Computer im Bauingenieurwesen

In der Eingabe-Stellung (ENTER MODE) ist es sogar möglich, den einzugebenden Grössen diejenigen Namen zu geben, die man ihnen in der mathematischen Ableitung gegeben hat. Nehmen wir an, in unserem Beispiel hätte der Koeffizient A den Namen Φ und B den Namen R, dann könnte man die 0. Zeile folgendermassen beginnen: ENT «PHI», A, «R», B, C; usw. Auf RUN PROGRAMM erscheint PHI, man gibt den Wert dafür ein und nach einem RUN PROGRAMM erscheint R. A wird übersprungen, aber der eingegebene Wert wird trotzdem dem Register A zugewiesen. Diese Möglichkeit erspart in vielen Fällen weitere Kommentare zum Benutzen der Programme.

7. Schlussbemerkung

Mit dieser kurzen Darstellung dürfte die Anpassungsfähigkeit und leichte Programmierbarkeit der HP20 deutlich geworden sein. Diese Eigenschaften verdankt man unter anderem dem Umstand, dass jede Zeile einzeln bei den Befehlen EXECUTE oder STORE mit Hilfe eines Übersetzers (Compiler genannt) in die Maschinensprache übersetzt und für die Anzeige der gespeicherten Zeile wieder mit einem Uncompiler zurückübersetzt wird. Deswegen sind die quittierten Zeilen oft einfacher als die eingetasteten. Ein aufmerksamer Benützer kann daraus lernen, welche Abkürzungen erlaubt sind; z.B. kommt JMP (—5) als JMP —5 zurück, die Klammern sind hier also nicht nötig. Ferner erlaubt die Übersetzung jeder Zeile, dass einstellige Operationen (Funktionen: f(x), $\sqrt{x}$, sin x usw.) nicht nach der logischen Folge, zuerst x dann f, sondern nach der in der Mathematik üblichen Schreibweise f(x) (also z.B. SIN 30) eingegeben werden können.

Die Entwicklung der Sprache der HP20 bedeutet eine grosse Leistung der Konstrukteure. Dank dieser Sprache kann der Benützer seine Aufmerksamkeit dem Problem zuwenden und erspart so viel Zeit bei der Ausarbeitung von anspruchsvollen Programmen. Das Aneignen dieser Sprache ist einfach und bildet eine gute Vorbereitung zum Erlernen höherer Programmiersprachen wie Fortran, Algol usw.

Adresse des Verfassers: Prof. Dr. *H. Schilt*, 2502 Biel, Höheweg 5.

Computer im Bauingenieurwesen

Von Dr. **E. Anderheggen**[1]), Zürich

DK 681.3:624

Einführung

Seit der Einführung von Computern gegen Mitte der fünfziger Jahre wurde in vielen Gebieten des Bauingenieurwesens eine grosse Zahl neuer numerischer Verfahren entwickelt. Besonders spektakulär waren die Erfolge auf dem Gebiet der Computerstatik, da es innerhalb weniger Jahre gelang, die meisten Probleme der Statik und der Kontinuumsmechanik in einer erstaunlich einheitlichen und theoretisch befriedigenden Art zu lösen. Dabei erweckte vor allem die Methode der Finiten Elemente weltweit Begeisterung.

Seit einiger Zeit jedoch, wenn man die technische Fachliteratur durchschaut oder wenn man sich mit Leuten trifft, die seit Jahren auf dem Gebiet der Computerstatik arbeiten, bekommt man immer deutlicher den Eindruck, dass eine solche optimistische Einstellung nicht ganz am Platz ist. Die Stimmung hat sich irgendwie geändert. Nachdem so viele Computerprogramme geschrieben worden sind, merkt man heute, wenn man zurückblickt, wie wenig rationell man vorgegangen ist und wie oft die Ziele übersehen wurden, für die man eigentlich arbeitete. Viel zu oft betrachtete man nämlich den Computer nur als Instrument zur Lösung von selten vorkommenden, besonders schwierigen Aufgaben, so dass die Probleme übersehen wurden, welche sich aus dem nicht immer leichten Zusammenleben von Computer, Programmierer und Programmbenützer ergeben, und zwar vor allem dann, wenn der Computer als Werkzeug des projektierenden Ingenieurs für alltägliche Aufgaben eingesetzt werden soll.

In der Zukunft wird der Computer einen noch viel breiteren Kreis von Anwendungen finden, nicht zuletzt als Folge dieser drei wichtigen technologischen Entwicklungen:

Erstens das sogenannte «Time Sharing», womit es möglich ist, aus entfernten und einfachen Ein- und Ausgabegeräten (Konsolen) mit dem Computer eine Art Dialog zu führen, zweitens die erweiterten Möglichkeiten der graphischen Datenverarbeitung und drittens die sogenannten Computernetzwerke, bei denen eine Koppelung zwischen verschiedenen Grosscomputern verwirklicht wird.

Heute stehen wir folglich Rationalisierungsproblemen gegenüber, die im wesentlichen auch Wachstumsprobleme sind:

Zuerst das Problem der Programmqualität und der Programmdokumentation. Soll ein Programm einen breiten Kreis von Benützern finden, treten die Mann-Maschine-Kommunikationsprobleme ganz in den Vordergrund. Dafür ist aber oft wenig Verständnis vorhanden, nicht zuletzt, weil bei Handrechnungen sich solche Probleme gar nicht stellen: nur der eigentliche Algorithmus ist dabei wichtig. Unzählige Programme liegen deswegen in irgendwelchen Schubladen begraben, ohne die geringste Hoffnung, je wieder aufgegriffen zu werden.

Dann das Problem der ungenügenden praktischen Anwendbarkeit vieler Programme: Gegenüber den Mann-Maschine-Kommunikationsproblemen wurde den Algorithmen oft zu viel Aufmerksamkeit geschenkt, mit dem bedauerlichen Ergebnis, dass als Folge der Einführung von Computern Theorie und Praxis in manchen Fällen noch weiter auseinander geraten sind.

Ein weiteres Problem ist das der Duplikationen, und zwar sind hier in erster Linie nicht die Duplikationen ganzer Programme wichtig. Viel mehr ins Gewicht fällt, dass beim Programmieren oft solche Probleme am meisten Mühe bereiten, die unabhängig von der spezifischen Anwendung immer wieder vorkommen. Viel Arbeit könnte eingespart werden, wenn zwischen Programmierern aus verschiedenen Anwendungsgebieten eine bessere Koordination vorhanden wäre. Wir werden sehen, dass dies einer der Hauptgründe für die Entwicklung von «integrierten» Systemen gewesen ist.

Ein verwandtes Problem ist das der Maschinenabhängigkeit vieler Programme. Da die Programmentwickler die Erreichung einer maximalen Effizienz oft als eines ihrer Hauptziele betrachten, wird gern übersehen, dass die Programme einmal auf einem anderen Computer laufen sollen und dies, was heute zu einer oft unwesentlichen Verkürzung der Rechenzeit führt, morgen einen grossen und undankbaren Mehraufwand verursachen kann.

Im folgenden sei zuerst über zwei Projekte berichtet, die in diesem Zusammenhang wesentlich erscheinen.

Das ICES-System

Anfangs der sechziger Jahre wurden am Massachusetts Institute of Technology zwei historisch wichtige Programme geschrieben. Das Programm COGO für Vermessungsaufgaben und das Programm STRESS zur Berechnung von elasti-

[1]) Antrittsvorlesung an der ETH Zürich vom 4. Dezember 1972.

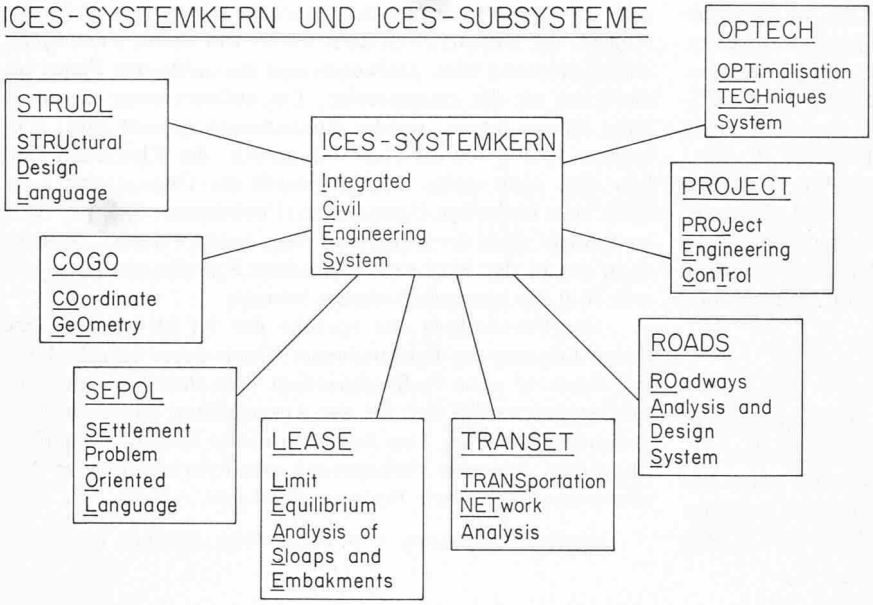

Bild 1. Die bekanntesten ICES-Subsysteme

schen Stabtragwerken. Obschon es sich um zwei grundverschiedene Programme handelt, haben sie folgende zwei gemeinsame Eigenschaften:

Erstens ist die Eingabe beider Programme so konzipiert, dass die Eingabeanweisungen möglichst an die technische Umgangssprache erinnern, welche Bauingenieure zur Beschreibung ihrer Probleme täglich verwenden. Dies wird als eine «problemorientierte Eingabesprache» bezeichnet.

Zweitens können beide Programme sowohl kleine Probleme, die keinen Sekundärspeicher benötigen, wie auch grosse Probleme mit sehr vielen Daten ohne wesentliche Effizienzverluste einheitlich behandeln. Dies verlangt eine komplizierte Organisation für die dynamische Speicherung von Daten und Programmteilen auf verschiedenen Speichermedien.

Diese beiden Eigenschaften wurden als notwendige Bedingungen betrachtet, um zu ermöglichen, dass die Programme zu alltäglich verwendeten Werkzeugen praktizierender Bauingenieure wurden.

Beim Schreiben von COGO und STRESS ergab sich, dass es bei derartig ausgesprochen praxisorientierten Programmen nicht wesentlich ist, ob die Koordinaten der Punkte eines Polygonzuges oder der Momentenverlauf in einem Rah-

```
STRUDL 'DYNAMISCHE  STAUDAMMBERECHNUNG'
TYPE  PLANE  STRAIN
JOINT  COORDINATES
1  X  60.  Y  45.
.
.
.
ELEMENT  INCIDENCES
1  9  32  33
.
.
.
ELEMENT  PROPERTIES
1 TO 180 TYPE 'CSTL' THICKNESS  1.0
CONSTANTS E 83100.  POISSON .35  ALL
DENSITY .075  ALL
.
.
.
DYNAMIC  LOADING 'EL CENTRO  ERDBEBEN  1940'
JOINT  1  TO  35  LOAD  X  FILE 'EL CENTRO'
INTEGRATE  FROM  0. TO  10. AT  .05
.
.
.
DYNAMIC  ANALYSIS  MODAL  15
LIST  DYNAMIC  EIGENVALUES  EIGENVECTORS  ALL
LIST  DYNAMIC  STRESSES  BY  TIME  ALL
FINISH
```

Bild 2. STRUDL: Problemorientierte Eingabeanweisungen

Die erste Anweisung beginnt mit dem Wort STRUDL und bewirkt, dass das STRUDL-Subsystem aufgerufen wird. Es folgt der Titel des Problems, welcher dann als Kopftitel bei der Ausgabe gedruckt wird. Dabei handelt es sich um die elastische dynamische Berechnung eines Staudammes für eine Erdbebenbelastung nach der Methode der Finiten Elemente.

Mit der Anweisung TYPE PLANE STRAIN wird dem Computer mitgeteilt, dass es sich um ein Scheibenproblem mit einem zweidimensionalen Dehnungszustand handelt. Die x- und y-Koordinaten aller Knotenpunkte der Elementmasche werden dann eingegeben sowie die «ELEMENT INCIDENCES», d.h. die globale Knotennummer aller Elemente.

Die Elementeigenschaften werden dann spezifiziert: Die Elemente mit den Nummern 1 bis 180 sind vom Typ «CSTL», einer der Elementtypen der STRUDL-Elementbibliothek, und haben eine einheitliche Dicke sowie die in den unteren Zeilen angegebenen Materialkonstanten: Elastizitätsmodul, Poissonscher Koeffizient und Dichte.

Der dynamische Lastfall wird dann angegeben. Sein Titel soll «EL CENTRO ERDBEBEN 1940» heissen; es ist dies ein in der kalifornischen Kleinstadt El Centro gemessenes Erdbeben, das oft als Testfall für dynamische Berechnungen verwendet wird. Die entsprechenden Knotenlasten in horizontaler Richtung x sollen von einem im System vorgespeicherten Datenfeld namens «EL CENTRO» abgelesen werden. Weiter soll die Zeitintegration von $t=0$ bis $t=10$ s in Intervallen von 1/20 s durchgeführt werden.

Mit der Anweisung «DYNAMIC ANALYSIS MODAL 15» wird verlangt, dass nur die ersten 15 Eigenschwingungen bei der entsprechenden dynamischen Analyse berücksichtigt werden. Es kommen dann Anweisungen, welche die gewünschte Ausgabe umschreiben: Alle 15 Eigenwerte, die Eigenschwingungsfrequenzen entsprechen, und alle 15 Eigenvektoren, die die Schwingungsformen beschreiben, sowie der zeitliche Verlauf aller Elementspannungen sollen gedruckt werden.

mentragwerk zu berechnen sind. Was vor allem einen grossen Programmieraufwand verlangt, sind nämlich die Programmteile, welche die problemorientierten Eingabeanweisungen interpretieren und sie auf syntaktische und logische Fehler prüfen, sowie die Programmteile für die dynamische Speicherorganisation.

Man überzeugte sich folglich, dass etwa dieselben Schwierigkeiten bei allen Arten von Computeranwendungen im Bauingenieurwesen vorkommen, sobald man dem Programmbenützer ein Maximum an Komfort bieten will. Und das nicht weil die Probleme ähnlich sind, sondern weil die Arbeitsweise von projektierenden Bauingenieuren ähnlich ist. Im Gegensatz zu den Grossbetrieben, z. B. der Maschinenindustrie, wo es Spezialisten gibt, die sich vom Morgen bis zum Abend nur mit Fragen der Computeranwendung befassen, arbeiten Bauingenieure meistens in kleinen Gruppen und werden ständig vor sehr verschiedenartige Aufgaben gestellt. Sie können folglich nicht gleichzeitig auch Computerfachleute sein, obwohl eigentlich sehr viele ihrer Probleme sich für eine automatisierte Computerbehandlung vorzüglich eignen. Sehr wichtig ist deswegen, dass ein übersichtliches und von Bauingenieuren leicht erlernbares Kommunikationssystem mit dem Computer vorhanden ist. Zudem treten die Probleme der Manipulation grosser Datenmengen bei Projektierungsarbeiten sehr in den Vordergrund: Projektieren bedeutet nämlich, die vielen Daten, die ein Projekt beschreiben, so lange zu manipulieren, bis die resultierende Variante zufriedenstellend ist.

Als die Firma IBM gegen Mitte der sechziger Jahre das neue Computersystem 360 einführte, entschloss man sich am Massachusetts Institute of Technology, ein allgemeines Software-System zu entwickeln, das derartige Probleme ein für alle Male auf möglichst allgemeine und effiziente Art lösen würde. Man gründete das Civil Engineering System Laboratory, das von Computerfachleuten verschiedener Institute der Abteilung für Bauingenieurwesen bemannt ist, und man begann mit der Entwicklung des heute berühmten ICES-Systems, wobei ICES für «Integrated Civil Engineering System» steht. Mehrere Jahre Arbeit und ein Aufwand von über drei Mio Dollar wurden zu dieser Entwicklung benötigt [1].

Es muss klar unterschieden werden zwischen den ICES-Subsystemen und dem eigentlichen ICES-Systemkern. Die ICES-Subsysteme sind Anwendungsprogramme aus verschiedenen Gebieten des Bauingenieurwesens. Computerbenützer haben immer nur mit Subsystemen zu tun. Der ICES-Systemkern hingegen stellt im wesentlichen eine Erweiterung des Betriebssystems des Computers dar.

Die bekanntesten Subsysteme heissen (vgl. Bild 1):

– STRUDL	Eine erweiterte Version des STRESS- Programmes für die Berechnung und die Bemessung beliebiger Tragwerke
– COGO	Ebenfalls eine erweiterte Version des ursprünglichen COGO-Programmes für Vermessungsprobleme
– SEPOL	Zur Berechnung von Bodensetzungen
– LEASE	Zur Bestimmung der Stabilität von Böschungen und Dämmen
– TRANSET	Für Netzwerkprobleme im Verkehrswesen
– ROADS	Für Strassenbauprobleme
– PROJECT	Für die Planung und die laufende Kontrolle des Baubetriebes nach der Methode des kritischen Weges
– OPTECH	Für allgemeine Optimalisierungsaufgaben.

Alle ICES-Subsysteme haben eine eigene problemorientierte Eingabesprache. Das Beispiel von Bild 2 zeigt, wie

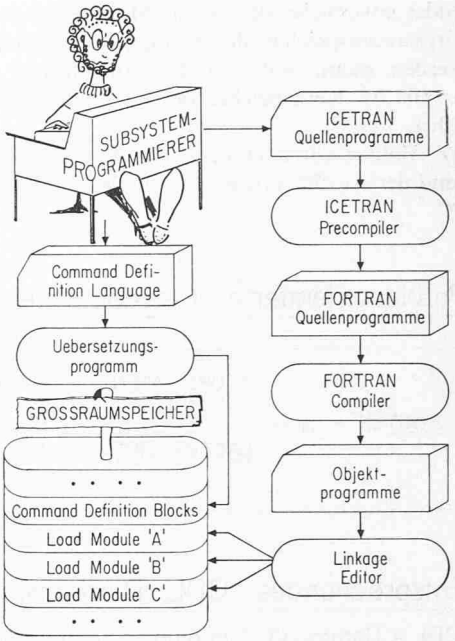

Bild 3. ICES, Subsystemprogrammierung

übersichtlich, leicht zu verstehen und folglich bequem anzuwenden eine solche problemorientierte Eingabesprache ist.

Das Konzept von ICES wird vielleicht am besten verstanden, wenn wir zuerst die Schritte verfolgen, die zur Entwicklung eines neuen Subsystems notwendig sind (Bild 3).

Zuerst muss der Subsystem-Programmierer seine eigene problemorientierte Eingabesprache in allen ihren syntaktischen Einzelheiten festlegen. Diese soll dann sowohl für die Eingabe von Problemdaten wie auch zur Steuerung des Programmablaufes dienen. Dazu verwendet er eine besondere Programmiersprache, die sogenannte «Command Definition Language». Das entsprechende ICES-Übersetzungsprogramm liest diese Sprache und erzeugt sogenannte «Command Definition Blocks», die dann als Daten in den Grossraumspeicher des Computers gebracht werden. Diese enthalten alle Informationen, die später zur Interpretation der problemorientierten Eingabeanweisungen notwendig sind. Ein Beispiel für die «Command Definition Language» ist in Bild 4 angegeben.

Der Subsystem-Programmierer muss dann die eigentlichen Unterprogramme seines Subsystems schreiben. Dazu verwendet er eine zweite besondere Programmiersprache, die rechnungsorientierte ICETRAN-Sprache. Diese entspricht FORTRAN, besitzt aber zusätzliche FORTRAN-ähnliche Befehle, die in erster Linie zur Steuerung der dynamischen Speicherorganisation dienen.

Die ICETRAN-Anweisungen werden von einem zweiten ICES-Übersetzungsprogramm, dem sogenannten ICETRAN-Precompiler, gelesen und interpretiert. Der ICETRAN-Precanpiler erzeugt nicht ein Objektprogramm, sondern ein normales FORTRAN-Quellenpropramm, indem spezielle ICETRAN-Anweisungen mit Aufrufen von besonderen ICES-Unterprogrammen ersetzt werden, währenddem Anweisungen, die sich von FORTRAN nicht unterscheiden, einfach ohne Änderungen weitergegeben werden. Das so erzeugte FORTRAN-Programm kann dann mit einem gewöhnlichen FORTRAN-Compiler übersetzt werden. Bild 5 zeigt als Beispiel einige Anweisungen der ICETRAN-Programmiersprache.

Ein weiteres ICES-Programm, der sogenannte «Linkage Editor», fasst verschiedene Unterprogramme zusammen und

bildet entsprechende «Load Modules», die ebenfalls in den Grossraumspeicher des Computers gebracht werden. Diese werden dann, während der Ausführung des Subsystemes, jeweils im Kernspeicher des Computers geladen und ausgeführt.

Folgen wir aber noch der Arbeitsweise von ICES während der Ausführung eines Subsystems (Bild 6).

Problemorientierte Eingabeanweisung:

Entsprechendes CDL-Programm (CDL = Command Definition Language):

```
REPLACE   'KNO'
NO  ID   INTEGER  'KNUMMER' REQUIRED
    MODIFIER  'KOO'
       ID  'X' REAL  'XKOORD '  STANDARD  0.
       ID  'Y' REAL  'YKOORD '  STANDARD  0.
       ID  'Z' REAL  'ZKOORD '  STANDARD  0.
            EXECUTE  'STKOORD'
    OR MODIFIER  'LAS'
       IGNORE  'KRA'
       EXISTENCE  'X' 'Y' 'Z'  SET 'M' STANDARD 1
       NO  ID  REAL 'WERT 'REQUIRED
            EXECUTE  'STLAST'
    END  MODIFIER
FILE
```

Bild 4. Eingabeanweisungen und CDL-Programm

Im oberen Rahmen wird die syntaktische Struktur einer möglichen Eingabeanweisung schematisch beschrieben. Im Laufe der Eingabe kann also eine Anweisung vorkommen, die mit dem Wort «KNOTEN» beginnt, wobei nur die drei unterzeichneten Buchstaben KNO relevant sind. Diesem Wort muss eine ganzzahlige Zahl *i* folgen, die die Knotennummer angibt. Es kann dann eines der beiden Worte «KOORDINATEN» oder «LASTEN» spezifiziert werden, wobei wieder nur die ersten drei Buchstaben relevant sind.

Wird das Wort «KOORDINATEN» spezifiziert, sollen die *x*-, *y*- und *z*-Koordinaten der entsprechenden Knoten angegeben werden, wobei jeweils die Buchstaben *x*, *y* oder *z*, gefolgt von einer reellen Zahl *r*, geschrieben werden müssen. Diese Eingabe ist jedoch fakultativ und wird deswegen in runden Klammern dargestellt: Der Standardwert Null soll vom Computer angenommen werden, wenn eine Koordinate nicht angegeben wird.

Wird das Wort «LASTEN» spezifiziert, kann es vom Wort «KRAFT» gefolgt sein. Dies ist jedoch nicht obligatorisch und wird nur zur Verbesserung der Übersichtlichkeit der Eingabe verwendet. Es muss dann die Lastrichtung *x*, *y* oder *z* spezifiziert werden, gefolgt von einer reellen Zahl *r*, welche die Lastintensität angibt. Wird keine Lastrichtung spezifiziert, nimmt der Computer die in Klammern dargestellte Richtung *x* an.

Damit wurde die syntaktische Struktur dieser problemorientierten Anweisung beschrieben. Dasselbe tut das im unteren Rahmen dargestellte CDL-Programm. Auf eine Besprechung der Einzelheiten kann in diesem Rahmen verzichtet werden. Mit der Command Definition Language wurde ein allgemein verwendbares Hilfsmittel zur Festlegung von kompliziert strukturierten Eingabeanweisungen geschaffen.

Der Subsystembenützer im Bild 6 links teilt auf interaktive Weise aus einer Time-Sharing-Konsole oder im Batch-Betrieb mit Hilfe von Lochkarten seine Anweisungen dem Computer mit.

Im Grossraumspeicher rechts im Bild sind die verschiedenen Hilfsprogramme des ICES-Systems gespeichert sowie auch die früher erwähnten «Command Definition Blocks» und die «Load Modules» des Subsystems.

```
SUBROUTINE   HANS
COMMON   A, B, C, D, VAR (P)
.
.
.
DEFINE   VAR, 100, DOUBLE, LOW, STEP=50
.
.
.
VAR (170) = 77.7
.
.
.
LINK   'FRITZ'
.
.
.
DESTROY   VAR
.
.
END
```

Bild 5. ICETRAN-Anweisungen

Das Beispiel gibt einige typische Anweisungen der ICETRAN-Programmiersprache. Es handelt sich um ein Unterprogramm namens HANS, das so beginnt wie eine gewöhnliche FORTRAN-Subroutine.

Es folgt eine COMMON-Anweisung, die äusserlich ähnlich wie die entsprechende FORTRAN-Anweisung aussieht. Ein Datenfeld für die indizierte Variable VAR wird dabei deklariert: VAR ist ein sogenanntes dynamisches Datenfeld, dessen Grösse im Laufe der Berechnung geändert werden kann. Der Buchstabe P steht für «POINTER» und deutet die besondere Speicherungsart dieses Datenfeldes an.

Die ICETRAN-Anweisung DEFINE enthält weitere Angaben über das Datenfeld VAR: Anfänglich soll es 100 Elemente umfassen, wobei jedes Element zwei Computerworte beansprucht. Das Feld hat zudem eine tiefe Priorität (Low Priority), so dass es vom Kernspeicher vorübergehend entfernt werden kann, falls Speicherplatz benötigt wird.

STEP=50 bedeutet, dass das Datenfeld VAR in Schritte von jeweils 50 Elemente sich vergrössern soll. Wenn z.B. eine Anweisung wie VAR (170) = 77.7 zur Ausführung kommt, wird die Speicherplatzreservation für das Feld VAR um 2 Schritte von je 50 Doppelwörtern, d.h. von den ursprünglichen 100 auf 200 Doppelworte, automatisch vergrössert.

In der 5. Zeile steht die ICETRAN-Anweisung LINK«FRITZ», wobei FRITZ der Name eines Unterprogrammes in einem bestimmten «Load Module» ist. Diese Anweisung bewirkt, dass das «Load Module», welches das Unterprogramm FRITZ enthält, im Kernspeicher des Computers geladen wird, falls es sich dort nicht schon aufhält. Wenn für die Vergrösserung eines dynamischen Datenfeldes oder für die Ladung eines «Load Modules» der Speicherplatz nicht ausreicht, findet automatisch eine Speicherreorganisation statt, bei der alle gegenwärtig nicht gebrauchten Datenfelder und Programmteile aus dem Kernspeicher entfernt werden.

Die Anweisung «DESTROY VAR» deutet darauf, dass das Datenfeld VAR nicht mehr gebraucht wird, so dass der für ihn reservierte Speicherplatz für andere Datenfelder und Programmteile freigegeben wird.

Im wesentlichen muss also der Subsystemprogrammierer nur jeweils angeben, welche Daten und welche Programme gerade gebraucht werden. ICES übernimmt dann automatisch die entsprechende dynamische Speicherorganisation.

Ein Benützer kann dann eigene Privatdaten im Grossraumspeicher behalten, wenn er diese Daten in verschiedenen unabhängigen Programmläufen verwenden will. Zudem gibt es Datenfelder, die zum Subsystem selber gehören: Im STRUDL sind z.B. die Stahlbautabellen auf diese Art dauernd gespeichert. Überlaufdaten aus dem Kernspeicher werden automatisch in den Grossraumspeicher gebracht, wenn bei einer Speicherorganisation gewisse Datenfelder mit tiefer Priorität, die aber später noch gebraucht werden, aus dem Kernspeicher temporär entfernt werden sollen.

Betrachten wir aber noch, was im Kernspeicher des Computers während der Ausführung eines Subsystems geschieht. Ein kurzes ICES-Exekutivprogramm bleibt ständig im Kernspeicher zur Steuerung der verschiedenen Operationen. Ein weiteres ICES-Programm, der sogenannte «Command-Interpreter», dient zur Interpretation der problemorientierten Eingabeanweisungen. Dafür wird jeweils eines der früher erwähnten «Command-Definition-Blocks» benötigt, bei dem die syntaktische Struktur der erlaubten Eingabeanweisungen festgelegt ist. Als Folge der Eingabeanweisungen werden entweder Daten gelesen und gespeichert, oder es werden «Load Modules» vom Grossraumspeicher geladen und ausgeführt.

Im Kernspeicher befindet sich somit eine Mischung von Daten- und Programmsegmenten. Es ist Aufgabe des ICES-Systems zu kontrollieren, dass keine Überlappung vorkommt und dass die Grenzen des zur Verfügung stehenden Speicherplatzes nicht überschritten werden.

Zwischen- und Endresultate werden je nach Subsystem und immer aufgrund entsprechender Eingabeanweisungen in alphanumerischer oder auch in graphischer Form auf verschiedene Ausgabegeräte oder auf Time-Sharing-Konsolen ausgegeben.

ICES ist heute weltweit auf über 400 IBM-Anlagen installiert, wobei aber vor allem die Subsysteme und in allererster Linie STRUDL verwendet werden.

ICES war das erste, ist aber heute nicht das einzige derartige System für Anwendungen im Bauingenieurwesen. In den USA wurden an den Universitäten Pennsylvania und Illinois ähnliche Systeme entwickelt. In Europa bestehen heute das belgische System SYSFAP, das englische System GENESIS und das deutsche System ISB, die alle ebenfalls an Universitäten entwickelt worden sind.

Alle diese integrierten Systeme sind ähnlich wie ICES gebaut. Im wesentlichen versuchen sie also, die am Anfang erwähnten Rationalisierungsprobleme durch eine geeignete Erweiterung des Betriebssystemes einzelner Computer zu lösen. Mit anderen Worten, sie versuchen software-mässig den Rahmen zu bilden, in dem praxisorientierte Anwendungsprogramme hoher Qualität auf verhältnismässig einfache Art entwickelt werden können.

Das neue amerikanische Projekt

Zurzeit wird in den USA noch ein weiteres Grossprojekt vorgeschlagen und diskutiert, das im wesentlichen die gleichen Rationalisierungsprobleme, jedoch auf globaler Ebene, zu lösen versucht.

Eine Gruppe von amerikanischen Professoren unter der Leitung von Professor *Fenves*, ein bekannter Name auf diesem Gebiet, schlägt vor, ein nationales Institut zu gründen mit der Aufgabe, Koordination zu erzielen zwischen allen Stellen, die mit der Entwicklung und mit der Verwendung von Computerprogrammen im Bauingenieurwesen zu tun haben [2]. Als Grund dafür wird die heutige, mit den Worten von Fenves «chaotische» Lage angegeben, mit den daraus sich ergebenden grossen Arbeits- und Geldverschwendungen. «Chaotische Lage» im wesentlichen als Folge der am Anfang des Aufsatzes angegebenen Gründe.

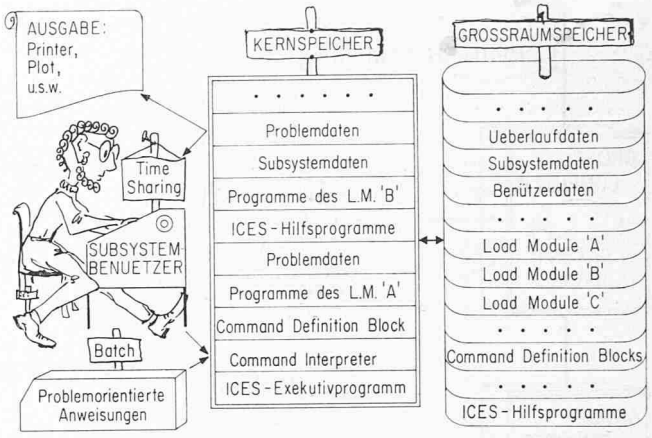

Bild 6. Arbeitsweise von ICES

Zentrale Aufgabe des neuen Institutes wäre der Aufbau und die Betreuung einer nationalen Programm-Bibliothek für alle möglichen Anwendungen des Bauingenieurwesens. Dabei würde man aber ganz anders vorgehen, als es heute bei Programm-Bibliotheken der Fall ist. Man würde nämlich zuerst systematisch die Probleme erforschen, die sich bei dem praktischen Einsatz von Computerprogrammen sowie bei der Implementation derselben auf verschiedenartige Computersysteme ergeben. Dabei spricht man von «Software-Engineering», ein neuer Ausdruck, der für diese sehr neuartige Wissenschaft erfunden wurde. Als Resultat dieser Forschung würden dann Normen und Richtlinien erarbeitet, an die sich die in der Bibliothek aufgenommenen Programme halten müssten.

Zum Beispiel müssten die Programme so aufgebaut werden, dass die Ein- und Ausgabe, die Datenorganisation und der eigentliche Algorithmus vollkommen voneinander getrennt werden können. Ein- und Ausgabe sowie Datenorganisation wären weitgehend standardisiert, so dass der Programmentwickler sich vor allem mit dem anwendungsabhängigen Algorithmus zu befassen hätte, währenddem die Spezialisten des Institutes Fragen der Implementation und der Wartung der Bibliothekprogramme behandeln würden.

Zudem sollten Datenbänke aufgebaut werden, die allgemein nützliche Daten enthalten, wie z.B. die Spezifikationen von vorfabrizierten Bauelementen oder die Eigenschaften und die Preise von Baumaterialien oder die zeitlich variablen Kostenindizes für bestimmte Bauarbeiten.

Das Institut hätte sich dann auch mit der Dokumentation von Programmen und Datenbänken zu befassen sowie mit der Verteilung derselben überall in den Vereinigten Staaten. Dafür würden Computer-Netzwerke eingesetzt, bei denen mehrere Grosscomputer, selbst in grossen Abständen, miteinander gekoppelt werden.

Bild 7 zeigt die Hardware-Konfiguration, die zur Verwirklichung des Projektes notwendig wäre, die aber auch abgesehen davon der zukünftigen Entwicklung der Computertechnologie etwa entsprechen dürfte.

Links ist das Computernetzwerk abgebildet, bei dem eine Reihe von Grosscomputern, möglicherweise von verschiedenen Fabrikaten, miteinander gekoppelt sind. Dabei soll jeder dieser Computer, auf Sekundärspeicher in direktem Zugriff, eine grosse Speicherkapazität aufweisen.

Rechts im Bild sind die mit dem Computernetzwerk in Verbindung stehenden Ingenieurbüros angedeutet. Dabei würden selbst die kleinsten Ingenieurbüros mindestens über eine Time-Sharing-Konsole verfügen, die durch gewöhnliche Telephonwahlleitungen an einen Grosscomputer angeschlossen wird. Es bestehen verschiedene Arten von Time-Sharing-

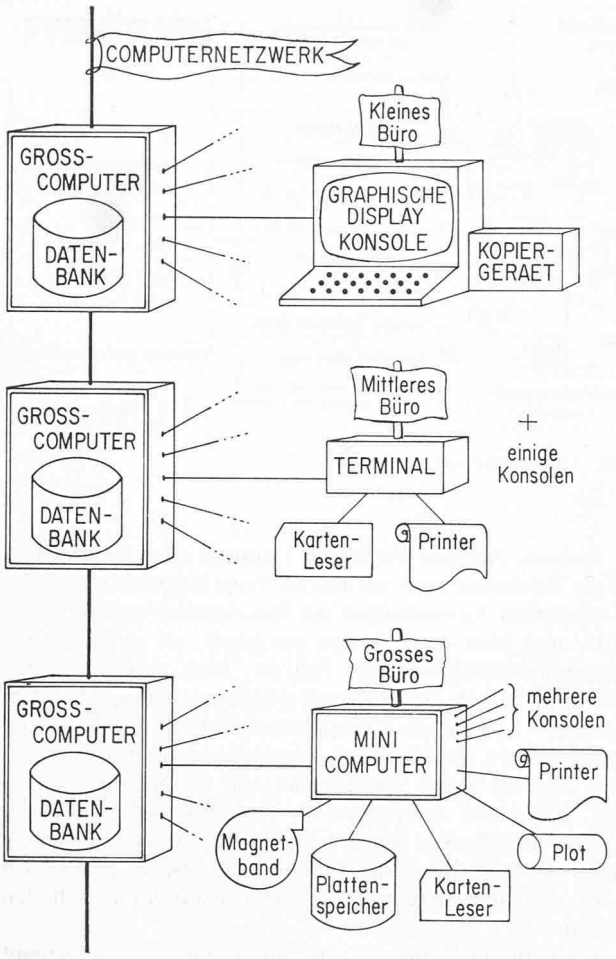

Bild 7. Computernetzwerk und Ingenieurbüros

Konsolen: Für Anwendungen im Bauingenieurwesen die sinnvollste Art wird ein Bildschirm sein, auf welchem Informationen sowohl in alphanumerischer wie auch in graphischer Form ausgegeben werden können, mit einer Tastatur für die Dateneingabe. Daneben steht ein einfaches Kopiergerät, womit der Bildschirminhalt auf Papier kopiert werden kann.

Werden geeignete interaktive Programme entwickelt, wäre es damit denkbar, die Verfassung von Berichten, z. B. bei der Projektierung und bei der Berechnung von Brücken und Hochbauten, so zu automatisieren, dass der projektierende Ingenieur praktisch nichts mehr selber zu schreiben hätte. Er würde vor der Konsole sitzen und könnte in konversationeller Arbeitsweise, zus mmen mit dem Computer, verschiedene Varianten seines Projektes analysieren. Wenn die auf dem Bildschirm erscheinenden Daten relevant sind, würde er einfach auf den Knopf des Kopiergerätes drücken. Ein Bericht würde somit entstehen, bei dem alle wesentlichen Schritte der Projektierung und des Sicherheitsnachweises angegeben sind.

Grössere Büros würden dann, neben einigen derartigen Konsolen, auch noch ein mehr oder weniger leistungsfähiges «Terminal» besitzen. Die Ausgabe von Resultaten könnte damit auf den Zeilendrucker umgeleitet werden, falls die Geschwindigkeit der Time-Sharing-Konsolen bei grossen Mengen von Ausgabedaten nicht mehr ausreicht. Zudem wäre es damit möglich, mit Lochkarten zu arbeiten, was in vielen Fällen immer noch sehr angenehm ist.

Noch grössere Büros könnten sich dann einen sogenannten Minicomputer leisten. Heute schon besteht eine ganze Reihe solcher verhältnismässig billiger und dennoch leistungsfähiger Kleincomputer, die sowohl selbständig wie auch als Satelliten eines Gros computers verwendet werden können. Da viele periphere Geräte an einen derartigen Minicomputer angeschlossen werden dürfen, könnte das Ingenieurbüro praktisch alle Möglichkeiten eines ausgebauten Rechenzentrums bei sich installiert haben.

Damit könnte jeder Bauingenieur von seinem Büro aus die Dokumentation betreffend Bibliothekprogramme und Datenbänke direkt über den Computer erhalten. Diese Dokumentation würde Programm- und Eingabebeschreibung umfassen sowie auch andere Informationen, z. B. über Rechenkosten und über Programmverwendungsgebühren.

Er könnte dann die Bibliothekprogramme selber verwenden und die vom Institut aufgestellten Datenbänke abfragen. Er könnte aber auch eigene Privatdaten im Grossraumspeicher eines Computers permanent behalten. Damit würde die Möglichkeit bestehen, dass verschiedene Benützer desselben Computernetzwerkes miteinander kommunizieren, indem die gleichen Daten verschiedenen Benützern zugänglich gemacht werden. Zum Beispiel könnte der Austausch von Informationen zwischen den projektierenden Ingenieuren und den Bauunternehmern sowie zwischen diesen und den Materiallieferanten über Computer direkt erfolgen. Eine wesentliche Rationalisierung vieler heute sehr zeitraubender Arbeiten wäre dadurch zu erreichen.

Ob, wann und in welcher Form das besprochene Projekt verwirklicht wird, ist vorläufig noch ungewiss. Abgesehen davon ist es aber nur eine Frage der Zeit, bis es gelingt, viele im Bauingenieurwesen täglich vorkommende Arbeiten zu automatisieren und damit die Arbeitsweise von Bauingenieuren wirklich zu revolutionieren.

An der Abteilung für Bauingenieurwesen der ETH

Zum Schluss sollen ein paar Punkte betreffend die Zukunft unserer Abteilung II für Bauingenieurwesen erwähnt werden, wobei heute natürlich die Planungsfragen im Zusammenhang mit der Versetzung der Abteilung auf den Hönggerberg im Vordergrund stehen.

Zweifellos zählt heute die ETH in bezug auf Computer zu den bestausgerüsteten Hochschulen der Welt. Trotzdem haben wir aber unerfüllte Wünsche, und ich möchte hier vor allem zwei erwähnen.

Erstens fehlt ein vollkommen interaktives Time-Sharing-System. Dabei ist zu bemerken, dass Time-Sharing trotz vielen Versprechungen immer noch gegen grosse Schwierigkeiten stösst und dass es ausserdem für reine ETH-interne Lehr- und Forschungsbedürfnisse eigentlich gar nicht so notwendig ist. Wenn wir es aber als unsere Aufgabe betrachten, mit den beschriebenen Entwicklungen Schritt zu halten, dann sollten wir auch an der ETH in der Lage sein, Programme zu schreiben, welche dann in der Praxis eine interaktive Arbeitsweise ermöglichen.

Zweitens fehlen an der ETH einfache Display-Konsolen, die eine graphische Ausgabe von Daten in Form von Kurven oder Diagrammen erlauben. Das ist um so bedauernswerter, als dieser Wunsch im Gegensatz zum Time-Sharing heute verhältnismässig leicht zu erfüllen ist. Diese Kriterien sollten berücksichtigt werden, wenn über die Planung der Zentralstelle für Datenverarbeitung im neuen Lehrgebäude auf dem Hönggerberg die Rede sein wird.

Als zweites folgen einige Überlegungen zur computerunterstützten Forschung innerhalb der Abteilung II. Es wurde hier versucht, etwas Licht in den manchmal dunklen Fragenkomplex zu werfen, der sich ergibt, wenn Computermethoden tatsächlich in die Praxis gelangen sollen. Wir haben weiter versucht, anhand dieser ausländischen Grossprojekte zu zeigen, wieviel Aufmerksamkeit solchen Fragen in der ganzen Welt geschenkt wird.

Die nähere Betrachtung der Abteilung für Bauingenieure führt unmittelbar zur Frage der Koordination zwischen allen Instituten, die mit der computerunterstützten Forschung arbeiten, da diese überall ähnlichen Problemen gegenüberstehen.

Wäre es beispielsweise sinnvoll, auch bei uns über ein integriertes System wie ICES zu verfügen, sei es, dass wir eines entwickeln oder dass wir ein schon vorhandenes System übernehmen und erweitern, so wie es mehrere ausländische Universitäten getan haben? Zumindest würde dieses eine gewisse Koordination zwischen System und Subsystem-Programmierer erzwingen.

Oder soll nicht eine Programm-Bibliothek auf Abteilungsebene aufgestellt werden, bei der die neu aufgenommenen Programme und ihre Dokumentation streng auf Qualität hin geprüft werden? Sicherlich würde die Praxis, die oft ungenügenden Zugang zu den an der ETH entwickelten Programmen findet, eine solche Programm-Bibliothek sehr begrüssen.

Es ist nicht die Absicht, hier konkrete Vorschläge in dieser Richtung anzubringen. Ein Punkt sei aber doch noch erwähnt. Damit zukünftige Bauingenieure alle Vorteile der modernen Computermethoden wirklich ausnützen können, müssen sie auch entsprechend ausgebildet werden. Während des Diplomstudiums haben alle Bauingenieurstudenten eine einsemestrige Vorlesung über Computer und Programmierung zu besuchen. Diese genügt aber offensichtlich nicht, falls einer später in der Praxis mit solchen Problemen wirklich in Berührung kommt. Hinsichtlich der Computeranwendungen ist deswegen die Einführung eines gut geregelten und entsprechend anerkannten Nachdiplomstudiums heutzutage eine dringende Notwendigkeit. Praxis und Forschung würden davon sehr viel profitieren.

Literaturverzeichnis

[1] *D. Roos:* ICES System Design. MIT-Press, 1966 Cambridge, Massachusetts.

[2] *S.J. Fenves:* Scenario for a Third Computer Revolution in Structural Engineering. Proceedings of the 5th Conference on Electronic Computation, ASCE, 1970, S. 829–845.

Adresse des Verfassers: Dr. sc. techn. *Edoardo Anderheggen,* Privatdozent an der ETHZ, Institut für Baustatik und Massivbau, Winterthurerstrasse 28, 8006 Zürich.

Vereinfachte Längenberechnung für Erdanker

DK 624.137.001.2

I. Grundgedanke und Methode

Von **Felix Adler,** Zürich

Es wird gezeigt, dass sich bei Annahme ebener Gleitflächen und der Eingabe von nur höchstens zwölf Werten die Ankerlängenberechnung einfach und unter Zuhilfenahme von Computern sehr rasch bewerkstelligen lässt. Die geometrischen Vereinfachungen erscheinen insofern vertretbar, als es sich bei den meisten Eingabegrössen (den Bodenkennwerten) lediglich um mehr oder weniger gut gewählte Annahmen handelt. Mit den abgeleiteten Formeln können entweder bei vorgegebener Ankerlänge die vorhandene Sicherheit berechnet oder bei eingegebenen Sicherheitskoeffizienten die minimale Ankerlänge (auf Grund einer Optimierung) ermittelt werden.

1. Einleitung

Über Erdanker liegen zahlreiche Veröffentlichungen vor; es sei hier lediglich auf den in der «Schweiz. Bauzeitung» (*83.* Jahrgang, Heft 35 vom 2. 9. 1965) erschienenen Aufsatz von Dr. *J. Huder* und das dort angeführte Literaturverzeichnis hingewiesen. Für das Teilproblem der Stabilitätsberechnung zur Ermittlung der notwendigen Ankerlängen sind dort mit Bild 16 drei Methoden dargelegt.

Wenn mit vorliegender Arbeit versucht wird, eine Weiterbearbeitung dieser Methoden (unter Mitverwendung derjenigen von Prof. Dr. *O. K. Fröhlich*) darzustellen, so entspringt dies der Erfahrung, dass mit der immer häufiger werdenden Ausführung tiefer Baugruben mit Erdankern die Berechnung der erforderlichen Ankerlängen so an Umfang zugenommen hat und so zeitraubend wurde, dass sich eine Programmierung und elektronische Berechnung aufdrängte.

Es war also ein Verfahren zu suchen, das computergerecht formuliert werden kann und bei einer Vielzahl von Ankerlagen und Spundwandschnitten einfach einzugeben ist und nur kurze Rechenzeiten erfordert.

2. Methode

Mit den Beziehungen, die in Bild 1 ersichtlich sind, und der Definition der Sicherheit:

$$(1) \quad F = \sum P_{passiv} / \sum P_{aktiv} \qquad \text{folgt:}$$

$$(2) \quad \sum P_{passiv} = R_\varphi + E_p \cos\alpha = (G\cos\alpha + E_p \sin\alpha) \cdot \tan\varphi + E_p \cos\alpha$$

$$(3) \quad \sum P_{aktiv} = G \sin\alpha$$

Setzt man (2) und (3) in (1) ein, ergibt sich nach entsprechender Zusammenfassung und Kürzen die einfache Beziehung:

$$(4) \quad F = \tan\varphi \cot\alpha + (\tan\varphi + \cot\alpha)\, E_p / G$$

Nach *Kranz* liegt der Ansatzpunkt der tiefen Gleitfuge bei Einspannung der Spundwand im Querkraftnullpunkt, bei freier Auflagerung im Fusspunkt der Wand. Das für die Ankerlängenbestimmung erforderliche Mass T ist demnach der vorgängig erfolgten Spundwandberechnung zu entnehmen oder allenfalls zu schätzen.

Als weitere Festwerte sind ausser den Bodenkennwerten φ, γ und λ_p noch die Höhen H und A sowie die Anker-

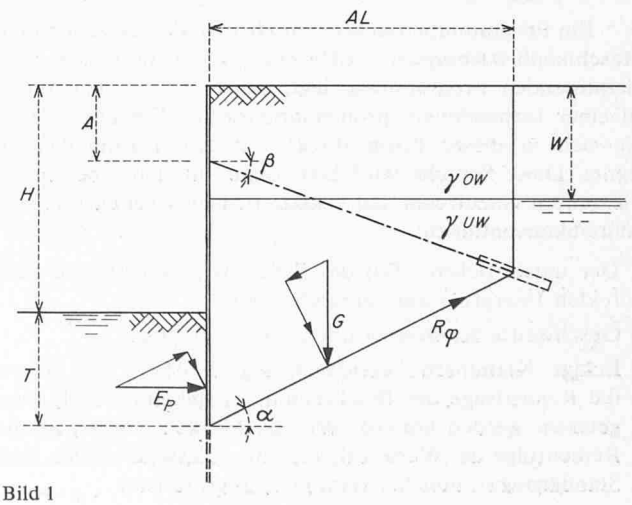

Bild 1