = 1900$ h liegt. Mit dieser Einschließung beginnen wir die Iteration im Programm »NEWTON« und bekommen augenblicklich das Resultat:

```
wertetabelle der funktion phi(tau)
nach formel (9)
phi(  100)=-.492221
phi(  200)=-.478171
phi(  300)=-.460294
phi(  400)=-.439564
phi(  500)=-.415862
phi(  600)=-.39064
phi(  700)=-.36384
phi(  800)=-.335738
phi(  900)=-.306563
phi( 1000)=-.27651
phi( 1100)=-.245743
phi( 1200)=-.214404
phi( 1300)=-.182614
phi( 1400)=-.15048
phi( 1500)=-.11809
phi( 1600)=-.0855248
phi( 1700)=-.0597718
phi( 1800)=-.0266991
phi( 1900)= 6.67524E-03
```

Bild 10. Wertetabelle für $\varphi(\tau)$ in Beispiel 3 b)

optimales instandhaltungsintervall
tau = 1880

Zur Kontrolle bestimmen wir wieder die Dauerverfügbarkeit für $\tau^* = 1880$ h und die Werte $\tau = 0{,}5\,\tau^*$, $\tau = 0{,}8\,\tau^*$, $\tau = 2\,\tau^*$, $\tau = 3\,\tau^*$, $\tau = 4\,\tau^*$ mittels »DAUERV«. Die Werte findet man im Computer-Ausdruck in Bild 11. Man erkennt, daß in einem sehr breiten Bereich um τ^* herum keine nennenswerten Verfügbarkeitsverluste auftreten, so daß es unwesentlich ist, ob nach 1880 h oder nach 7520 h prophylaktische Instandhaltung betrieben wird. Wenn auch die Optimierung des Instandhaltungsintervalls aus diesem Grund sinnlos zu sein scheint, hat uns aber gerade die verwendete Optimierungsmethode auf diesen Sachverhalt aufmerksam gemacht. Für den Praktiker ist das Ergebnis eine wichtige Information über die Unempfindlichkeit seiner Erneuerungsstrategie gegenüber Abweichungen vom optimalen Wert τ^*.

```
dauerverfuegbarkeit  a(  940)=0.950
dauerverfuegbarkeit  a( 1504)=0.953

dauerverfuegbarkeit  a( 1880)=0.953

dauerverfuegbarkeit  a( 3760)=0.953
dauerverfuegbarkeit  a( 5640)=0.953
dauerverfuegbarkeit  a( 7520)=0.953
```

Bild 11. Wertetabelle der Dauerverfügbarkeit in Beispiel 3 b)

Beispiel 4. Die Lebensdauer sei weibullverteilt mit $a = 1{,}6 \cdot 10^4$ h und $b = 1{,}5$. $\mathbf{E}\,\eta_H$ und $\mathbf{E}\,\eta_P$ seien wie in Beispiel 1 gewählt. Gegenüber Beispiel 1 ist jetzt die mittlere Lebensdauer der

```
wertetabelle der funktion phi(tau)
nach formel (9)
phi(  200)=-.0902106
phi(  400)=-.0889348
phi(  600)=-.0872855
phi(  800)=-.0853363
phi( 1000)=-.0831305
phi( 1200)=-.0806978
phi( 1400)=-.0780604
phi( 1600)=-.0752357
phi( 1800)=-.0722382
phi( 2000)=-.06908
phi( 2200)=-.0657714
phi( 2400)=-.0623217
phi( 2600)=-.058739
phi( 2800)=-.0550305
phi( 3000)=-.0512029
phi( 3200)=-.0472622
phi( 3400)=-.0432139
phi( 3600)=-.0390632
phi( 3800)=-.0348148
phi( 4000)=-.0304731
phi( 4200)=-.0260423
phi( 4400)=-.0215263
phi( 4600)=-.0169287
phi( 4800)=-.0115972
phi( 5000)=-6.77063E-03
phi( 5200)=-1.86771E-03
phi( 5400)= 3.10868E-03
```

Bild 12. Wertetabelle für $\varphi(\tau)$ in Beispiel 4

betrachteten Anlage deutlich höher, und das optimale Erneuerungsintervall wird deshalb größer sein als im ersten Beispiel. Mit dem Programm »TABELLE« und einer Schrittweite $h = 200$ erhalten wir folgende Ergebnisliste (Bild 12). Mit der Einschließung $u = 5200$ h und $o = 5400$ h beginnen wir die Berechnung des optimalen Instandhaltungsintervalls τ^* mittels »NEWTON« und bekommen nach wenigen Iterationsschritten das Ergebnis:

optimales instandhaltungsintervall
tau = 5275

Die Berechnung der Dauerverfügbarkeit für $\tau^* = 5275$ h und die Werte $\tau = 0{,}2\,\tau^*$, $\tau = 0{,}5\,\tau^*$, $\tau = 1{,}5\,\tau^*$, $\tau = 1{,}8\,\tau^*$, $\tau = 2\,\tau^*$, $\tau = 3\,\tau^*$, $\tau = 4\,\tau^*$

```
dauerverfuegbarkeit  a( 1055)=0.993
dauerverfuegbarkeit  a( 2638)=0.996

dauerverfuegbarkeit  a( 5275)=0.996

dauerverfuegbarkeit  a( 7913)=0.996
dauerverfuegbarkeit  a( 9495)=0.996
dauerverfuegbarkeit  a(10550)=0.996
dauerverfuegbarkeit  a(15825)=0.996
dauerverfuegbarkeit  a(21100)=0.995
```

Bild 13. Wertetabelle der Dauerverfügbarkeit in Beispiel 4

mit Hilfe des Programms »DAUERV« liefert die in Bild 13 angegebenen Werte.
Es treffen hier dieselben Bemerkungen wie bei Beispiel 3 b) zu.

Literatur

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

[2] Kiesewetter, H.; Maess, G.: Elementare Methoden der numerischen Mathematik. – Berlin: Akademie-Verlag, 1974

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

[4] Preuss, W.: Formulas for the Availability of Units with Age Replacement Policy. – In: Elektronische Inf. verarb. Kybern. EIK 22 (1986) 2/3. – S. 125–129

Autoren:

Dr. rer. nat. Andreas Kossow
Prof. Dr. sc. nat. Wolfgang Preuß

Ingenieurhochschule Wismar
Abteilung Mathematik/Naturwissenschaften

(Fortsetzung von S. 43)

Das heißt, unsere Zahl 1/10 aus dem Dezimalsystem ist ein *unendlicher Dualbruch*. Und da für die Mantisse dieses Bruches nur 23 Bit zur Verfügung stehen, können von den unendlich vielen Dualstellen nur die ersten 23 geltenden Ziffern

11001100110011001100110

im Speicher untergebracht werden. Es ist also nicht die genaue Zahl 1/10 im Rechner vorhanden, mit der wir rechnen wollen, sondern eine sich davon geringfügig unterscheidende Zahl. Wenn dann aber dieser Näherungswert 10000mal addiert werden soll, dann darf man sich nicht wundern, wenn am Ende nicht das herauskommt, was man erwartet hat.

Damit muß man sich abfinden ! *Und das muß man wissen*, um bereits beim Programmieren einer Aufgabe solchen unliebsamen »Ungenauigkeiten« des Rechners von vornherein entgegenwirken zu können.

HK

Komfortable Z 1013-Tastatur

Der Mikrorechnerbausatz robotron Z 1013 wird mit einer Folientastatur ausgeliefert, die bei einer intensiveren Beschäftigung mit dem Rechner nicht befriedigen kann. Dieses betrifft sowohl die Bedienfreundlichkeit sowie die Kontaktgabe. Für die Arbeit mit einem komfortablen BASIC-Interpreter erwies sich die Tastatur wegen der Prellerscheinungen als unzuverlässig.
Der Artikel gibt Hinweise, wie die zeitweilig im Amateuerbedarfshandel angebotene Tastatur K 7659 den Erfordernissen des Mikrorechnerbausatzes angepaßt werden kann. Es wurde versucht, mit Bauelementen auszukommen, die relativ gut erhältlich sind. Durch Verwendung von Bastelbauelementen liegt der Aufwand in vertretbaren Grenzen. Es wurde eine Lösung angestrebt, die *weder Änderungen am Rechner noch an der Tastatur* verlangt, um nicht gegen die Garantiebestimmungen zu verstoßen. Unter diesen Voraussetzungen ist ein Kompromiß bezüglich der Shift-Taste notwendig, der aber beim Programmieren, was ohnehin mit Großbuchstaben erfolgt, wenig störend wirkt.
Die Tastatur wurde in zwei Baugruppen realisiert. Dadurch wird es möglich, die Tastatur sowohl am Z 1013-Bausatz zu betreiben als auch an Rechnern, die die ASCII-Zeichenfolge am Tastatureingang verlangen. Weiterhin ist hierbei auch an eine spätere Implementierung einer anderen Tastaturroutine gedacht (z. B. CP/M-Monitor).

Funktion der Z 1013-Tastatur

Um die Funktion der komfortablen Tastatur besser verstehen zu können, ist es nötig, die Funktion der Z 1013-Tastatur genauer zu betrachten. Wie in [1], [2] und [3] beschrieben, handelt es sich um eine Kontaktmatrix mit 8 Spalten und 4 Zeilen. Ein BCD-zu-1-aus-10-Dekoder MH 7442 gibt zur Abfrage der Tastatur nacheinander auf die Spalten der Matrix Low-Potential. Mit 4 Bit des Ports B der PIO wird überprüft, ob eine Taste gedrückt ist. Wenn dies nicht der Fall ist, sind die PIO-Eingänge durch Ziehwiderstände auf High-Potential gezogen. Bei gedrückter Taste gelangt das Low-Potential von der Spaltenleitung über die gedrückte Taste auf einen PIO-Eingang. Aus den Daten, die dem Tastaturspaltentreiber MH 7442 durch einen Outbefehl übergeben wurden (sie kennzeichnen die gerade Low-aktivierte Spalte) und dem auf Low-Potential liegenden PIO-Eingang berechnet der Rechner den Code der jeweils betätigten Taste. Er stellt diesen zunächst im A-Register und dann im Speicher auf Adresse 0004 bereit. Gleichzeitig wird überprüft, ob eine der Shift-Tasten

S1 – S4 gedrückt war. In diesem Fall wird zu dem aus der Tastaturspaltenadresse und PIO-Bit berechneten Wert noch ein Wert addiert. Dieser Wert ist abhängig von der gedrückten Shift-Taste. Es ist weiterhin zu beachten, daß die Shiftinformation vor der gedrückten Taste bereitgestellt wird.

0000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	F5	FF	FF	FF	FF
0020	FF	F7	FF	FE	FF	FF	FF	FF	FF	FF	F7	FF	FF	FF	FE	FF	FF
0030	F7	FF	FF	FE	FF	FF	FF	FF	F7	FF	FF	FF	FF	FF	FE	FF
0040	FF	FF	FF	FF	F7	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	F7	FF
0050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0060	FF	FF	FF	F7	FB	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	F7
0070	F7	FF	FF	FF	FF	FE	FF	FF	FF	F7	FF	FF	FE	FF	FF	FF
0080	FF	FE	FF	F7	FF	FF	FF	FF	FF	FF	FF	F7	FF	FF	FF	FE
0090	FF	F7	FF	FF	FF	FF	FE	FF	F7	FF	FF	FF	FE	FF	FF	FF
00A0	F7	FF	FF	FF	FF	FF	FF	FE	FF	FF	FF	FF	FF	FF	FF	FF
00B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0100	FF	FF	FF	FF	FF	F7	FF	FF	FF	F5	FF	FF	FF	FF	FF	FF
0110	FF	F7	FD	FF	FF	FF	FF	FF	FF	F7	FF	FD	FF	FF	FF	FF
0120	FF	F7	FF	FF	FD	FF	FF	FF	FF	F7	FF	FF	FF	FD	FF	FF
0130	FF	F7	FF	FF	FF	FF	FD	FF	FF	F7	FF	FF	FF	FF	FF	FD
0140	FB	F7	FF	FF	FF	FF	FF	FF	FF	F3	FF	FF	FF	FF	FF	FF
0150	FF	F7	FB	FF	FF	FF	FF	FF	FF	F7	FF	FB	FF	FF	FF	FF
0160	FF	F7	FF	FF	FB	FF	FF	FF	FF	F7	FF	FF	FF	FB	FF	FF
0170	FF	F7	FF	FF	FF	FF	FB	FF	FF	F7	FF	FF	FF	FF	FF	FB
0180	F5	FF	FF	FF	FF	FF	FF	FF	F7	FD	FF	FF	FF	FF	FF	FF
0190	F7	FF	FD	FF	FF	FF	FF	FF	F7	FF	FF	FD	FF	FF	FF	FF
01A0	F7	FF	FF	FF	FD	FF	FF	FF	F7	FF	FF	FF	FF	FD	FF	FF
01B0	F7	FF	FF	FF	FF	FF	FD	FF	F7	FF	FF	FF	FF	FF	FF	FD
01C0	F3	FF	FF	FF	FF	FF	FF	FF	F7	FB	FF	FF	FF	FF	FF	FF
01D0	F7	FF	FB	FF	FF	FF	FF	FF	F7	FF	FF	FB	FF	FF	FF	FF
01E0	F7	FF	FF	FF	FB	FF	FF	FF	F7	FF	FF	FF	FF	FB	FF	FF
01F0	F7	FF	FF	FF	FF	FF	FB	FF	F7	FF	FF	FF	FF	FF	FF	FB
	00	01	02	03	04	05	06	07	08	09	OA	OB	OC	OD	OE	OF

Tabelle 1

Prinzip der komfortablen Tastatur

Wie schon in der Einleitung erwähnt, bestimmen zwei Baugruppen die Schaltung zur Modifizierung der Tastatur K 7659.

- Die erste Platine wandelt den Tastaturcode in ASCII-Zeichen um.
- Die zweite Platine wandelt die ASCII-Zeichen in den für den Mikrorechnerbausatz Z 1013 erforderlichen Code um (vgl. Tabelle 1).

Am Ausgang der Platine steht außerdem ein High-aktives Statussignal bereit, das z. B. für einen CP/M-Monitor erforderlich wäre. Mittels eines Generators wird ein Zähler mit Dekoder angesteuert, der nacheinander an alle Tastaturspalten Low-Potential anlegt. An den Zeilen der Kontaktmatrix liegt ein 1-aus-8-zu-BCD-Dekoder. Aus dem Zählerstand und dem Ausgangssignal des Dekoders wird eine Adresse gebildet, die einen EPROM ansteuert. Die Speicherzelle des EPROM ist mit dem Hexadezimalcode des ASCII-Zeichens programmiert, welches auf der Tastatur gedrückt wurde.
Die höchstwertige Adresse des EPROMs wird durch die Shift-Taste angesteuert. So steht zum Beispiel das kleine »h« (Hexadezimalcode 68) auf Adresse 92 und das große »H« (Hexadezimalcode 48) auf Adresse 12.
Da die Shift-Tasten bei der Tastatur K 7659 inmitten der Matrix liegen, mußte eine Lösung gefunden werden, wie diese erkannt werden können. Die verwendete Schaltung hat zur Folge, daß eine gedrückte Shift-Taste so lange gespeichert wird, bis die Shift-Lock-Taste gedrückt wird. Nur bei Textverarbeitungsprogrammen könnte dies als Nachteil empfunden werden. Der Vorteil, daß keine Veränderungen an der Tastatur vorgenommen werden müssen, gab den Ausschlag für die Wahl dieser Lösung. Will man eine solche Lösung nicht akzeptieren, sind die Shift- und Shift-Lock-Taste aus der Matrix herauszutrennen.
Der Funktionsablauf beim Betätigen einer Taste ist wie folgt. Wird eine Taste gedrückt, so wird durch den 1-aus-8-zu-BCD-Dekoder ein Status-Signal gebildet. Es wird durch ein RC-Glied verzögert und dient zum Stoppen des Zählers. Die Verzögerung ist nötig, um Prellerscheinungen der Tastatur zu unterdrücken. Nachdem der Generator gestoppt wurde, liegt eine stabile Adresse am EPROM und das Statussignal am Ausgang der Platine. Am Ausgang des EPROMs liegt der ASCII-Code der gedrückten Taste. Bild 1 zeigt die Blockschaltung dieses Teils.

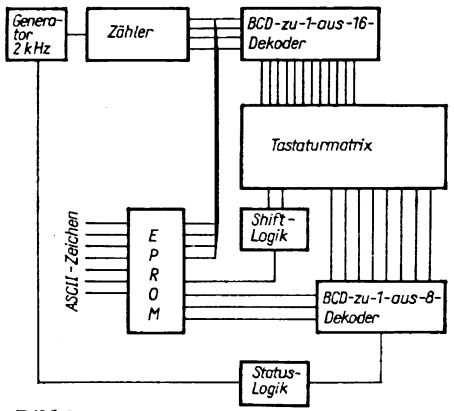


Bild 1

Der zweite Schaltungsteil übernimmt die Umcodierung des ASCII-Codes in den Z 1013-Tastaturcode. Bild 2 zeigt das Blockschaltbild.
Der 1-aus-8-zu-BCD-Dekoder stellt den Low-Teil der EPROM-Adresse bereit. Dieser ergibt sich direkt aus der vom Z 1013 angesteuerten Spalte. Daraus folgt, daß pro ASCII-Zeichen 8 Speicherplätze zu je 4 bit angesprochen werden. Durch Verwendung eines Bastel-EPROMs S 555 läßt sich dieser Aufwand, der eine einfachere Kodierung

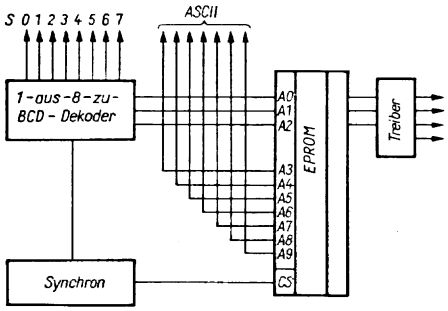


Bild 2

erlaubt, vertreten. Da viele Zellen des S 555 ohnehin mit FF programmiert werden, stören die Fehler des S 555 nicht, da meist nicht eine ganze Speicherhälfte defekt ist, sondern nur einige Zellen im unteren bzw. oberen Bereich. Der hochwertige Adreßteil wird direkt aus dem ASCII-Teil des ersten Schaltungsteils gewonnen. Die Synchronlogik stellt durch eine Verzögerung sicher, daß dem Z 1013 bei entsprechen-

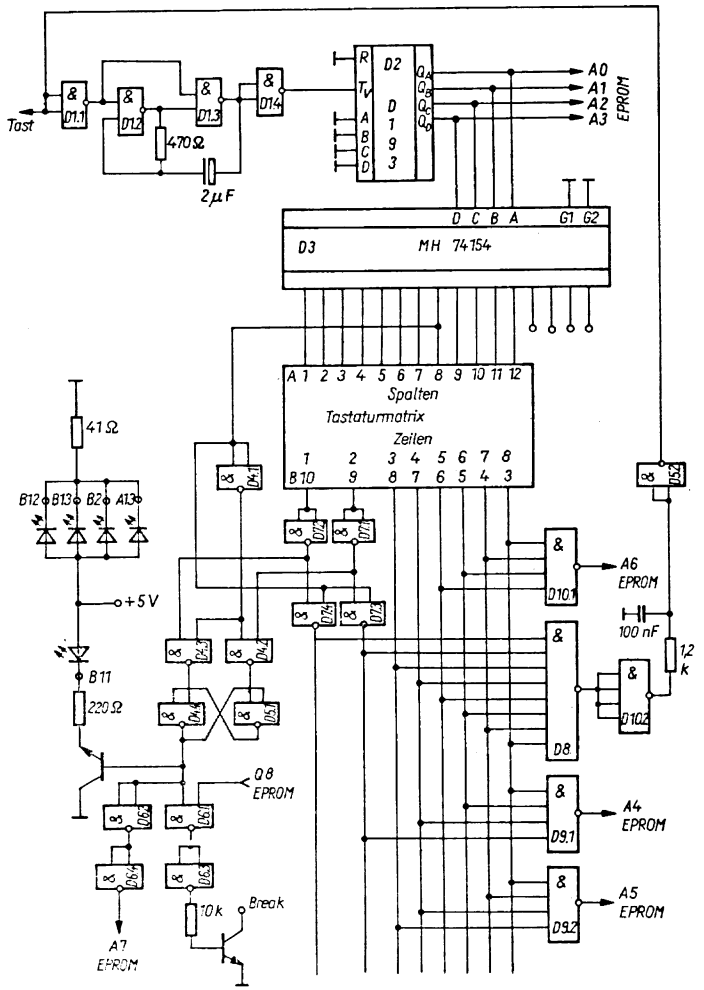


Bild 3

dem Zeichencode zuerst eine gedrückte Shift-Taste »vorgetäuscht« wird.
Bild 3 zeigt die Schaltung des ersten Tastaturteils.

Schaltungsbeschreibung

Der Generator wird durch D1 gebildet. Er schwingt mit etwa 2 kHz, wobei die genaue Frequenz unkritisch ist. Durch seinen Ausgang wird ein DL 193 (oder D 193 bzw. P 193) angesteuert. Seine Ausgänge bilden den Low-Teil der EPROM-Adresse sowie die Eingangsinformation zur Ansteuerung des BCD-zu-1-aus-16-Dekoders. Sollte der MH 74154 (SU-Äquivalent K 155 ИД 3) nicht erhältlich sein, kann dieser durch zwei DS 8205 ersetzt werden. Obwohl nur 12 Ausgänge dieses Dekoders gebraucht werden, wurde auf eine Begrenzung des Zählumfangs verzichtet. Die Ausgänge des Dekoders steuern die Spalten der Tastaturmatrix an. An den Zeilenleitungen der Tastaturmatrix liegt der aus D4 bis D6.1 gebildete 1-aus-8-zu-BCD-Dekoder. Sein Ausgangssignal bildet zum einen das Statussignal »Taste gedrückt« sowie einen Teil des High-Adreßteils des EPROMs. Für D4 bis D6 sollten unbedingt DL- bzw. PL-Schaltkreise eingesetzt werden. Ansonsten kann es Probleme mit dem sicheren Erkennen des Low-Potentials an den Zeilenausgängen geben.
D6.2 negiert das Statussignal. R2 und C2 bilden ein Verzögerungsglied von etwa 0,1 ms, um die Wirkung des Prellens der Tasten zu unterdrücken. Nach Ablauf dieser Zeit wird der Generator bei gedrückter Taste gestoppt. Die am EPROM anliegende Adresse ist jetzt stabil, und das Statussignal führt High-Potential. Am Ausgang des EPROMs liegt der ASCII-Code der gedrückten Taste.
Die Gatter D7.1 bis D7.4 sorgen dafür,

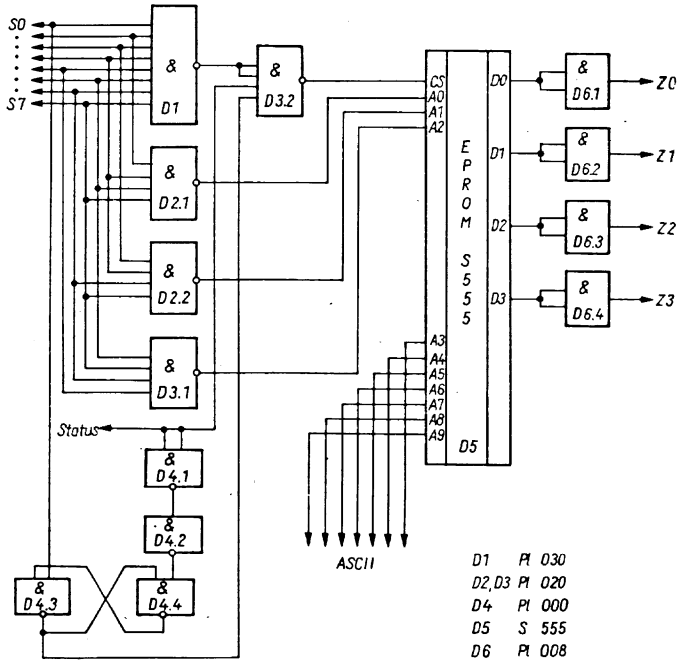


Bild 4

daß die Shift- und Shift-Lock-Taste keine Auswirkung auf das Statussignal haben. Unter der Voraussetzung, daß keine Zeichentaste gedrückt ist, wird durch Drücken der Shift-Taste der aus D4.4 und D5.1 gebildete Flip-Flop gesetzt oder rückgesetzt. Dementsprechend ist das Adreßbit A9, das über Groß- und Kleinbuchstaben bzw. Sonderzeichen entscheidet, auf High- oder Low-Potential gesetzt.

Die Schaltung des zweiten Schaltungsteils zeigt Bild 4.

Hier wird mit den IS D1 bis D3.1 wiederum ein 1-aus-8-zu-BCD-Dekoder realisiert. Seine Ausgänge steuern den niederwertigen Adreßteil des EPROMs an. Der hochwertige Adreßteil wird direkt von den sieben ASCII-Ausgängen des ersten Schaltungsteils angesteuert. Solange das Statussignal von der ersten Platine nicht geliefert wird, sind die Ausgänge des EPROMs hochohmig, da das CS-Signal auf High liegt. Außerdem sind die hochwertigen Adressen zu diesem Zeitpunkt nicht stabil. Bei anliegendem Statussignal (Taste gedrückt) wird durch die Synchronlogik dafür gesorgt, daß das CS-Signal erst freigegeben wird, wenn der Z 1013 ein Low-Potential auf Spalte Null ausgibt. Dadurch wird sichergestellt, daß eventuell betätigte Shift-Tasten rechtzeitig erkannt werden. Von Z 1013 erfolgt jetzt das Durchzählen von 8 Speicherzellen. Diese sind so programmiert, daß sie die für das jeweilige Zeichen notwendigen Low-Signale an die Zeilenleitungen des Z 1013 legen. Da nur 4 Bit gebraucht werden, kann, falls sich auf einem S 555 diese Bits als defekt herausstellen, auf die anderen Datenausgänge ausgewichen werden.

Netzteilschaltung

Zum Betrieb der Tastatur sind 3 Spannungen von -5 V, $+5$ V und $+12$ V nötig. Dabei muß unbedingt für ein gemeinsames Massepotential zwischen Rechner und Tastatur gesorgt werden. Als einzige Änderung am Rechner war es nötig, die Masseleitung und die 12-V-Leitung auf den Tastaturkamm der Rechnerleiterplatte zu führen. Dazu werden die freien Anschlüsse 12 und 11 auf der Bestückungsseite genutzt. Die Schaltung des Netzteils zeigt Bild 5. Damit sind andere Schaltungen nicht ausgeklammert.

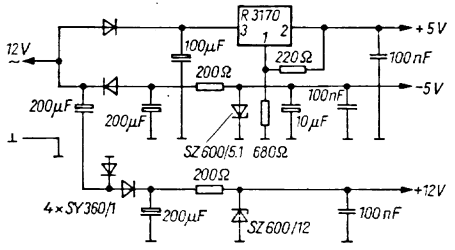


Bild 5

Programmierungshinweise

Bild 6 zeigt die *Adreßzuordnung* des ersten EPROMs zu den einzelnen Matrixplätzen der Tastatur.

Die Anordnung entspricht den Herstellerangaben in [4]. Bei gedrückter Shift-Taste wird zu jeder Adresse 80H addiert.

Bild 7 zeigt die *Datenzuordnung*, die gewählt wurde.

Graphikzeichen wurden nicht gesondert festgelegt, sondern sind durch Umschaltung der Klapptaste (+/−) auf den Buchstabentasten erreichbar. Zur Programmierung des EPROMs ist auf der jeweiligen Adresse der Inhalt der Datentabelle nach Bild 7 einzutragen.

Der zweite EPROM ist nach Tabelle 1 programmiert. Im Falle des Defekts der Datenbits 0 bis 3 kann auf die Datenbits 4 bis 7 ausgewichen werden, indem in jedem Byte High- und Low-Teil getauscht werden. Es ergibt sich eine Belegung nach Bild 8.

Adressenbelegung

70 H	71 H	72 H	73 H	74 H	75 H	76 H			79 H	7A H	7B H
60 H	61 H	62 H	63 H	64 H	65 H	66 H		68 H	69 H		
50 H	51 H	52 H	53 H	54 H	55 H	56 H		58 H	59 H	5A H	
40 H	41 H	42 H	43 H	44 H	45 H	46 H		48 H	49 H	4A H	
30 H	31 H	32 H	33 H	34 H	35 H	36 H		38 H	39 H	3A H	
20 H	21 H	22 H	23 H	24 H	25 H	26 H		28 H	29 H	2A H	
10 H	11 H	12 H	13 H	14 H	15 H	16 H		18 H	19 H		1B H
00 H	01 H	02 H	03 H	04 H	05 H	06 H		08 H	09 H	0A H	

ohne Shift

F0 H	F1 H	F2 H	F3 H	F4 H	F5 H	F6 H			F9 H	FA H	FB H
E0 H	E1 H	E2 H	E3 H	E4 H	E5 H	E6 H		E8 H	E9 H		
D0 H	D1 H	D2 H	D3 H	D4 H	D5 H	D6 H		D8 H	D9 H	DA H	
C0 H	C1 H	C2 H	C3 H	C4 H	C5 H	C6 H		C8 H	C9 H	CA H	
B0 H	B1 H	B2 H	B3 H	B4 H	B5 H	B6 H		B8 H	B9 H	BA H	BB H
A0 H	A1 H	A2 H	A3 H	A4 H	A5 H	A6 H		A8 H	A9 H	AA H	
90 H	91 H	92 H	93 H	94 H	95 H	96 H		98 H	99 H		9B H
80 H	81 H	82 H	83 H	84 H	85 H	86 H		88 H	89 H	8A H	

mit Shift

Bild 6

1 31 H	3 33 H	5 35 H	7 37 H	9 39 H	ß	akust. Signal 07 H			17 H	15 H	Zeile Lö. ab Kurser 03 H
Q 51 H	E 45 H	T 54 H	U 55 H	O 4F H	Ü	↵ 0D H		Kursor home 01 H	18 H		
A 41 H	D 44 H	G 47 H	J 4A H	L 4C H	Ä	Zeile Löschen 04 H			19 H	← 0B H	
Y 59 H	C 43 H	B 42 H	M 4D H	. 2E H	'	Space einfügen 05 H		Anf. d. f. Zeile 06 H		↓ 0A H	
2 32 H	4 34 H	6 36 H	8 38 H	Ø 30 H	+	Space 20 H		12 H		→ 09 H	normale Zeichen 10 H
W 57 H	R 52 H	Z 5A H	I 49 H	P 50 H	#	Doppel Fkt. 0F H		13 H		↑ 08 H	
S 53 H	F 46 H	H 48 H	K 4B H	Ö	µ	Zeichen löschen 04 H	Shift Shift	14 H			Inverses Zeichen 11 H
X 58 H	V 56 H	N 4E H	, 2C H	- 2D H	②	Space einfügen 05 H	Shift Look	16 H	Kursor-Positionierg 0E H		

ohne Shift

! 21 H	§ 40 H	% 25 H	/ 2F H	) 29 H	? 3F H						
q 71 H	e 65 H	t 74 H	u 75 H	o 6F H	ü	↵ 0D H		Break 80 H			
a 61 H	d 64 H	g 67 H	j 6A H	l 6C H	ä	Zeichen Löschen 04 H		Schirm Löschen 0C H		← 0B H	
y 79 H	c 63 H	b 62 H	m 6D H	: 3A H	`	Space einfügen 05 H				↓ 0A H	
" 22 H	$ 24 H	& 26 H	(28 H	= 3D H	✳ 2A H	Space 20 H				→ 09 H	normale Zeichen 10 H
w 77 H	r 72 H	z 7A H	i 69 H	p 70 H	'					← 08 H	
s 73 H	f 66 H	h 68 H	k 6B H	ö	O	Zeichen Löschen 04 H	Shift Shift				unverse Zeichen 11 H
x 78 H	v 76 H	n 6E H	; 3B H	—	③	Space einfügen 05 H	Shift Look			Schirm Löschen 02 H	

mit Shift

Bild 7

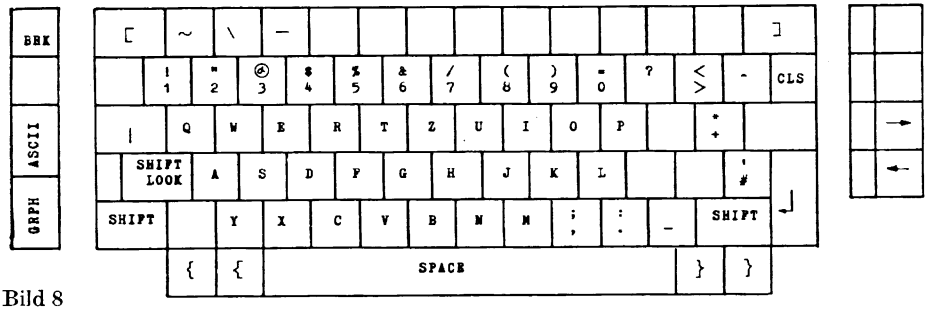


Bild 8

Abschließende Hinweise

Es hat sich als zweckmäßig erwiesen, Tastatur und Rechner trennen zu können. Auf der Leiterplatte des Rechners sind zwar Bohrungen für einen 26poligen Steckverbinder vorgesehen, jedoch lehnt der Hersteller in den Garantiebedingungen im Falle einer Nachrüstung die Instandsetzung ab. Deshalb wurde ein Teil nach Bild 9 angefertigt und mittels zweier Schrauben M 2,5 × 8 befestigt.

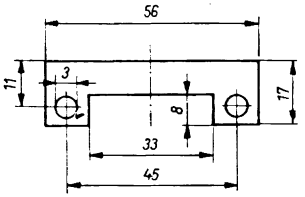


Bild 9

Als Steckverbinder wird ein auf 2 × 13 Anschlüsse gekürzter direkter Steckverbinder verwendet, der auf die Leiterplatte aufgesteckt wird. Durch Teil 9 wird ein Verrutschen unmöglich gemacht.

Die vorgestellte Lösung kann bei entsprechender Änderung der EPROM-Inhalte an jede beliebige Tastatur angepaßt werden.

Literatur

[1] Handbuch zum Z 1013-Bausatz. – Firmenschrift Robotron Elektronik Riesa

[2] Schwitalla, D.; Müller, E.: Mikrorechnerbausatz Z 1013. – In: Kleinstrechner-TIPS, H. 7. – Leipzig, 1987

[3] Heinke, T.: Tastatur K 7659 am Mikrorechner Z 1013. – In: radio fernsehen elektronik 35, H. 10. – S. 623–624

[4] Bedienungsanleitung für Tastatur K 7659. – Firmenschrift VEB robotron Elektroschaltgeräte Auerbach

Autor:

Andreas Köhler

Student an der
Hochschule für Verkehrswesen
Dresden

Hut ab vor dem Computer! (?)

Dieser Tage ging die folgende Meldung durch die Tagespresse:

> Auf 201326000 Stellen hinter dem Komma hat jetzt Prof. Yasumasa Kaneda aus Japan die Zahl Pi mit einem Supercomputer berechnet. Pi gibt das Verhältnis des Kreisumfangs zum Durchmesser an und wird von Laien mit 3,14 beziffert. Nach 5 Stunden und 57 Minuten lag das Ergebnis auf 40266 Druckseiten vor. Mit dieser Leistung übertraf der Wissenschaftler seinen im Vorjahr aufgestellten Rekord, der bei 135550000 Stellen hinter dem Komma lag.
>
> Neues Deutschland vom 18. 3. 1988

Da werden Zahlen genannt, von denen man sich wegen ihrer Größe nur sehr schwer einen richtigen Begriff machen kann. Wir wollen uns diese Zahlen einmal etwas genauer ansehen, um uns ein reales Bild von der Leistungsfähigkeit des verwendeten Supercomputers machen zu können.

Der Computer arbeitete rund 6 Stunden, um die gestellte Aufgabe zu lösen. Er muß demzufolge *in einer Sekunde* etwa 9000 Stellen der Zahl Pi *berechnet und ausgedruckt* haben. Zum Niederschreiben aller ermittelten Dezimalstellen wurden 40000 Druckseiten benötigt. Das bedeutet, daß *in einer Sekunde zwei Druckseiten* voll mit Ziffern beschrieben worden sind!

Dabei handelt es sich keinesfalls um einfache Heft- oder gewöhnliche A 4-Seiten, denn es müssen auf einer Seite rund 5000 Ziffern untergebracht worden sein. Würde man eine A 4-Seite von oben bis unten ohne Rand engzeilig mit einer Schreibmaschine beschreiben, so würde man auf dieser Seite gerade etwa 5000 Zeichen unterbringen.

Eine derartig eng beschriftete A 4-Seite wurde also *innerhalb einer halben Sekunde vollgeschrieben*, wobei zu bedenken ist, daß alle Ziffern in dieser Zeitspanne ja auch vorher noch *berechnet* werden mußten!

Das ist eine ganz erstaunliche Leistung des Computers!

Nun wird sich mancher Leser fragen, welchen Sinn es denn wohl haben mag, die Zahl Pi auf so viele Dezimalstellen zu ermitteln. Bei technischen oder naturwissenschaftlichen Aufgabenstellungen reichen doch 4 oder 5 Stellen nach dem Komma meistens aus, um die Aufgabe mit der erforderlichen Genauigkeit lösen zu können.

Gewiß, das mag stimmen, aber für den *Mathematiker* – insbesondere für den Zahlentheoretiker – geben diese Berechnungen wesentliche Aufschlüsse über die Struktur und den Aufbau der transzendenten Zahl Pi. Und schließlich stellen derartige Berechnungen ein wesentliches Hilfsmittel für den *Informatiker* dar, um die Leistungsfähigkeit des verwendeten Computers testen

und mit der anderer Computer vergleichen zu können. Unter diesem Aspekt betrachtet, sind derartige Berechnungen durchaus sinnvoll und nützlich.

Sollte jemand auf den Gedanken kommen: »Das versuche ich auch einmal mit meinem KC 87 oder einem anderen Kleincomputer«, so würde ich dringend raten, die Finger davon zu lassen. Selbst wenn man ein superschnell konvergierendes Verfahren zur Berechnung der Zahl Pi verwenden würde, würde es Monate dauern, bis das von Prof. Kaneda gefundene Ergebnis erreicht ist.

Nachdem wir nun also gesehen haben, zu welchen kaum vorstellbaren Leistungen ein Computer heutzutage bereits fähig ist, sei mir noch ein abschließender Gedanke gestattet.

Wir lesen und hören fast tagtäglich davon, daß der eine Computer dies »kann« und der andere noch viel mehr. Aber wir erfahren kaum etwas über die Menschen, die diese Computer konstruiert haben und über jene, die sich bemühen, leistungsfähige Algorithmen und Programme zu entwickeln, damit diese Computer erst einmal dazu befähigt werden, die ihnen gestellten Aufgaben zu lösen. Wieviel Mühe aufgewendet, Forschungsarbeit geleistet und Schweiß geopfert werden mußte, bis die hochintegrierten Schaltkreise entwickelt und in die Produktion übergeführt worden sind, ohne die ein Computer nicht mehr denkbar ist, das kann sich der Laie wohl noch viel weniger vorstellen, als das, was dann von diesen intelligenten Maschinen hervorgebracht wird. Und derjenige, der vielleicht schon das eine oder das andere kleine BASIC-Programm selbständig an einem Kleincomputer zum Laufen gebracht hat, der wird erahnen können, welche enormen geistigen Leistungen erforderlich sind, wenn Algorithmen und Programme für umfangreichere wissenschaftliche, technische oder ökonomische Aufgabenstellungen erarbeitet werden müssen, denn diese Programme sind ja erst die Voraussetzungen dafür, daß der Computer das tut, was wir von ihm erwarten. Sollten wir daher statt »Der Computer kann . . .« nicht lieber öfter einmal sagen »Der Mensch kann mit Hilfe des Computers . . .«?

Diese Menschen, die die Hard- und die Software für die Computer entwickeln, etwas mehr als gewöhnlich in den Blickpunkt zu rücken, das ist der eigentliche Sinn dieses Kommentars, denn

SIE HABEN ES VERDIENT!

Hans Kreul

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).

ISBN 3-343-00386-7

1. Auflage
Lizenznummer 114-210/2/88
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328
Redaktionsschluß: 15. 3. 1988
Bestellnummer: 5473717
00780

Kleinstrechner-TIPS/Hrsg. von
Hans Kreul u. a. – Leipzig: Fachbuchverl.,
H. 9. – 1. Aufl. – 1988. – 64 S.: 60 Bild.

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

Vorschau auf die nächsten Hefte

Hänsel/Wagenknecht: Der binäre Suchbaum

Lipp/Vieweg: Nullstellenbestimmung nichtlinearer Funktionen mit einer Veränderlichen

Magerl/Borchardt: TESY – eine Rechnerkopplung mit dem Kleincomputer KC 85/2

Girlich: Polynomgrafik

Görgens: Spielprogrammierung

Kühnel: FORTH – ein Softwarekonzept für Mikro- und Minicomputer

Bockhacker: Programm zum chemischen Rechnen

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

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 3-343-00386-7