Jürgen Schwarz

Behandlung von Polynomen – Teil 3*

Im ersten Teil dieser Serie wurde die Darstellung von Polynomen im Computer behandelt und SUB-Programme für die rationalen Funktionen abgeleitet. Der zweite Teil enthielt SUB-Programme zur Berechnung der Wurzeln von Polynomen und zwar ein spezielles für reelle Koeffizienten und nur reelle einfache Nullstellen und ein universelles für Polynome mit komplexen Koeffizienten.

Stichwörter:

Polynom, HP-BASIC, Laplace-Transformation, Approximation, Ausgleichsrechnung

Hardware:

HP 9845 B mit SP ROM

6 Größter gemeinsamer Teiler zweier Polynome

Werden zwei Polynome f(x) und g(x) zur Darstellung einer gebrochen rationalen Funktion

$$F(x) = \frac{f(x)}{g(x)} \qquad (43)$$

verwendet, so ist es vor der Weiterverarbeitung zweckmäßig, eventuell vorhandene gemeinsame Teiler beider Polynome „herauszukürzen". Aus Gl. (43) wird also

$$F(x) = \frac{f(x)}{g(x)} = \frac{f^*(x)\ h(x)}{g^*(x)\ h(x)} = \frac{f^*(x)}{g^*(x)}, \qquad (44)$$

wobei hier Fragen der Definition von F(x) an den Nullstellen von h(x) beiseite gelassen werden sollen. Das Polynom h(x) soll mindestens von erstem Grade sein; von trivialen gemeinsamen Teilern nullten Grades sehen wir ab.

Die folgenden Algorithmen können außerdem noch zur Bestimmung und Elimination von mehrfachen Wurzeln in Polynomen verwendet werden, indem für g(x) die erste Ableitung von f(x) eingesetzt wird, da überall dort, wo gemeinsame Nullstellen von f(x) und f'(x) vorliegen, Mehrfachwurzeln sind.

Hier ist jetzt die Aufgabe gestellt, den größten gemeinsamen Teiler h(x) der beiden Polynome f(x) und g(x) zu bestimmen, also das Polynom h(x) mit dem größtmöglichen Grad.

6.1 *Der Euklidische Algorithmus*

Bezeichnet man das Polynom mit dem größeren Grad mit r_0 und das andere mit r_1, so läßt sich r_0 durch r_1 dividieren

$$\frac{r_0}{r_1} = t_1 + \frac{r_2}{r_1} \text{ mit } gr(r_2) < gr(r_1), \qquad (45)$$

wobei dabei im allgemeinen ein Restpolynom r_2 auftritt. Durch die Forderung, daß der Grad von r_2 kleiner als der Grad von r_1 sein soll, ist die Aufgabe auch eindeutig. Tritt kein Restpolynom auf, d.h. ergibt sich für r_2 das Nullpolynom (ein formales Polynom, bei dem alle Koeffizienten null sind und das keinen bestimmten Grad hat), dann ist die Aufgabe gelöst und r_1 ist der gesuchte größte gemeinsame Teiler der beiden Polynome.

Aus Gl. (45) folgen nach Multiplikation mit r_1 und formaler Weiterführung des Algorithmus die folgenden Gleichungen:

$$\begin{aligned} r_0 &= t_1 r_1 + r_2 \\ r_1 &= t_2 r_2 + r_3 \\ r_2 &= t_3 r_3 + r_4 \\ &\cdots \\ &\cdots \\ r_{k-2} &= t_{k-1} r_{k-1} + r_k \\ r_{k-1} &= t_k r_k , \end{aligned} \qquad (46)$$

bei denen die Bedingungen

$$gr(r_{n+1}) < gr(r_n) \qquad (47)$$

eingehalten werden müssen. Aus Gl. (47) ergibt sich, daß der Euklidische Algorithmus nach

$$k \leqslant gr(r_1) \qquad (48)$$

Schritten abgeschlossen sein muß, da spätestens dann $gr(r_k) = 0$ ist. Zum Verständnis des Rechenverfahrens nehmen wir die *n*-te Gleichung aus (46) heraus

$$r_{n-1} = t_n r_n + r_{n+1} \qquad (49)$$

und verdeutlichen uns folgenden Satz: Ist das Polynom s gemeinsamer Teiler von r_n und r_{n+1}, dann ist s auch Teiler von r_{n-1}. Dies wird mit

$$\underbrace{r^*_{n-1}\, s}_{r_{n-1}} = t_n \underbrace{r^*_n s}_{r_n} + \underbrace{r^*_{n+1}\, s}_{r_{n+1}} \qquad (50)$$

deutlich, wobei offensichtlich

$$r^*_{n-1} = t_n r^*_n + r^*_{n+1} \qquad (51)$$

ist. Aus der letzten Teilgleichung von (46) folgt aber, daß r_k ein Teiler von r_{k-1} ist und damit ist r_k ein gemeinsamer, und zwar auch der größte gemeinsame Teiler von allen r_n und damit auch von r_0 und r_1 [7].

Jürgen Schwarz, Wiesbadener Str. 59 c, 1000 Berlin 33

* Teil 2: s. CAL – Comp. Anw. Lab. 3 (1985) 76.

Listing 7 zeigt die Implementation des EUKLIDISCHEN Algorithmus für Polynome mit reellen Koeffizienten in HP-BASIC. Nach dem Vereinbarungsteil und der Sortierung der Ausgangspolynome nach ihrem Grad erfolgt die Berechnung streng nach Gl. (46). Es erfolgt keine Prüfung, ob die mit *n* und *m* angegebenen Polynomgrade auch stimmen, ob also a(*n*) ≠ 0 und b(*m*) ≠ 0 sind. Ist dies beim kleineren Polynom der Fall, dann versagt das SUB-Programm wegen einer Division durch null beim Aufruf von Rest_division. Dieser in Listing 8 wiedergegebene Algorithmus lehnt sich weit an Listing 4 (Teil 1 dieser Serie) an. Allerdings wird hier nicht nur geprüft, ob das Nullpolynom als Rest auftritt, sondern dieses Restpolynom wird ausgegeben, da es zur weiteren Berechnung benötigt wird. Die auf den ersten Blick merkwürdig erscheinende LOOP-Konstruktion in den Zeilen 420 bis 460 ist erforderlich, weil in Zeile 450 bis auf $p = -1$ subtrahiert wird, die entsprechende Abfrage auf Zeile 440

Listing 7: Berechnung des größten gemeinsamen Teilers von zwei Polynomen mit reellen Koeffizienten nach Euklid

```
10    SUB Euklid(INTEGER N,M,K,REAL A(*),B(*),C(*),D(*),E(*),Epsilon)
20
30      SUB-Programm zur Berechnung der gemeinsamen Nullstellen von zwei
40      Polynomen mit reellen Koeffizienten f(x) und g(x) und Bestimmung des
50      größten gemeinsamen Teilers [ggT] h(x) mit dem Euklidischen Algorithmus:
60
70                    f(x) = f_stern(x) * h(x)
80                    g(x) = g_stern(x) * h(x)
90
100     Eingabedaten: n          ... Grad           des Polynoms f(x)
110                   m          ... Grad           des Polynoms g(x)
120                   a(0:n)     ... Koeffizienten des Polynoms f(x)
130                   b(0:m)     ... Koeffizienten des Polynoms g(x)
140                   epsilon    ... zulässige relative Abweichung zur Prüfung
150                                   auf das Auftreten des Nullpolynoms
160   ! Ergebnis:     n-k        ... Grad           des Polynoms f_stern(x)
170   !               m-k        ... Grad           des Polynoms g_stern(x)
180   !               k          ... Anzahl der gemeinsamen Nullstellen
190   !                              von f(x) und g(x) = Grad von h(x)
200   !               c(0:n-k)   ... Koeffizienten des Polynoms f_stern(x)
210   !               d(0:m-k)   ... Koeffizienten des Polynoms g_stern(x)
220   !               e(0:k)     ... Koeffizienten des Polynoms h(x) mit e(k)=1
230   !
240   ! Programmierer:    Jürgen Schwarz      Datum/Variante:   12.11.84 / 01
250   ! Programm-Name:          Euklid        Speichermedium: Kassetten 57/58
260   !
270     INTEGER P,R
280     REAL R_0(0:MAX(N,M)),R_1(0:MIN(N,M)),R_2(0:MAX(0,MIN(N,M)-1))
290     REAL T(0:MAX(1,MIN(N,M)-1,MAX(N,M)-MIN(N,M)))
300     !
310     IF (N<0) OR (M<0) OR (Epsilon<0) THEN PAUSE     ! widersprüchliche Daten
320     REDIM A(0:N),B(0:M)
330     IF N>=M THEN
340       MAT R_0=A
350       MAT R_1=B
360     ELSE
370       MAT R_0=B
380       MAT R_1=A
390     END IF
400     MAT E=ZER
410     !
420     P=MAX(N,M)
430     K=MIN(N,M)
440     LOOP
450       CALL Rest_division(P,K,R,R_0(*),R_1(*),T(*),R_2(*),Epsilon)
460     EXIT IF R=-1
470       P=K
480       K=R
490       MAT R_0=R_1
500       MAT R_1=R_2
510     END LOOP
520     MAT E=(1/R_1(K))*R_1          ! Bezug auf e(k), um e(k)=1 zu gewährleisten
530     CALL Rest_division(N,K,R,A(*),E(*),C(*),R_2(*),Epsilon)
540     CALL Rest_division(M,K,R,B(*),E(*),D(*),R_2(*),Epsilon)
550     !
560   SUBEND
```

mit dem Computer zu verbinden, Sie brauchen aber nicht mehr das Signal zu konvertieren.

Die hier vorliegende Implementation hat den Vorteil, daß sie auch bei konjugiert komplexen Nullstellen ohne Rechnungen im **C** auskommt. Nachteilig ist die geringe numerische Stabilität, die auf die vielen Divisionen zurückzuführen ist. Deshalb ist es im allgemeinen vorteilhafter, ein anderes Verfahren zu verwenden.

6.2 Anwendung des Fundamentalsatzes der Algebra

Da die Normalform jeder ganzen rationalen Funktion *n*-ten Grades nach Gl. (10) in ein Produkt von *n* Linearkombinationen zerlegt werden kann, besteht bei Kenntnis der einzelnen Faktoren, d.h. der Wurzeln des Polynoms, die Möglichkeit, den größten gemeinsamen Teiler durch einen Vergleich der Wurzeln beider Polynome f(*x*) und g(*x*) zu berechnen:

Listing 8: Division von zwei Polynomen mit reellen Koeffizienten und Berechnung des Restpolynoms

```
10    SUB Rest_division(INTEGER N,M,P,REAL A(*),B(*),C(*),D(*),Epsilon)
20    !
30    ! SUB-Programm zur Durchführung einer Division von zwei Polynomen
40    ! mit reellen Koeffizienten und Berechnung des Restes r(x):
50    !    f(x) = g(x) * h(x) + r(x)    |    f(x)/g(x) = h(x) + r(x)/g(x)
60    !
70    ! Eingabedaten: n          ... Grad           des Polynoms f(x) (>= m)
80    !               m          ... Grad           des Polynoms g(x)
90    !               a(0:n)     ... Koeffizienten des Polynoms f(x)
100   !               b(0:m)     ... Koeffizienten des Polynoms g(x)
110   !               epsilon    ... zulässige relative Abweichung zur Prüfung
120   !                              auf das Auftreten des Nullpolynoms
130   ! Ergebnis:     n-m        ... Grad           des Polynoms h(x)
140   !               p          ... Grad           des Polynoms r(x)
150   !                              p=-1 ===> Rest ist das Nullpolynom
160   !               c(0:n-m)   ... Koeffizienten des Polynoms h(x)
170   !               d(0:p)     ... Koeffizienten des Polynoms r(x)
180   !
190   ! Programmierer:   Jürgen Schwarz        Datum/Variante:   12.11.84 / 01
200   ! Programm-Name:   unter   Euklid        Speichermedium: Kassetten 57/58
210   !
220     INTEGER I,J
230     REAL A_stern(0:N),B_stern(0:M)
240     !
250     IF (N<M) OR (M<0) OR (Epsilon<0) THEN PAUSE     ! widersprüchliche Daten
260     !
270     REDIM A(0:N),B(0:M)
280     MAT A_stern=A
290     MAT B_stern=B
300     MAT C=ZER
310     MAT D=ZER
320     !
330     REDIM C(0:N-M)
340     FOR I=N-M TO 0 STEP -1
350       C(I)=A_stern(I+M)/B_stern(M)
360       FOR J=I+M TO I STEP -1
370         A_stern(J)=A_stern(J)-C(I)*B_stern(J-I)
380       NEXT J
390     NEXT I
400     !
410     P=M-1
420     LOOP
430     EXIT IF P=-1 ! ==> Ergebnis ist das Nullpolynom
440     EXIT IF ABS(A_stern(P))>Epsilon*ABS(C(0)*B_stern(P))
450       P=P-1
460     END LOOP
470     IF P>=0 THEN
480       REDIM A_stern(0:P)
490       MAT D=A_stern
500     ELSE
510       REDIM D(0:0)
520     END IF
530     !
540   SUBEND
```

$$F(x) = \frac{f(x)}{g(x)} = \frac{a_n \cdot (x-f_1)\,(x-f_2)\,\ldots\,(x-f_n)}{b_m \cdot (x-g_1)\,(x-g_2)\,\ldots\,(x-g_m)}. \quad (52)$$

Ist beispielsweise (im Rahmen der Rechengenauigkeit) $f_i = g_j$, dann wird der Linearfaktor $(x - f_i)$ dem Polynom $h(x)$ zugeschlagen.

Listing 9 zeigt ein Programmbeispiel, welches zum Einsatz kommt, wenn bekannt ist, daß die Wurzeln von $f(x)$ und $g(x)$ jeweils nicht mehrfach auftreten, reell und negativ sind. Das Nullstellenberechnungsprogramm Newton_mod wurde im 2. Teil dieser Serie nach Gl. (13) kurz erwähnt und ist in [8] vollständig veröffentlicht. Ist nur bekannt, daß die Wurzeln reell sind und nicht mehrfach auftreten, so kann in den Zeilen 440 und 450 das SUB-Programm Newton nach Listing 4 aufgerufen werden. Allerdings müßten dann die Vergleichsbedingungen in den Zeilen 540 und 550 entsprechend Listing 4 verändert werden. Der Rest des Programms ist ohne weitere Erklärungen verständlich.

Listing 10 zeigt ein ähnliches, universelles Programm für Polynome mit komplexen Koeffizienten. Selbstverständlich kann es auch für Polynome mit nur reellen Koeffizienten zum Einsatz kommen, wenn die entsprechenden Imaginärteile zu null gesetzt werden. Im Unterschied zu Listing 9 erfolgt hier die Ausgabe der Ergebnisse über dieselben Felder, über die auch die Eingabe erfolgt. Die Originalwerte gehen also verloren. Ein Aufruf zur Kürzung von zwei Polynomen hat also die Form

```
CALL Komplex_kuerzg(N,M,K,A(*),B(*),
C(*),D(*),E(*),F(*),Eps),
```

wobei nach der Abarbeitung die gekürzten Polynomkoeffizienten wieder in A(*), B(*), C(*) und D(*) stehen. Auch N und M ändern (bei einer Kürzung) ihren Wert. Sonst arbeitet das SUB-Programm ohne Besonderheiten.

Ein zum obigen analoger Aufruf für mit dem SUB-Programm Kuerzung nach Listing 9 verarbeitete Polynome hat die Form

```
CALL Kuerzung(N,M,K,A(*),B(*),A(*),
B(*),E(*),Epsilon).
```

Auch das SUB-Programm Euklid kann genauso aufgerufen werden. Dies wird durch die Datensicherung (Zwischenspeicherung) nach dem Vereinbarungsteil gewährleistet. Beim Aufruf von Kuerzung und Euklid müssen aber N und M, im Gegensatz zu Komplex_kuerzg, mit zwei folgenden Befehlen um K reduziert werden.

7 Zweckmäßige Datensicherung bei SUB-Programmen

Bei dieser Gelegenheit ein paar Bemerkungen zu der im SUB-Programm Kuerzung, aber auch in den anderen SUB-Programmen, z.B. in Produkt, verwendeten Form der Datensicherung. Im allgemeinen haben die Programme folgende Struktur:

```
10  SUB Programm(INTEGER N,M,
    REAL A(*),B(*))
20  REAL A_stern(0:N)
30  REDIM A(0:N)
40  MAT A_stern = A
50  MAT B = ZER
60  REDIM B(0:N - M)
70  ...
```

In Zeile 20 wird Speicherplatz für die Zwischenspeicherung reserviert. Mit dem Befehl REDIM A(0:N) werden die Indexgrenzen des Feldes A(*) auf den aktuellen Wert eingestellt. Erfolgt dies nicht, dann kann es bei der folgenden Programmabarbeitung zu Fehlern kommen. Stehen die Indexgrenzen von A(*) beim Aufruf des SUB-Programms z.B. auf (1:N + 1), dann stehen die Grenzen von A_stern(*) nach dem Kopieren genauso. Bei der Redimensionierung werden die Werte verschoben, d.h. bei dem obigen Zahlenbeispiel wird A(1) → A(0), A(2) → A(1), usw. In Zeile 40 wird das Feld A(*) dann in das Feld A_stern(*) kopiert. Der Befehl MAT B = ZER bewirkt die Löschung aller Feldelemente innerhalb der aktuellen Indexgrenzen. Würde dieser Befehl vor Zeile 40 stehen, und ein Aufruf der Art CALL Programm(N,M,A(*),A(*)) erfolgen, würden alle Ausgangswerte von A(*) gelöscht und es könnte nichts mehr gerechnet werden.

Die Befehle in den Zeilen 50 und 60 könnte man eigentlich auch mit dem Befehl MAT B = ZER(0:N - M) zusammenfassen. Dieser Befehl wirkt aber genau umgekehrt wie die obige Anordnung. Zuerst wird redimensioniert, dann wird nulliert. Auf die vorher vorhandenen Elemente (falls das Feld vorher größer war) kann dann zwar nicht unmittelbar zugegriffen werden, aber die Zahlenwerte sind noch abgespeichert. Wird das Feld bei der nächsten Redimensionierung vergrößert, dann sind die „auftauchenden" Werte nicht null (wie erhofft), sondern haben irgendwelche Größen. Insbesondere bei der mehrfachen Verwendung von Feldern zur Zwischenspeicherung sind diese Konstruktionen erforderlich und haben sich bewährt.

Ein Beispiel möge den Sachverhalt verdeutlichen. Es sollen das Zähler- und das Nennerpolynom $z(x)$ und $n(x)$ der Gleichung

$$F(x) = \frac{z(x)}{n(x)} = \frac{a(x)\,b(x) + c(x)\,d(x) + e(x)}{f(x)\,g(x) - h(x)\,i(x)}, \quad (53)$$

wobei $a(x)$, $b(x)$, ..., $i(x)$ Polynome sein sollen, berechnet werden. Ein entsprechendes Programmstück hat dann folgendes Aussehen:

```
100  CALL Produkt(0,0,A(*),B(*),
     Hilf_1(*))
110  CALL Produkt(0,0,C(*),D(*),
     Hilf_2(*))
120  CALL Addition(1,Hilf_1(*),1,
     Hilf_2(*),Hilf_1(*))
130  CALL Addition(1,Hilf_1(*),1,
     E(*),Z(*))
140  CALL Produkt(0,0,F(*),G(*),
     Hilf_1(*))
150  CALL Produkt(0,0,H(*),I(*),
     Hilf_2(*))
160  CALL Addition(1,Hilf_1(*), - 1,
     Hilf_2(*),N(*)).
```

Ohne die oben erwähnten Datensicherungsoperationen würde die so relativ übersichtliche Programmkonstruktion versagen, insbesondere deshalb, weil die Hilfsfelder Hilf_1(*) und Hilf_2(*) immer wieder mit unterschiedlichen Werten und aktuellen Indexgrenzen belegt werden.

Literatur

[7] EUKLID „Die Elemente", III. Teil, VII. Buch, § 2. C. Thaer (Übersetzer und Herausgeber). Ostwald's Klassiker der exakten Wissenschaften, Bd. 240. Akad. Verlagsgesellschaft Leipzig (1935).

[8] J. SCHWARZ „Thermische Ersatzschaltbilder". Elektronik Journal **20** (1985) 7, 72 – 82.

CAL

Listing 9: Bestimmung des größten gemeinsamen Teilers von zwei Polynomen mit reellen Koeffizienten und reellen, negativen und nichtmehrfachen Nullstellen durch Zerlegung in Linearkombinationen

```
10    SUB Kuerzung(INTEGER N,M,K,REAL A(*),B(*),C(*),D(*),E(*),Epsilon)
20    !
30    ! SUB-Programm zur Berechnung der gemeinsamen Nullstellen von zwei
40    ! Polynomen mit reellen Koeffizienten f(x) und g(x) und nur reellen,
50    ! einfachen Wurzeln und Bestimmung des größten gemeinsamen Teilers [ggT]:
60    !                   f(x) = f_stern(x) * h(x)
70    !                   g(x) = g_stern(x) * h(x)
80    !
90    ! Eingabedaten: n         ... Grad           des Polynoms f(x)
100   !               m         ... Grad           des Polynoms g(x)
110   !               a(0:n)    ... Koeffizienten des Polynoms f(x)
120   !               b(0:m)    ... Koeffizienten des Polynoms g(x)
130   !               epsilon   ... zulässige Abweichung
140   !                             bei der Wurzelberechnung
150   ! Ergebnis:     n-k       ... Grad           des Polynoms f_stern(x)
160   !               m-k       ... Grad           des Polynoms g_stern(x)
170   !               k         ... Anzahl der gemeinsamen Nullstellen
180   !                             von f(x) und g(x) = Zahl der Kürzungen
190   !               c(0:n-k)  ... Koeffizienten des Polynoms f_stern(x)
200   !               d(0:n-k)  ... Koeffizienten des Polynoms g_stern(x)
210   !               e(0:k)    ... Koeffizienten des Polynoms h(x) mit e(k)=1
220   !
230   ! Programmierer:   Jürgen Schwarz     Datum/Variante:   06.11.84 / 02
240   ! Programm-Name:          Kuerzg      Speichermedium: Kassetten 57/58
250   !
260     INTEGER I,J,Mm,Nn
270     REAL A_stern(0:N),B_stern(0:M)
280     REAL Null_f(1:MAX(1,N)),Null_g(1:MAX(1,M)),Nullstelle(0:1)
290     !
300     IF (N<0) OR (M<0) OR (Epsilon<0) THEN PAUSE      ! widersprüchliche Daten
310     REDIM A(0:N),B(0:M)
320     MAT A_stern=A
330     MAT B_stern=B
340     K=0
350     MAT C=ZER
360     MAT C=A_stern
370     MAT D=ZER
380     MAT D=B_stern
390     MAT E=ZER
400     MAT E=CON(0:0)                                  ! Initialisiert e(0) mit 1
410     IF (N=0) OR (M=0) THEN SUBEXIT                  ! Kürzung unmöglich
420     !
430     Nullstelle(1)=1
440     CALL Newton_mod(N,A_stern(*),Null_f(*),Epsilon)  ! Wurzelberechnung f(x)
450     CALL Newton_mod(M,B_stern(*),Null_g(*),Epsilon)  ! Wurzelberechnung g(x)
460     Nn=N
470     Mm=M
480     I=0
490     WHILE I<Nn
500       I=I+1
510       J=0
520       REPEAT
530         J=J+1
540       UNTIL (ABS(Null_f(I)-Null_g(J))<=10*Epsilon) OR (J>=Mm)
550       IF ABS(Null_f(I)-Null_g(J))<=10*Epsilon THEN
560         Nullstelle(0)=-.5*(Null_f(I)+Null_g(J))
570         CALL Produkt(K,1,E(*),Nullstelle(*),E(*))
580         K=K+1                                  ! gemeinsame Nullstelle gefunden
590         CALL Streichen(Nn,I,C(*),Null_f(*))
600         CALL Streichen(Mm,J,D(*),Null_g(*))
610       END IF
620     END WHILE
630     SUBEXIT
```

Listing 9: Fortsetzung

```
640     SUB Streichen(INTEGER N,I,REAL A(*),N(*))
650       INTEGER L
660       REAL A,B
670       !
680       A=A(N)
690       A(N)=0
700       FOR L=N-1 TO 0 STEP -1
710         B=A(L)
720         A(L)=A+N(I)*A(L+1)                    ! Division durch die Nullstelle
730         A=B
740       NEXT L
750       N=N-1
760       REDIM A(0:N)
770       !
780       FOR L=I TO N
790         N(L)=N(L+1)                           ! Streichen der Nullstelle
800       NEXT L
810       I=I-1
820     SUBEND
830   SUBEND
```

Listing 10: Bestimmung des größten gemeinsamen Teilers von zwei Polynomen mit komplexen Koeffizienten durch Zerlegung in Linearkombinationen

```
10    SUB Komplex_kuerzg(INTEGER N,M,K,REAL A(*),B(*),C(*),D(*),E(*),F(*),Eps)
20    !
30    ! SUB-Programm zur Berechnung der gemeinsamen Nullstellen von zwei
40    ! Polynomen mit komplexen Koeffizienten f(z) und g(z) und Bestimmung
50    ! des größten gemeinsamen Teilers [ggT] über eine Wurzelbestimmung
60    ! mit Ausgabe der Ergebnisse über die Eingabefelder:
70    !                f(z) = f_stern(z) * h(z)
80    !                g(z) = g_stern(z) * h(z)
90    !
100   ! Eingabedaten: n         ... Grad           des Polynoms f(z)
110   !               m         ... Grad           des Polynoms g(z)
120   !               a(0:n)    ... Realteil      der Koeffizienten von f(z)
130   !               b(0:n)    ... Imaginärteil der Koeffizienten von f(z)
140   !               c(0:m)    ... Realteil      der Koeffizienten von g(z)
150   !               d(0:m)    ... Imaginärteil der Koeffizienten von g(z)
160   !               eps       ... zulässige Abweichungen
170   !                             bei der Wurzelberechnung
180   ! Ergebnis:     n         ... Grad           des Polynoms f_stern(z)
190   !               m         ... Grad           des Polynoms g_stern(z)
200   !               k         ... Anzahl der gemeinsamen Nullstellen
210   !                             von f(z) und g(z) = Zahl der Kürzungen
220   !               a(0:n)    ... Realteil      der Koeffizienten von f_stern(z)
230   !               b(0:n)    ... Imaginärteil der Koeffizienten von f_stern(z)
240   !               c(0:m)    ... Realteil      der Koeffizienten von g_stern(z)
250   !               d(0:m)    ... Imaginärteil der Koeffizienten von g_stern(z)
260   !               e(0:k)    ... Realteil      der Koeffizienten von h(z)
270   !               f(0:k)    ... Imaginärteil der Koeffizienten von h(z)
280   !
290   ! Programmierer:   Jürgen Schwarz     Datum/Variante:   26.11.84 / 01
300   ! Programm-Name:          Kpl_Ku      Speichermedium: Kassetten 57/58
310   !
```

Listing 10: Fortsetzung

```
320     INTEGER I,J,Boo_r,Boo_i
330     REAL F_r(1:MAX(1,N)),F_i(1:MAX(1,N)),G_r(1:MAX(1,M)),G_i(1:MAX(1,M))
340     REAL N_r(0:1),N_i(0:1)
350     !
360     IF (N<0) OR (M<0) OR (Eps<0) THEN PAUSE          ! widersprüchliche Daten
370     REDIM A(0:N),B(0:N),C(0:M),D(0:M)
380     K=0
390     MAT E=ZER
400     MAT F=ZER
410     MAT E=CON(0:0)                                   ! Initialisiert E(0) mit 1
420     REDIM F(0:0)
430     IF (N=0) OR (M=0) THEN SUBEXIT                   ! Kürzung unmöglich
440     !
450     N_r(1)=1
460     N_i(1)=0
470     CALL Siljak(N,100,A(*),B(*),Eps,Eps,F_r(*),F_i(*))     ! Wurzeln von f(x)
480     CALL Siljak(M,100,C(*),D(*),Eps,Eps,G_r(*),G_i(*))     ! Wurzeln von g(x)
490     I=0
500     WHILE I<N
510       I=I+1
520       J=0
530       REPEAT
540         J=J+1
550         Boo_r=(ABS(F_r(I)-G_r(J))<=10*Eps)
560         Boo_i=(ABS(F_i(I)-G_i(J))<=10*Eps)
570       UNTIL Boo_r AND Boo_i OR (J>=M)
580       IF Boo_r AND Boo_i THEN                 ! gemeinsame Nullstelle gefunden
590         N_r(0)=-.5*(F_r(I)+G_r(J))
600         N_i(0)=-.5*(F_i(I)+G_i(J))
610         CALL Komplex_produkt(E(*),F(*),N_r(*),N_i(*),E(*),F(*))
620         K=K+1
630         CALL Streichen(N,I,A(*),B(*),F_r(*),F_i(*))
640         CALL Streichen(M,J,C(*),D(*),G_r(*),G_i(*))
650         I=I-1
660       END IF
670     END WHILE
680     SUBEXIT
690     SUB Streichen(INTEGER N,I,REAL A_r(*),A_i(*),N_r(*),N_i(*))
700       INTEGER L
710       REAL A,B,C,D
720       !
730       A=A_r(N)
740       B=A_i(N)
750       A_r(N)=A_i(N)=0
760       FOR L=N-1 TO 0 STEP -1
770         C=A_r(L)
780         D=A_i(L)
790         A_r(L)=A+N_r(I)*A_r(L+1)-N_i(I)*A_i(L+1)
800         A_i(L)=B+N_r(I)*A_i(L+1)+N_i(I)*A_r(L+1)
810         A=C
820         B=D
830       NEXT L                                  ! Division durch die Nullstelle
840       N=N-1
850       REDIM A_r(0:N),A_i(0:N)
860       !
870       FOR L=I TO N
880         N_i(L)=N_i(L+1)                       ! Streichen der Nullstelle
890         N_r(L)=N_r(L+1)                       ! Streichen der Nullstelle
900       NEXT L
910     SUBEND
920   SUBEND
```

Abstract

Jürgen Schwarz

136 Behandlung von Polynomen Teil 3

In diesem dritten Teil der fortlaufenden Serie bringt der Autor Subprogramme für die Division von zwei Polynomen und für die Bestimmung des größten gemeinsamen Teilers von zwei Polynomen.