STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING TOEGEPASTE WISKUNDE

Cursus

Algebraïsche Bewerkingen met behulp van de Computer

door

Dr. R.P. van de Riet

The Mathematical Centre at Amsterdam, founded the 11th of February, 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.) and the Central Organization for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

Inhoudsopgave

Woord vooraf.

Deze cursus is bedoeld als een informele inleiding voor [1] hoofdstuk 3, alwaar op preciese wijze de syntax en de semantiek van een formule - programma wordt beschreven.

1. De elementaire bouwstenen.

Gesteld, men staat voor het volgende probleem:

> Bewijs dat $\frac{z+1}{z-1}$ zuiver imaginair is als de absolute waarde van de complexe variabele z gelijk is aan 1.

1e aanpak:

Substitueer een aantal waarden: z = +1, +i, -1, -i en, bijvoorbeeld, $z = \frac{1}{2}\sqrt{2} + i\,\frac{1}{2}\sqrt{2}$.

Vervolgens construeert men een formule - programma als volgt:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 1
(bovengrens aantal formules: 8192,
 bovengrens aantal reele getallen: 100,
 bovengrens aantal complexe getallen: 100,
 bovengrens aantal afgebroken machtreeksen: 0,
 bovengrens graad afgebroken machtreeksen: 0,
 absolute nauwkeurigheid: ₁₀-10,
 relatieve nauwkeurigheid: ₁₀-10,
 bovengrens aantal identifiers: 2,
 bovengrens aantal speciale afgeleiden: 0 )
z:= 1; f:= (z + 1)/(z - 1);
OUTPUT R(f(1):= f);
z:= i ; f:= (z + 1)/(z - 1);
OUTPUT R(f(i):= f);
z:= -1; OUTPUT R(f(-1):= (z + 1)/(z - 1));
z:= - i ; OUTPUT R(f(-i ):= (z + 1)/(z - 1));
NLCR; OUTPUT R(?:= ((1 + i)/sqrt(2) + 1)/((1 + i)/sqrt(2) - 1));
END;
```

Afgezien van de heading is het ongeveer duidelijk wat het formule-programma (i.h.v: f.p.) moet doen.

Opmerkelijke verschillen met een ALGOL 60 programma zijn:

1e geen begin, end;

2e geen declaraties;

3e complexe getallen mogen wel;

4e een willekeurige tekst als parameter van OUTPUT R:

Als de willekeurige tekst de tekst "?" is dan wordt de waarde van de expressie na het ":=" teken zonder meer geprint;
als de willekeurige tekst niet de tekst "?" is, dan wordt eerst op een nieuwe regel overgegaan, vervolgens wordt de letterlijke tekst tot en met het ":=" teken geprint, vervolgens wordt de waarde van de expressie na het ":=" teken geprint en tenslotte wordt een ";" geprint.

Behalve dat de resultaten geprint worden, worden alle resultaten ook in een band geponst. Deze band is gebruikt om de resultaten in deze syllabus te reproduceren. Als een formule te lang is voor één regel, dan last het systeem geen overgangen op nieuwe regel in; de met de hand aangebrachte overgangen op nieuwe regel zijn te herkennen aan het inspringen van de tekst.

5e geen "for - statement", want:

```
"for z := 1, i, -1, -i, (1 + i) /  sqrt(2) do
 OUTPUT R (f := (z + 1) / (z - 1));"
```

is verboden. Ook is een "goto - statement" of een conditionele - statement" verboden.

Het resultaat van het f.p. is als volgt:

f(1):= $.1_{10}11$;
f(i):= -i;
f(-1):= 0;
f(-i):= i;
-i×$.241421356237_{10}1$
ready
line number = 8 execution time = 4 sec

Bovenstaand programma had weinig te maken met algebraïsche bewerkingen, behalve dat er met de imaginaire eenheid i werd gerekend.

2e aanpak:

Substitueer $z = e^{i\phi}$, met ϕ willekeurig.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 2
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 3, 0)
z:= exp(i × phi); f:= (z + 1)/(z - 1);
OUTPUT R(f als functie van phi:= f);
END;
```

met als resultaat:

```
f als functie van phi:= (exp(i×phi)+1)/(exp(i×phi)-1);
ready
line number = 3 execution time = 4 sec
```

Veel succes is er niet geboekt.

3e aanpak:

Substitueer $z = e^{i\phi}$, met ϕ willekeurig en bereken

$w = \bar{f} + f$, waarin $\bar{f}$ de complex geconjugeerde van f is.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 3
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 3, 0)
PR STRING(resultaten fp 3);
NLCR; PR STRING(dit keer boeken we succes.);
z:= exp(i × phi); f:= (z + 1)/(z - 1);
OUTPUT R(Reele deel van f:= (CC(f) + f)/2);
END;
```

met als resultaat:

```
resultaten fp 3
dit keer boeken we succes.
Reele deel van f:= 0;
ready
line number = 5 execution time = 8 sec
```

Dit keer is er succes geboekt: kennelijk geldt

$$\bar{f} = -f$$

zodat Re f = 0.

<u>Conclusie</u>: we zien dat bij executie van het programma de waarden van de variabelen, zoals z en f, niet altijd getallen moeten zijn. Er mogen expressies met symbolen blijven staan.

De computer ontdekt dat

$$\bar{f} = (e^{-i\phi} + 1) \;/\; (e^{-i\phi} - 1)$$

en dat

$$\bar{f} + f = \frac{e^{-i\phi} + 1}{e^{-i\phi} - 1} + \frac{e^{i\phi} + 1}{e^{i\phi} - 1} = 0.$$

We hebben bovendien kennis gemaakt met de PR STRING operator.

Het lijkt merkwaardig dat we niet de symbolen "‘" en "’" of "≮" en "≯" gebruiken ter afsluiting van de string ; immers, hoe kunnen we nu een ")" via de string laten printen.

Het antwoord is eenvoudig:

Om een ")" te laten printen moet men
twee ")" tekens in de string plaatsen.

Voorbeeld:

" PR STRING (a:= sin(x)); PR STRING (b:= cos(x)));));");

heeft als effect dat het volgende geprint wordt:

"a:= sin(x); PR STRING (b:= cos(x));)"

Het voordeel van deze aanpak is dat nu elke willekeurige string geprint kan worden en dat kan met de string quotes "‘" en "’" niet. We zullen deze mogelijkheid ook hard nodig hebben als we met een f.p. een andere f.p. willen laten genereren.

<u>4e aanpak</u>:

Bepaal het argument van $\frac{z + 1}{z - 1}$:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 4
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 6, 0)
PR STRING(resultaten fp 4);
z:= exp(i × phi); f:= (z + 1)/(z - 1);
Im:= (CC(f) - f)/(2 × i); Re:= (CC(f) + f)/2;
OUTPUT R(het argument van f:= arctan(Im/Re));
```

```
OUTPUT R(het complement van het argument van f:=
  pi/2 - arctan(Re/Im));
END;
```

resultaten fp 4

het argument van f:= $-\underline{i}/2\times\ln((6-.20000000004_{10}11\times\exp(-\underline{i}\times phi)+.19999999996_{10}11\times$
$\exp(\underline{i}\times phi)+.10000000001_{10}11\times\exp(-\underline{i}\times2\times phi)-.9999999999_{10}10\times\exp(\underline{i}\times2\times phi))/$
$(6+.19999999996_{10}11\times\exp(-\underline{i}\times phi)-.20000000004_{10}11\times\exp(\underline{i}\times phi)-.9999999999_{10}10\times$
$\exp(-\underline{i}\times2\times phi)+.10000000001_{10}11\times\exp(\underline{i}\times2\times phi)));$

het complement van het argument van f:= $1/2\times pi$;

ready

line number = 7 execution time = 69 sec

Uit deze aanpak zien we dat oppassen geboden is. Het systeem ziet niet dat "iets" / 0 = ∞ , en kan dat natuurlijk ook niet; immers "iets zou 0 kunnen zijn. Bovendien zien we dat de arctan als een logarithme wordt behandeld.

Dit is nit altijd het geval; het hangt er vanaf of het systeem in de zogenaamde "expand" toestand is of niet.

In de "expand" toestand worden:

1e haakjes zoveel mogelijk weggewerkt.
   d.w.z. (a + b) * c wordt behandeld als a * c + b * c ;

2e de sin, cos, arctan en sqrt functies worden getransformeerd in exponentiele en logathmische functies;

3e quotienten worden vereenvoudigd;

4e afgebroken macht reeksen worden vereenvoudigd;

5e alvorens bij een OUTPUT R statement met printen wordt begonnen. wordt de expressie die geprint zal worden vereenvoudigd.

Is het systeem niet in de "expand" toestand, dan zijn de enige bewerkingen die betrekking hebben op het vereenvoudigen de zogenaamde triviale vereenvoudigingen:

$$a + 0 \to a \ , \ a * 0 \to 0 \ , \ a * 1 \to a \ , \ 0 \,/\, a \to 0 \quad \text{etc.}$$

bovendien worden elementaire functies van getallen uitgerekend (bijv. sin (0) → 0 , exp (i * 3.14159265359) → -1).
Het systeem verkeert in de "expand" toestand bij het begin van de verwerking van een f.p.

Om het systeem in de niet-"expand" toestand te zetten kan men de statement "NOT EXP" gebruiken.
Vanaf deze statement worden de volgende statements in de niet-"expand" toestand verwerkt. De overgang naar de "expand" toestand wordt bewerkstelligd door de statement "EXPAND"

We draaien f.p.3 opnieuw:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 5
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 5, 0)
PR STRING(resultaten fp 5);
NLCR; PR STRING(In de niet-'expand' toestand lukt het niet);
NOT EXP;
z:= exp(i × phi); f:= (z + 1)/(z - 1);
OUTPUT R(w1:= CC(f) + f);
NLCR; PR STRING(het is gevaarlijk om de, in de niet-'expand' toestand
behandelde, f in de herstelde 'expand' toestand te gebruiken);
EXPAND;
OUTPUT R(w2:= CC(f) + f);
END;
```

```
resultaten fp 5
In de niet-'expand' toestand lukt het niet
w1:= (exp((-i)×phi)+1)/(exp((-i)×phi)-1)+(exp(i×phi)+1)/(exp(i×phi)-1);
het is gevaarlijk om de, in de niet-'expand' toestand
behandelde, f in de herstelde 'expand' toestand te gebruiken
```

w2:= ((exp(i×phi)-1)×exp(-i×phi)+(exp(i×phi)-1)-exp(i×phi)+exp(-i×phi))/((exp(i×phi)-
1)×exp(-i×phi)-(exp(i×phi)-1));
ready
line number = 10 execution time = 14 sec

N.B.1. in de "expand" toestand wordt de formule "a - b" behandeld als " a + (-1) * b ", terwijl in de niet-"expand" toestand "a - b" wordt behandeld als " a - b ".
N.B.2. de niet-"expand" toestand is van belang bij "OUTPUT C - statements", bij berekeningen van afgeleiden, en bij het volgende:
Zoals reeds vermeld, worden in de "expand" toestand quotienten en afgebroken macht reeksen automatisch vereenvoudigd; ook "OUTPUT R" bewerkstelligt een vereenvoudiging. Om nu een dubbele vereenvoudiging te voorkomen kan men de "OUTPUT R" statement in de niet-"expand" toestand aanroepen; het effect is tijd- en geheugenbesparing, maar het geprinte resultaat is niet zo mooi.
Voorbeeld:

Formule programma t.b.v. cursus ABC. RPR 250968/fp 6
(8192, 100, 100, 0, 0, $_{10}$-10, $_{10}$-10, 3, 0)
PR STRING(resultaten fp 6);
z:= (u + i)/(u - i); f:= (z + i)/(z - i);
OUTPUT R(f:= f);
NOT EXP; OUTPUT R(een minder fraaie
f:= f);
END;

resultaten fp 6
f:= ((1+i)×u+1+i)/((1-i)×u+i-1);
een minder fraaie
f:= ((1+i)×u+(1+i))/((1-i)×u+(i-1));
ready
line number = 6 execution time = 9 sec

Nog een voorbeeld dat het verschil tussen de "expand" toestand en de niet-"expand" toestand laat zien:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 7
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 7, 0)
PR STRING(resultaten fp 7);
NOT EXP;
z:= (u + i)/(u - i); f:= (z + i)/(z - i); OUTPUT R(f1:= f);
f:= (a + b) × (a - b); OUTPUT R(f2:= f);
f:= (x↑2 - 2 × x + 1)/(x↑2 - 1); OUTPUT R(f3:= f);
f:= sin(x+y)↑2  + cos(y+x)↑2; OUTPUT R(f4:= f);
f:= exp(ln(x)); OUTPUT R(f5:= f);
EXPAND; NLCR; PR STRING(nu dezelfde opgaven in de 'expand'-toestand);
z:= (u + i)/(u - i); f:= (z + i)/(z - i); OUTPUT R(f1:= f);
f:= (a + b) × (a - b); OUTPUT R(f2:= f);
f:= (x↑2 - 2 × x + 1)/(x↑2 - 1); OUTPUT R(f3:= f);
f:= sin(x+y)↑2  + cos(y+x)↑2; OUTPUT R(f4:= f);
f:= exp(ln(x)); OUTPUT R(f5:= f);
END;
```

```
resultaten fp 7
f1:= ((u+i)/(u-i)+i)/((u+i)/(u-i)-i);
f2:= (a+b)×(a-b);
f3:= (x↑2-2×x+1)/(x↑2-1);
f4:= sin(x+y)↑2+cos(y+x)↑2;
f5:= x;
nu dezelfde opgaven in de 'expand'-toestand
f1:= ((1+i)×u+1+i)/((1-i)×u+i-1);
f2:= a↑2-b↑2;
f3:= (-1/2×x+1/2)/(-1/2×x-1/2);
f4:= 1;
f5:= x;
ready
line number = 14 execution time = 21 sec
```

Als slot van dit hoofdstuk behandelen we het volgende programma:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 8
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 3, 0)
PR STRING(resultaten fp 8);
f:= (z + 1)/(z - 1); z:= x + 1; OUTPUT R(f:= f); END;
```

Indien men verwacht dat er als output: " f : = (x + 2) / x ";
wordt geprint, komt men bedrogen uit.
Het resultaat is:

```
resultaten fp 8
f:= (z+1)/(z-1);
ready
line number = 2 execution time = 4 sec
```

Men moet bovenstaande situatie vergelijken met het volgende ALGOL 60 programma:

```
begin real x, z, f; x : = 0 ; z : = 0 ;
      f : = (z + 1) / (z - 1) ; z : = x + 1 ; print (f)
end.
```

Kennelijk is het zo dat als "f" eenmaal een waarde heeft gekregen, n.l. de formule " (z + 1) / (z - 1) ", dan behoudt hij deze waarde. M.b.v. een substitutie (zie volgende hoofdstuk) is de waarde van "f" wel te veranderen (en, natuurlijk, m.b.v. een nieuwe assignment).

2. Afgeleiden, substituties.

De afgeleide van f (z) berekenen we met de operator "DER".
Voorbeeld:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp 9
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 3, 0)
PR STRING(resultaten fp 9);
f:= (z + 1)/(z - 1); derf:= DER(f,z); OUTPUT R(derf:= derf);
NLCR; PR STRING(of als volgt:);
OUTPUT R(derivative:= DER((z + 1)/(z - 1), z));
END;
```

```
resultaten fp 9
derf:= (-2)/(z↑2-2×z+1);
of als volgt:
derivative:= (-2)/(z↑2-2×z+1);
ready
line number = 5 execution time = 6 sec
```

Als $z = e^{i\phi}$ en $f(z) = (1 + z) / (1 - z)$ dan kunnen we $\frac{df}{d\phi}$ als volgt berekenen:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp10
(8192, 100, 100, 0, 0, ⏨-10, ⏨-10, 4, 0)
PR STRING(resultaten fp10);
z:= exp(i × phi);
f:= (z + 1)/(z - 1); derfphi:= DER(f,phi); OUTPUT R(derfphi:= derfphi);
NLCR; PR STRING(of als volgt:);
OUTPUT R(derfphi2:= DER(f,z) × DER(z,phi));
END;
```

```
resultaten fp10
derfphi:= -i×2×exp(i×phi)/(exp(i×2×phi)-2×exp(i×phi)+1);
of als volgt:
derfphi2:= 0;
ready
line number = 6 execution time = 10 sec
```

De tweede formule voor afgeleide:

"DER (f, z) * DER (z, phi)"

is gevaarlijk om te gebruiken omdat z geen variabele (algebraic variabele) is, maar de naam van een formule (expressie).

We zien dat DER (f, z) fout berekend wordt; ook in het volgende voorbeeld gaat het fout:

Formule programma t.b.v. cursus ABC. RPR 250968/fp11
(8192, 100, 100, 0, 0, $_{10}-10$, $_{10}-10$, 3, 0)
PR STRING(resultaten fp11);
z:= x + 1; f:= (z + 1)/(z - 1); OUTPUT R(derf:= DER(f,z));
END;

resultaten fp11
derf:= 0;
ready
line number = 3 execution time = 3 sec

De reden van het fout gaan is dat f niet meer expliciet van z afhangt, maar als de formule "(x + 2) / x" wordt behandeld.

N.B. In de niet-"expand" toestand zou bovenstaand sommetje als volgt gaan:

Formule programma t.b.v. cursus ABC. RPR 250968/fp12
(8192, 100, 100, 0, 0, $_{10}-10$, $_{10}-10$, 2, 0)
PR STRING(resultaten fp12);
NOT EXP; z:= x + 1; OUTPUT R(der:= DER((z + 1)/(z - 1), z));
END;

resultaten fp12
der:= (x+1-1-(x+1+1))/((x+1-1)×(x+1-1));
ready
line number = 3 execution time = 3 sec

De eerste parameter van DER mag een willekeurige formule zijn. De tweede, in principe, alleen maar een variabele. Verder mag DER in een formule voorkomen: zoals in:

Formule programma t.b.v. cursus ABC. RPR 250968/fp13
(8192, 100, 100, 0, 0, $_{10}-10$, $_{10}-10$, 2, 0)
PR STRING(resultaten fp13);
z:= sin(x); OUTPUT R(differentiaal identiteit voor sin(x) is
z'' + z := DER(DER(z,x),x) + z);
END;

```
resultaten fp13
differentiaal identiteit voor sin(x) is
z'' + z := 0;
ready
line number = 4 execution time = 4 sec
```

Als een functie f niet van x afhangt, dan levert "DER" de waarde nul. Als de functie f wel afhangt van de variabele z, welke op zijn beurt weer impliciet van x afhangt, dan kan men dit aangeven door een "speciale - afgeleide - statement".

Voorbeeld:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp14
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 4, 1)
PR STRING(resultaten fp14);
SPEC DER(x, z,z accent);
f:= (z + 1)/(z - 1); OUTPUT R(afgeleide naar x:= DER(f,x));
END;
```

```
resultaten fp14
afgeleide naar x:= -2×zaccent/(z↑2-2×z+1);
ready
line number = 4 execution time = 5 sec
```

Als het systeem z tegenkomt en z wil differentieren naar x, dan weet het systeem dat de afgeleide naar z gegeven is door z'.

Een differentiaal vergelijking, bijv. $\frac{dz}{dx} = x^2 + z^2$, is op deze wijze gemakkelijk te behandelen:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp15
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 4, 3)
PR STRING(resultaten fp15);
SPEC DER(x, z,x↑2 + z↑2); OUTPUT R(z acc:= DER(z,x));
OUTPUT R(z dacc:= DER(DER(z,x),x));
SPEC DER(x, z,z acc, z acc,z, y,cos(x));
OUTPUT R(z acc + dy/dx:= DER(z + y,x));
OUTPUT R(z 4dacc:= DER(DER(DER(DER(z,x),x),x),x));
END;
```

```
resultaten fp15
z acc:= x↑2+z↑2;
z dacc:= 2×x↑2×z+2×z↑3+2×x;
z acc + dy/dx:= zacc+1/2×exp(i×x)+1/2×exp(-i×x);
z 4dacc:= z;
ready
line number = 7 execution time = 5 sec
```

Opmerking 1. een volgende "speciale - afgeleide - statement" vernietigt het effect van een voorgaande.

2. speciale afgeleiden kunnen maar naar één variabele worden gespecificeerd. Formele partiele afgeleiden kunnen dus niet gebruikt worden.

De "speciale - afgeleide - statements" lenen zich goed voor een Taylor-reeks berekening van (de) functie(s) die aan een gegeven differentiaal vergelijking voldoet (aan een systeem van differentiaal vergelijkingen voldoen).
Daartoe is eerst nodig de substitutie operator "SUBST" in te voeren.

Als f een formule is in de variabelen $x_1, \ldots, x_n$ en $y_1, \ldots, y_n$ zijn andere formules, dan is g = SUBST $(f, x_1, y_1, \ldots, x_n, y_n)$ een formule waarin de $x_1, \ldots, x_n$ vervangen zijn door $y_1, \ldots, y_n$.

Of: als $f = f\ (x_1, \ldots, x_n)$ dan geldt:

$g = f\ (y_1, \ldots, y_n)$

Voorbeeld (zie f.p.1 en f.p.3):

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp16
(8192, 100, 100, 0, 0, ⏨-10, ⏨-10, 3, 0)
PR STRING(resultaten fp16);
f:= (z + 1)/(z - 1); f:= (CC(f) + f)/2;
OUTPUT R(f(i):= SUBST(f, z,i));
OUTPUT R(f(exp(i phi)):= SUBST(f, z, exp(i × phi)));
END;
```

```
resultaten fp16
f(i):= -i;
f(exp(i phi)):= (exp(i×phi)+1)/(exp(i×phi)-1);
ready
line number = 5 execution time = 10 sec
```

De differentiaal vergelijking $y'' = x^2 + y^2$, $y\ (0) = a$, $y'\ (0) = b$ kan nu als volgt m.b.v. een Taylor-reeks ontwikkeling tot oplossing (benaderde) gebracht worden:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp17
(8192, 100, 100, 0, 0, ⏨-10, ⏨-10, 6, 2)
PR STRING(resultaten fp17);
SPEC DER(x, y,y acc, y acc,x↑2 + y↑2);
der:= y;
OUTPUT R(c0:= SUBST(der, y,a, y acc,b, x,0)); der:= DER(der,x);
OUTPUT R(c1:= SUBST(der, y,a, y acc,b, x,0)); der:= DER(der,x);
OUTPUT R(c2:= SUBST(der, y,a, y acc,b, x,0)/2); der:= DER(der,x);
OUTPUT R(c3:= SUBST(der, y,a, y acc,b, x,0)/6); der:= DER(der,x);
OUTPUT R(c4:= SUBST(der, y,a, y acc,b, x,0)/24); der:= DER(der,x);
OUTPUT R(c5:= SUBST(der, y,a, y acc,b, x,0)/120); der:= DER(der,x);
OUTPUT R(c6:= SUBST(der, y,a, y acc,b, x,0)/(120×6));
END;
```

```
resultaten fp17
c0:= a;
c1:= b;
c2:= 1/2×a↑2;
c3:= 1/3×a×b;
c4:= 1/12×a↑3+1/12×b↑2+1/12;
c5:= 1/12×a↑2×b;
c6:= 1/72×a↑4+1/36×a×b↑2+1/180×a;
ready
line number = 11 execution time = 13 sec
```

N.B. in bovenstaand voorbeeld missen we de for-statement; in het volgende hoofdstuk worden afgebroken machtreeksen behandeld die voor dit voorbeeld nuttige diensten kunnen bewijzen.

Een minder triviaal voorbeeld voor het gebruik van de "speciale - afgeleide - statement" is het volgende:

Bewijs dat

$$\phi(x) = x^{1/2} J_{1/3} \left(\frac{2}{3} i \, x^{3/2} \right)$$

voldoet aan:

$$\frac{d^2 \phi}{d x^2} + x \phi = 0 \, ,$$

waarin J_n de Bessel functie is welke voldoet aan:

$$\frac{d^2 J_n(z)}{d z^2} + \frac{1}{z} \frac{d J_n(z)}{d z} + \left(1 - \frac{n^2}{z^2}\right) J_n(z) = 0$$

<u>Oplossing</u> Voer in:

$$s = \sqrt{x} ,$$

$$z(x) = \frac{2}{3} i\, s^3 ,$$

$$n = 1/3 ,$$

$$I(x) = J_n(z(x)),$$

$$\frac{d\,I(x)}{d\,x} = \left.\frac{d\,J_n(z)}{d\,z}\right|_{z = z(x)} \cdot \frac{d\,z}{d\,x}$$

$$= I_1(x)\,\frac{d\,z}{d\,x} ,$$

$$\frac{d\,I_1(x)}{d\,x} = \left.\frac{d^2\,J_n(z)}{d\,z^2}\right|_{z = z(x)} \cdot \frac{d\,z}{d\,x}$$

$$= \left.\left(-\frac{1}{z}\,\frac{d\,J_n(z)}{d\,z} + \left(1 - \frac{n^2}{z^2}\right) J_n(z)\right)\right|_{z = z(x)} \cdot \frac{d\,z}{d\,x}$$

In het programma worden de speciale afgeleiden van s, z, I, en I_1 naar x gespecificeerd.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp18
(8192, 200, 200, 0, 0, ₁₀-10, ₁₀-10, 7, 4)
PR STRING(resultaten fp18);
SPEC DER(x,
  s, .5/s,
  z, i × s,
  I, I1 × DER(z,x),
  I1, - SUBST((I1/z + (1 - n↑2/z↑2) × I) × DER(z,x),
              z, 2/3 × i × s↑3,
              n, 1/3)
  ); phi:= s × I; OUTPUT R(phi'' + x phi:= DER(DER(phi,x),x) + s↑2 × phi);
END;
```

```
resultaten fp18
phi'' + x phi:= 2×s↑3×I;
ready
line number = 10 execution time = 45 sec
```

Opmerking Het verkregen resultaat laat zien dat we een tekenfout gemaakt hebben; ϕ voldoet aan $\frac{d^2\,\phi}{d\,x^2} - x\,\phi = 0$.

3. Afgebroken machtreeksen en het oplossen van vergelijkingen.

Het systeem kan met afgebroken machtreeksen manipuleren.

Zij m_1 de afgebroken machtreeks:

$$m_1 = a_0 + a_1x + a_2x^2 + a_3x^3 + O(x^4),$$

en

$$m_2 = b_0 + b_1x + b_2x^2 + b_3x^3 + O(x^4),$$

dan is het resultaat van de berekening van:

de som, het verschil, het product, het quotient

van m_1 en m_2 weer een afgebroken machtreeks.

Bovendien is $m_1 + f$, $m_1 - f$, $m_1 * f$, m_1 / f en $m_1 \uparrow n$, met n geheel niet-negatief, ook een afgebroken machtreeks.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp19
(8192, 100, 100, 100, 3, ⏨-10, ⏨-10, 13, 0)
PR STRING(resultaten fp19);
m1:= TPS(x,a0,a1,a2,a3); m2:= TPS(x,b0,b1,b2,b3);
OUTPUT R(m1 + m2:= m1 + m2);
OUTPUT R(m1 - m2:= m1 - m2);
OUTPUT R(m1 × m2:= m1 × m2);
OUTPUT R(m1/m2:= m1/m2);
OUTPUT R(m1↑3:= m1↑3);
OUTPUT R(m1 × c + d:= m1 × c + d);
END;
```

```
resultaten fp19
m1 + m2:= a0+b0+(a1+b1)×x+(a2+b2)×x↑2+(a3+b3)×x↑3+O(x↑4);
m1 - m2:= a0-b0+(a1-b1)×x+(a2-b2)×x↑2+(a3-b3)×x↑3+O(x↑4);
m1 × m2:= a0×b0+(a0×b1+a1×b0)×x+(a0×b2+a1×b1+a2×b0)×x↑2+(a0×b3+a1×b2+a2×b1+
        a3×b0)×x↑3+O(x↑4);
m1/m2:= a0/b0+(-a0×b1+a1×b0)/b0↑2×x+(-a0×b0×b2+a0×b1↑2-a1×b0×b1+a2×b0↑2)/
        b0↑3×x↑2+(-a0×b0↑2×b3+2×a0×b0×b1×b2-a0×b1↑3-a1×b0↑2×b2+a1×b0×b1↑2-
        a2×b0↑2×b1+a3×b0↑3)/b0↑4×x↑3+O(x↑4);
m1↑3:= a0↑3+(3×a0↑2×a1)×x+(3×a0↑2×a2+3×a0×a1↑2)×x↑2+(3×a0↑2×a3+6×a0×a1×a2+
        a1↑3)×x↑3+O(x↑4);
m1 × c + d:= a0×c+d+(a1×c)×x+(a2×c)×x↑2+(a3×c)×x↑3+O(x↑4);
ready
line number = 9 execution time = 76 sec
```

<u>Opmerking 1.</u> Het resultaat van een operatie op twee machtreeksen m_1 en m_2 van verschillende graad is een machtreeks met als graad: het minimum van de graden van m_1 en m_2.

<u>Opmerking 2.</u> Blijkt tijdens executie van het programma dat alle coëfficiënten van een machtreeks, op de nulde coëfficiënt na, gelijk aan nul zijn, en is deze nulde coëfficiënt zelf geen machtreeks, dan wordt deze machtreeks niet meer als machtreeks beschouwd, maar wordt gelijk gemaakt aan zijn nulde coëfficiënt.

<u>Opmerking 3.</u> Wordt er in het programma met machtreeksen in, zeg, x gewerkt dan wordt een "los voorkomende" x <u>niet</u> als de speciale machtreeks variabele herkend.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp20
(8192, 100, 100, 100, 3, ⏨-10, ⏨-10, 10, 0)
PR STRING(resultaten fp20);
m1:= TPS(x,a0,a1,a2,a3); m2:= TPS(x,b0,b1,b2);
OUTPUT R(m1 + m2:= m1 + m2);
m2:= TPS(x,b0,0,0,0); OUTPUT R(zie opm. 2  m2:= m2);
OUTPUT R(m1 - TPS(x,0,a1,a2,a3):= m1 - TPS(0,a1,a2,a3));
OUTPUT R(zie opm. 3
m1 - (a0 + a1 × x + a2 × x↑2 + a3 × x↑3):=
m1 - (a0 + a1 × x + a2 × x↑2 + a3 × x↑3));
END;
```

resultaten fp20

m1 + m2:= a0+b0+(a1+b1)×x+(a2+b2)×x↑2+O(x↑3);

zie opm. 2 m2:= b0;

m1 - TPS(x,0,a1,a2,a3):= a0-a1+(a1-a2)×x+(a2-a3)×x↑2+O(x↑3);

zie opm. 3

m1 - (a0 + a1 × x + a2 × x↑2 + a3 × x↑3):= -x↑3×a3-x↑2×a2-x×a1+a1×x+a2×x↑2+
a3×x↑3+O(x↑4);

ready

line number = 9 execution time = 7 sec

<u>Voorbeeld</u> bereken de machtreeksontwikkeling van

$$f(z) = (z+1)/(z-1) \quad \text{voor} \quad z = 0$$

en van

$$\tan(z) = \sin(z)/\cos(z) = (z - z^3/6 + z^5/120 + \ldots) / (1 - z^2/3 + z^4/24 - z^6/720 + \ldots).$$

Formule programma t.b.v. cursus ABC. RPR 250968/fp21

<u>(</u>8192, 100, 100, 100, 10, $_{10}$-10, $_{10}$-10, 3, 0<u>)</u>

PR STRING(resultaten fp21);

Z:= TPS(z,0,1,0,0,0,0,0,0,0,0,0); f:= (Z + 1)/(Z - 1);

OUTPUT R(de m. r. ontwikkeling van f:= f);

OUTPUT R(de m. r. ontwikkeling van tan(z)
:= TPS(z,0,1,0,-1/6,0,1/120,0,-1/5040)/
TPS(z,1,0,-1/2,0,1/24,0,-1/720,0));

END;

resultaten fp21

de m. r. ontwikkeling van f:= -1+(-2)×z+(-2)×z↑2+(-2)×z↑3+(-2)×z↑4+(-2)×z↑5+
(-2)×z↑6+(-2)×z↑7+(-2)×z↑8+(-2)×z↑9+(-2)×z↑10+O(z↑11);

de m. r. ontwikkeling van tan(z)
:= 0+1×z+0×z↑2+1/3×z↑3+0×z↑4+2/15×z↑5+0×z↑6+17/315×z↑7+O(z↑8);

ready

line number = 7 execution time = 6 sec

Machtreeksen in meerdere variabelen verkrijgt men door de coëfficiënten van een machtreeks gelijk te stellen aan andere machtreeksen.

Stel we hebben

$$m = \sum_{i,j,k} a_{ijk} \; x^i \, y^i \, z^k,$$

dan is m op te vatten als machtreeks in z:

$$m = \sum_k (\sum_j (\sum_i a_{ijk} \, x^i) \, y^i) \, z^k.$$

Voorbeeld: ontwikkel exp $(\sum a_{ijk} \; x^i \, y^i \, z^k)$

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp22
(8192, 100, 0, 1000, 3, ⏨-10, ⏨-10, 26, 0)
PR STRING(resultaten fp22);
m:=
  TPS(z,
    TPS(y,
      TPS(x,a000,a100,a200,a300),
      TPS(x,a010,a110,a210),
      TPS(x,a020,a120),
      TPS(x,a030)),
    TPS(y,
      TPS(x,a001,a101,a201),
      TPS(x,a011,a111),
      TPS(x,a021)),
    TPS(y,
      TPS(x,a002,a102),
      TPS(x,a102)),
    TPS(y,
      TPS(x,a003)));
OUTPUT R(m:= m);
OUTPUT R(exp:= 1 + (1 + (1/2 + 1/6 × m) × m) × m);
END;
```

```
resultaten fp22
m:= a000+a100×x+a200×x∧2+a300×x∧3+O(x∧4)+(a010+a110×x+a210×x∧2+O(x∧3))×y+
        (a020+a120×x+O(x∧2))×y∧2+a030×y∧3+O(y∧4)+(a001+a101×x+a201×x∧2+O(x∧3)+
        (a011+a111×x+O(x∧2))×y+a021×y∧2+O(y∧3))×z+(a002+a102×x+O(x∧2)+a102×
        y+O(y∧2))×z∧2+a003×z∧3+O(z∧4);
exp:=
array too small857   101   0   55   4
line number = 19 execution time = 153 sec
```

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp22a
(8192, 1000, 0, 500, 3, ⏨-10, ⏨-10, 24, 0)
PR STRING(resultaten fp22a);

zie voor de rest  fp22.
```

```
resultaten fp22a
m:= a000+a100×x+a200×x∧2+a300×x∧3+O(x∧4)+(a010+a110×x+a210×x∧2+O(x∧3))×y+(a02
  0+a120×x+O(x∧2))×y∧2+a030×y∧3+O(y∧4)+(a001+a101×x+a201×x∧2+O(x∧3)+(a011+a111
×x+O(x∧2))×y+a021×y∧2+O(y∧3))×z+(a002+a102×x+O(x∧2)+a102×y+O(y∧2))×z∧2+a003×
  z∧3+O(z∧4);
exp:= 1/6×a000∧3+1/2×a000∧2+a000+1+(1/2×a000∧2×a100+a000×a100+a100)×x+(1/2×
  a000∧2×a200+1/2×a000×a100∧2+a000×a200+1/2×a100∧2+a200)×x∧2+(1/2×a000∧2×a300+
  a000×a100×a200+1/6×a100∧3+a000×a300+a100×a200+a300)×x∧3+O(x∧4)+(1/2×a000∧2×
  a010+a000×a010+a010+(1/2×a000∧2×a110+a000×a100×a010+a000×a110+a100×a010+a110
  )×x+(1/2×a000∧2×a210+a000×a100×a110+a000×a200×a010+1/2×a100∧2×a010+a000×a210
  +a100×a110+a200×a010+a210)×x∧2+O(x∧3))×y+(1/2×a000∧2×a020+1/2×a000×a010∧2+a000
  ×a020+1/2×a010∧2+a020+(1/2×a000∧2×a120+a000×a100×a020+a000×a010×a110+1/2×a100
  ×a010∧2+a000×a120+a100×a020+a010×a110+a120)×x+O(x∧2))×y∧2+(1/2×a000∧2×a030+a000
  ×a010×a020+1/6×a010∧3+a000×a030+a010×a020+a030+(a000×a100×a030+a000×a010×a120
```

```
 +a000×a110×a020+a100×a010×a020+1/2×a010↑2×a110+a100×a030+a010×a120+a110×a020
 )×x+O(x↑2))×y↑3+O(y↑4)+(1/2×a000↑2×a001+a000×a001+a001+(1/2×a000↑2×a101+a000×
 a100×a001+a000×a101+a100×a001+a101)×x+(1/2×a000↑2×a201+a000×a100×a101+a000×
 a200×a001+1/2×a100↑2×a001+a000×a201+a100×a101+a200×a001+a201)×x↑2+O(x↑3)+(1/2
 ×a000↑2×a011+a000×a010×a001+a000×a011+a010×a001+a011+(1/2×a000↑2×a111+a000×
 a100×a011+a000×a010×a101+a000×a110×a001+a100×a010×a001+a000×a111+a100×a011+
 a010×a101+a110×a001+a111)×x+O(x↑2))×y+(1/2×a000↑2×a021+a000×a010×a011+a000×
 a020×a001+1/2×a010↑2×a001+a000×a021+a010×a011+a020×a001+a021+(a000×a100×a021
 +a000×a010×a111+a000×a110×a011+a000×a020×a101+a000×a120×a001+a100×a010×a01
 1+a100×a020×a001+1/2×a010↑2×a101+a010×a110×a001+a100×a021+a010×a111+a110×a
 011+a020×a101+a120×a001)×x+O(x↑2))×y↑2+O(y↑3))×z+(1/2×a000↑2×a002+1/2×a000×
 a001↑2+a000×a002+1/2×a001↑2+a002+(1/2×a000↑2×a102+a000×a100×a002+a000×a001×
 a101+1/2×a100×a001↑2+a000×a102+a100×a002+a001×a101+a102)×x+O(x↑2)+(1/2×a000
 ↑2×a102+a000×a010×a002+a000×a001×a011+1/2×a010×a001↑2+a000×a102+a010×a002+a
 001×a011+a102+(a000×a100×a102+a000×a010×a102+a000×a110×a002+a000×a001×a111+
 a000×a101×a011+a100×a010×a002+a100×a001×a011+a010×a001×a101+1/2×a110×a001↑2
 +a100×a102+a010×a102+a110×a002+a001×a111+a101×a011)×x+O(x↑2))×y+O(y↑2))×z↑2+
 (1/2×a000↑2×a003+a000×a001×a002+1/6×a001↑3+a000×a003+a001×a002+a003+(a000×
 a100×a003+a000×a001×a102+a000×a101×a002+a100×a001×a002+1/2×a001↑2×a101+a100
 ×a003+a001×a102+a101×a002)×x+O(x↑2)+(a000×a010×a003+a000×a001×a102+a000×a011
 ×a002+a010×a001×a002+1/2×a001↑2×a011+a010×a003+a001×a102+a011×a002+(a000×
 a110×a003+a000×a101×a102+a000×a011×a102+a000×a111×a002+a100×a010×a003+a100×a0
 01×a102+a100×a011×a002+a010×a001×a102+a010×a101×a002+a110×a001×a002+1/2×a001
 ↑2×a111+a001×a101×a011+a110×a003+a101×a102+a011×a102+a111×a002)×x+O(x↑2))×y+
 O(y↑2))×z↑3+O(z↑4);
ready
line number = 20 execution time = 307 sec
```

N.B. We merken op dat het zinvoller was geweest om bovenstaande a000 gelijk aan nul te kiezen.

We bespreken nu de combinatie van <u>substitutie</u> en <u>afgebroken machtreeksen</u>.

Het effect van de statement:

" g : = SUBST (f, a, b) ",

als f een afgebroken machtreeks: $\sum_{i=0}^{n} c_i x^i + 0\ (x^{n+1})$ is,
is als volgt:

1e als a = x

en als b een algebraïsche variabele, dan is g een afgebroken machtreeks:

$$g = \sum_{i=0}^{n} \text{SUBST}\ (c_i, a, b)\ b^i + 0\ (b^{n+1}); \quad (1)$$

en als b geen algebraïsche variabele is, dan is g niet de machtreeks (1) maar een som van n termen:

$$g = \text{SUBST}\ (c_0, a, b) + \text{SUBST}\ (c_1, a, b) * b + \ldots. + \text{SUBST}\ (c_n, a, b) * b^n.$$

N.B. er wordt door het systeem gebruik gemaakt van:

$$d_0 + d_1\ x + \ldots + d_n\ x^n = d_0 + (\ d_1 + (d_2 + \ldots + (d_{n-1} + d_n\ x)\ x \ldots.)\ x)\ x.$$

2e als a ≠ x, dan is g de machtreeks:

$$g = \sum_{i=0}^{n} \text{SUBST}\ (c_i, a, b)\ x^i + 0\ (x^{n+1}).$$

<u>Voorbeeld:</u> bereken ln (1 + x) - arctan (x) voor kleine x.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp23
(8192, 100, 100, 100, 3, ₁₀-10, ₁₀-10, 13, 0)
PR STRING(resultaten fp23);
NOT EXP; f:= ln(1 + x); g:= arctan(x);
  f1:= DER(f,x); g1:= DER(g,x);
  f2:= DER(f1,x); g2:= DER(g1,x);
  f3:= DER(f2,x); g3:= DER(g2,x);
```

```
EXPAND;
  LN:= SUBST(TPS(X,f,f1,f2/2,f3/6), x,0);
  AT:= SUBST(TPS(X,g,g1,g2/2,g3/6), x,0);
  OUTPUT R(LN:= LN); OUTPUT R(AT:= AT);
  LN M AT:= LN - AT;
  OUTPUT R(LN - AT:= LN M AT);
  OUTPUT R(nauwkeurige waarde voor x = ⏨-3:=
      SUBST(LN M AT, X, ⏨-3));
  OUTPUT R(onnauwkeurige waarde voor x = ⏨-3:=
      ln(1 + ⏨-3) - arctan(⏨-3));
END;
```

```
resultaten fp23
LN:= 0+1×X+(-1/2)×X↑2+1/3×X↑3+O(X↑4);
AT:= 0+1×X+0×X↑2+(-1/3)×X↑3+O(X↑4);
LN - AT:= 0+0×X+(-1/2)×X↑2+2/3×X↑3+O(X↑4);
nauwkeurige waarde voor x = ⏨-3:= -.49933333333⏨-6;
onnauwkeurige waarde voor x = ⏨-3:= -.49933284707⏨-6;
ready
line number = 16 execution time = 23 sec
```

<u>Voorbeeld</u> Bereken f (z (x, y)), met $f(z) = \sum_{i=0}^{3} a_i z^i + O(z^4)$

en $z = \sum b_{kl} x^k y^l$.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp24
(8192, 100, 100, 1000, 3, ⏨-10, ⏨-10, 18, 0)
PR STRING(resultaten fp24);
f:= TPS(z,a0,a1,a2,a3);
Z:= TPS(y,TPS(x,  0,b10,b20,b30),
        TPS(x,b01,b11,b21),
        TPS(x,b02,b12),b03);
OUTPUT R(f(x,y):= SUBST(f, z,Z));
NLCR; PR STRING(Als f(z)) = exp(z)));
f:= TPS(z,1,1,1/2,1/6);
OUTPUT R(exp(z(x,y)):= SUBST(f, z,Z));
END;
```

```
resultaten fp24
f(x,y):= a0+(a1×b10)×x+(a2×b10↑2+a1×b20)×x↑2+(a3×b10↑3+2×a2×b10×b20+a1×b30)×x↑3+
   O(x↑4)+(a1×b01+(2×a2×b10×b01+a1×b11)×x+(3×a3×b10↑2×b01+2×a2×b10×b11+2×a2×b20×
   b01+a1×b21)×x↑2+O(x↑3))×y+(a2×b01↑2+a1×b02+(3×a3×b10×b01↑2+2×a2×b10×b02+2×a2×
   b01×b11+a1×b12)×x+O(x↑2))×y↑2+(a3×b01↑3+2×a2×b01×b02+a1×b03+(6×a3×b10×b01×b02
   +3×a3×b01↑2×b11+2×a2×b10×b03+2×a2×b01×b12+2×a2×b11×b02)×x+O(x↑2))×y↑3+O(y↑4);
Als f(z) = exp(z)
exp(z(x,y)):= 1+b10×x+(1/2×b10↑2+b20)×x↑2+(1/6×b10↑3+b10×b20+b30)×x↑3+O(x↑4)+(b01
   +(b10×b01+b11)×x+(1/2×b10↑2×b01+b10×b11+b20×b01+b21)×x↑2+O(x↑3))×y+(1/2×b01↑2
   +b02+(1/2×b10×b01↑2+b10×b02+b01×b11+b12)×x+O(x↑2))×y↑2+(1/6×b01↑3+b01×b02+b03
   +(b10×b01×b02+1/2×b01↑2×b11+b10×b03+b01×b12+b11×b02)×x+O(x↑2))×y↑3+O(y↑4);
ready
line number = 10 execution time = 66 sec
```

Aangezien we soms behoefte hebben om met de coëfficiënten van een berekende afgebroken machtreeks verder te werken, is er een mogelijkheid om die coëfficiënten namen te geven.

Zij $f(x) = \sum_{i=0}^{n} a_i x^i + 0(x^{n+1})$,

dan zijn na de statement:

COEFF (f, c0, c1, c2, ..., cn)

de waarden van c0, c1, c2, ..., cn gelijk aan $a_0, a_1, a_2, \ldots, a_n$.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp25
(8192, 100, 100, 1000, 3, ₁₀-10, ₁₀-10, 10, 0)
PR STRING(resultaten fp25);
f:= TPS(z,a0,a1,a2,a3);
COEFF(f↑2 + 2 × f , c0,c1,c2,c3);
OUTPUT R(c0:= c0); OUTPUT R(c1:= c1);
OUTPUT R(c2:= c2); OUTPUT R(c3:= c3);
COEFF(DER(f,z),c0,c1,c2,c3); OUTPUT R(dc3:= c3);
END;
```

```
resultaten fp25
c0:= a0/\2+2×a0;
c1:= 2×a0×a1+2×a1;
c2:= 2×a0×a2+a1/\2+2×a2;
c3:= 2×a0×a3+2×a1×a2+2×a3;
degree of tr pow series too small
line number = 6 execution time = 7 sec
```

De foutmelding geeft aan dat de graad van "DER (f, z)" kleiner is dan 3. Deze graad is in feite 2, hetgeen logisch is aangezien "DER" de constante term in f laat verdwijnen.

Met behulp van de nu te behandelen statement voor <u>het oplossen van vergelijkingen in een aantal onbekenden</u> zullen we het belang van de "COEFF" operator leren kennen.
Het systeem kan (quasi-) lineaire vergelijkingen in een aantal onbekenden oplossen.
<u>Voorbeeld</u>:

$$\begin{aligned} a\,y + b\,z - c &= 0 \\ a\,x + b\,y \qquad - d &= 0 \\ b\,x \qquad + a\,z - f &= 0 \end{aligned}$$

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp26
(8192, 100, 100, 0, 0, ₁₀-10, ₁₀-10, 8, 0)
PR STRING(resultaten fp26);
SOL LIN EQ(3, x,y,z,
              a × y + b × z - c,
      a × x + b × y         - d,
      b × x         + a × z - f);
END;
```

```
resultaten fp26
y:= (a/\2×c-a×b×f+b/\2×d)/(a/\3+b/\3);
x:= (a/\2×d-a×b×c+b/\2×f)/(a/\3+b/\3);
z:= (a/\2×f-a×b×d+b/\2×c)/(a/\3+b/\3);
ready
line number = 6 execution time = 59 sec
```

We zien dat het resultaat van de berekening meteen wordt geprint, zonder dat OUTPUT R moet worden aangeroepen.

Opmerking 1: als de orde van het stelsel (in bovenstaand voorbeeld: 3) als negatief getal wordt meegegeven, dan is de berekening efficienter m.b.t. geheugenruimte; echter, dan zijn de oplossingen (x, y, en z boven) later in het programma niet meer beschikbaar (dit zal meestal geen bezwaar zijn omdat het juist ging om deze oplossingen).

Opmerking 2: de oplossingsmethode is sucessieve eliminatie; d.w.z. als de vergelijkingen zijn:

$$f_i\ (x_1, \ldots, x_n) = 0\ ,\quad i = 1, \ldots, n,$$

dan wordt x_1 uit $f_1 = 0$ berekend en in de overige f_i, $i = 2, \ldots, n$ gesubstitueerd, zodat het systeem één vergelijking en één onbekende minder telt (mocht blijken dat $\partial f_1 / \partial x_1 = 0$, dan wordt een andere f_i genomen).

Het zal duidelijk zijn dat de vergelijkingen niet strict lineair behoeven te zijn; immers, als f_1 slechts van x_1 afhangt, zodat na eliminatie van x_1, de x_1 bekend is, dan mag f_i, $i > n$, best op niet-lineaire wijze van x_1 afhangen.

Van deze eigenschap maken we gebruik in het volgende programma dat de Taylor-reeks ontwikkeling berekent van de functie y (x) die voldoet aan de differentiaalvergelijking en beginconditie:

$$y' = x^2 + y^2\ ,\ y\ (0) = a\ .$$

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp27
(8192, 1000, 0, 50, 4, ₁₀-10, ₁₀-10, 10, 0)
PR STRING(resultaten fp27);
y:= TPS(x,a,a1,a2,a3,0);
COEFF(DER(y,x) - y↑2 - TPS(x,0,0,1,0),
  c0,c1,c2,c3);
SOL LIN EQ(-3, a1,a2,a3, c1,c2,c3);
END;
```

```
resultaten fp27
a1:= (128/3×a↑7+256/3×a↑4×a2+192×a↑3×a3-32×a↑3+128/3×a×a2↑2)/(-256/3×a↑9-256×
  a↑6×a2-384×a↑5×a3+128×a↑5-256×a↑3×a2↑2-384×a↑2×a2×a3+128×a↑2×a2-256/3×a2↑3
  );
a2:= (-512×a↑9-1024×a↑6×a2-2304×a↑5×a3+384×a↑5-512×a↑3×a2↑2)/(1024×a↑10+3072×
  a↑7×a2+.4608⏨4×a↑6×a3-1536×a↑6+3072×a↑4×a2↑2+.4608⏨4×a↑3×a2×a3-1536×a↑3×
  a2+1024×a×a2↑3);
a3:= 32×a↑3/(-128×a↑7-256×a↑4×a2-576×a↑3×a3+192×a↑3-128×a×a2↑2);
ready
line number = 6 execution time = 131 sec
```

N.B. De resultaten geven aan dat er ergens iets fout is! (Wat?)

Een ander voorbeeld is de berekening van de machtreeks ontwikkeling van een inverse functie, of nog algemener een impliciet gegeven functie y (x), welke voldoet aan

$$f\ (x,\ y\ (x)) = 0.$$

<u>Voorbeeld</u>: $e^{y(x)} - 1 + \sin\ (x) = 0.$

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp28
(8192, 1000, 0, 400, 4, ⏨-10, ⏨-10, 14, 0)
PR STRING(resultaten fp28);
EXP:= TPS(t,1,1,1/2,1/6,1/24);
SIN:= TPS(t,0,1,0,-1/6,0);
y:= TPS(x,0,y1,y2,y3,y4);
COEFF(SUBST(EXP, t,y) - 1 + SUBST(SIN, t,x),
        c0,c1,c2,c3,c4);
OUTPUT R(c0:= c0); OUTPUT R(c1:= c1); OUTPUT R(c2:= c2);
OUTPUT R(c3:= c3); OUTPUT R(c4:= c4);
SOL LIN EQ(-4,y1,y2,y3,y4, c1,c2,c3,c4);
END;
```

```
resultaten fp28
c0:= 0;
c1:= y1+1;
c2:= 1/2×y1↑2+y2;
c3:= 1/6×y1↑3+y1×y2+y3-1/6;
c4:= 1/24×y1↑4+1/2×y1↑2×y2+y1×y3+1/2×y2↑2+y4;
y1:= -1;
y2:= -1/2;
y3:= -1/6;
y4:= -1/12;
ready
line number = 10 execution time = 15 sec
```

4. Efficientie m.b.t. geheugenruimte.

In het vorige formule-programma (f.p.28) zijn we onnodig kwistig met geheugenruimte geweest. Immers, nadat y als machtreeks in x was berekend, komt hij niet meer aan bod; toch blijft hij geheugenruimte innemen. Bovendien blijven allerlei anonieme tussen-resultaten in het geheugen opgeborgen.
Beschouwen we, bijvoorbeeld: SUBST (EXP, t, y) dan wordt de volgende som berekend:

$$(((1/24 * y + 1/6) * y + 1/2) * y + 1) * y + 1.$$

Alle hier voorkomende subformules, zoals $1/24 * y$, worden opgeborgen in het geheugen. Alleen het eindresultaat is uiteindelijk van belang en ook dat eindresultaat kan, nadat de y1, ..., y4 berekend en geprint zijn uit het geheugen verdwijnen.
Een van de nadelen van het algebraïsche-bewerkingen-systeem is nu dat het de gebruiker verplicht zelf voor het verwijderen van overbodige formules zorg te dragen.
Daartoe is de syntactische eenheid: "formule-blok" ingevoerd, die als volgt wordt gedefinieerd:

< formule-blok > :: = FIX;
< rij statements >;
< formule-blok-einde >

< formule-blok-einde > :: = ERASE |
ER B RET (< rij namen van formules >)

Een formule-blok is syntactisch een productie van statement.

Allereerst de werking van een formule-blok van de vorm:

"FIX; < rij statements > ; ERASE".

Als we de verzameling formules die opgeborgen worden in het geheugen, voordat het systeem "FIX;" gelezen heeft, met V_1 aanduiden; Als we de verzameling formules die opgeborgen worden in het geheugen, voordat het systeem "ERASE" gelezen heeft, met V_2 aanduiden, dan is het effect van "ERASE", dat alle formules uit $V_2 \setminus V_1$ uit het geheugen verwijderd worden. De namen (identifiers) die naar de formules uit $V_2 \setminus V_1$ verwezen, hebben na "ERASE" hun betekenis verloren.

Bij een formule-blok van de vorm:

"FIX; < rij statements > ; ER B RET (f_1, ..., f_n)"

is de situatie analoog als boven: "ERASE" moet vervangen worden door "ER B RET (f_1, ..., f_n)" en $V_2 \setminus V_1$ moet vervangen worden door $V_2 \setminus (V_1 \vee \{f_i, i = 1, \ldots, n\})$.

N.B.1. "ER B RET" is een afkorting voor "ERase But RETain".

N.B.2. Elk formule-blok-einde dient voorafgegaan te worden door "FIX;"

N.B.3. De formules "0", "1", "-1", en "i" worden door het systeem zelf, alvorens met de executie van het formule-programma te beginnen, in het geheugen opgeborgen. Als de naam "z", bijvoorbeeld, verwijst naar i dan is deze verwijzing niet ongedaan te maken door ERASE. (Wel door een nieuwe assignment aan "z").

N.B.4. Een vergeten "FIX;" leidt tot de foutmelding:

"FIX missing".

N.B.5. "Formule-bloks" kunnen "genest" voorkomen.

Voorbeeld: Alternatief voor f.p.28, waarin ook x als functie van y wordt berekend en tenslotte de gevonden x (y) in de oorspronkelijke reeks voor y (x) gesubstitueerd wordt. Het zal duidelijk zijn dat in dit programma het verwijderen van niet meer interessante formules een belangrijke rol speelt.

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp29
(8192, 1000, 0, 400, 4, ₁₀-10, ₁₀-10, 20, 0)
PR STRING(resultaten fp29);
EXP:= TPS(t,1,1,1/2,1/6,1/24);
SIN:= TPS(t,0,1,0,-1/6,0);
FIX;
  FIX;
    y:= TPS(x,0,y1,y2,y3,y4);
    COEFF(SUBST(EXP, t,y) - 1 + SUBST(SIN, t,x),
        c0,c1,c2,c3,c4);
    FIX; OUTPUT R(het resultaat als m.r. in x
      := TPS(x,c0,c1,c2,c3,c4));
    ERASE;
  ER B RET(c1,c2,c3,c4);
  OUTPUT R(y is niet meer, immers y:= y);
  SOL LIN EQ(4,y1,y2,y3,y4, c1,c2,c3,c4);
ER B RET(y1,y2,y3,y4);
FIX;
  FIX;
    X:= TPS(Y,0,x1,x2,x3,x4);
    COEFF(SUBST(EXP, t,Y) - 1 + SUBST(SIN, t,X),
        c0,c1,c2,c3,c4);
    FIX; OUTPUT R(het resultaat als m.r. in Y
      := TPS(Y,c0,c1,c2,c3,c4));
    ERASE;
  ER B RET(c1,c2,c3,c4);
  SOL LIN EQ(4,x1,x2,x3,x4, c1,c2,c3,c4);
ER B RET(x1,x2,x3,x4);
NLCR; PR STRING(De proef op de som verkrijgen we door
de verkregen x in de oude y te substitueren.);
OUTPUT R(y(x(Y)):= SUBST(TPS(x,0,y1,y2,y3,y4),
                         x,TPS(Y,0,x1,x2,x3,x4)));
END;
```

```
resultaten fp29
het resultaat als m.r. in x
       := 0+(y1+1)×x+(1/2×y1↑2+y2)×x↑2+(1/6×y1↑3+y1×y2+y3-1/6)×x↑3+
  (1/24×y1↑4+1/2×y1↑2×y2+y1×y3+1/2×y2↑2+y4)×x↑4+O(x↑5);
y is niet meer, immers y:= y;
y1:= -1;
y2:= -1/2;
y3:= -1/6;
y4:= -1/12;
het resultaat als m.r. in Y
       := 0+(x1+1)×Y+(x2+1/2)×Y↑2+(-1/6×x1↑3+x3+1/6)×Y↑3+
  (-1/2×x1↑2×x2+x4+1/24)×Y↑4+O(Y↑5);
x1:= -1;
x2:= -1/2;
x3:= -1/3;
x4:= -7/24;
De proef op de som verkrijgen we door
de verkregen x in de oude y te substitueren.
y(x(Y)):= 0+1×Y+0×Y↑2+0×Y↑3+0×Y↑4+O(Y↑5);
ready
line number = 31 execution time = 26 sec
```

N.B. Om begrijpelijke redenen mag de combinatie "< formule-blok-einde>; END;" vervangen worden door "END;".

Voor meer details omtrent de "ERASE" en de "ER B RET" operator verwijzen we naar [1], sectie 3.2.4.3 en sectie 3.2.7, voorbeeld 1 en 2.

Het kan soms nodig zijn een gesteld probleem te splitsen over een aantal formule programma's: $fp_1, \ldots, fp_n$.
M.b.v. de "PR STRING" operator is het mogelijk om fp_1 zodanig te maken dat hij fp_2 genereert die op zijn beurt, bij executie, weer fp_3 genereert, enz. Deze splitsing is bijvoorbeeld van belang als voor de oplossing van het probleem zowel machtreeksen nodig zijn,die erg veel geheugenruimte vergen, als een aantal "SOL LIN EQ" operaties op de coëficiënten van de berekende machtreeksen.
Men zie voor een dergelijke techniek [1] sectie 3.2.7, voorbeeld 7 en 8.

We merken tenslotte op dat men een formule programma kan opstellen dat ALGOL 60 statements genereert voor een ALGOL 60 programma dat met complexe getallen rekent.
Ook voor deze mogelijkheid verwijzen we naar [1] sectie 3.2.4.9 en voorbeeld 4 van sectie 3.2.7.

[1] R.P. van de Riet, Formula Manipulation in ALGOL 60, part 1.
Mathematical Centre Tracts 17
Mathematisch Centrum, Amsterdam, 1968.

[2] R.P. van de Riet, Formula Manipulation in ALGOL 60, part 2.
Mathematical Centre Tracts 18
Mathematisch Centrum, Amsterdam, eind 1968.

Appendix

De correcte formulering van formule programma fp.27 luidt:

```
Formule programma t.b.v. cursus ABC. RPR 250968/fp27a
(8192, 1000, 0, 50, 4, ⏨-10, ⏨-10, 11, 0)
PR STRING(resultaten fp27a);
y:= TPS(x,a,a1,a2,a3,a4);
COEFF(DER(y,x) - y↑2 - TPS(x,0,0,1,0),
  c0,c1,c2,c3);
SOL LIN EQ(-4, a1,a2,a3,a4, c0,c1,c2,c3);
END;

resultaten fp27a
a1:= a↑2;
a2:= a↑3;
a3:= a↑4+1/3;
a4:= a↑5+1/6×a;
ready
line number = 6 execution time = 10 sec
```