J. Dankert

Numerische Methoden der Mechanik

Springer-Verlag
Wien New York

Numerische Methoden der Mechanik

Festigkeits- und Schwingungsberechnung mittels elektronischer Rechentechnik

J. Dankert

Springer-Verlag
Wien New York

Dr.-Ing. Jürgen Dankert
Wissenschaftlicher Oberassistent
an der Technischen Hochschule „Otto von Guericke",
Magdeburg, Deutsche Demokratische Republik

Das Werk erscheint gleichzeitig im
VEB Fachbuchverlag Leipzig und im
Springer-Verlag Wien — New York

Vertriebsrechte für die sozialistischen Länder:
VEB Fachbuchverlag Leipzig

Vertriebsrechte für alle Staaten mit Ausnahme der sozialistischen Länder.
Springer-Verlag Wien — New York

Mit 162 Abbildungen

Printed in GDR
Satz und Druck: Volksdruckerei Zwickau

Library of Congress Cataloging in Publication Data.
Dankert, J., 1941-. Numerische Methoden der Mechanik.
Bibliography: p. Includes index. 1. Engineering mathematics.
I. Title. TA 330. D35. 510'. 2' 462. 77-7180.

ISBN 3-211-81439-6 Springer-Verlag Wien–New York
ISBN 0-387-81439-6 Springer-Verlag New York–Wien

Vorwort

Die Verfahren der numerischen Mathematik werden in den unterschiedlichsten Bereichen von Naturwissenschaft und Technik angewendet. Von den in der Mechanik gebräuchlichen Methoden (und in diesem Sinne ist der Titel dieses Buches zu verstehen) sollen einige besonders häufig angewendete hier vorgestellt werden.

Die zwangsläufig zu treffende Auswahl war nicht einfach. Die Entscheidungen wurden schließlich von der Absicht diktiert, erprobte und vielseitig anwendbare Verfahren anzugeben, die den Praktiker in die Lage versetzen, eine möglichst große Palette von Problemen zu behandeln.

Der Ingenieur ist es gewöhnt, auf den gesicherten Erkenntnissen der Physik und der Mathematik aufzubauen, nicht kritiklos, wohl aber im (berechtigten) guten Vertrauen auf deren Gültigkeit. Er ist für Existenz- und Konvergenzbeweise dankbar, scheut jedoch gegebenenfalls auch das "numerische Experiment" nicht. Sein Interesse konzentriert sich in erster Linie auf die Realisierbarkeit der angebotenen Methoden. Dieser Aspekt stand bei der Beschreibung der Berechnungsverfahren im Mittelpunkt. Dem Leser, der an theoretischen Fragen stärker interessiert ist, steht in der mathematischen Literatur eine Vielzahl ausgezeichneter Publikationen zur Verfügung.

Das vorliegende Buch setzt die Kenntnisse der Grundausbildung in der Technischen Mechanik voraus, die an einigen Stellen jedoch nicht ganz ausreichen. In jedem Fall werden die benötigten Formeln und Differentialbeziehungen sowie die Voraussetzungen der zugrunde liegenden Theorien in knapper Form zusammengestellt. Ausführlichere Informationen liefert die Literatur zur Technischen Mechanik und ihrer Spezialgebiete, die Literaturstellen /1/ bis /10/ werden (neben vielen anderen) in dieser Hinsicht empfohlen.

Auf die Herleitung der Verfahren der numerischen Mathematik wird nur dann eingegangen, wenn daraus nützliche Erkenntnisse für ihre Anwendung zu ziehen sind. Im übrigen wird auch hier auf die Spezialliteratur verwiesen (z. B. /11/ bis /20/).

Die erfolgreiche Anwendung moderner Berechnungsverfahren setzt die Nutzung des elektronischen Digitalrechners voraus. Bei allen behandelten Verfahren wird deshalb auch auf Programmierungsprobleme eingegangen. Dem Ingenieur werden Entscheidungshilfen gegeben, wann er auf Bibliotheksprogramme zurückgreifen sollte und in welchen Fällen die Handrechnung bzw. eine Kombination von Hand- und Automatenrechnung günstiger ist.

Die angegebenen Unterprogramme sind weitgehend rechenzeit- und speicheroptimal. Sie sind der vom Verfasser geschriebenen Programmbibliothek "Grundaufgaben der angewandten Mathematik und Mechanik" entnommen. Um Druckfehler zu vermeiden, wurden von den getesteten Programmen Lochstreifenkopien hergestellt, die maschinell auf die Druckvorlage übertragen wurden.

Die bewußt einfach gewählten Beispiele sollen das Wesentliche verdeutlichen und durch den Vergleich mit analytischen Lösungen die Leistungsfähigkeit der numerischen Verfahren demonstrieren. Das induktive Vorgehen, bei dem gelegentlich sogar ein Beispiel an die Spitze gestellt wird, erleichtert sicher die Durcharbeitung des Stoffes. Die zum Teil kurz gehaltene Darstellung kommt dem Leser entgegen, der sich einen Überblick über die Verfahren verschaffen will. Ein tieferes Eindringen ist ohne aktive Mitarbeit (mit Bleistift und Papier, im Idealfall unter Einbeziehung des Computers) ohnehin wohl kaum möglich.

Für zahlreiche wertvolle Hinweise bei der Erarbeitung des Manuskripts dankt der Verfasser den Herren Prof.Dr.sc.techn. J. Altenbach, Prof. Dr.sc.techn. U. Fischer, Dr.-Ing. U. Gabbert, Doz. Dr.-Ing. S. Koczyk, Doz. Dr.-Ing. W. Wenzke und nicht zuletzt seiner Frau Dr.-Ing. Ingrid Dankert. Herr Dipl.-Ing. H. Mohr hat mit großer Mühe und viel Fleiß das Manuskript durchgesehen und die Beispiele nachgerechnet. Das druckreife Manuskript wurde von Frau Heinemann geschrieben. Frau Kersten hat die Zeichnungen angefertigt und die Formeln eingesetzt. Für die sorgfältige Arbeit soll auch ihnen an dieser Stelle gedankt werden.

Magdeburg

J. Dankert

INHALTSVERZEICHNIS

Seite

1. Numerische Methoden und Digitalrechentechnik

1.1. Vorbemerkungen

Der Ingenieur nutzt die Technische Mechanik, um Beanspruchungen, Verformungen, Funktionsweise und Sicherheit von technischen Systemen zu untersuchen. Er ist dabei immer gezwungen, das reale Objekt durch Vereinfachungen und Idealisierungen in ein Modell zu überführen, das unter Ausnutzung gesicherter Gesetze und bewährter Hypothesen berechnet werden kann.

Die zur Verfügung stehenden mathematischen Methoden und die Hilfsmittel zu ihrer praktischen Realisierung beeinflussen den Prozeß der Modellfindung, zumal der Praktiker in der Regel die Ergebnisse möglichst schnell erhalten will. Erst durch den Einsatz der elektronischen Rechentechnik ist es möglich, Berechnungsmodelle zu verwenden, die auch die Verhaltensweise komplizierter Gebilde ausreichend genau widerspiegeln.

Dabei haben besonders die numerischen Methoden an Bedeutung gewonnen. Viele dieser Verfahren wurden erst durch den Einsatz von Rechenautomaten effektiv, andere wurden der Arbeitsweise dieses neuen Hilfsmittels angepaßt oder direkt dafür entwickelt.

Berechnungsverfahren, für die ein Rechenprogramm vorliegt, können genutzt werden, ohne daß der Nutzer den zugrunde liegenden Algorithmus kennt. Vielfach braucht er nicht einmal zu programmieren, die Beachtung der Vorschriften zur Dateneingabe und einige elementare Kenntnisse über das Betriebssystem des verwendeten Rechners genügen.

Für die wichtigsten mathematischen Verfahren (Lösung von Gleichungssystemen, Differentialgleichungen usw.) sind in den Bibliotheken vieler Rechenzentren solche Programme vorhanden. Es ist jedoch auch möglich, unter Verwendung spezieller Verfahren (z.B. der Finite-Elemente-Methode) Programme zu schreiben, die die unterschiedlichsten Aufgabenstellungen ganzer Problemklassen der Technischen Mechanik erledigen können. Auch solche Programme (z.T. umfangreiche Programmsysteme) existieren schon in relativ

großer Zahl. Trotz dieses Angebots ist es zweckmäßig, die verwendeten Algorithmen zu kennen, so daß alle Möglichkeiten ausgeschöpft und die Grenzen und Nebeneffekte der Verfahren berücksichtigt werden können.

Für zahlreiche Aufgaben ist es jedoch immer wieder erforderlich, spezielle Programme zu schreiben. Die noch weit verbreitete Meinung, daß dafür Programmierer eingesetzt werden sollten, ist falsch. Um einen Programmierer zu befähigen, einen Algorithmus in die Sprache des Rechenautomaten umzusetzen, muß dieser exakt beschrieben werden. Dazu stehen aber die leicht erlernbaren problemorientierten Programmiersprachen zur Verfügung, die von speziellen Programmen automatisch in die Maschinensprache übersetzt werden.

Der Ingenieur muß auch selbst programmieren, um alle Vorteile der elektronischen Rechentechnik zu nutzen. Dabei bedient er sich selbstverständlich der in Programmbibliotheken für Standardprozesse vorhandenen Unterprogramme.

1.2. Programmierung

1.2.1. Assembler- und Compilersprachen

Eine programmgesteuerte elektronische Digitalrechenanlage kann nur in ihrer Maschinensprache geschriebene Programme verarbeiten. Diese bestehen aus Maschinenbefehlen, die als Binärzahl verschlüsselt im Operativspeicher stehen und die arithmetischen, logischen, Kontroll- und Steueroperationen sowie die Ein- und Ausgabe initiieren. Ein Maschinenbefehl enthält den Befehlscode und ein oder mehrere Adressen von Speicherplätzen, auf deren Inhalt sich der Befehl bezieht.

Das Programmieren in dieser anlagenspezifischen Sprache ist sehr mühsam und fehleranfällig und wird selbst von den dazu befähigten Spezialisten vermieden, wenn, wie für moderne Anlagen selbstverständlich, eine sogenannte Assemblersprache zur Verfügung steht. In dieser Sprache wird der Befehlscode durch leicht verständliche Symbole ersetzt (z.B. ADD für die Addition zweier Zahlen), und statt der Speicheradressen werden Namen (symbolische Adressen)

geschrieben, für die zum Beispiel Formelsymbole verwendet werden können. Vor der Abarbeitung werden diese Symbole vom Assembler, einem Programm im Maschinencode, ersetzt, wobei in der Regel aus jedem Befehl der Assemblersprache ein Maschinenbefehl entsteht. Bei einigen Anlagen können häufig vorkommende, aus mehreren Maschinenbefehlen bestehende Routinen durch einen Assemblerbefehl (sog. MAKRO) beschrieben werden.

Im Gegensatz zu den maschinenabhängigen Assemblersprachen sind die problemorientierten Compilersprachen weitgehend maschinenunabhängig. Von den speziell für wissenschaftlich-technische Rechnungen entwickelten Sprachen sind ALGOL und FORTRAN besonders verbreitet. Ihre Befehlsstruktur ist der in der Mathematik üblichen Formelsprache ähnlich und wird durch leicht verständliche Wortsymbole ergänzt.

Vor der eigentlichen Rechnung müssen die in einer Compilersprache geschriebenen Programme (Quellen-Programme) von einem Übersetzerprogramm (Compiler) in die Maschinensprache übertragen werden. Leistungsfähige FORTRAN- und ALGOL-Compiler existieren für alle modernen Rechenanlagen.

Dies ist leider für die Sprache PL/1 noch nicht gegeben. PL/1 vereinigt in sich die von ALGOL, FORTRAN und der auf ökonomische Probleme orientierten Sprache COBOL gegebenen Möglichkeiten.

Selbst der wenig geübte Programmierer ist in der Lage, seine Aufgaben mit Hilfe der problemorientierten Sprachen für die Arbeit einer Rechenanlage aufzubereiten. Er wird dabei von den Compilern unterstützt, die die Programme auf (syntaktische) Richtigkeit untersuchen und entsprechende diagnostische Informationen ausgeben. Der Nachteil, daß schon die Übersetzung Rechenzeit benötigt, wird durch den wesentlich geringeren manuellen Arbeitsaufwand ausgeglichen. Darüber hinaus ist im allgemeinen der gesamte Rechenzeitverbrauch für das Einfahren (Test auf syntaktische und algorithmische Richtigkeit) des Programms geringer. Schwerwiegender kann der Nachteil sein, daß ein vom Compiler erzeugtes Maschinenprogramm nicht so effektiv arbeitet wie ein entsprechendes Assemblerprogramm. Dies gilt besonders für Programme, die für häufige Nutzung mit unterschiedlichen Eingabedaten vorgesehen sind.

Der Grund dafür, daß heute FORTRAN die meistgenutzte Sprache für die Programmierung wissenschaftlich-technischer Probleme ist, liegt sicher darin, daß sie einen besonders glücklichen Kompromiß darstellt zwischen den Einschränkungen, die dem Programmierer auferlegt werden müssen und der dem Compiler gegebenen Möglichkeit, ein leistungsfähiges Maschinenprogramm zu erzeugen.

Die weitaus größeren Freiheiten, die ALGOL dem Programmierer läßt, machen diese Sprache ideal für die Beschreibung von Algorithmen, führen aber zu weniger effektiven Maschinenprogrammen. Hinzu kommt, daß im ersten ALGOL-Bericht (1960) die Ein- und Ausgabe von Daten noch nicht definiert war. Dies führte dazu, daß fast alle Rechenanlagen dafür andere Routinen vorsehen, was die Verwendung der Programme auf unterschiedlichen Rechnertypen erschwert.

PL/1 bietet für rein numerische Rechnungen gegenüber FORTRAN kaum Vorteile, obwohl das wesentlich größere Befehlsspektrum ein bequemeres Programmieren gestattet. Dies muß jedoch mit langsamer arbeitenden Maschinenprogrammen bezahlt werden. Die Universalität dieser Sprache kann ihr jedoch trotzdem schon sehr bald zum endgültigen Durchbruch verhelfen.

Die Problematik des Programmierens in problemorientierten Sprachen besteht nicht in der Beachtung der syntaktischen Regeln einer bestimmten Sprache, sondern fast ausschließlich in der Formulierung des Algorithmus. Man sollte sich deshalb nicht scheuen, sich der jeweils gegebenen Situation (zur Verfügung stehende Rechenanlagen, Compiler usw.) anzupassen. Das Erlernen der Programmierung erfordert einige praktische Übung, der Übergang zu einer neuen Programmiersprache ist demgegenüber unproblematisch.

1.2.2. Einige allgemeine Bemerkungen zur Programmierung

Eine klare und übersichtliche Formulierung des zu programmierenden Problems ist die beste Voraussetzung für das Schreiben eines Programms. Für umfangreiche Aufgaben empfiehlt es sich, einen Programmablaufplan anzufertigen, der in groben Schritten den Rechengang verdeutlicht. Detaillierte Programmablaufpläne sind bei der Anwendung problemorientierter Sprachen nicht erforderlich.

Vor der Festlegung der zu verwendenden numerischen Verfahren sollte man überprüfen, ob geeignete Routinen aus einer Programmbibliothek oder der Literatur genutzt werden können. Die Wahl der Programmiersprache muß mit Rücksicht auf die Möglichkeiten des zu nutzenden Rechenautomaten getroffen werden.

Ein Programm, das für die Lösung eines aktuellen Problems geschrieben wird, kann oft ohne wesentlichen Mehraufwand so angelegt werden, daß es gleichartige oder ähnliche Aufgaben erledigen kann, wenn nur wenige Parameter variabel gehalten werden, wobei die aktuelle Wertzuweisung über die Dateneingabe gesteuert wird. Man sollte auch überprüfen, ob Teile des Programms für andere Programmierungsprobleme verwendbar sein könnten. Diese werden dann zweckmäßig als Unterprogramme geschrieben.

Für Programme, die zur mehrfachen Nutzung vorgesehen sind, müssen Dokumentationen erarbeitet werden. Diese enthalten Informationen über die Möglichkeiten und Grenzen des Programms, den verwendeten Algorithmus, benötigte Unterprogramme, die Programmiersprache, reguläre und irreguläre Programmstopps, eine ausführliche Beschreibung der Dateneingabe und, falls erforderlich, Hinweise über die Ausgabe der Ergebnisse. Im Zusammenhang mit den Angaben über die Speicherung des Programms (z.B. Quellenprogramm oder ladefähiges Programm auf Lochkarten oder Magnetband) mit Hinweisen auf die erforderlichen Steuerbefehle wird so die Nutzung durch verschiedene Anwender ermöglicht. Mitteilungen über Erfahrungen bei der Programmnutzung, die etwa zu erwartenden Rechenzeiten und ein Beispiel zur Dateneingabe erleichtern die Anwendung.

Der Umgang mit größeren Eingabedatenmengen erfordert besonders sorgfältiges Arbeiten. Prinzipiell sollten alle Eingabedaten sofort nach dem Einlesen gedruckt werden. Ein Druckbild, das mit genügend Textinformation versehen ist, erhöht die Nutzerfreundlichkeit. Programme, die für größere Rechnungen vorgesehen sind, müssen die Eingabewerte auf Vollständigkeit und, wenn möglich, auch darauf kontrollieren, ob ein sinnvolles Berechnungsmodell beschrieben wird. Fällt ein Test negativ aus, wird die nachfolgende Rechnung verhindert.

Die Zeit, die für eine sorgfältige Programmierung aufgewendet

wird, spart man beim Einfahren des Programms ganz sicher wieder ein. Man programmiere übersichtlich und spare nicht mit Kommentar im Programm.

In der Regel erfordert das Testen wesentlich mehr Zeit als das Programmieren. Geeignete Testbeispiele mit bekannten Lösungen sollten so ausgewählt werden, daß jeder Programmteil wenigstens einmal durchlaufen wird. Die Werte 1 und 0 werden bei den Eingabedaten und, wenn das überschaubar ist, als Zwischenergebnisse in der Testphase möglichst vermieden, da sich sonst ein vergessener Faktor bzw. Summand eventuell nicht bemerkbar macht. Zusätzliche Druckanweisungen, die bei der Nutzrechnung entweder nicht angesteuert oder ganz aus dem Programm herausgenommen werden, erleichtern die Fehlersuche.

Für besonders komplizierte Teile des Programms empfiehlt sich ein "Trockentest", die Simulierung der Automatenrechnung durch den Programmierer.

Trotz sorgfältigster Testung können Programme bei Nutzrechnungen gelegentlich versagen. Gute Programme reagieren auf solche Ausnahmesituationen (z.B. schlechte Konvergenz iterativer Verfahren) mit einer entsprechenden Druckanweisung.

1.2.3. FORTRAN-Programmierung numerischer Methoden

Charakteristisch für zahlreiche Programme, die numerische Verfahren enthalten, ist, daß sie den weitaus größten Teil der Rechenzeit in einem relativ kurzen Programmstück, das sehr oft durchlaufen wird, verbrauchen. Dies gilt gleichermaßen für direkte Verfahren wie für Iterationsalgorithmen. Solche Programmstrecken muß man mit besonderer Sorgfalt programmieren, wofür nachfolgend einige Hinweise gegeben werden. Sie werden an FORTRAN-Beispielen demonstriert, gelten aber zu einem großen Teil sinngemäß auch für andere problemorientierte Sprachen. Für diese müßte der Katalog der Empfehlungen für rechenzeitoptimales Programmieren allerdings erweitert werden, da viele der dort erlaubten Freiheiten, die mit längeren Rechenzeiten bezahlt werden müssen, in FORTRAN nicht vorgesehen sind.