CALCULUS

S. M. Rump
Institut für Informatik III
Technische Universität Hamburg-Harburg

Zusammenfassung

Das im folgenden vorgestellte System CALCULUS ist ein interaktives Programmierwerkzeug für numerisches Rechnen, dessen Hauptaufgabe es ist, dem Benutzer eine möglichst einfache, natürliche Notation für die Entwicklung von Numerik-Software zur Verfügung zu stellen. Die Zielsetzung dabei ist, daß ein Arbeiten mit dem System CALCULUS nahezu dem Entwerfen auf dem Papier gleichkommt.

1 Einleitung

Vor der Einführung in das System CALCULUS noch einige Grundüberlegungen zum Aufgabenbereich von Programmiersprachen. Bei der Entwicklung von Software kann man unterscheiden die Entwicklung von Algorithmen und die Entwicklung von Programmen. Die wichtigsten Anforderungen, die an ein entsprechendes Entwicklungswerkzeug gestellt werden sind für Algorithmen

- einfache Notation
- schnelle Antwortzeit
- „Ein-Knopf"-Operationen.

Für Programme sind dies hingegen

- schneller Code
- speichereffizienter Code
- Bedienerfreundlichkeit (im Hintergrund).

Das System CALCULUS soll nun hauptsächlich der Entwicklung von Algorithmen dienen und dies in möglichst natürlicher Form; es soll mathematische Notation auf dem Rechner ermöglichen. Die Ausführung der entwickelten Algorithmen soll außerdem möglichst auf Knopfdruck, also ohne zusätzliche Compilierungszeiten, erfolgen können.

2 Aufbau des Systems

Das Programm CALCULUS entstand im IBM-Forschungslabor Böblingen in Zusammenarbeit mit H. Böhm und D. Husung. Letzterer besorgte die komplette Implementierung. Das Werkzeug ist interaktiv und besitzt einen

interpretativen Teil, der vollständig in Pascal geschrieben ist und einen Zwischencode erzeugt, einen

Numerik-Teil, der neben den herkömmlichen numerischen Verfahren auch Einschließungsverfahren bereitstellt und in FORTRAN implementiert ist, und einen

Bibliotheksteil, der die Programmbibliotheken LINPACK, EISPACK und ACRITH benutzt.

CALCULUS ist auf allen IBM /370-Anlagen lauffähig. Das System stellt die erste Version der Neuentwicklung ABACUS dar, die mittlerweile in Hamburg gestartet wurde. Diese neue Version läuft unter DOS/UNIX und damit auf vielen gängigen PC's und *Work-Stations*. ABACUS wird Anfang 1989 fertiggestellt sein.

3 Vergleich von CALCULUS mit herkömmlichen Programmiersprachen

Ein ganz wesentliches Merkmal herkömmlicher Programmiersprachen bzw. Programmiersysteme ist die Redundanz. Jedes noch so kleine Programmstück enthält sehr viel redundante Information, da Programmiersprachen wie FORTRAN, PASCAL oder C zur *Programm*entwicklung geschrieben wurden und diese zusätzlichen Informationen zur Steigerung der Effizienz benötigt werden. An einem einfachen Beispiel soll der Unterschied zwischen CALCULUS und anderen Systemen verdeutlicht werden.

Will man den Ausdruck $3 * 5 + 7$ auf dem Rechner auswerten, so muß in einem FORTRAN-Programm

```
WRITE(6,*) 3*5+7
END
```

geschrieben werden, wobei schon alles weggelassen ist, was weggelassen werden kann. Trotzdem stecken in diesem kurzen Programmstück noch redundante Informationen, denn wenn man diesen Ausdruck berechnet ohne das

Ergebnis einer Variablen zuzuweisen, so ist klar, daß das Ergebnis ausgegeben werden soll und danach die Aufgabe beendet ist. Das bedeutet, die **WRITE**-Anweisung und **END** sind überflüssig.
Programmiert man das ganze in PASCAL, so muß man sogar schreiben

```
PROGRAM dummy;
  BEGIN
    writeln(3*5+7)
  END.
```

Man hat also noch mehr redundante Informationen im Programmstück. Selbst in BASIC kommt man nicht ohne das PRINT-Statement aus, es gibt also auch dort noch eine gewisse Redundanz.
In CALCULUS hingegen ist es möglich, einfach einzugeben

```
3*5+7
```

und die *ENTER*-Taste zu drücken; das Ergebnis wird berechnet und ausgedruckt. Dies zeigt schon die Arbeitsweise von CALCULUS; man kann einfach an den Rechner herangehen, einen Algorithmus eingeben und erhält das Ergebnis sofort.

4 Weitere Anwendungsbeispiele

Im folgenden wird die Eingabezeile immer mit „==>“, das Ende der Ausgabe durch „<>“ gekennzeichnet.

CALCULUS erlaubt die Eingabe von Daten und Anweisungen in mathematischer Schreibweise. Für die Berechnung des Umfangs eines Kreises mit Radius $r = 2$ genügt zum Beispiel folgende Eingabe

```
==> r=2; 2*pi*r
```

Das System antwortet dann mit

```
ANS =
        12.5664
<>
```

ANS steht dabei für *ANSWER*, also für die Antwort des Systems. Weist man dem Umfang einen Variablennamen zu, so antwortet das System mit diesem Namen anstatt mit **ANS**.

```
==> r=2; u=2*pi*r

u =
     12.5664
<>
```

Zu beachten ist, daß durch das Semikolon die Ausgabe des Wertes von r nach der Zuweisung unterdrückt wird. Wird ein Ausdruck durch ein Komma abgeschlossen oder folgt kein Zeichen mehr, so wird der Wert ausgegeben.

Will man die Wurzel aus 5 berechnen, so kann man einfach hinschreiben `SQRT(5)`, wobei zu bemerken ist, daß die Standardfunktionen, anders als zum Beispiel in FORTRAN, auf jeden numerischen Argumenttyp angewendet werden können. Die Eingabe

```
==> SQRT(2-5i)
```

berechnet z. B. die Quadratwurzel aus der komplexen Zahl $2 - 5i$.

Natürlich kann man die Wurzel aus einer Zahl a auch mit dem *Newton-Verfahren* bzw. dem *Babylonischen Wurzelziehen* gemäß der Iteration

$$x_{n+1} = (x_n + a/x_n)/2, \quad n = 0, 1, 2, \ldots$$

berechnen, wobei für einen geeigneten Startwert x_0 gilt

$$\lim_{n \to \infty} x_n = a$$

In CALCULUS läßt sich dies für $a = 5$ einfach hinschreiben durch

```
==> x=5; x=(x+5/x)/2
```

und durch wiederholtes Durchführen der zweiten Anweisung nähert sich x dem gesuchten Wert $\sqrt{5}$.

Das CALCULUS-System kennt aber auch Vektoren und Matrizen. Den Zeilenvektor $x = (1, 2, 3, 4, 5)$ erhält man z. B. durch

```
==> x=<1,2,3,4,5>

x = (1,5)-real-point-matrix
     1 2 3 4 5
<>
```

oder durch

```
==> 1:5

ANS = (1,5)-real-point-matrix
        1 2 3 4 5
<>
```

Durch die mathematische, natürliche Notation unterstützt CALCULUS die Lesbarkeit von Programmen bzw. Algorithmen. Um den Vektor **x** z. B. zu transponieren schreibt man nur

```
==> x'

ANS = (5,1)-real-point-matrix
        1
        2
        3
        4
        5
<>
```

Genauso einfach können die Elemente des Vektors x aufsummiert oder deren Produkt berechnet werden:

```
==> sum(x)

ANS =
       15
<>

==> prod(x)

ANS =
      120
<>
```

Ein Skalarprodukt, also ein Produkt aus einem Zeilenvektor und einem Spaltenvektor wird durch das gewöhnliche Multiplikationssymbol * gekennzeichnet:

```
==> x*x'

ANS =
        55
<>
```

Berechnet man dagegen das *dyadische Produkt* von `x'` und `x` so ergibt sich

```
==> A=x'*x

A = (5,5)-real-point-matrix
     1  2  3  4  5
     2  4  6  8 10
     3  6  9 12 15
     4  8 12 16 20
     5 10 15 20 25
<>
```

Auch auf eine Matrix kann `sum` angewendet werden, was eine Summierung über die Spalten und als Ergebnis einen Zeilenvektor liefert.

Für Matrizen und Vektoren bietet CALCULUS außerdem die Möglichkeit, beliebige Teilfelder herauszugreifen und auch zuzuweisen.

```
==> A(2:3,3:5)

ANS = (2,3)-real-point-matrix
       6  8 10
       9 12 15
<>

==> v=3:5; A(v,v) = <1,2,3;4,5,4;3,2,1>

A = (5,5)-real-point-matrix
     1  2  3  4  5
     2  4  6  8 10
     3  6  1  2  3
     4  8  4  5  4
     5 10  3  2  1
<>
```

Für die Matrixmultiplikation gibt es zwei Möglichkeiten der Ausführung: die normale über die Skalarproduktbildung oder aber die elementweise Produktbildung. In CALCULUS wird die elementweise Produktbildung durch einen Doppelpunkt (`:`) hinter dem Multiplikationszeichen gekennzeichnet. Schreibt man also `A*A`, so wird die normale Matrixmultiplikation durchgeführt, während durch `A*:A` die Quadrate der Matrixelemente gebildet werden.

```
==> A = <1,2,3;4,5,4;3,2,1>

A = (3,3)-real-point-matrix
     1 2 3
     4 5 4
     3 2 1
<>

==> A*A

ANS = (3,3)-real-point-matrix
       18 18 14
       36 41 36
       14 18 18
<>

==> A*:A

ANS = (3,3)-real-point-matrix
       1  4  9
      16 25 16
       9  4  1
<>
```

Dieses Doppelpunkt-Konzept ist generisch im gesamten System. Es läßt sich z. B. auch bei den Standardfunktionen anwenden. `SQRT:(A)` liefert die elementweise berechnete Quadratwurzel aus `A`, während `B=SQRT(A)` eine komplexe Matrix B erzeugt, für die `B*B`$\approx$`A` gilt.

Eine einfache Schreibweise für die Berechnung einer Näherungsinverse ist ebenfalls vorhanden, `R=1/A` liefert das Gewünschte. Um die Genauigkeit der so berechneten Matrix zu überprüfen, würde man ja normalerweise `R*A-Id` berechnen, wobei `Id` die generische Einheitsmatrix des CALCULUS-Systems darstellt, die automatisch an die Dimension des zweiten Operanden in einem Ausdruck angepaßt wird. Will man den Ausdruck `R*A-Id` maximal genau, also mit einem einzigen Skalarprodukt berechnen, so reicht es in CALCULUS den Ausdruck zu klammern und ein „!“ vorzustellen, welches die maximale Genauigkeit symbolisiert:

```
!(R*A-Id) .
```

Ebenso können interaktiv die Eigenwerte einer Matrix A durch die Eingabe von `eig(A)` bestimmt werden. Man kann also z. B. mit `eig(Id-R*A)` un-

mittelbar feststellen, ob eine Residueniteration mit den Matrizen R und A konvergiert, denn als Ergebnis von **eig** erhält man die Eigenwerte und damit den Spektralradius von Id-R*A.

Das bereits erwähnte Babylonische Wurzelziehen kann man auch auf Matrizen anwenden. Will man die Matrix B bestimmen, für die gilt $B*B \approx A$, so iteriert man in CALCULUS einfach durch

```
B=(B+A/B)/2 .
```

Dabei entspricht A/B der Berechnung von $A*1/B=A*B^{-1}$. Zu beachten ist lediglich, daß für B ein komplexer Startwert gewählt werden muß, da sonst die Iterierten immer reell bleiben. Zum Beispiel könnte man mit

```
B=A+2i
```

starten. Der Buchstabe i wird vom CALCULUS-System als imaginäre Einheit erkannt, da zwischen dem Faktor 2 und dem i kein Operatorzeichen auftritt. In Verbindung mit einem Operator wird i als Variable angesehen.

Die Eingabe von Intervallen kann in CALCULUS unmittelbar durch Toleranz

```
==> 2+/-.001
```

erfolgen. Vom System wird dieses Intervall im Format F5 (Fixpunktformat mit 5 Stellen) als

```
ANS =
      2.000__
<>
```

ausgegeben, wobei der Strich nach der letzten angezeigten Ziffer jeweils bedeutet, daß diese Ziffer ± 1 zu nehmen ist. Eine alternative Schreibweise (I-Format) wäre

```
(1.999,2.001)
```

Für die Invertierung von Matrizen gibt es neben dem Aufruf der LINPACK-Routine durch 1/A auch die Möglichkeit, eine Einschließung für die Inverse zu berechnen. Der Aufruf erfolgt durch

```
INV(A),
```

und intern wird die entsprechende ACRITH-Routine benutzt, die die Inverse maximal genau in ein Intervall einschließt.

Will man einen beliebigen Ausdruck intervallmäßig auswerten, so gibt es in CALCULUS die Möglichkeit, dies mit der Funktion **ival** zu kennzeichnen. Bei der Residueniteration z. B. würde

```
ival(R*(b-A*x))
```

die intervallmäßige Auswertung des Ausdrucks bewirken. Tritt irgendwo in einem Ausdruck ein Intervall auf, so wird dieser grundsätzlich intervallmäßig ausgewertet.

5 Zusammenfassung und Kurzübersicht zum CALCULUS-System

Datentypen

In der folgenden Tabelle sind die in CALCULUS verwendeten Datentypen mit entsprechenden Beispielen aufgeführt.

	point	interval
real	`-1.23E3`	`1.23+/-0.01` bzw. `HULL (1.22, 1.24)`
complex	`1+2i`	`1+2i+/-(0.02+0.05i)` bzw. `HULL (0.98+1.95i,1.02+2.05i)`

Datenstrukturen

Der Aufbau eines Zeilenvektors $v = (\ 1 \quad 1.5 \quad 2 \quad 2.5\ )$ erfolgt durch

```
v = <1,1.5,2,2.5>  oder durch  v = 1:2.5:0.5
```

Eine (3,4)-Matrix $M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$ entsteht durch

```
M = <1, 2, 3, 4;
     5, 6, 7, 8;
     9,10,11,12>
```

Auswahl der 3. Komponente der 2. Zeile von M:

```
s = M(2,3)
```

Auswahl der 1. Zeile von M:

```
v = M(1,:)
```

Auswahl der Komponenten 2 bis 4 der 1. Zeile von M:

```
v = M(1,2:4)  oder  v = M(1,<2,3,4>)
```

Auswahl der Komponenten 1, 2 und 4 der 1. und 3. Zeile von M:

```
T = M(<1,3>,<1,2,4>)
```

Tauschen von 2. mit 3. Spalte von M:

```
v = M(:,2); M(:,2) = M(:,3); M(:,3) = v;  oder
M = M(:,<1,3,2>)
```

(Anstelle eines Indexes können Indexvektoren angegeben werden.)

CALCULUS kennt eine generische Einheitsmatrix `Id`, die z. B. wie folgt benutzt werden kann:

`M+3*Id`	Der Wert Drei wird zu den Diagonalelementen von M addiert
`Id(m,n)`	Eine (m,n)-Matrix wird erzeugt, deren Diagonalelemente den Wert Eins und deren übrige Elemente den Wert Null besitzen
`Id(M)`	Eine der Dimensionen von M entsprechende Einheitsmatrix wird erzeugt

Arithmetik

Alle einfachen Skalar-, Vektor- und Matrixoperationen werden unter Verwendung der IBM-Programmbibliothek ACRITH maximal genau ausgeführt. Zusammengesetzte Ausdrücke in Skalarproduktform wie z. B.

```
a*b-c*d+e
```

werden maximal genau ausgeführt, wenn dies durch ein vorangestelltes „!" gekennzeichnet wird:

```
!(a*b-c*d+e).
```

Die Elemente des Skalarprodukts können reelle oder komplexe, Punkt- oder Intervalldaten, aber auch Vektoren und Matrizen aller dieser Typen sein.

Ist ein Operand innerhalb eines Ausdrucks vom Typ *interval*, so wird der gesamte Ausdruck intervallmäßig berechnet.

Operatoren

In der folgenden Tabelle sind die CALCULUS-Operatoren gemäß ihrer Priorität angegeben.

' (Hochkomma)	Transposition (Vektoren u. Matrizen)
!	maximal genaue Auswertung von Skalarproduktausdrücken
+,- (monadisch)	Vorzeichen
**	Potenzierung
*, /, \	Multiplikation, Division (Inverse), umgek. Division (lin. Gls. lösen)
+, -, +/-	Addition, Subtraktion, Toleranzangabe
==, <>, <, <=, >=, >, IN, INO	Vergleiche: Gleich, Ungleich, Kleiner, Kleinergleich, Größergleich, Größer Enthalten in, Enthalten im Innern
NOT AND OR	logisch Nicht logisch Und logisch Oder
:	Operator elementweise anwenden
#	intervallmäßige Auswertung

Als Ersatzzeichen für \ kann auch % vewendet werden. Die intervallmäßige Auswertung wird auch mit IVAL anstatt # aufgerufen.

Ein-/Ausgabe

CALCULUS zeigt den Wert eines Ausdrucks an, wenn auf den Ausdruck ein Komma oder das Zeilenende folgt. Ein Semikolon hingegen unterdrückt die Anzeige. Der Änderung des Anzeigeformats dient die Anweisung

FORMAT Xn mit X = F: Fixpunktformat (voreingest. F5)
E: Exponentialformat
I: Intervallformat
H: Hex-Format
n = gewünschte Ziffernzahl

Intervalle werden im F- und E-Format z. B. als 3.1415_ (Format F5) dargestellt, was dem Intervall [3.1414,3.1416] entspricht, d. h. die letzte angezeigte Ziffer ist „± 1" zu nehmen.

Kontrollstrukturen

Jede CALCULUS-Anweisung darf eine Blockanweisung sein, d. h. eine Folge von Anweisungen, eingefaßt in geschweifte Klammern. Die IF-Anweisung, die allgemeine Schleife (LOOP) und die FOR-Schleife haben folgende Syntax:

```
IF scalar_expression THEN statement [ELSE statement]
LOOP statement
FOR identifier = vector_expression DO statement
```

Zu beachten ist daß die Blockanweisung des LOOP eine EXIT-Anweisung enthalten muß. Das Kommando EXIT beendet jeweils die innerste FOR- oder LOOP-Anweisung bzw. die Ausführung des betreffenden Moduls. Mit RETURN wird ein Modul definitiv beendet.

Vordefinierte Konstanten und Variablen

Vordefiniert stehen in CALCULUS die Konstanten PI, EULER und EPS zur Verfügung. Außerdem ist die Variable ANS für die Antwort des Systems vordefiniert.

Intervallfunktionen

INF	: untere Intervallgrenze
SUP	: obere Intervallgrenze
MID	: Intervallmittelpunkt
RAD	: Intervallradius
DIAM	: Intervalldurchmesser
HULL	: Intervallhülle
ALL	: Intervalldurchschnitt

Allgemeine Standardfunktionen

RE	IM	CONJ	ABS	ARG
ROUND	TRUNC			
SQR	SQRT			
EXP	LOG	LOG10		
SIN	ARCSIN	SINH	ARSINH	
COS	ARCCOS	COSH	ARCOSH	
TAN	ARCTAN	TANH	ARTANH	
COT	ARCCOT	COTH	ARCOTH	

Typabfragen

```
COMPLEX
INTERVAL
MATRIX   VECTOR   SCALAR   EMPTY
```

Größenabfragen und -änderungen

```
SIZE     SHAPE    RESIDUE  RESHAPE
PACK     UNPACK
```

Datenanalyse

```
MIN      MAX      SUM      PROD     RAT
```

Funktionen zur Matrix-Manipulation

Matrix-Erzeugung:

```
RAND                      RANDOMIZE
UNIFORM_DISTRIBUTION      NORMAL_DISTRIBUTION
NULLS                     ONES
HILB                      MAGIC
DIAG                      TRIL                    TRIU
```

Matrix-Eigenschaften:

```
RANK     DET      RCOND    COND
NORM     FNORM    INORM
```

Matrix-Zerlegung:

```
CHOL     LU       QR
HESS     SCHUR    ORTH
SVD
```

Weitere Matrix-Funktionen:

```
INF      PINV     EIG      LSS
```

Funktionen zur Polynom-Manipulation

```
POLY     ROOTS    PEVAL    CHARPOLY
```

Programmaufbau

CALCULUS unterstützt über interaktive Eingaben hinaus ein einfaches Unterprogrammkonzept in Form von **EXEC**-Modulen.

- Den interaktiven Eingabemodus kennzeichnet die Eingabeanforderung „<>"; **EXIT** oder **RETURN** führen an dieser Stelle zurück zur Betriebssystemebene.
- **EXEC**-Module sind Quelltextdateien, die mit dem Schlüsselwort **MODULE** beginnen, gefolgt von einer optionalen Parameterliste sowie den Anweisungen des Moduls. Diese **EXEC**-Module können interaktiv oder (rekursiv) von beliebigen Modulen mittels **EXEC**-Anweisungen aufgerufen werden; Modulname ist dabei der Filename der betreffenden Datei.
- Dem ersten Modul einer Datei können weitere lokale Module folgen, eingeleitet durch das Schlüsselwort **MODULE**, gefolgt von einem Unterprogrammnamen und einer optimalen Parameterliste sowie den Modulanweisungen. Diese lokalen Module werden gleichfalls mittels **EXEC**-Anweisung (rekursiv) aufgerufen, sind jedoch nur innerhalb der betreffenden Datei sichtbar. (Lokale Module verdecken gleichnamige globale Module.)

6 Ausblick

In der Praxis hat sich die vorliegende Version des interaktiven Programmierwerkzeuges CALCULUS bestens bewährt. Es wurden eine Reihe der ACRITH-Routinen in CALCULUS nachprogrammiert mit beträchtlichen Effektivitätssteigerungen. Die Entwicklungszeit der Programme reduzierte sich auf einen Bruchteil, die Anzahl der Programmzeilen schrumpfte oft um mehr als zwei Zehnerpotenzen.

In Übungen und ganz besonders in der Forschung ist CALCULUS bzw. ABACUS von unschätzbarem Wert. Es gelingt, die entwickelten Algorithmen in praktisch der gleichen Notation, wie sie auf dem Papier entwickelt werden, in den Rechner einzugeben. Oftmals schreckt man vor der Implementierung in einer herkömmlichen Programmiersprache wegen des Aufwandes zurück. Mit CALCULUS bzw. ABACUS ist eine Implementierung schnell und vor allem übersichtlich durchführbar. Wegen der kurzen Programme wird die Fehlerrate beim Programmieren verschwindend klein.

Die neue Version ABACUS wird auf IBM PC unter DOS und Workstations unter UNIX Anfang 1989 laufen. Im neuen System ABACUS sind

allfällige Schwächen von CALCULUS eliminiert und vor allem der Funktionsumfang wesentlich erweitert worden. Mit ABACUS steht dem Benutzer ein äußerst mächtiges Werkzeug zur Verfügung.

Literatur

[1] Husung, D.: *ABACUS, Programmierwerkzeug mit hochgenauer Arithmetik für Algorithmen mit verifizierten Ergebnissen.* Diplomarbeit, Universität Karlsruhe, 1988.

TU-BH