Technische Universität Ilmenau
Fakultät für Mathematik
und Naturwissenschaften
Institut für Mathematik

Postfach 10 0565
98684 Ilmenau
Germany
Tel.: 03677/692652
Fax: 03677/691241
Telex: 33 84 23 tuil d.

Preprint No. M 9/95

Kondition eines Problems und angepaßte Lösungsmethoden

Werner Neundorf

April 1995

‡MSC (1991): 65-01, 65Y99, 68U99, 92J99

Die erfolgreiche Anwendung von Lösungsverfahren hängt nicht nur von ausgereiften Softwaretools ab, sondern in der ersten Phase der Lösung ist eine breite fachliche Grundbildung für Problemanalyse, für Möglichkeiten der Umformung einer Aufgabe in eine andere adäquate Darstellung sowie für die Auswahl von entsprechenden Lösungsmethoden notwendig. Dieser Kenntnisstand kann zu einer höheren Qualität an Verfahren und darauf aufbauenden Computerprogrammen führen. Andererseits verfügen auch Softwaretools über zahlreiche tutorielle Elemente oder Bausteine, und sie können in vielfältiger Weise zu tiefergreifenden überlegungen und neuen Ideen führen.
Beobachtungen und Erfahrungen zeigen, daß es für Studenten der ersten Semester in naturwissenschaftlichen und technischen Fachrichtungen nicht unbedingt notwendig ist, das große Szenarium eines noch größeren Tools nahezubringen. Vielmehr erweisen sich überschaubare Programme als wirksame und moderne didaktische Werkzeuge. Sie gestatten insbesondere die weitgehende Konfigurierbarkeit und damit kognitive Manipulationsmöglichkeiten, auch bezüglich graphischer und algebraisch-symbolischer Elemente, Darstellungsweisen und Umfang der Ergebnisse, Funktions- und Methodenauswahl, interaktive Arbeitsweise und Wiederholbarkeit.
Computer und Software können als kognitive Technologie beim Aufbau von und insbesondere auch beim Denken in und mit mentalen Modellen eine wichtige Rolle übernehmen.
Ausgewählte Demonstrationen aus den Gebieten Matrixkondition, Subtraktionskatastrophe, Termumformung, Fehlerfortpflanzung, rekursive Algorithmen, Näherungszahlen und Computereffekte untersetzen dies.

Successful application of numerical solvers depends not only on fully-developed software tools, but during the primary phase of solution there is necessary a wide basic education which includes the ability for problem modelling and analysis, possibilities of transforming the problem into another form and the choice of appropriate solution methods. This knowledge level can lead to higher quality of methods and computer programs, which are based on the algorithms. On the other hand, software tools, too, dispose of tutorial components and they can promote deeper reflections and new powerful ideas in varied ways.
Observations and experiences have shown, that it is not necessary to confront students of the first semesters studying natural sciences and technology with big shells of very extensive software tools. Instead, limited programs prove as effective and modern didactic means. They allow especially the far-reaching changes of configuration and in this way some possibilities of doing cognitive manipulations with regard to graphical and algebraic-symbolic elements, presentation and scope of results, selection of functions and methods, interactive mode of operation and repetition.
Computers and software belong to the cognitive technologies and can play an important role in the creation and development of mental models.
This is supported by selected examples and demonstrations from different areas like matrix condition, dilemma of subtraction, transformation of expression, error propagation, recursive algorithms, approximate numbers and special computer effects.

1 Einleitung

Es gibt viele Möglichkeiten, mathematische und andere Aufgabenstellungen in Ersatzprobleme umzuformen, die dann praktisch und mit Computereinsatz gelöst werden können. Dabei zeigt sich, daß die erfolgreiche Anwendung von Verfahren nicht nur von ausgereiften Softwaretools abhängt, sondern in der ersten Phase der Lösung eine breite fachliche Grundbildung für Problemanalyse, für Möglichkeiten der Umformung einer Aufgabe in eine andere adäquate Darstellung sowie für die Auswahl von entsprechenden Lösungsmethoden notwendig ist. Dieser Kenntnisstand kann zu einer höheren Qualität der Verfahren und der darauf aufbauenden Computerprogrammen führen. Andererseits verfügen auch die Softwarewerkzeuge über zahlreiche tutorielle Elemente und sie können in vielfältiger Weise zu tiefergreifenden überlegungen und neuen Ideen führen.
Beobachtungen und Erfahrungen zeigen, daß es für Studenten der ersten Semester in naturwissenschaftlichen und technischen Fachrichtungen nicht unbedingt notwendig ist, das große Szenarium eines noch größeren Tools nahezubringen. Vielmehr erweisen sich überschaubare Programme als wirksame und moderne didaktische Werkzeuge. Sie gestatten insbesondere die weitgehende Konfigurierbarkeit und damit kognitive Manipulationsmöglichkeiten, auch bezüglich graphischer und algebraisch-symbolischer Elemente, Darstellungsweisen und Umfang der Ergebnisse, Funktions- und Methodenauswahl, interaktive Arbeitsweise und Wiederholbarkeit (vgl. [4]).

Auswahl einiger Softwareaspekte als methodisch-didaktische Hilfsmittel in kognitiven Modellen (vgl. [2]) :
- Multiple-window Darstellungen, Desk-Top-Publishing Systeme,
- Veranschaulichung des Stellenwertsystems und Operationen darin,
- Computergraphik,
- Nutzung von Gleichungen und Funktionsgraphen,
- algebraische, geometrische und graphische Symbolsysteme,
- Darstellung von Formeln und Tabellen,
- Datenspeicherung und -manipulation,
- Wiederholbarkeit, Reflexion, Interaktion, Verstärkeraspekt, Reorganisationsaspekt.

Während rein rechnerisch-technische Aufgabenteile mit Hilfe des Computers sehr oft gelöst werden können, gewinnen die anspruchsvolleren Aspekte an Bedeutung. Dazu gehören natürlich die geschickte Umsetzung einer Anwendungssituation in ein passendes (mathematisches) Modell sowie die kreative Entwicklung und sachgemäße Beurteilung und Begründung mathematischer Begriffe und Verfahren.
Zu einigen ausgewählten Aufgabentypen werden im Abschnitt 3 verschiedene Methoden vorgestellt sowie Möglichkeiten, Grenzen und Effekte bei der Nutzung von Programmen (in Turbo Pascal oder Pascal-XSC) diskutiert.

2 Fehlerquellen in Lösungsmethoden

Im Rahmen der Analyse des Problems als auch der Methode zu seiner Lösung ist es notwendig, Fehler abzuschätzen und in Abhängigkeit von den verfügbaren Rechenhilfsmitteln den Rechenaufwand zu bestimmen. Dies unterstützt die sachgerechte Entscheidung über die Verfahrensauswahl.
Hier soll zunächst eine kurze Übersicht mit Beispielen über die Arten von Fehlern gegeben werden.

2.1 Verfahrensfehler

Diskretisierungsfehler (vgl. [5]).
- Ersetzen einer Funktion durch endlich viele Zahlen,
  numerische Integration auf der Basis der Funktionswerte $f_0, f_1, ..., f_n$,
  quadratische Polynominterpolation mit den Koeffizienten $a_0, a_1\ und\ a_2$.
- Approximation einer Ableitung durch einen Differenzenquotienten.

Abbruchfehler, Iterationsfehler.
- Ersetzen eines infiniten Prozesse durch eine finites Verfahren.
- Partialsumme einer unendlichen Reihe.
- Grenzwert einer Iterationsfolge näherungsweise nach endlich vielen Schritten.

2.2 Eingangsfehler

Fehler treten in den Eingangsdaten als auch bei Rundung von Eingabedaten in maschinenkonforme Zahlen auf. Beide sind vom Charakter her analog zu behandeln.

2.3 Rundungsfehler und Kondition

Rundungsfehler können bei allen Rechenoperationen entstehen, sofern mit fester endlicher Stellenzahl und irgendeiner Rundung gerechnet wird.
Das Verhalten gegenüber Rundungsfehlern (und damit auch Eingangsfehlern) ist ein wesentliches Charakteristikum eines numerischen Algorithmus. Man studiert die Fehlerfortpflanzung von Störungen auf Zwischenergebnisse und das Schlußresultat, wie sich also Fehler vergrößern.
Die Kondition eines Problems ist der im ungünstigsten Falle auftretende Vergrößerungsfaktor für den Einfluß von relativen Eingangsfehlern auf relative Resultatsfehler. Unvermeidbare Eingangsfehler durch Störungen und Rundungen bewegen sich entweder in abschätzbaren beschränkten Größenordnungen oder sie wachsen über Maßen an, je nachdem, ob das Problem gut oder schlecht konditioniert ist.
Analog ist die Kondition des mathematischen Ersatzproblems sowie des Näherungsverfahrens zu sehen (z.B. Differentialgleichung, Differenzenschema, Gleichungssystemlöser).

Man bemerke also :

- Bei gegebener Aufgabe (mit eindeutiger Lösung) können zwei verschiedene Berechnungsverfahren völlig andere Ergebnisse liefern, weil sie unterschiedliches Fehlerverhalten haben.

- Die gute Kondition einer Aufgabe sollte möglichst in das entsprechende Ersatzproblem "hinübergerettet" werden.
  Die Ersetzung einer schlecht gestellten Aufgabe durch ein "gut gestelltes" Ersatzproblem heißt Regularisierung. Dabei muß man einen zusätzlichen Verfahrensfehler in Kauf nehmen.

- Wenn für ein Ersatzproblem mit gegebener (guter) Kondition das Verfahren eine wesentliche Verstärkung der Fehlerfortpflanzung bewirkt, ist es schlecht konditioniert.

- Computerrealisierungen in Gleitkommaarithmetik (GK-Arithmetik) erzeugen notwendigerweise Rundungsfehler.

- Wichtig ist die Herausarbeitung von Begriffen und Prinzipien der Fehleranalyse und die Untersuchung des Fehlerverhaltens beginnend bei einfachen Basisalgorithmen der linearen Algebra (vgl. [6]).

- Erstellung einer Klassifikation von gut konditionierten Lösungsverfahren in den verschiedenen Teilgebieten der Numerischen Analysis.
  Die numerische Stabilität ist eine Mindestforderung an ein vernünftiges Verfahren. Liegt sie nicht vor, so können für gewisse Probleme beliebig große Genauigkeitsverluste in bezug auf das unvermeidliche Fehlerniveau auftreten. Die bestmögliche Qualität eines Algorithmus stellt die numerische Gutartigkeit dar (vgl. [6]).

3 Anwendungen und Demonstrationen

Die hier durchgeführten Überlegungen sollen nunmehr an einigen ausgewählten Beispielen demonstriert werden. Sie beinhalten Anwendungsfälle, Probleme, Begriffe, Eigenschaften, Darstellungen oder Verfahren wie

- Matrixkondition,
- korrekt gestellte Aufgaben,
- Subtraktionskatastrophe (vgl. [3]),
- Termumformungen, mathematische und numerische Äquivalenz (vgl. [3]),
- Fehlerfortpflanzung und Fehleranalyse,
- Stabilität,
- Näherungszahlen und Stellenwertsysteme,
- interne Zahlendarstellung und Zahlenformate,

- Kurvendiskussion,
- rekursive und iterative Algorithmen,
- Verarbeitung von umfangreichen Datenmengen,
- spezielle Computereffekte.

Die Beispiele weisen darauf hin, wie komplex die Verwendung des Computers als Werkzeug und Medium der Lernens und Lehrens allgemein und besonders in der Mathematikausbildung zu sehen ist.
Der Komfort des Werkzeugs Computer einschließlich der Software kann sich aber nur "entfalten" durch den an Ideen reichen und kreativen Nutzer in der Einheit mit seiner gezielten Nutzung, gründlichen Analyse und ständigen Weiterentwicklung.

3.1 Matrixkondition und Lösung eines linearen Gleichungssystems

Gesucht sei die Lösung des linearen Gleichungssystems $Ax = b$ mit $x, b \in \mathbb{R}^n$ und $A \in \mathbb{R}^{n,n}$ regulär der Form

$$\left(\begin{array}{cc|c} 0.780 & 0.563 & 0.217 \\ 0.913 & 0.659 & 0.254 \end{array} \right).$$

Um einfach festzustellen, ob ein Vektor x Lösung des Systems ist, prüft man, ob das Residuum $r = Ax - b$ einen Nullvektor liefert. Nehmen wir also zwei Kandidaten für die Lösung und zwar

$$\begin{aligned} \bar{x} &= (0.341,\ -0.087)^T, \\ \hat{x} &= (0.999,\ -1.001)^T, \end{aligned}$$

und berechnen dafür das Residuum. Es gilt entsprechend

$$\begin{aligned} \bar{r} &= (-0.000001,\ 0)^T, \\ \hat{r} &= (-0.001343,\ -0.001572)^T. \end{aligned}$$

Die "Güte" des Fehlers könnte den Betrachter dazu verleiten, $\bar{x}$ als den besseren Vorschlag zu akzeptieren. Das ist jedoch ein Trugschluß bei Kenntnis der exakten Lösung $x^* = (1, -1)^T$. Der Grund ist, daß die Matrix A eine schlechte Kondition hat. Kennzeichen dafür sind u.a.

- Die Matrix A ist fast singulär.
- Die Determinante von A ist nahe Null, denn $detA = 10^{-6}$.
- Wenn die Matrix A Elemente der Größenordnung $O(1)$ besitzt, dann hat die inverse Matrix A^{-1} betragsmäßig große Elemente, hier

$$A^{-1} = \left(\begin{array}{cc} 659000 & -563000 \\ -913000 & 780000 \end{array} \right), \quad det(A^{-1}) = 10^6.$$

- Das Spektrum der Eigenwerte von A ist sehr "breit". Es liegen Größenordnungen zwischen dem betragsmäßig kleinsten ($\neq 0$) und größten Eigenwert, denn

$$\lambda_1 = -0.00000069492670,$$
$$\lambda_2 = \quad 1.43900069492670.$$

Mit der Zeilensummennorm $\|A\|_\infty = \max\limits_{1\le i\le n} \sum\limits_{j=1}^{n} |a_{ij}|$ beträgt die Kondition

$$cond(A) = \|A\|_\infty \|A^{-1}\|_\infty \approx 2.5 \cdot 10^6.$$

Explizite oder direkte Gleichungssystemlöser müssen numerisch stabil, wenn nicht gutartig sein, um bei schlecht konditionierter Systemmatrix noch vertretbare Lösungen zu erzeugen.
Aber schon für solche Matrizen, wie die Hilbermatrix $A = (a_{ij}),\ a_{ij} = 1/(i+j-1)$, oder der Boothroyd/Dekker-Matrix (vgl. [9]) mit den Elementen

$$a_{ij} = \binom{n+i-1}{i-1}\binom{n-1}{n-j}\ \frac{n}{i+j-1}$$

und der Inversen

$$A^{-1} = (\alpha_{ij}), \quad \alpha_{ij} = (-1)^{i+j} a_{ij},$$

sind in der Gleitpunktarithmetik bei nicht allzugroßer Dimension n Grenzen gesetzt.

3.2 Iterationsverfahren für lineare Gleichungssysteme

Iterationsverfahren (IV) für das Gleichungssystem $Ax = b$ sind, wenn sie konvergieren, selbstkorrigierende Methoden. Sie können mit einem angenommenen Startvektor sofort bzw. im Nachgang an ein direktes Verfahren eingesetzt werden. In jedem Fall erhoffen wir uns eine Korrektur der Näherungslösung. Deshalb spricht man auch von Defekt- oder Residuenkorrekturverfahren oder einfach Residueniteration.
Sei A regulär mit $a_{ii} > 0$ und der Zeilensummennorm $\|A\|_\infty = 1$.
Dies ist erreichbar durch entsprechende Zeilenvertauschung von A und eine sinnvolle Skalierung (Normalisierung, Äquilibrierung) gemäß

$$\begin{aligned} A' &= DA, \quad A' = (a'_{ij}), \\ D &= diag\ (d_1, d_2, ...d_n), \\ d_i &= \frac{sign\ a_{ii}}{\sum\limits_{j=1}^{n} |a_{ij}|}, \quad i = 1, 2, ..., n. \end{aligned}$$

Damit haben wir mittels der vorgeschlagenen Zeilenskalierung auch eine erste Variante der sogenannten Vorkonditionierung der Matrix durchgeführt und erhalten

$$\sum_{j=1}^{n} |a'_{ij}| = 1 \quad und \quad a'_{ii} \ge 0\ , \quad i = 1, 2, ..., n.$$

Die Kondition von A, ausgedrückt mit der Zeilensummennorm, wird sich dabei nicht verschlechtern, oftmals jedoch deutlich besser werden (vgl. [6]). Wenn wir die Vorzeichenbedingung fallen lassen, ist die Skalierung weit weniger effizient.
Das Problem formt man nun häufig um in eine Fixpunktgleichung der Form

$$x = x - (Ax - b)$$

und zur numerischen Implementierung in die Iterationsgestalt

$$x^{(m+1)} = x^{(m)} - (Ax^{(m)} - b) = (I - A)x^{(m)} + b.$$

Dieses "naive" IV mit der Iterationsmatrix $H = I - A$ konvergiert nicht, wenn der Spektralradius $\rho(H) > 1$ ist, und kann auch leicht bei diagonal dominanter Matrix A versagen. Somit wird einfach auf die modifizierte Form der Fixpunktpunktgleichung

$$x = x - \omega(Ax - b),\ \omega > 0 \quad Parameter,$$

zurückgegriffen, die nunmehr schon eine ganze Reihe von Vorzügen besitzt.
Das entsprechende IV wird als
- Richardson-Iteration
- Relaxation
- ω-Jacobi-Verfahren
- Jacobi Overrelaxation (JOR)

bezeichnet.
Wenn man noch einen Schritt weitergeht zu

$$\begin{aligned} x &= x - W^{-1}(Ax - b), \quad W \in \mathbb{R}^{n,n}\ regulär, \\ x &= x - \omega W^{-1}(Ax - b), \end{aligned}$$

ist man bei der Klasse der semiterativen Verfahren, Newtonverfahren, Gradientenverfahren mit/ohne Vorkonditionierung unter Ausnutzung von Zerlegungen der Matrix A für die Wahl von W.
Für ein Gleichungssystem mit positiver und diagonal dominanter Matrix können wir die konvergente Richardson-Iteration

$$\begin{aligned} x^{(m+1)} &= x^{(m)} - \frac{1}{\|A\|_\infty}(Ax^{(m)} - b), \quad m = 0, 1, ..., \\ & \quad x^{(0)}\ beliebiger Startvektor, \\ & \quad \|A\|_\infty\ Zeilensummennorm, \end{aligned}$$

anwenden.
Wenn der Spektralradius der Iterationsmatrix kleiner als 1 ist, dann haben wir nicht nur die gesicherte Konvergenz des IV, sondern irgendwelche Fehler im Lösungsprozeß - auch künstlich eingebrachte Störungen - werden stets gedämpft. Darüber hinaus sind garantierte Abschätzungen der Konvergenzrate und der Fehler möglich.

Bemerkung :
Robustheit (Stabilität, Konvergenz) und Effizienz (Zeitaufwand, Speicherbedarf) sind zwei Seiten eines Gleichungssystemlösers. Gerade für großdimensionierte Systeme leben diese Eigenschaften "in unterschiedlichen Welten" und erfordern einen Kompromiß. Sogenannte Polyalgorithmen, die zwischen verschiedenen Verfahren geeignet und geschickt umschalten, sind ein möglicher Ausweg, verlangen aber auch einen reichen Erfahrungsschatz.
Bei iterativen Methoden spielt die schnelle Matrix*Vektor-Multiplikation eine große Rolle.

3.3 Fehlerfortpflanzung bei Rekursion und Stabilität

Für die Berechnung des bestimmten Integrals

$$I_n = \frac{1}{e}\int_0^1 e^x x^n dx \ , \quad n = 0, 1, ...,$$

mittels Rekursionsformeln nehmen wir zunächst eine Abschätzung von I_n durch Einschachtelung vor.

$$\begin{aligned} ex &\le e^x \le e, \quad x \in [0,1] \ , \\ \tfrac{1}{e}\int_0^1 ex x^n dx &< I_n < \tfrac{1}{e}\int_0^1 e x^n dx, \\ \tfrac{1}{n+2} &< I_n < \tfrac{1}{n+1}, \end{aligned}$$

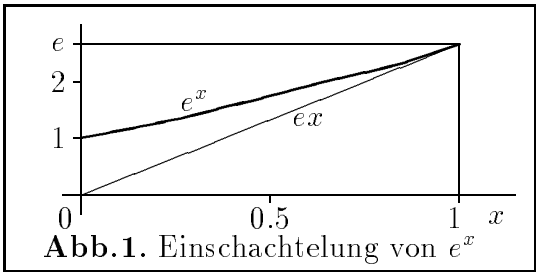

Abb.1. Einschachtelung von e^x

und somit gilt

$$1 > I_0 > \frac{1}{2} > I_1 > \frac{1}{3} > I_2 > ... \ , \quad \lim_{n\to\infty} I_n = 0 \quad und \quad \lim_{n\to\infty} nI_n = 1.$$

Durch partielle Integration erhalten wir die typische **lineare Rekursion**

$$\begin{aligned} I_n &= 1 - nI_{n-1}, \quad falls\ n > 0, \\ I_0 &= 1 - \frac{1}{e} = 0.63212\ 05588\ 28557\ 67840... \ . \end{aligned}$$

Diese kann man auch als Iteration $I_n = 1 - nI_{n-1},\ n = 1, 2, ...,\ I_0$ gegeben, interpretieren, zumal theoretisch ja auch ein Grenzwert der Folge $\{I_n\}$ existiert. Durch Entrekursivierung folgt

$$I_n = 1 - n + n(n-1) - n(n-1)(n-2) + ... + (-1)^n n(n-1)\cdot ...\cdot 4\cdot 3 - (-1)^n n!/e.$$

Bei Ausführung der Rekursion mit einem Näherungswert für I_0 bemerkt man sehr bald, daß die Einschließung $I_n \in \left(\frac{1}{n+2}, \frac{1}{n+1}\right)$ und damit die Monotonie der Folge verletzt werden.

Die Rekursionsformel ist numerisch instabil. Ein Eingangsfehler, der bei der Näherung der Eulerschen Zahl e gemacht wird, wirkt sich durch die Multiplikation mit $n!$ sehr drastisch aus.
In der GK-Arithmetik in Turbo Pascal mit den Zahlenformaten *double* (15-16 dezimale Mantissenstellen) und *extended* (19-20 Mantissenstellen, 18 Stellen werden angezeigt) sowie mit einem bestmöglichen Startwert I_0, wandert pro Rekursionsschritt die erste ungültige Mantissenstelle um durchschnittlich eine Position nach links.

n	I_n auf 20 Dezimalstellen genau	I_n mit Rekursion und GK-Format	
		double 16 Nachkommastellen	*extended* 18 Nachkommastellen
0	0.63212055882855767840	**0.6321205588285577**	**0.632120558828557678**
1	0.36787944117144232160	**0.3678794411714423**	**0.367879441171442322**
2	0.26424111765711535681	**0.2642411176571153**	**0.264241117657115357**
3	0.20727664702865392957	**0.207276647028654** 0	**0.207276647028653930**
4	0.17089341188538428171	**0.170893411885384** 0	**0.17089341188538428** 1
5	0.14553294057307859146	**0.14553294057308** 01	**0.14553294057307859** 3
6	0.12680235656152845122	**0.1268023565615** 195	**0.1268023565615284** 41
7	0.11238350406930084144	**0.1123835040693** 635	**0.112383504069300** 915
8	0.10093196744559326848	**0.100931967445** 0921	**0.10093196744559** 2678
9	0.09161229298966058367	**0.09161229299** 41707	**0.09161229298966** 5896
10	0.08387707010339416334	**0.0838770700** 582927	**0.0838770701033** 41042
11	0.07735222886266420323	**0.077352229** 3587803	**0.077352228863** 248537
12	0.07177325364802956125	**0.07177324** 76946367	**0.07177325364** 1017554
13	0.06694770257561570369	**0.0669477** 799697233	**0.066947702** 666771802
14	0.06273216394138014834	**0.06273** 10804238732	**0.06273216** 2665194768
15	0.05901754087929777487	**0.0590** 337936419019	**0.0590175** 60022078475
16	0.05571934593123560215	**0.055** 4593017295701	**0.055719** 039646744406
17	0.05277111916899476344	**0.05** 71918705973076	**0.05277** 6326005345098
18	0.05011985495809425803	**-0.0** 294536707515363	**0.050** 026131903788240
19	0.04772275579620909737	1.5596197442791890	**0.04** 9503493828023437
20	0.04554488407581805262	-30.1923948855837807	**0.0** 09930123439531258

Tab.1. Rekursion $I_n = 1 - nI_{n-1}, \quad n = 1, 2, ..., 20, \quad I_0$ maximal genau.

Rechnet man die Rekursionsformel mit Intervallarithmetik und Pascal-XSC mit dem GK-Format *real* (entspricht dem Format *double* in TP), so ist die instabile Fehlerfortpflanzung ähnlich.
Die relative Länge des Startintervalls [I01 , I02], ε = (I02-I01)/I01, vergrößert sich ständig für die nachfolgenden Intervalle $[\mathrm{I01}\,,\mathrm{I02}]^{(n)}$ und die Intervallgrenzen bekommen unbrauchbare Werte.
Nur wenn I01=I02 ist, dann gilt $(\mathrm{I02}^{(n)}\text{-}\mathrm{I01}^{(n)})/\mathrm{I01}^{(n)} = O(10^{-15})$.

Startintervall I_0 : [0.632120 , 0.632121]		
n	[inf (I_n) , sup (I_n)]	
0	[	6.32119E-001, 6.32122E-001]
1	[	3.67878E-001, 3.67881E-001]
2	[	2.64239E-001, 2.64243E-001]
3	[	2.0727E-001, 2.0729E-001]
4	[	1.708E-001, 1.710E-001]
5	[	1.454E-001, 1.457E-001]
6	[	1.26E-001, 1.28E-001]
7	[	1.1E-001, 1.2E-001]
8	[	7.8E-002, 1.2E-001]
9	[	-6.9E-002, 3.0E-001]
10	[	-2.0E+000, 1.7E+000]
11	[	-1.8E+001, 2.3E+001]
12	[	-2.7E+002, 2.2E+002]
13	[	-2.8E+003, 3.5E+003]
14	[	-4.9E+004, 3.9E+004]
15	[	-5.8E+005, 7.4E+005]

Startintervall I_0 : [0.63212 , 0.63212]		
n	[inf (I_n) , sup (I_n)]	
0	[	6.321199999999999E-001, 6.321200000000001E-001]
1	[	3.678799999999999E-001, 3.678800000000001E-001]
2	[	2.642399999999998E-001, 2.642400000000001E-001]
3	[	2.07279999999999E-001, 2.07280000000001E-001]
4	[	1.7087999999999E-001, 1.7088000000001E-001]
5	[	1.4559999999999E-001, 1.4560000000002E-001]
6	[	1.263999999999E-001, 1.264000000001E-001]
7	[	1.15199999999E-001, 1.15200000001E-001]
8	[	7.8399999996E-002, 7.8400000001E-002]
9	[	2.943999999E-001, 2.944000001E-001]
10	[	-1.944000001E+000, -1.943999999E+000]
11	[	2.238399999E+001, 2.238400001E+001]
12	[	-2.676080001E+002, -2.676007999E+002]
13	[	3.479903999E+003, 3.479904001E+003]
14	[	-4.871765601E+004, -4.871765599E+004]
15	[	7.307658399E+005, 7.307658402E+005]

Tab.2. Rekursion $I_n = 1 - n I_{n-1}$, $n = 1, 2, ..., 15$, I_0 gegeben, mit Pascal-XSC und Intervallarithmetik.

Bemerkung :
Es liegt eine gewisse Verwandtschaft der Rekursion zur Fixpunktiteration auf der Basis von $x = g(x),\ g(x) = 1 - nx$ *abhängig vom Iterationsindex*, $n \geq 1$, vor. Die Instabilität der Rekursion korrespondiert sozusagen mit der Divergenz des allgemeinen Iterationsverfahrens $I_n = g(I_{n-1}),\ g'(I) = -n \leq -1$. Der Fixpunkt wandert mit dem Index von 1/2 nach 0, kann aber wegen wachsenden $|g'| \geq 1$ nicht angenähert werden.

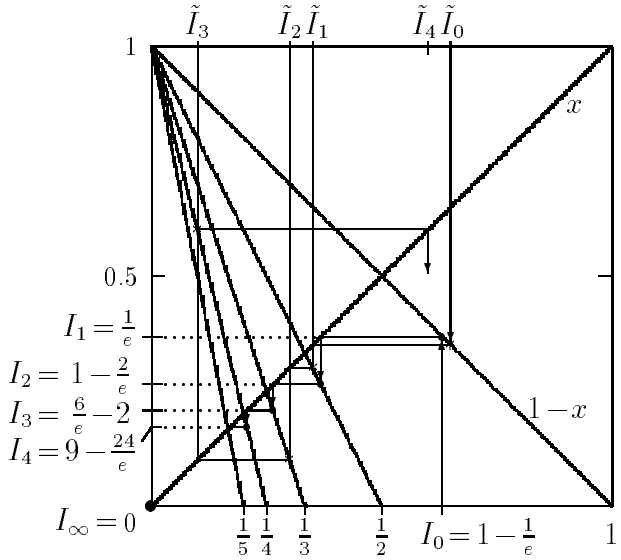

Abb.2.
Interpretation der Rekursion $I_n = 1 - nI_{n-1},\ n = 1, 2, ...,\ I_0 = 1 - \frac{1}{e}$, als "wandernde" Fixpunktiteration, Divergenz bei Startwert $\tilde{I}_0 \neq I_0$ sichtbar.

So problematisch und eigentlich nicht anwendbar die Rekursionsformel ist, die Rückwärtsrekursion erweist sich als das positive Gegenteil.
Mit $I_{n-1} = (1 - I_n)/n,\ n = m, m-1, ...1,$ könnte man bei einem guten Startwert I_m die vorherliegenden Werte bestimmen.

n	$\tilde{I}_n$ mit Rückwärtsrekursion	
	$\tilde{I}_{10} = 1/12$	$\tilde{I}_{10} = 1/11$
10	**0.083** 3333333333333	**0.0** 909090909090909
9	**0.0916** 666666666667	**0.09** 09090909090909
8	**0.1009** 259259259259	**0.101** 0101010101010
7	**0.11238** 42592592593	**0.1123** 737373737374
6	**0.126802** 2486772487	**0.12680** 37518037518
5	**0.1455329** 585537919	**0.145532** 7080327080
4	**0.17089340** 82892416	**0.1708934** 583934584
3	**0.207276647** 9276896	**0.2072766** 354016354
2	**0.264241117** 3574368	**0.26424112** 15327882
1	**0.367879441** 3212816	**0.36787943** 92336059
0	**0.632120558** 6787184	**0.63212056** 07663941

Tab.3.
Rückwärtsrekursion $\tilde{I}_{n-1} = (1 - \tilde{I}_n)/n$, $n = 10, 9, ..., 1$, $\tilde{I}_{10}$ gegeben, Turbo Pascal mit GK-Format *double*.

Was passiert aber, wenn I_m fehlerbehaftet ist? Wir rechnen also mit $\tilde{I}_m = I_m - \delta_m$.

$$\begin{aligned}
\tilde{I}_{m-1} &= \frac{1-\tilde{I}_m}{m} = \frac{1-(I_m-\delta_m)}{m} = \frac{1-I_m}{m} + \frac{\delta_m}{m} = I_{m-1} - \delta_{m-1}, \\
\delta_{m-1} &= -\frac{\delta_m}{m}, \\
\tilde{I}_{m-2} &= \frac{1-\tilde{I}_{m-1}}{m-1} = \frac{1-(I_{m-1}-\delta_{m-1})}{m-1} = \frac{1-I_{m-1}}{m-1} + \frac{\delta_{m-1}}{m-1} = I_{m-2} - \delta_{m-2}, \\
\delta_{m-2} &= -\frac{\delta_{m-1}}{m-1} = \frac{\delta_m}{(m-1)m} \quad usw.
\end{aligned}$$

Ein Anfangsfehler δ_m wird bei hinreichend großem m nach wenigen Schritten sehr klein werden. Die Anzahl der Schritte der Vorwärtsrekursion, die alle gültigen Mantissenstellen auslöscht, charakterisiert den Wert m, mit dem die Rückwärtsrekursion bei $\tilde{I}_m \approx 0$ den (genauen) Wert I_0 liefert.

n	$\tilde{I}_n$ mit Rückwärtsrekursion und GK-Format	
	double 16 Nachkommastellen	*extended* 18 Nachkommastellen
20		**0.** 000000000000000000
19		**0.05** 0000000000000000
18		**0.050** 000000000000000
17	**0.** 0000000000000000	**0.05277** 7777777777778
16	**0.05** 88235294117647	**0.055718** 954248366013
15	**0.058** 8235294117647	**0.0590175** 65359477124
14	**0.0627** 450980392157	**0.06273216** 2309368192
13	**0.06694** 67787114846	**0.066947702** 692187986
12	**0.071773** 3247145012	**0.0717732536** 39062463
11	**0.07735222** 29404582	**0.07735222886** 3411462
10	**0.083877070** 6417765	**0.083877070103** 326231
9	**0.0916122929** 358223	**0.0916122929896** 67377
8	**0.1009319674** 515753	**0.10093196744559** 2514
7	**0.11238350406** 85531	**0.112383504069300** 936
6	**0.126802356561** 6353	**0.1268023565615284** 38
5	**0.1455329405730** 608	**0.14553294057307859** 4
4	**0.17089341188538** 78	**0.17089341188538428** 1
3	**0.20727664702865** 30	**0.207276647028653930**
2	**0.264241117657115** 7	**0.264241117657115357**
1	**0.367879441171442** 2	**0.367879441171442322**
0	**0.632120558828557** 9	**0.632120558828557678**

Tab.4. Rückwärtsrekursion $\tilde{I}_{n-1} = (1-\tilde{I}_n)/n,\ n = m(-1)1,\ \tilde{I}_m = 0,$
$m = 17, 20,$ Turbo Pascal.

Bei Rückwärtsrekursion wird mit wachsendem Index m zuerst $\tilde{I}_0$ immer genauer bis die Genauigkeit eine immer größere Anzahl von anfänglichen Werten erfaßt.

m	$\tilde{I}_0$	$\delta_0 = I_0 - \tilde{I}_0$
2	0.5	0.13
4	0.625	0.0071
6	0.63194	0.00017
8	0.63211 805	0.00000 25
10	0.63212 05357	0.00000 0023
12	0.63212 05586 78718	0.00000 00001 5
14	0.63212 05586 27838	0.00000 00000 0072
16	0.63212 05586 28555	0.00000 00000 0000 3
17	0.63212 05586 28558	0

Tab.5. Verbesserung der Genauigkeit bei Rückwärtsrekursion
$\tilde{I}_{n-1} = (1 - \tilde{I}_n)/n, \quad n = m, m-1, ..., 1, \quad \tilde{I}_m = 0,$
mit wachsendem Startindex m,
Turbo Pascal mit GK-Format *double*.

m	k	$\delta_k = I_k - \tilde{I}_k$
20	4	1E-18
21	6	-1E-18
22	7	-1E-18
23	9	1E-18
24	11	-3E-18
25	12	-1E-18
26	14	7E-18
27	14	-1E-18
28	16	2E-18
29	17	1E-18
30	17	-1E-18
31	20	1E-18
32	20	1E-18
33	21	

Tab.6. Vergrößerung der Mengen
der genauen Werte $\tilde{I}_0, \tilde{I}_1, ..., \tilde{I}_{k-1}$
bei Rückwärtsrekursion mit
wachsendem m,
$\tilde{I}_m = 0$,
Turbo Pascal mit GK-Format
extended.

Bezüglich des stark gedämpften Fehlers betrachten wir noch einige Abschätzungen. Für die Werte $\{I_m, \tilde{I}_m\}$ gilt

$$\frac{1}{m+2} < I_m < \frac{1}{m+1} < I_{m-1} < \frac{1}{m} < I_{m-2} < \frac{1}{m-1} < ...$$

und mit $\tilde{I}_m = 0$

$$0 = \tilde{I}_m < \frac{1}{m+1} < \tilde{I}_{m-1} = \frac{1}{m} = \tilde{I}_{m-2} < \frac{1}{m-1} < ...$$

und weiter

$$\begin{aligned} |I_m - \tilde{I}_m| &< \frac{1}{m+1} = O(1/m), \\ |I_{m-1} - \tilde{I}_{m-1}| &= \frac{1}{m} - \frac{1}{m+1} = \frac{1}{m(m+1)} = O(1/m^2), \\ |I_{m-2} - \tilde{I}_{m-2}| &= \frac{1}{m-1} - \frac{1}{m} = \frac{1}{(m-1)m}. \end{aligned}$$

Die wachsende Größenordnung der nächsten Verbesserung ist in der Abschätzung nicht zu erkennen. Einen Hinweis darauf erhält man jedoch schon, wenn man z.B. mit der Annahme $I_m = 1/(m+\frac{3}{2})$ rechnet. Daraus ergeben sich

$$I_{m-1} = \frac{2m+1}{m(2m+3)}, \quad I_{m-2} = \frac{2m^2+m-1}{(m-1)m(2m+3)},$$

und somit

$$|I_{m-2} - \tilde{I}_{m-2}| = \frac{2m^2+m-1}{(m-1)m(2m+3)} - \frac{1}{m} = \frac{2}{(m-1)m(2m+3)} = O(1/m^3).$$

Wir fassen die wichtigsten Schlußfolgerungen der Rückwärtsrekursion zusammen:

- Je größer m ist oder je genauer $\tilde{I}_m$, desto kleiner $|I_0 - \tilde{I}_0|$ bzw. desto mehr Anfangselemente $\tilde{I}_0, \tilde{I}_1, ..., \tilde{I}_{k-1}$ sind genau.
- Je größer m ist, desto näher "rückt" der erste genaue Werte $\tilde{I}_{k-1}$.
- Bei $m = 10^5$ und $\tilde{I}_m = 0$ gilt für den relativen Fehler $\varepsilon_m = |(I_m - \tilde{I}_m)/I_m|$

  $$\begin{aligned} \varepsilon_m &= 1, \\ \varepsilon_{m-1} &= O(1/m) = O(10^{-5}), \\ \varepsilon_{m-2} &= O(1/m^2) = O(10^{-10}), \\ \varepsilon_{m-3} &= O(10^{-15}), \end{aligned}$$

  so daß man bei 15-16 Mantissenstellen nach 3 Rekursionsschritten den "genauen Bereich" erreicht. Mittels dieser Rückwärtsrekursion kann man mit geringem Aufwand für jedes n das Integral I_n mit gegebener Maschinengenauigkeit berechnen.

- In der Intervallarithmetik ist die Stabilität noch ausgeprägter.
  Vergleicht man die Rückwärtsrekursion von Tab.4 ($\tilde{I}_{20} = 0$, *extended*) mit den (genauen) Werten $\tilde{I}_0, \tilde{I}_1, \tilde{I}_2, \tilde{I}_3, (\tilde{I}_4)$ und die der nachfolgenden Tabelle 7 mit dem Startintervall $\tilde{I}_{20} = [0, 0.09] \approx [0, 2I_{20}]$, so bemerkt man ein weiteres Anwachsen der Zahl der genauen Werte (hier $\tilde{I}_0, ..., \tilde{I}_6$).

n	Startintervall $\tilde{I}_{20}$	
	[0,	0.09]
20	[0.0E-000,	9.1E-002]
19	[4.5E-002,	5.1E-002]
18	[4.99E-002,	5.03E-002]
17	[5.276E-002,	5.278E-002]
16	[5.5718E-002,	5.5720E-002]
15	[5.901751E-002,	5.901757E-002]
14	[6.2732162E-002,	6.2732166E-002]
13	[6.69477024E-002,	6.69477027E-002]
12	[7.177325363E-002,	7.177325366E-002]
11	[7.7352228861E-002,	7.7352228864E-002]
10	[8.38770701033E-002,	8.38770701035E-002]
9	[9.161229298965E-002,	9.161229298967E-002]
8	[1.0093196744559E-001,	1.0093196744560E-001]
7	[1.12383504069300E-001,	1.12383504069301E-001]
6	[1.268023565615284E-001,	1.268023565615285E-001]
5	[1.455329405730785E-001,	1.455329405730787E-001]
4	[1.708934118853842E-001,	1.708934118853844E-001]
3	[2.072766470286539E-001,	2.072766470286540E-001]
2	[2.642411176571153E-001,	2.642411176571154E-001]
1	[3.678794411714422E-001,	3.678794411714424E-001]
0	[6.321205588285576E-001,	6.321205588285578E-001]

Tab.7. Rückwärtsrekursion
$\tilde{I}_{n-1} = (1 - \tilde{I}_n)/n, \quad n = 20, 19, ..., 1, \quad \tilde{I}_{20} = [0, 0.9]$,
mit Pascal-XSC und Intervallarithmetik.

3.4 Subtraktionskatastrophe bei Bestimmung der Zahl π nach Archimedes

Durch iterative Bestimmung der Umfänge von ein- und umbeschriebenen regelmäßigen Vielecken kann man die Kreiszahl π einschachteln.
Die Seitenlänge s_{2n} des einbeschriebenen 2n-Ecks im Kreis mit dem Radius $r = 1$ kann man mittels Rekursion aus der Seitenlänge s_n des n-Ecks ermitteln. Es gilt

$$s_{2n} = \sqrt{2 - \sqrt{4 - s_n^2}} = \frac{s_n}{\sqrt{2 + \sqrt{4 - s_n^2}}}, \; n = 6, 12, ..., \; s_6 = 1,$$

$$Umfang \;\; des \;\; 2n\text{–}Ecks \quad U_{2n} = 2ns_{2n},$$
$$Kreiszahl \quad \pi \approx \tfrac{1}{2}\, U_{2n} = ns_{2n}.$$

Die algebraisch gleichwertigen Formeln für s_{2n} verhalten sich numerisch sehr verschieden. Wegen $\lim_{n\to\infty} s_n = 0$ führt die erstere zu einer Subtraktion von annähernd gleichen Zahlen im Computer (mit endlicher Mantissenlänge des GK-Formats) und damit zu unerwünschten und katastrophalen Stellenauslöschung. Nicht nur in diesem Fall läßt sich die Subtraktionskatastrophe durch eine geeignete Termumformung umgehen.

j	Variante 1		Variante 2	
	$s_{2n}=\sqrt{2-\sqrt{4-s_n^2}}$ $n = 6 * 2^j$	$\pi^{(1)}$	$s_{2n}=\frac{1}{\sqrt{2+\sqrt{4-s_n^2}}}$	$\pi^{(2)}$
0	1.0000000000E-00	3.0000000000	1.0000000000E-00	3.0000000000
1	5.1763809020E-01	3.1058285412	5.1763809020E-01	3.1058285412
2	2.6105238444E-01	3.1326286132	2.6105238444E-01	3.1326286133
3	1.3080625846E-01	3.1393502029	1.3080625846E-01	3.1393502030
4	6.5438165642E-02	3.1410319508	6.5438165643E-02	3.1410319509
5	3.2723463242E-02	3.1414524712	3.2723463253E-02	3.1414524723
6	1.6362279155E-02	3.1415575977	1.6362279208E-02	3.1415576079
7	8.1812079465E-03	3.1415838514	8.1812080524E-03	3.1415838921
8	4.0906125332E-03	3.1415904255	4.0906125823E-03	3.1415904632
9	2.0453070448E-03	3.1415916208	2.0453073607E-03	3.1415921060
10	1.0226528554E-03	3.1415895717	1.0226538140E-03	3.1415925167
11	5.1132598302E-04	3.1415868397	5.1132692372E-04	3.1415926194
12	2.5566299151E-04	3.1415868397	2.5566346395E-04	3.1415926450
13	1.2782793831E-04	3.1414994119	1.2783173224E-04	3.1415926514
14	6.3903295775E-05	3.1409747940	6.3915866151E-05	3.1415926530
15	3.1944530915E-05	3.1402751671	3.1957933079E-05	3.1415926534
16	1.5958023577E-05	3.1374750995	1.5978966540E-05	3.1415926535
17	7.9790117887E-06	3.1374750995	7.9894832701E-06	3.1415926535
18	3.8146972656E-06	3.0000000000	3.9947416351E-06	3.1415926536
19	1.9073486328E-06	3.0000000000	1.9973708175E-06	3.1415926536
20	0.0000000000E-00	0.0000000000	9.9868540877E-07	3.1415926536
21	0.0000000000E-00	0.0000000000	4.9934270438E-07	3.1415926536

Tab.8. π nach Archimedes mit einbeschriebenen Vielecken,
Turbo Pascal mit GK-Format *real*
(6 Byte, 11-12 gültige Dezimalen der Mantisse).

3.5 Termumformung und mathematische Äquivalenz von Formeln

Empfiehlt man dem Betrachter für die Berechnung eines Wertes die mathematisch äquivalenten Formeln

$$\textbf{(1)} \qquad \frac{1}{(3+\sqrt{10})^4} \;=\; 6.93481\ 60951\ 23042\ 52272\text{E-}4,$$

$$\textbf{(2)} \qquad (3-\sqrt{10})^4,$$

$$\textbf{(3)} \qquad (19-6\sqrt{10})^2,$$

$$\textbf{(4)} \qquad 721-228\sqrt{10},$$

$$\textbf{(5)} \qquad \frac{1}{(19+6\sqrt{10})^2},$$

$$\textbf{(6)} \qquad \frac{1}{721+228\sqrt{10}},$$

so wird er auf den ersten Blick vom ästhetischen Standpunkt aus sich für die Verwendung der Formel (3) entscheiden. Daß er damit fast den numerisch schlechtesten Ausdruck gewählt hat, wird verursacht durch das Subtraktionsdilemma (siehe Abschnitt 3.4).
Im folgenden Turbo-Pascal-Programm mit wahlweiser "künstlicher" Festlegung der Mantissenlänge auf 2...5 Byte kann man die "Güte" der Formeln schnell herausfinden.

```
program Aequivalenz;
{von Quadratwurzel-Bruchtermen bei kuenstlich gekuerzter Mantisse}

uses crt;

type float = real;

var  arr   : array[0..5] of byte;
     x     : float absolute arr;
     n,m   : integer;

{kuenstliches Abschneiden der Mantisse auf n-1 Bytes}
procedure clear;
var i:integer;
begin
  for i:=1 to 6-n do arr[i]:=0
end;

begin
  clrscr;
  writeln('Aequivalenz von Termen');
  writeln;
```

```
  writeln(
   'Byteanzahl der GKZ/Laenge der Mantisse/gueltige Dez.stellen');
  writeln('  n = 3 / 2 / 4                ');
  writeln('      4 / 3 / 7   Format single');
  writeln('      5 / 4 / 9                ');
  writeln('      6 / 5 / 11  Format real  ');
  writeln;
  write('  n = ');
  readln(n);
  writeln;
  case n of  { m --> Format m+6,  6 = Vorzeichen & Punkt & E+00 }
    3 : m:=4;
    4 : m:=7;
    5 : m:=9;
    6 : m:=11
  end;
  {Formel (1), Formeln (2),...,(6) analog}
  x:=10;      clear;
  x:=sqrt(x); clear;
  x:=3+x;     clear;
  x:=sqr(x);  clear;
  x:=sqr(x);  clear;
  x:=1/x;     clear;
  write(' (1)  ',x:m+6);
  x:=1/sqr(sqr(3+sqrt(10)));
  writeln('    real  : ',x);
  writeln;
  writeln(' Guete : (6) >= (5) >= (1) > (2) > (3) >> (4)');
  writeln;
  writeln('     exakt :  6.934816095123042522 72E-04');
  readln
end.
```

Hier sind die Ergebnisse zur numerischen Äquivalenz von Termen mit obigen Turbo-Pascal-Programm im Vergleich der GK-Formate *single* und *real*.

```
Aequivalenz von Termen
Byteanzahl der GKZ/Laenge der Mantisse/gueltige Dez.stellen
  n = 3 / 2 / 4
      4 / 3 / 7   Format single
      5 / 4 / 9
      6 / 5 / 11  Format real
  n = 4
 (1)   6.934818E-04    real  :  6.9348160951E-04
 (2)   6.934781E-04    real  :  6.9348160952E-04
 (3)   6.936202E-04    real  :  6.9348160973E-04
 (4)   7.934570E-04    real  :  6.9348141551E-04
 (5)   6.934817E-04    real  :  6.9348160951E-04
 (6)   6.934816E-04    real  :  6.9348160951E-04
 Guete : (6) >= (5) >= (1) > (2) > (3) >> (4)
      exakt :  6.9348160951230425 22E-04
```

Mit wachsender Mantisse können auch mit den problematischen Formeln (2), (3), (4) bessere Ergebnisse erzielt werden, wie bei Turbo Pascal mit GK-Format *extended*.

```
Aequivalenz von Termen in TP7.0/BP7.0

Format extended = 10Byte => 19-20stellige Mantisse

 (1) :  6.93481609512304252E-0004
 (2) :  6.93481609512304252E-0004
 (3) :  6.93481609512304262E-0004
 (4) :  6.93481609512325292E-0004
 (5) :  6.93481609512304252E-0004
 (6) :  6.93481609512304252E-0004

 Guete : (6) >= (5) >= (1) > (2) > (3) >> (4)

      exakt :  6.934816095123042522 72E-04
```

Natürlich kann man in Turbo Pascal analoge Rechnungen mit den GK-Formaten *single*, *real* oder *double* durchführen. Dabei ist jedoch folgendes zu beachten. Arithmetische Ausdrücke werden bei ihrer Berechnung mit der Stellenzahl der *extended*-Mantisse behandelt (Rechnung jedoch nicht unbedingt in dieser Gemauigkeit) und erst durch Zuordnung eines Wertes zu einer Variablen kommt es zur Stellenabschneidung.
Bei komplizierten Formeln müßte man also auch "zwischendurch" immer wieder die Mantisse kürzen (siehe TP-Programm).
Es ist also ein Unterschied zwischen den ausgegebenen Größen bei folgender Anweisungsfolge

```
...
var x:single;
...
x:=721-228*sqrt(10);
writeln(x);                    { --> 6.93481590.........E-004 }

writeln(721-228*sqrt(10):26);  { --> 6.93481609512325292E-004 }
...
```

Zum Vergleich sei noch eine Rechnung mit Pascal-XSC mit dem GK-Format *real* angegeben.

```
Aequivalenz von Termen in Pascal-XSC
Format real = 8Byte => 15-16stellige Mantisse

 (1) :  6.934816095123040E-004
 (2) :  6.934816095123075E-004
 (3) :  6.934816095122907E-004
 (4) :  6.934816094599228E-004
 (5) :  6.934816095123043E-004
 (6) :  6.934816095123043E-004

 Guete : (6) >= (5) >= (1) > (2) > (3) >> (4)

      exakt :  6.9348160951230425 2272E-04
```

3.6 Kurvendiskussion, Funktionswertberechnung und Approximation von Ableitungen

Betrachten wir die rationale Funktion

$$f(x) = \frac{4970x - 4923}{4970x^2 - 9799x + 4830}$$

mit der Nullstelle $x_1 = 4923/4970 = 0.9905...$ und den nahe beieinanderliegenden Polstellen $p_1 = 0.9857142823...$, $p_2 = 0.9859154963...$.
Zunächst untersuchen wir den Funktionsgraphen, den wir z.B. mit dem Computeralgebrasystem ***Mathematica*** oder dem interaktiven Plotterprogramm **GNUPLOT** erzeugen können. Ein grober Graph mittels

```
        Plot [f[x],{x,0,1.2}]
bzw.    plot [0:1.2] [-150:75] f(x)
```

läßt zwar die Nullstelle gut erkennen, aber die Polstellen fehlen.
Geht man mittels "Fenstertechnik" langsam in den interessanten Bereich, z.B. mit

```
        Plot [f[x],{x,0.985,0.987}]
bzw.    plot [0.9854:0.9864] [-3e6:3e6] f(x)
```

so werden die Sprungstellen (bei eingezeichneten Asymptoten) sichtbar. Aber über diese Stellen mit ihren beidseitig betragsgroßen Werten wird interpoliert. Die Nullstelle ist nun rechts außerhalb des Fensters. Im letzten Schritt

```
        plot [0.98565:0.98600] [-3e6:3e6] f(x)
```

sind die Asymptoten ebenfalls zusätzlich eingezeichnet worden.

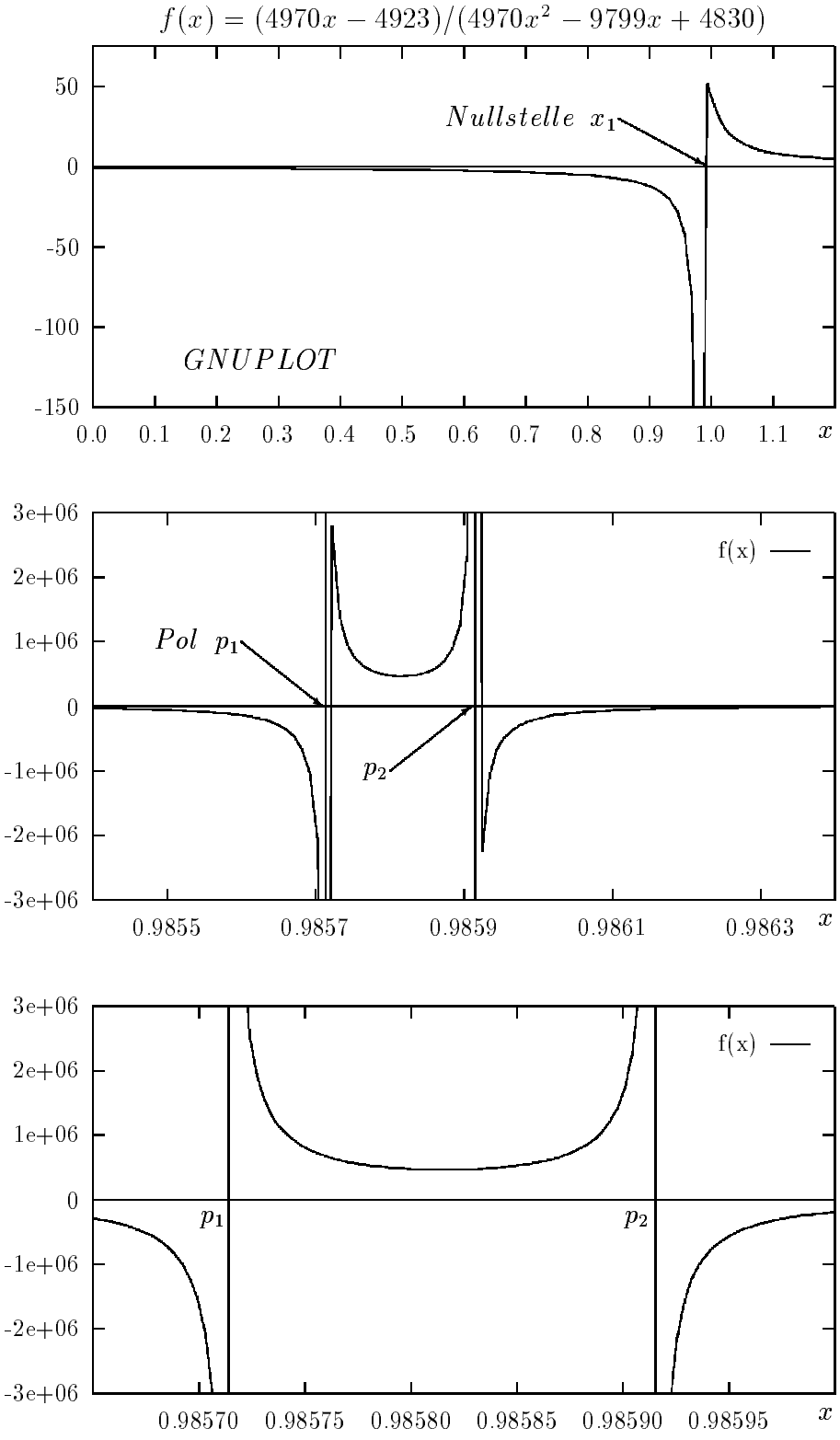

Abb.3. Graph der Funktion $f(x)$ in Fenstern.

Als nächstes interessieren wir uns für die näherungsweise Berechnung und Darstellung der zweiten Ableitung von $f(x)$. Es gilt

$$f(1) = 47, \quad f''(1) = 94, \quad f''''(1) = -27851401752,$$

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

$$zentraler \;\; Differenzenquotient \;\; 2.\, Ordnung$$

$$\lim_{h \to 0} \Delta_h^2 f(x) = f''(x)$$

Nun untersuchen wir die graphische Darstellung von $\Delta_h^2 f(x)$ als Funktion von h bei beliebigen aber festen x.
Bei der Berechnung der notwendigen Differenzen in der GK-Arithmetik mit der Mantissenlänge t weiß man, daß die Stellenauslöschung bei $h \approx \sqrt{10^{-t}}$ beginnt. Dies steht dem Wunsch entgegen, mit möglichst sehr kleiner Schrittweite h eine gute Genauigkeit der Approximation der 2. Ableitung zu erreichen. Diese **Zwickmühle** ist in [5] für die erste Ableitung dargestellt worden. In unserem Fall stehen sich

- der relative Diskretisierungsfehler $\dfrac{h^2 f''''}{12 f''}$,
- und der Auslöschungsfehler $\varepsilon \dfrac{3f}{h^2 f''}, \quad \varepsilon = 10^{-t}$,

gegenüber.
Vorausgesetzt man kann $f(x)$ mit einem relativen Fehler nahe der Mantissengenauigkeit 10^{-t} berechnen, dann erhalten wir für die Schrittweite h bei einem ausgewogenen Verhältnis der Kontrahenten

$$h = \sqrt[4]{\varepsilon \, \frac{48f}{f''''}},$$

$$h \approx \sqrt[4]{\varepsilon \, 10^{-7}}, \quad falls \;\; x = 1.$$

Für die verschiedenen GK-Formate ergeben sich aus der folgenden Tabelle zulässige Bereiche für die Schrittweite h.

GK-Format	t	h-Bereich
extended	19...20	2E-7...1E-5
double	15...16	5E-7...1E-5
real	11...12	3E-5...4E-5
single	7...8	versagt wegen ungenauer Berechnung von f(x)

Tab.9. Zulässige Schrittweiten h für Approximation der 2. Ableitung.

Graphische Darstellungen von $\Delta_h^2 f(1)$ machen den Sachverhalt noch anschaulicher.

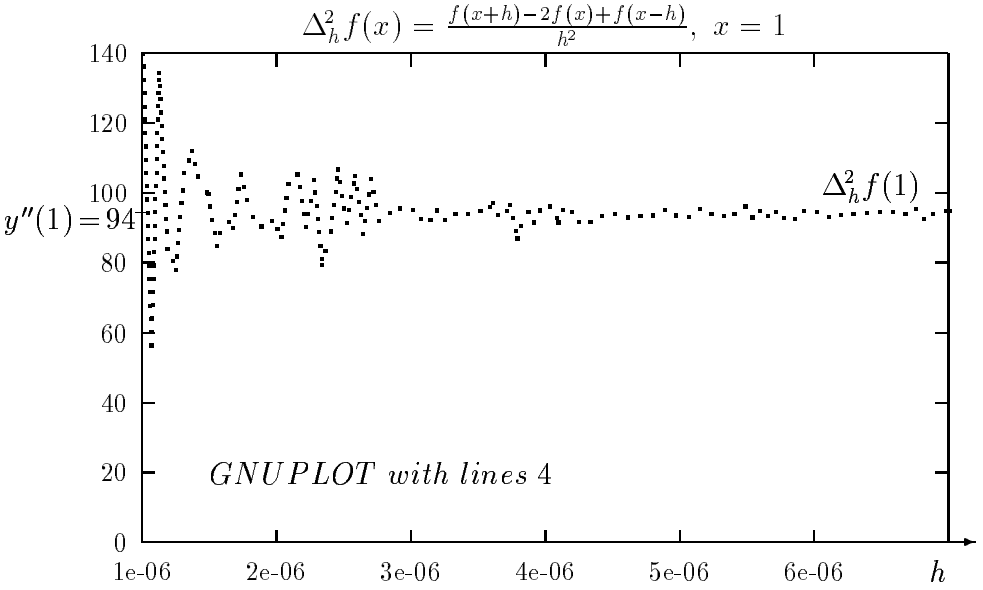

Abb.4. GNUPLOT-Graph.

Jetzt wird die Punktfolge $(h, \Delta_h^2 f(1))$ für die zweite Ableitung mit einem TP-Programm erzeugt und als Text-File "cg23.dat" bereitgestellt. Dieses Datenfile kann man in GNUPLOT graphisch darstellen und in analoger Weise auch als TEX-File "cg23.tex" aufbereiten, um es wie hier in einen Text (Artikel) als Input-File einzubinden.
Für die folgende Abb.5 (Teil 2) notieren wir die GNUPLOT-Kommandos.

```
set terminal latex
set output "cg23.tex"
set size 1.0, 1.0
set title "$\Delta^{\!2}_{h}f(x)=\frac{f(x+h)-2f(x)+f(x-h)}{h^2},\
          x=1 $"
set nokey
set arrow from 0.0000002, 0 to 0.000000205, 0
set label "$h$" at 0.0000002, -8 right
set format x "%g"
set xtics 0.00000002, 0.00000004, 0.00000018
set label "$\Delta^{\!2}_{h}f(1)$" at 0.00000019, 102 right
set label "$y''(1)\!=\!94$" at 0.0000000192, 92 right
set label "--" at 0.0000000215, 94 right
set label "$TP\!-\!Format~~extended$" at 0.00000018, 20 right
f(x) = (4970*x-4923)/(4970*x**2-9799*x+4830)
plot [0.00000002:0.00000020] [0:140] "cg23.dat" with points 1 10
```

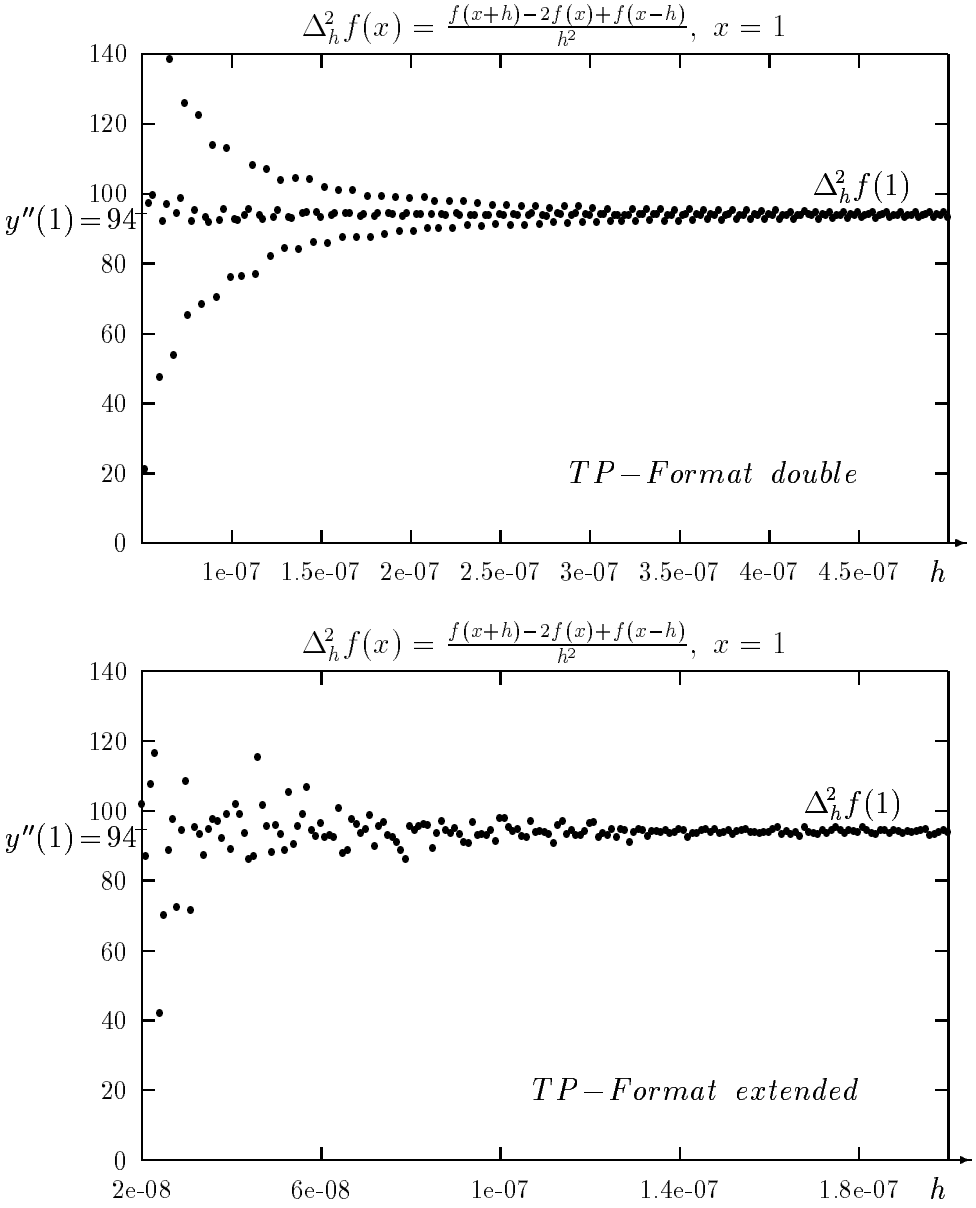

Abb.5. GNUPLOT-Graphen von TP-Dateien.

3.7 Computergenauigkeit und PC-Null

Für manche Aufgabenstellungen ist es wünschenswert oder sogar notwendig, sich einige Informationen über die Computergenauigkeit in der gegebenen GK-Arithmetik zu verschaffen. Für das Format *real* in Turbo Pascal gilt z.B.

(1) z = kleinste positive reelle Zahl = 2.9E-39,

(2) aus $1 + x \neq 1$ und $1 + x/2 = 1$, ergibt sich x=1.8E-12,

(3) Anzahl der gültigen dezimalen Mantissenstellen beträgt 11...12.

Die Bestimmung der Zahl z kann mittels einer Schleifenanweisung erfolgen.

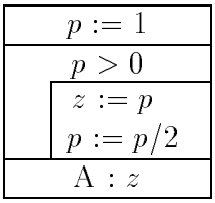

Abb.6. Struktogrammteil für $z = \min_{p>0} p$.

Natürlich ist auch ein anderer Quotient als 2 für die Reduktion möglich.
In TP kann man dieses Konstrukt durch eine Rekursivität in zwei Formen darstellen.

```
type float = real;
function REKURSION1(P:float):float;
var Z:float;
begin
  Z:=P;
  P:=P/2;
  if P>0 then REKURSION1:=REKURSION1(P)
         else REKURSION1:=Z
end;

function REKURSION2(P:float):float;
var Z:float;
begin
  Z:=P;
  if P/2>0 then REKURSION2:=REKURSION2(P/2)
           else REKURSION2:=Z
end;
```

Mit dem Aufruf und der Ausgabe **writeln(REKURSIONi(1)); i=1,2** erhalten wir den Wert 2.9387358771E-39.
Anders verhält sich die Situation in den GK-Formaten *single*, *real*, *double* oder *extended* mit der Compilerdirektive **$N+** (Numerik-Koprozessor). Hier muß man beachten, daß es einen sogenannten **d-Bereich** gibt, der von der kleinsten positiven normierten reellen Zahl (1.Ziffer der Mantisse $\neq 0$) bis zur kleinsten positiven intern darstellbaren Zahl reicht.Es kommt also die Mantissenlänge im Exponenten hinzu.

GK-Format	t	d-Bereich
single	7...8	1.40E-45.....1.18E-38
real	11...12	.....2.93E-39
		kein d-Bereich
double	15...16	4.94E-324.....2.23E-308
extended	19...20	1.90E-4951....3.30E-4932

Tab.10. d-Bereiche der GK-Formate in Turbo Pascal.

Werte von Ausdrücken sind für alle 4 Formate bis zur Größenordnung *extended* möglich. Variablen vom Typ *single*, *double* bzw. *extended* können Werte aus dem d-Bereich aufnehmen und werden erst zu Null, wenn dieser Wertebereich unterschritten wird. Mit dem Funktionsaufruf **REKURSION1(1)** erhalten wir folgende kleinste positive Werte z.

GK-Format	Schrittzahl	z
single	150	1.4E-45
real	129	2.9E-39
double	1075	4.9E-324
extended	16446	1.9E-4951

Tab.11. $z = \min\limits_{p>0} p$ mittels REKURSION1(1) und $N+ .

Weitere Besonderheiten sind :
- Beim GK-Format *extended* werden die Werte aus dem d-Bereich als 0 ausgegeben.
- Im d-Bereich "verlieren" die Variablen gültige Mantissenstellen . Deshalb sollten Rechnungen dort möglichst unterbleiben.

Beim Funktionsaufruf **REKURSION2(1)** kann der Ausdruck $p/2$ im d-Bereich liegen. Beim wiederholten Aufruf von **REKURSION2(p/2)** wird eine lokale Hilfsvariable "h" im Keller bereitgestellt, die erstens denselben Typ wie p hat und zweitens den Wert $p/2$ aufnimmt. Damit wird h=0 , falls $p/2$ erstmalig kleiner als der d-Bereich ist, und **REKURSION2** kehrt mit dem Wert 0 zurück. Es wird also im Vergleich zu **REKURSION1(1)** ein Schritt zuviel ausgeführt.

4 Was leistet der Computer?

An solchen Beispielen zeigt sich, was ein Computer schon und noch nicht leisten kann. Der wirklich kreative Anteil am Problemlösungsprozeß bleibt beim Menschen. Der Phantasie sind aber keine Grenzen gesetzt. Natürlich ist es zumeist nicht einfach, komplizierte Sachverhalte methodisch geschickt und didaktisch wirksam für Präsentationen vor einem Hörerkreis aufzubereiten.
Denkt man bei Computerunteranwendungen zum Beispiel an adaptive und selbstkorrigierende Verfahren, Intervallarithmetik, Konzept der hohen und optimalen Genauigkeit, wissenschaftliches Rechnen mit Ergebnisverifikation oder an paralleles und verteiltes Rechnen (parallel and distibuted computing), so sind hier schon neue und erfolgversprechende Wege gegangen worden.
So erhält man mit Systemen des wissenschaftlichen Rechnens, wie es zum Beispiel Pascal-XSC ist, neben einem Ergebnis auch sehr nützliche Informationen über seine Bewertung und Brauchbarkeit.
Solche und noch nicht vorstellbare Entwicklungen sind ernst zu nehmen, um die komplexen und wachsenden Probleme zu meistern.

Literatur

[1] BAUMANN, R.: *Mathematik mit BASIC.* Klett-Verlag, Stuttgart 1985.

[2] DÖRFLER, W.: *Der Computer als kognitives Werkzeug und kognitives Medium.* In : Computer-Mensch-Mathematik, Schriftenreihe Didaktik der Mathematik Universität Klagenfurt, Band 21, 1994.

[3] HERGET, W.: *Algorithmen und Computer im Mathematikunterricht.* In : Computer-Mensch-Mathematik, Schriftenreihe Didaktik der Mathematik Universität Klagenfurt, Band 21, 1994.

[4] WINKELMANN, B.: *Software für den Mathematikunterricht.* In : Computer-Mensch-Mathematik, Schriftenreihe Didaktik der Mathematik Universität Klagenfurt, Band 21, 1994.

[5] SCHABACK, R.; WERNER, H.: *Numerische Mathematik.* Springer-Verlag, Berlin 1992.

[6] KIELBASINSKI, A.; SCHWETLICK, H.: *Numerische lineare Algebra.* Mathematik für Naturwissenschaft und Technik Band 18, DVW, Berlin 1988.

[7] KULISCH, U.(Hrsg.): *Wissenschaftliches Rechnen mit Ergebnisverifikation - Eine Einführung.* Mathematical Research Vol. 58, Akademie-Verlag, Berlin, Vieweg, Wiesbaden 1989.

[8] KLATTE, R.; KULISCH, U.; NEAGA, M.; RATZ, D.; ULLRICH, CH.: *PASCAL-XSC.* Sprachbeschreibung mit Beispielen. Springer-Verlag, Berlin 1991.

[9] ZURMÜHL, R.; FALK, S.: *Matrizen und ihre Anwendungen,* Teil 2 : *Numerische Methoden.* Springer-Verlag, Berlin 1984.

[10] KOTZ,D.: *LaTeX and the GNUPLOT Plotting Program.* GNUPLOT LaTeX Tutorial Version 3.0, 1991.

Anschrift:

Dr. Werner Neundorf
Institut für Mathematik
Technische Universität Ilmenau
PSF 10 0565
D - 98684 Ilmenau

e-mail : neundorf@mathematik.tu-ilmenau.de