Wolfgang Hackbusch

Numerische Mathematik und ihre Wechselwirkung mit der gegenwärtigen Rechnerentwicklung

(Vortrag in der gemeinsamen Sitzung der Biowissenschaftlich-medizinischen und der Mathematisch-naturwissenschaftlichen Klasse am 14. Februar 1997)

In: Berichte und Abhandlungen / Berlin-Brandenburgische Akademie der Wissenschaften (vormals Preußische Akademie der Wissenschaften) ; 4.1997, S. 55-67

Persistent Identifier: urn:nbn:de:kobv:b4-opus-29716

Wolfgang Hackbusch

Numerische Mathematik und ihre Wechselwirkung mit der gegenwärtigen Rechnerentwicklung

(Vortrag in der gemeinsamen Sitzung der Biowissenschaftlich-medizinischen und der Mathematisch-naturwissenschaftlichen Klasse am 14. Februar 1997)

1 Zur Geschichte der Numerischen Mathematik

Einerseits ist die Numerische Mathematik eine sehr junge Disziplin, andererseits stehen numerische Aspekte, d. h. Aspekte des rechnerischen Umgehens mit Zahlen am Anfang der Mathematikgeschichte überhaupt. Daß sich die Mathematik in der sumerisch-akkadisch-babylonischen Kultur so erfolgreich entwickeln konnte, lag nicht zuletzt an der Tatsache, daß das babylonische Zahlensystem in etwa unserer modernen Gleitkommaarithmetik entsprach. Ein auf der Basis 60 beruhendes System wurde sowohl für den ganzzahligen wie auch für den gebrochenen Anteil verwendet, wobei allerdings der „Dezimal"-Punkt bzw. der Exponentialteil nicht angegeben wurden (vgl. Neugebauer [3]). Eine ähnlich konsequente Zahlendarstellung im 10er-System wurde erst 1585 von dem Holländer S. Stevin eingeführt, vorher wurde im Nachkommabereich weiterhin das 60er-System (Minuten, Sekunden, ...) verwendet. Stevin war ein Vertreter der „Rechenmeister", die sich um 1600 einer regen Nachfrage nach Rechenfertigkeiten erfreuen konnten. Der bis heute tradierte Name von Adam Ries (Riese) spricht für die Popularität der Rechenmeister. Die Entwicklung der Logarithmen ist z. B. eine wichtige Entwicklung jener Zeit.

Im Zuge der Entwicklung der Analysis wurden numerische Algorithmen mitentwickelt und waren inhaltlich nicht von der Analysis getrennt. So waren etwa Reihenentwicklungen von Funktionen kein Selbstzweck, sondern wurden als Berechnungsverfahren verwendet, wobei selbstverständlich Wert auf schnelle Konvergenz gelegt wurde. Stellvertretend seien zwei Namen genannt, die bis heute mit wichtigen und grundlegenden numerischen Algorithmen verbunden sind. Von I. Newton (1643–1727) stammen u. a.

- das „Newton-Schema" (1675): Interpolation mit Hilfe dividierter Differenzen,
- die „Newton-Cotes-Formeln" (Cotes 1711) für die Quadratur von

(1) $\int_a^b f(x)\,dx \sim \sum_j \omega_j f(x_j)$ (ω_j Gewichte, x_j Stützstellen)

mit gleichabständigen Stützstellen,

- das „Newton-Verfahren" (1669) zur Lösung einer nichtlinearen Gleichung.

Die genannten Verfahren sind auch heute noch Gegenstand jeder einführenden Vorlesung „Numerische Mathematik", ebenso wie die folgenden Verfahren, die auf C. F. Gauß (1777–1855) zurückgehen:

- „Gauß-Elimination" zur Lösung eines linearen Gleichungssystems,
- „Gauß-Seidel-Iteration" (Gerling 1845) für die gleiche Aufgabe,
- „Gauß-Quadratur" (1814), Quadratur von (1) mit spezifisch gewählten Stützstellen,
- „Methode der kleinsten Quadrate" (1795) in der Ausgleichsrechnung.

Wer Genaueres zur Numerik jener Zeit erfahren möchte, sei auf Goldstine [1] verwiesen.
Zu einer eigenständigen Disziplin ist die Numerische Mathematik erst Mitte dieses Jahrhunderts in dem Moment geworden, als elektronische Rechner eingesetzt werden konnten. Warum gerade die elektronischen Rechner (und nicht etwa die schon länger verfügbaren mechanischen Rechner) den Anstoß zur Eigenständigkeit gaben, wird im Abschnitt 3 näher erläutert werden.

2 Die Beziehungen der Numerischen Mathematik zur Angewandten Mathematik

Eine Einteilung der verschiedenen mathematischen Bereiche in Reine und Angewandte Mathematik ist nicht sehr hilfreich. Jeder Teil der Mathematik kann dadurch angewandt sein, daß eine Beziehung zu einem außermathematischen Bereich hergestellt wird, der im oberen Teil der Abb. 1 mit „Anwendung" bezeichnet ist. Diese Anwendung braucht noch nicht in der Sprache der Mathematik formuliert zu sein, so daß eine mathematische Modellierung erforderlich sein kann. Danach muß versucht werden, die entstehende mathematische Fragestellung im Sinne der Anwendung zu beantworten. Besonders fruchtbar für die Mathematik waren Anwendungen, die auf Fragen führten, welche im Rahmen der bisher entwickelten Mathematik nicht formalisiert werden konnten und so den Anstoß zu neuen Strukturen innerhalb der Mathematik gaben.
Die Numerische Mathematik hat ebenfalls einen Bezugspunkt außerhalb der Mathematik: den Rechner, der erst in jüngster Zeit der elektronische Rechner ist und davor der mechanische bzw. der menschliche Rechner war (vgl. unterer Teil

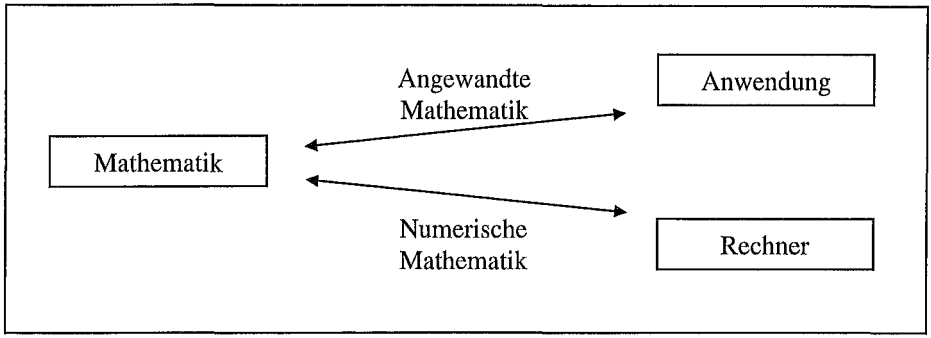

Abb. 1

der Abb. 1). Anders als die Anwendung stellt der Rechner keine Fragen an die Mathematik, sondern gibt im Gegenteil Antworten im Sinne der Lösung elementarer arithmetischer Aufgaben. Damit ist die Numerische Mathematik nicht mit der Angewandten Mathematik gleichzusetzen oder ein Teil von ihr. Allerdings gehen die Numerische Mathematik und die Angewandte Mathematik in den meisten Fällen gemeinsame Wege, da die Nachfrage nach numerischen Antworten typisch für Anwendungen von außen und eher untypisch für innermathematische Aufgaben ist.

3 Numerische Mathematik als selbständige Disziplin

Auf den ersten Blick erscheint es selbstverständlich, daß die Einführung moderner Computer den Anstoß zu einer sich schnell entfaltenden Numerischen Mathematik gegeben hat. Auf der anderen Seite kann man Argumente für das Gegenteil finden: Wie in Abschnitt 1 beschrieben, sind die grundlegenden numerischen Algorithmen schon seit mehr als 100 Jahren bekannt. War die Ausführung der Rechnungen vor der Einführung moderner Computer noch sehr mühsam, können heute die Rechnungen mit atemberaubender Geschwindigkeit durchgeführt werden. Dies deutet eher einen zufriedenstellenden Zustand an und keinen Mangel, der durch wissenschaftliche Fortschritte in der Numerischen Mathematik abzubauen wäre. Dafür, daß in der Tat wissenschaftliche Fortschritte erforderlich sind, seien im folgenden drei unterschiedliche Antworten angegeben:

1. Neuartige Fragestellungen
2. Stabilitätsproblematik
3. Problem der stets zu knappen Rechnerressourcen.

Zu 1. Viele numerische Aufgaben lassen sich erst bei umfangreichen Datenmengen sinnvoll stellen und können daher erst im Zeitalter der modernen Computer in Angriff genommen werden. Ein Beispiel dieser Art ist etwa die Bildrekonstruktion in der Computertomographie und die Klasse der „schlechtgestellten Probleme" im allgemeinen.

Zu 2. Selbst bei klassischen Verfahren hat es bei der Einführung der elektronischen Rechner unangenehme Überraschungen gegeben. Bei der Mehrzahl der Algorithmen gibt es eine „Dimension" oder „Problemgröße", die im folgenden mit n bezeichnet sei. Solange man per Hand oder mechanisch gerechnet hat, war n erzwungenermaßen klein. Erst der Rechner eröffnete die Möglichkeit, den Grenzprozeß $n \to \infty$ besser auszuschöpfen. Welche Schwierigkeiten dabei entstehen können, sei anhand der schon oben erwähnten Newton-Cotes-Quadratur erläutert. Als Aufgabe sei die Integration von (1) für $f(x) = e^x$ und $a=0$, $b=1$ gestellt. Der Integralwert $e-1 = 1.718281828...$ soll durch Newton-Cotes-Formeln mit n Teilintervallen berechnet werden, wobei die Werte der Exponentialfunktion e^x einer Tafel mit 6 Dezimalstellen entnommen seien. Für $n=1$ Teilintervall liefert die „Trapezformel" den Wert

$$0.5 * exp(0) + 0.5 * exp(1) = 1.86... \qquad \text{(Fehler: } 0.14\text{)}.$$

$n = 2$ Teilintervalle („Simpson-Formel") ergeben den wesentlich besseren Wert

$$1/6 * exp(0) + 2/3 * exp(0.5) + 1/6 * exp(1) = 1.7189... \quad \text{(Fehler: } 0.00058\text{)}.$$

Für $n=4$ (Wert = $1.718282...$) und $n=8$ (Wert = $1.7182816...$) sind alle 6 Stellen exakt, was das optimale Resultat bei Eingabewerten ist, die nur auf 6 Stellen genau sind. Läßt man den Rechner nun die Näherung mit einer höheren Teilintervallanzahl n berechnen, so ergeben sich zunehmend unsinnigere Werte, z. B. $n=30$: 1.67, $n=38$: -4.5, $n=40$: $+4.9$. Dieses Fehlverhalten ist keineswegs eine generelle Eigenschaft von Quadraturverfahren. Das oben erwähnte Gauß-Quadraturverfahren, das nur anders definierte Stützstellen x_i (und damit auch andere Gewichte) besitzt, liefert für beliebig große n gute Resultate.

Die Newton-Cotes-Quadratur ist nur ein Beispiel für ein instabiles Verfahren. Auf vielen Feldern haben sich ähnliche Instabilitätsphänomene ergeben. Dies führte dazu, daß sich kurz nach Einführung der elektronischen Rechner Stabilitätstheorien der verschiedenen Bereiche entwickelt haben. Für den numerischen Laien ist die Stabilitätsproblematik oft schwer einsehbar. Noch brisanter ist die Frage, ob eine eventuelle Instabilität als eine solche erkannt werden kann. In vielen Fällen äußert sich die Instabilität durch offensichtlich unsinnige Werte, aber es gibt auch schleichende Instabilitäten, die sogar Konvergenz (gegen einen falschen

Wert) vortäuschen können. Im Zweifelsfall ist die numerische Expertise erforderlich, um festzustellen, ob den numerischen Resultaten zu trauen ist oder nicht.

Zu 3. Das Problem der stets zu knappen Rechnerressourcen ist wohl der stärkste Antrieb, den die Numerische Mathematik erfährt. Der folgende Abschnitt ist dieser Thematik gewidmet.

4 Permanent knappe Rechnerressourcen

4.1 Gleichungssysteme als Modellbeispiel

Die Lösung eines Systems von n Gleichungen mit n Unbekannten

$$
\begin{aligned}
& a_{11}\, x_1 + \ldots + a_{1n}\, x_n = b_1 && (x\text{: gesuchte Lösung}) \\
(2)\quad & \ldots \\
& a_{n1}\, x_1 + \ldots + a_{nn}\, x_n = b_n && (n\text{: Dimension})
\end{aligned}
$$

gehört zu den „simplen" Problemen, auf die der oben erwähnte Eliminationsalgorithmus von Gauß eine generelle Antwort gibt. Lineare Gleichungssysteme treten auch auf, wenn die zu behandelnde Aufgabe nichtlinear ist, da dann die Lösung durch das Newton-Verfahren auf eine Folge linearer Probleme zurückgeführt wird. Eine besonders häufige Quelle linearer Gleichungssysteme sind partielle Differentialgleichungen, die Funktionen in zwei oder drei Raumdimensionen beschreiben. Im einfachsten Falle diskretisiert man die partiellen Differentialgleichungen, indem man ein zwei- oder dreidimensionales (quadratisches bzw. kubisches) Gitternetz der Schrittweite h über das betrachtete Gebiet spannt (vgl. Abb. 2).

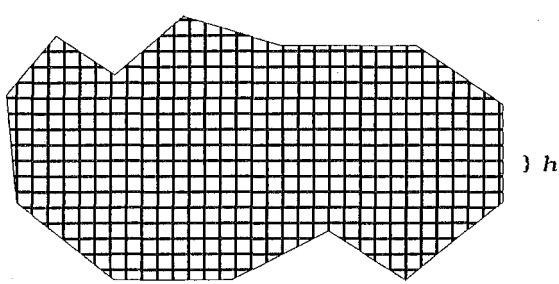

Abb. 2
Gitter der Schrittweite h

Im Falle eines beschränkten Gebietes führt die Schrittweite h zu $n = O(h^2)$ bzw. $n = O(h^3)$ Gitterpunkten (hier bedeutet $O(...)$ „proportional zu ...“). Jedem Gitterpunkt entspricht mindestens eine Unbekannte im entstehenden linearen Gleichungssystem. Schrittweiten $h=1/100$ oder $h=1/1000$ führen somit auf Gleichungssysteme bis zur Dimension $n\sim 1.000.000$ in zwei Raumdimensionen und $n\sim 1.000.000.000$ in drei Raumdimensionen. Es lassen sich weitere Beispiele anfügen, die eine noch höhere Größenordnung von n ergeben.

4.2 Zwei Thesen

Die hohen Dimensionen sind der Grund, warum lineare Gleichungssysteme zu einer schwierigen Aufgabe werden. Die Rechnung mit der zuletzt genannten Dimension $n\sim 1.000.000.000$ scheitert schon deshalb, weil ein entsprechender Speicher im Rechner nicht verfügbar ist. Die Tatsache, daß man Probleme rechnen möchte, die in ihrem Umfang die Kapazität der vorhandenen Rechner weit übersteigen, führt dazu, daß die in der Praxis auftretende Dimension der Gleichungssysteme durch die Kapazität des Rechners begrenzt ist. Hieraus läßt sich die folgende These ableiten:

These I. Die Dimension n für praktisch relevante Probleme ist von der Größenordnung des Maximums, das durch die Speicherkapazität des Rechners gegeben ist.

Die Rechnerkapazität ist insbesondere durch zwei Größen gekennzeichnet, die Rechengeschwindigkeit und den zentralen Speicherplatz. Beide steigen mit fast konstanter Rate, die in etwa mit dem Faktor 100 über 10 Jahre beziffert werden kann. Hierzu einige Daten (nach Siemens-Zeitschrift Special – FuE – Frühj. 1996) in Abb. 3.
Diese Daten belegen die zweite These:

These II. Die Rechnergeschwindigkeit und der zur Verfügung stehende Speicherplatz steigen proportional.

Die Konsequenzen der beiden Thesen seien wieder anhand eines linearen Gleichungssystems erläutert. Der Gaußsche Eliminationsalgorithmus benötigt zur Lösung eines Systems der Dimension n im allgemeinen Fall $2n^3/3$ arithmetische Operationen $(+,-,*,/)$. Seine Realisierung scheitert aber im allgemeinen an dem Speicherplatz, der für die Matrix benötigt wird. Die bei der Diskretisierung partieller Differentialgleichungen auftretenden Matrizen sind zwar schwachbesetzt (d. h. anstelle von n^2 Koeffizienten sind nur $O(n)$ Daten abzuspeichern), aber die Matrix entwickelt sich während der Elimination zu einer eher vollbesetzten Struktur. Als Alternative sei im folgenden die iterative Lösung mit Hilfe der Gauß-

Jahr	Speicher	Prozessor
1973	1 kbit	8080
1975	4 kbit	80186
1978	16 kbit	
1982	64 kbit	80286
1985	1 Mbit	80386
1989	4 Mbit	80486
1994	16 Mbit	Pentium

Tabelle 1
Zeitliche Entwicklung des Speichers

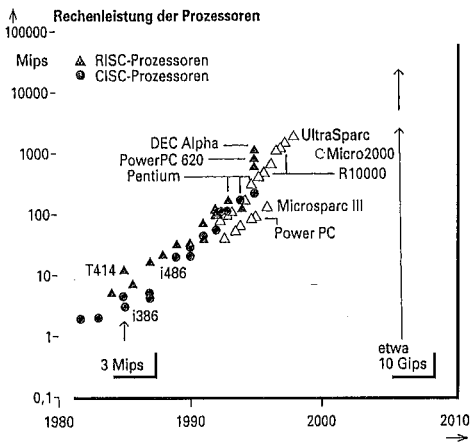

Abb. 3
Zeitliche Entwicklung der Rechenleistung

Seidel-Methode (s. o.) verwandt. Für Standardaufgaben wie die Lösung der Poisson- oder Laplace-Gleichung führt die relativ langsame Konvergenz dieser Iteration dazu, daß ebenfalls $O(n^3)$ arithmetische Operationen zur Gleichungslösung erforderlich sind. Dafür bleibt der Speicheraufwand bei $O(n)$. Dieser Algorithmus soll im folgenden zugrundegelegt werden.

4.3 ... und ihre Folgen

Nach den oben angegebenen Daten werden nach 5 Jahren Rechner angeboten, die sich um den Faktor 10 verbessert haben. Wir nehmen daher an, eine Rechner-Neuinvestition habe gemäß These II sowohl den verfügbaren Speicher als auch die Rechengeschwindigkeit um den Faktor 10 ansteigen lassen. Nach These I möchte man nun aber Probleme der Dimension $10*n$ statt n lösen. Die Zahl der arithmetischen Operationen betrug $O(n^3)$ und steigt deshalb um den Faktor 1.000 an; wegen der gleichzeitig verbesserten Rechengeschwindigkeit bleibt ein Anstieg der Rechenzeit um den Faktor 100. Dies führt zu dem paradoxen Fazit, daß der schnellere Rechner mehr Zeit benötigt. Dieses Paradox stellt sich solange ein, wie Verfahren verwendet werden, deren Aufwand stärker als linear wächst: $O(n^\alpha)$ mit

$\alpha > 1$. Erst wenn ein Algorithmus lineare Komplexität besitzt, d. h. wenn er einen Aufwand proportional zur Dimension n besitzt, kompensieren sich der größere Aufwand und die gestiegene Rechengeschwindigkeit. Allerdings ist lineare Komplexität zugleich die bestmögliche Komplexitätsordnung, da schon eine einzige Operation pro Unbekannte zu einem $O(n)$-Aufwand führt. Für das Weitere halten wir fest:

Bemerkung: Für Aufgaben, auf die These I zutrifft, benötigt man Algorithmen mit (möglichst) linearer Komplexität.

Diese Bemerkung zeigt deutlich, daß die Entwicklung im Bereich der Rechner einen wesentlichen Impuls auf die Numerische Mathematik ausübt. Es zeigt sich, daß der Rechner (vgl. Skizze in Abb. 1) nicht nur Antworten gibt, sondern der Numerischen Mathematik auch spezifische Aufgaben stellt. Man kann aber in bezug auf die Rechnerentwicklung auch den Umkehrschluß ziehen: Wäre die Numerik nicht in der Lage gewesen, entsprechende Algorithmen anzubieten, gäbe es wegen der immer größer werdenden Rechenzeiten (siehe obiges Paradoxon) kaum Interesse, die Speicherkapazitäten zu erhöhen.
Es sei angemerkt, daß es andere Aufgaben gibt, bei denen man froh wäre, einen Algorithmus mit z. B. kubischer Komplexität zu kennen. Für diese Probleme trifft allerdings These I nicht zu. In diesen Fällen ist die Rechenzeit auch schon für mäßige Dimensionen das Nadelöhr.

4.4 Helfen Parallelrechner?

Ein Verbund von p Prozessoren (mit jeweiliger Speicherkapazität n) liefert den (verteilten) Gesamtspeicher vom Umfang $p*n$. Geht man optimistischerweise davon aus, daß ein Algorithmus optimal parallelisiert werden kann (optimaler „speed-up"), so reduziert sich seine Rechenzeit um dem Faktor $1/p$. Man sieht, daß auch in diesem Fall die These II gültig bleibt: Bezüglich Speicher und Geschwindigkeit trifft der gleiche Faktor p zu. Damit stellt der Parallelrechner keine neue Situation her. Die Algorithmen müssen jetzt aber nicht nur optimale Komplexität besitzen, sondern auch möglichst gut parallelisierbar sein und mit verteiltem Speicher arbeiten können. Man kann einen Parallelrechner mit p Einzelprozessoren auch als einen Vorschuß auf die Zukunft ansehen: Nach entsprechender Wartezeit würden sequentielle Rechner angeboten, die hinsichtlich Speicher und Geschwindigkeit um den Faktor p verbessert sind.

4.5 Lineare Komplexität bei linearen Gleichungssystemen

Die Frage, ob lineare Gleichungssysteme in $O(n)$ Operationen gelöst werden können, ist in dieser Form zunächst zu verneinen. Im folgenden sei die Situation vorausgesetzt, die bei der Diskretisierung partieller Differentialgleichungen vorliegt: Die Matrix A des Gleichungssystems $Ax=b$ (Kurzschreibweise für (2)) sei schwachbesetzt. Dies stellt sicher, daß die Matrixvektormultiplikation $(A,x) \rightarrow A*x$ mit $O(n)$ Operationen durchführbar ist (für eine vollbesetzte Matrix wären es $2n^2$ Operationen). Auch für (beliebige) schwachbesetzte Matrizen ist kein allgemeiner Algorithmus bekannt, der das lineare Gleichungssystem in $O(n)$ Operationen löst. Zur Illustration der Schwierigkeiten sei folgendes angemerkt. Die inverse Matrix A^{-1} einer schwachbesetzten Matrix A ist im allgemeinen vollbesetzt. Auch wenn die Inverse A^{-1} kostenlos zur Verfügung stände, erforderte die Lösung von $Ax=b$ mittels der Matrix-Vektor-Multiplikation $x = A^{-1}*b$ immer noch $O(n^2)$ Operationen. Um Algorithmen linearer Komplexität zu erhalten, muß man entweder spezielle Matrixeigenschaften (im Sinne der linearen Algebra) ausnutzen oder aber von der Herkunft der Matrizen Gebrauch machen. Letzteres ist erfolgreich für Matrizen, die mittels Diskretisierung aus elliptischen Differentialgleichungen entstehen. Die speziellen Eigenschaften elliptischer Differentialgleichungen findet man in diskreter Form in den Matrizen wieder. Das Mehrgitterverfahren ist von dieser Form und hat sich in den verschiedensten Anwendungen als sehr erfolgreich erwiesen (vgl. Hackbusch [2] sowie die Tabelle im nachfolgenden Unterabschnitt).

4.6 Numerischer Fortschritt versus Fortschritte in der Rechnerentwicklung

Die Ausführungen haben gezeigt, daß bei der Lösung hochdimensionaler Probleme („Large Scale Computations") sowohl die moderne Rechnerentwicklung als auch die Entwicklung numerischer Verfahren wesentliche Beiträge geliefert haben. Das HPCC-Programm der US-Regierung hat 1992 versucht, die Fortschritte zu quantifizieren. Die folgende Tabelle stammt aus dieser Untersuchung und belegt, daß die Verbesserungen aufgrund moderner numerischer Algorithmen sogar noch größer sind als die infolge der Hardwareentwicklungen.

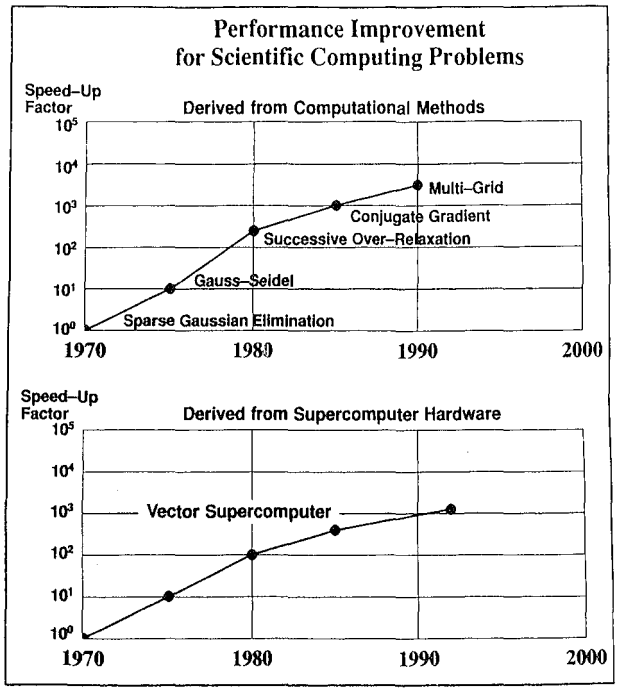

Tabelle 2
Aus dem HPCC-Programm

5 Adaptivität

5.1 Noch einmal: Dilemma der permanent knappen Ressourcen

In dem vorangehenden Abschnitt wurde die Aufgabe (eine partielle Differentialgleichung) diskretisiert und für das resultierende lineare Gleichungssystem ein möglichst schneller Algorithmus gesucht. Die Art der Diskretisierung (d. h. Näherung durch ein endlich-dimensionales Problem) stand dabei nicht zur Disposition. Nachhaltiger ist es, eine der folgenden Fragen zu beantworten:

a) Gegeben eine Differentialgleichung und eine Genauigkeitsanforderung ε. Wie ist zu diskretisieren, damit die Dimension einer diskreten Lösung mit der Genauigkeit $<\varepsilon$ möglichst klein ist?

b) Gegeben eine Differentialgleichung und eine Dimension n (z. B. maximale Speichergröße). Wie sieht eine n-dimensionale Näherung aus, die eine möglichst gute Genauigkeit besitzt.

In beiden Fällen lautet die Antwort, daß man mit Finite-Element-Methoden arbeiten muß, die problem-angepaßte Gitter verwenden. Welchen Gewinn man aus einer geeigneten Anpassung ziehen kann, sei an einem einfachen Beispiel im folgenden Abschnitt illustriert. Weiteres zu Anpassungstechniken folgt in Abschnitt 5.3.

5.2 Einführendes Beispiel zu angepaßten Gittern

Zur Quadratur von $f(x)=x^{0.1}$ über dem Intervall $[0,1]$ sei die summierte Trapezmethode angewandt. Dabei wird in jedem Teilinterval $[x_i,x_{i+1}]$ die Trapeznäherung $(x_{i+1}-x_i)*(f(x_i)+f(x_{i+1}))/2$ verwendet. Pro Teilintervall entsteht dabei ein Fehler der Form

$$h^2 f''/12, \qquad \text{wobei } h = x_{i+1} - x_i \tag{3}$$

die Teilintervallänge und f'' die zweite Ableitung von f an einer Zwischenstelle in $[x_i, x_{i+1}]$ ist. Die einfachste Realisierung der Trapezformel benutzt gleichabständige x_i, also $x_i = i/n$ für $0 \le i \le n$. In diesem Fall sind die Fehler (3) von sehr unterschiedlicher Größenordnung: Für $[x_i, x_{i+1}]$ mit kleinem i sind sie wesentlich größer als für $i \sim n$. Eine bessere Strategie ist es, Teilintervalle verschiedener Länge zu wählen, wobei die Längen so einzurichten sind, daß die Fehler (3) für alle i vergleichbare Größenordnung haben. Im Falle von $f(x) = x^{0.1}$ findet man, daß $x_i = (i/n)^{3/1.1}$ für $0 \le i \le n$ eine geeignete Wahl der Stützstellen x_i ist. Versucht man, mittels der summierten Trapezformel einen Integralwert mit einer Genauigkeit von mindestens $\varepsilon = 10^{-6}$ zu berechnen, so beträgt n im gleichabständigen Fall: $n=128.600$, im angepaßten Fall: $n=391$. Man sieht, daß sich Größenordnungen in der Rechenzeit gewinnen lassen, ohne an Genauigkeit einzubüßen, wenn man nur eine geeignet angepaßte Diskretisierung gefunden hat.

5.3 Adaptive Gitter bei Finiten Elementen

Im vorherigen Beispiel (Integration von f) war die Funktion f explizit bekannt einschließlich ihrer zweiten Ableitung, die in (3) auftrat. Bei Differentialgleichungen ist man in einer schwierigeren Lage, da die zu approximierende Funktion Gegen-

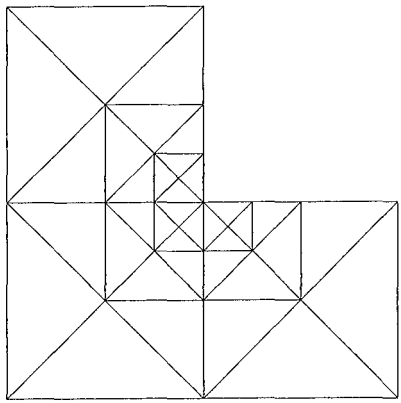

Abb. 4
Adaptives Gitter im L-Gebiet

stand des Problems und somit noch unbekannt ist. Für die Adaption gibt es zwei ganz unterschiedliche Strategien:

a) Adaption mittels a-priori-Kenntnissen (z. B. über die Lage und Art möglicher Singularitäten),
b) Automatische Kontrolle durch den Algorithmus selbst.

Abb. 4 zeigt die Triangulierung eines Gebietes, wobei in der Nähe der einspringenden Ecke die Dreiecke verfeinert werden. Je kleiner die Dreiecke sind, umso genauer läßt sich die gesuchte Funktion mit Hilfe der Finite-Element-Methode approximieren. Im Falle der Laplace-Gleichung hat man a-priori-Aussagen über eine Singularität der Lösungsgradienten an der einspringenden Ecke und kann anhand dieser Kenntnisse beschreiben, wie sich die Dreiecke verfeinern sollten, wenn ihr Abstand von der Ecke kleiner wird.
Gegen die Adaption mittels a-priori-Kenntnissen spricht,

- daß sie nur von Anwendern benutzt werden kann, die über entsprechende analytische Kenntnisse verfügen,
- daß sie programmtechnisch eher aufwendiger ist als die Alternative b) und
- daß die Mehrzahl der Gründe für Adaption nicht a priori vorhersagbar sind.

Im Falle b) („Selbstadaptivität") besteht die Aufgabe des Algorithmus nicht nur in der Lösung des diskreten Problems, sondern auch in der Bestimmung der Diskretisierung. Die Werkzeuge, die hierfür zur Verfügung stehen sind sogenannte Fehlerschätzer (oder auch nur Fehlerindikatoren) und a-posteriori-Fehlerabschätzungen. Während eine a-priori-Abschätzung den Fehler vor einer Rechnung auf-

grund der gegebenen Problemgrößen anzugeben versucht, wird mit einer a-posteriori-Fehlerabschätzung aufgrund der aus der Rechnung gewonnenen diskreten Daten eine Schätzung des Fehlers angestellt. Damit können zwei Ziele erreicht werden:

- Stoppkriterium: Falls die gewünschte Genauigkeit aufgrund der a-posteriori-Fehlerabschätzung erreicht ist, kann die Rechnung mit positivem Resultat beendet werden.
- Unterstützung der Adaptivität: Fehlerschätzer bzw. Fehlerindikatoren geben an, wo der Fehler besonders groß ist.

Gemäß der auch in Abschnitt 5.2 verwandten Strategie der Gleichverteilung der lokalen Fehler wird man das Gitter dort verfeinern, wo große Fehler angezeigt werden. Dieser Verfeinerungsprozeß ist so lange zu wiederholen, bis das Stoppkriterium erreicht ist. Die zur Verfeinerung angemerkten Dreiecke der Triangulierung werden dabei in der Regel in vier ähnliche Dreiecke halber Länge zerlegt, wobei weitere Maßnahmen in den Nachbardreiecken nötig werden können. Details zu Fehlerschätzern bei elliptischen Differentialgleichungen findet man z. B. bei Verfürth [5] oder übersichtsweise in [4, Abschnitt 7.7.6].

Literatur

[1] Goldstine, H. H.: A History of Numerical Analysis. New York: Springer-Verlag, 1977.

[2] Hackbusch, W.: Multi-Grid Methods and Applications. Berlin: Springer-Verlag, 1985.

[3] Neugebauer, O.: Vorlesungen über Geschichte der antiken mathematischen Wissenschaften. Berlin: Springer-Verlag, 1934.

[4] Teubner-Taschenbuch Mathematik, Band I (Hg.: E. Zeidler). Stuttgart – Leipzig: Teubner, 1996.

[5] Verfürth, R.: A Review of a posteriori Error Estimation and Adaptive Mesh-Refinement Techniques. Chichester – Stuttgart: Wiley – Teubner, 1996.