STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING TOEGEPASTE WISKUNDE

TW 99

An application of a method for algebraic manipulation in ALGOL 60

R.P. van de Riet

january 1966

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 Pure Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

1. Summary

In this paper we describe an application of a set of ALGOL 60 procedures for algebraic formula manipulation.

This application concerns the Cauchy problem (sections 3 and 4).

Another application is described in ref [1].

2. Algebraic manipulation in ALGOL 60

This section is devoted to the ALGOL 60 procedures, by means of which one can manipulate formulae in an ALGOL program.

We require that the operations used in a formula be the sum, the difference, the product and the quotient, which are assumed to be dyadic.

Moreover, a formula may contain variables, numbers and the function symbols sin, cos, exp, ln, sqrt and arctan.

Since the symbols +, −, ×, / and the function symbols are reserved in ALGOL for numerical purposes, we can not use these symbols to write a formula in the program.

Therefore we use a different notation, called the program notation, by means of which we can write a formula in the program. The correspondence between this

notation and the ordinary notation is given in the table below, in which a and b are formulae and c an arithmetic expression representing a (real) number.

ordinary notation	a + b	a - b	a × b	a / b	c	function symbol in small letters
program notation	S(a,b)	D(a,b)	P(a,b)	Q(a,b)	NUMBER(c)	function symbol in capital letters

table 1

We require, moreover, that for each variable occurring in a formula an integer variable or an integer array element, having an identifier directly corresponding with the variable, be declared in the program.

Example 1 in ordinary notation we have the formula $((U_{2} \times U_{1\,y}) + x)$

which in program notation is S(P(U[2],Uy[1]),x).

Example 2 in ordinary notation we have the formula $(3.14 - \arctan(U_{2\,y} / U_{1\,x}))$

which in program notation is D(NUMBER(3.14),ARCTAN(Q(Uy[2],Ux[1]))).

Remark one can write an ALGOL program which converts a formula written in the ordinary notation into a formula written in program notation.

In describing the procedures we use a set of non-local integer variables, declared by integer sum, product, quotient, number, variable, function,

Sin, Cos, Exp, Ln, Sqrt, Arctan,

one, zero;

The values of the integer variables on each line of this declaration will be different. The variables one and zero correspond to the numbers 1 and 0. They are used as the unit element and the zero element in the system.

First, the procedure INT REPR is introduced, the heading of which runs as follows:

procedure INT REPR(case,formula,left side,type,right side,numb);

integer case,formula,left side,type,right side; real numb;

A call of this procedure has the following effect:

If case = 1, the own variables declared in INT REPR, get values.

If case = 2, information stored in the parameters left side, type, right side and numb is stored in own arrays and to the parameter "formula" a value (the address of the stored information) is assigned.

If case = 3, dependent on the value of the parameter "formula", the parameters

left side, type, right side and numb get values.

INT REPR is called within the procedure bodies of STORE, TYPE, VALUE OF NUMBER and NUMBER.

It has the property: for each arbitrary triple of integers a, b and c and each arbitrary real number d the following Boolean expressions:

TYPE(STORE(a,b,c),A,C) = b ∧ A = a ∧ C = c

and VALUE OF NUMBER(NUMBER(d)) = d

in which A and C are auxiliary integer variables, have the value true.

The body of the procedure is not given here in order to avoid specifying unnecessary details. Moreover, the following procedures are introduced:

```
integer procedure STORE(left side,type,right side); value left side,right side;
integer left side,type,right side;
begin integer a; INT REPR(2,a,left side,type,right side,0); STORE:= a end;
integer procedure TYPE(formula,left side,right side); value formula;
integer formula,left side,right side;
begin integer t; real c; INT REPR(3,formula,left side,t,right side,c); TYPE:=t end;
real procedure VALUE OF NUMBER(formula); value formula; integer formula;
begin real numb; INT REPR(3,formula,0,0,0,numb); VALUE OF NUMBER:= numb end;
```

We now describe the procedures for storing a formula.

```
integer procedure S(i,j); value i,j; integer i,j;
begin comment S applies the rule (i + 0) = (0 + i) = i;
  S:= if i = zero then j else if j = zero then i else STORE(i,sum,j)
end S;
integer procedure D(i,j); integer i,j; D:= S(i,P(NUMBER(-1),j));
integer procedure P(i,j); value i,j; integer i,j;
begin comment P applies the rules (i × 0) = (0 × j) = 0 and (i × 1) = (1 × i) = i;
  P:= if i = zero ∨ j = zero then zero else if i = one then j else if j = one then i
  else STORE(i,product,j)
end P;
integer procedure Q(i,j); integer i,j; Q:= STORE(i,quotient,j);
integer procedure NUMBER(c); real c;
begin integer a; INT REPR(2,a,0,number,0,c); NUMBER:= a end;
```

Note that the procedure D converts the difference (i - j) into (i + ((-1) × j)). By means of the procedure NUMBER, the values of one and zero are defined as follows: one:= NUMBER(1); zero:= NUMBER(0);

There are six procedures for storing functions, of which we describe only the

```
integer procedure ARCTAN(i); integer i; ARCTAN:= STORE(Arctan,function,i);
```

Moreover, there are procedures for evaluating, outputting and differentiating a formula. Let us merely sketch the differentiation procedure DER(f,z), which stores the derivative of f with respect to z, by giving only a typical part of the procedure body.

```
integer procedure DER(f,z); value f,z; integer f,z;
if f = z then DER:= one else
begin integer a,type,b; type:= TYPE(f,a,b);
  if type = product then DER:= S(P(DER(a,z),b),P(a,DER(b,z))) else
  Similar statements for a sum and a quotient should be inserted:
  if type = function then
  begin We treat only the arctan function:
    if a = Arctan then DER:= P(Q(one,S(one,P(b,b))),DER(b,z))
  end else DER:= zero
end DER;
```

Remark to each stored formula there corresponds an integer (the address of the in INT REPR stored information), which we will call the index of the formula.

3. The Cauchy problem

In this section we will be concerned with the Cauchy problem, i.e. the determination of functions satisfying the partial differential equations

$$F_h(U_k, U_{k\,x}, U_{k\,y}, x, y) = 0, \text{ with } h = 1,\ldots, M \text{ and } k = 1,\ldots, M, \quad (1)$$

(in which $U_{k\,x}$ and $U_{k\,y}$ are the partial derivatives of U_k with respect to x and y)

and the initial conditions

$$U_k(x, 0) = G_k(x). \quad (2)$$

If we require

1) the F_h and the G_k are analytic in a neighbourhood of $x = 0$, $y = 0$,

2) $$\det \left\{ \frac{\partial F_h}{\partial U_{k\,y}} \right\} \neq 0, \text{ in } x = 0, y = 0, \quad (3)$$

3) the quantities $U_{k\,y}(0, 0)$ are explicitly solvable from (1) and (2),

then we can calculate the Taylor series for the functions U_k, i.e.

$$U_k(x, y) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} u_{k,n,m} x^n y^m . \qquad (4)$$

If the left-hand sides of (1) are of the form described in section 2, this calculation can be performed on a computer by means of the ALGOL program which will be sketched in section 4.

Let all formulae which build up the F_h be enumerated, say, $A_1, \ldots, A_K$. For the example 1, e.g., we have

$A_1 = U_2$, $A_2 = U_{1y}$, $A_3 = A_1 \times A_2$, $A_4 = x$ and $A_5 = A_3 + A_4$.

For each A_q the Taylor coefficient $a_{q,n,m}$ (n and m fixed for the moment) is defined using rules for the sum, the product and the quotient of two formulae. These rules are, together with rules for a number and the variables, listed in table 2. Rules for the function symbols sin, cos, etc. will be given separately.

$A_3 =$	$a_{3,n,m} =$
$A_2 + A_1$	$a_{2,n,m} + a_{1,n,m}$
$A_2 \times A_1$	$\sum_{i=0}^{n} \sum_{j=0}^{m} a_{2,i,j}\, a_{1,n-i,m-j}$
A_2 / A_1	$(a_{2,n,m} - a_{3,0,0}\, a_{1,n,m} - \sum_{i=0}^{n} \sum_{j=0}^{m}{}' a_{3,i,j}\, a_{1,n-i,m-j})/a_{1,0,0}$ the symbol $\sum\sum'$ is defined by $\sum_{i=0}^{n} \sum_{j=0}^{m}{}' b_{i,j} = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{i,j} - b_{0,0} - b_{n,m}$.
c (a real number)	$c\, \delta_{n,0}\, \delta_{m,0}$ ($\delta_{i,j}$ is the Kronecker delta)
U_k	$u_{k,n,m}$
$U_{k\,x}$	$(n + 1)\, u_{k,n+1,m}$
$U_{k\,y}$	$(m + 1)\, u_{k,n,m+1}$
x	$\delta_{n,1}\, \delta_{m,0}$
y	$\delta_{n,0}\, \delta_{m,1}$

table 2

The function symbols are treated quite generally in the program.

As part of the input of the program, we require that for each function symbol ϕ functions ψ_k be stored, defined by means of:

$$\frac{d\,\phi_k(X)}{d\,X} = \psi_k(X, \phi_1, \ldots, \phi_p),\ k = 1, \ldots, p \tag{5}$$

and $\phi_1(X) = \phi(X)$.

The ψ_k are formulae in which no function symbols occur.

Besides the mentioned function symbols, the program can also treat other function symbols, for which the ψ_k are known.

Consider the function $A_2 = \phi(A_1)$; the $a_{2,n,m}$ can be defined in the following way.

Let $\varphi_{k,i,j}$ and $\Psi_{k,i,j}$ be the Taylor coefficients of $\phi_k(A_1)$ and $\psi_k(A_1, \phi_1(A_1), \ldots, \phi_p(A_1))$ respectively.

Then we have:

1) $$\varphi_{k,0,0} = \phi_k(a_{1,0,0}), \tag{6}$$

2) for $n \neq 0$,

$$\varphi_{k,n,m} = \Big(\sum_{i=1}^{n}\sum_{j=0}^{m} \Psi_{k,n-i,m-j}\ i\ a_{1,i,j}\Big)/n, \tag{7}$$

which follows from

$$\frac{\partial \phi_k}{\partial x} = \psi_k(A_1, \phi_1(A_1), \ldots, \phi_p(A_1)) \cdot \frac{\partial A_1}{\partial x},$$

3) for $n = 0$ and $m \neq 0$,

$$\varphi_{k,n,m} = \Big(\sum_{i=0}^{n} \sum_{j=1}^{m} \Psi_{k,n-i,m-j} \; j \; a_{1,i,j} \Big)/m, \tag{8}$$

found by differentiating $\phi_k(A_1)$ with respect to y.

From this $a_{2,n,m}$ follows, since $a_{2,n,m} = \varphi_{1,n,m}$.

The $\Psi_{k,n,m}$, can be defined using table 2.

The Taylor coefficients $a_{q,n,m}$ with $q = 1, \ldots, K$ are, according to the above remarks, indirectly defined in terms of the $u_{k,n,m}$, $u_{k,n+1,m}$ and $u_{k,n,m+1}$.

The calculation process is such that the index pair (n,m) is successively equal to

(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ... , (i,j), if i = 0 then (j+1,0) else (i-1,j+1), ...

Assume now that the $a_{q,i,j}$ are already calculated for those index pairs (i,j), which precede in this sequence the index pair (n,m).

The variable $U_{k\,y}$ corresponds to some formula A_s, and $a_{s,i,j} = (j+1)\, u_{k,i,j+1}$.

Since, for $m \neq 0$, the index pairs (n,m-1) and (n+1,m-1) precede the (n,m), it follows from our assumption that the $u_{k,n,m}$ and the $u_{k,n+1,m}$ are already

calculated (in the case m = 0 these coefficients follow from the initial conditions). Thus the only coefficients which are not calculated are the $u_{k,n,m+1}$, which are replaced by the symbols Z_k.

The program then calculates algebraically with these symbols and builds up formulae linear in the Z_k for the left-hand sides of (1), from which the Z_k and thus the $u_{k,n,m+1}$ and finally the $a_{q,n,m}$ can be calculated.

Since the $a_{q,0,0}$ can be calculated, using the initial conditions, it follows that all the $a_{q,n,m}$ can be calculated by means of the recurrent calculation process defined above.

4. The program for the Cauchy problem

In accordance with the remarks in section 2 the program contains the declaration integer x, y; integer array U, Ux, Uy, Z[1:M];

in which M and p are defined in (1) and (5).

These variables get values by means of the following statements

x:= STORE(1,variable,1); y:= STORE(2,variable,1);

```
for k:= 1 step 1 until M do

begin U[k]:= STORE(-1,variable,k); Ux[k]:= STORE(-2,variable,k);

  Uy[k]:= STORE(-3,variable,k); Z[k]:= STORE(-4,variable,k)

end;
```

It does not matter which information is set in the first and third parameter of STORE, except that it should be non-ambiguous.

In this section, the function symbols will not be treated.

The presentation of the program is correspondingly simplified (see ref [2]).

The differential equations are stored by execution of e.g. the statements

```
Left hand side of DE[1]:= S(P(U[2],Uy[1]),x);

Left hand side of DE[2]:= D(NUMBER(3.14),ARCTAN(Q(Uy[2],Ux[1])));
```

In the program we need the arrays u, a and fa, declared by

```
array u[1:M,0:N,0:N], a[1:K,0:N-1,0:N-1]; integer array fa[1:K];
```

K is defined in section 3. N defines the number of Taylor coefficients which should be calculated.

In the array u, the calculated Taylor coefficients of U_k are stored.

The core of the program is the procedure CALC COEFF, which defines for each formula F its Taylor coefficient with a fixed index pair (n,m). This coefficient is stored in the real array element a[F,n,m] if it actually can be calculated. However, if it can not be calculated, a formula is built up and stored instead. The index of this formula is assigned to the integer array element fa[F]. (Note that F is used here also as the index of the formula F).

Let us sketch this procedure.

```
procedure CALC COEFF(F); value F; integer F;
begin integer A,type,B; Boolean procedure KNOWN(F); integer F;
  begin comment KNOWN becomes true or false dependent on whether a[F,n,m]
    has been calculated or not;
  end KNOWN;
  type:= TYPE(F,A,B);
  if type = sum then
  begin comment F is of the form A + B; CALC COEFF(A); CALC COEFF(B);
    if KNOWN(A) ∧ KNOWN(B) then a[F,n,m]:= a[A,n,m] + a[B,n,m] else
```

```
fa[F]:= S(if KNOWN(A) then NUMBER(a[A,n,m]) else fa[A],
          if KNOWN(B) then NUMBER(a[B,n,m]) else fa[B])
end else
```

Similar statements for the product and quotient should be inserted:

```
if type = variable then
begin if A = -1 then F is of the form U: a[F,n,m]:= u[B,n,m] else
      if A = -2 then F is of the form Ux: a[F,n,m]:= (n+1) × u[B,n+1,m] else
      if A = -3 then F is of the form Uy: fa[F]:= P(NUMBER(m+1),Z[B]);
  comment For other variables similar statements should be inserted;
end
end CALC COEFF;
```

By means of CALC COEFF we get a formula linear in the Z[k] for the Taylor coefficients of the left-hand sides of (1) and the Z[k] can be calculated, using the evaluation and the differentiation procedures.

The Taylor coefficients which have not yet been determined, can now be calculated and the process can be repeated for another index pair (n,m).

References

[1] R.P.van de Riet — Algebraic operations in ALGOL 60. A second order problem
Report TW 96, Mathematisch Centrum, Amsterdam (1965)

[2] R.P.van de Riet — Algebraic operations in ALGOL 60. The Cauchy problem I
Report TW 97, Mathematisch Centrum, Amsterdam (1965)