STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING TOEGEPASTE WISKUNDE

TW 97

Algebraic operations in ALGOL 60

The Cauchy problem I

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 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.

The ALGOL 60 program

and the results

(Algebraic operations in ALGOL

The Cauchy problem I

TW 97)

```
if H[i,1] > 0 then VADIPUFUNC(i,0,3) else
if H[i,1] = -1 then
begin PUTEXT1(‹(›); FLOP(3,1,HC[H[i,3]]); PUTEXT1(‹)›) end else VADIPUST(i,0,3);
real procedure VADIPUCOMPFORM(i,n,l); value i; integer i,n,l;
begin real a,b; integer p,q; switch SS:= value,diff,output,sum,product,quotient;
  procedure A(a,b,st); real a; integer b; string st;
  begin if l = 1 then VADIPUCOMPFORM:= a else if l = 2 then l:= b else
    begin PUTEXT1(st); OUTPUT(H[i,3]); PUTEXT1(‹)›) end; goto END
  end; VADIPUCOMPFORM:= 1; goto SS[l];
value: a:= VALUE(H[i,1]); b:= VALUE(H[i,3]); goto SS[H[i,2]+3];
diff: p:= DIFF(H[i,1],n); q:= DIFF(H[i,3],n); goto SS[H[i,2]+3];
output: PUTEXT1(‹(›); OUTPUT(H[i,1]); goto SS[H[i,2]+3];
sum: A(a + b,S(p,q),‹+›);
product: A(a × b,S(P(p,H[i,3]),P(H[i,1],q)),‹×›);
quotient: A(a / b,if p = 0 ∧ q = 0 then 0 else Q(D(P(p,H[i,3]),P(H[i,1],q)),
  P(H[i,3],H[i,3])),‹/›);
END: end;
real procedure VADIPUFUNC(i,n,l); value i; integer i,n,l; if l = 3 then
begin VADIPUFUNC:= 1; INFORM FUNC(0,-H[i,1],1); PUTEXT1(‹(›);
  OUTPUT(H[i,3]); PUTEXT1(‹)›)
end;
real procedure VADIPUST(i,n,l); value i; integer i,n,l;
begin VADIPUST:= 0; if l = 3 then
  begin if H[i,1] = -2 ∨ H[i,1] = -3 then
    begin if H[i,1] = -2 then PUTEXT1(‹x^(›) else
      PUTEXT1(‹y^(›); ABSFIXP(2,0,H[i,3]); PUTEXT1(‹)›)
    end else if H[i,1] = -10 then
    begin if H[i,2] < 0 then PUTEXT1(‹d›); PUTEXT1(‹U[›);
      ABSFIXP(1,0,H[i,3]); PUTEXT1(‹]›); if H[i,2] < 0 then
      begin PUTEXT1(‹/d›); if H[i,2] = -1 then PUTEXT1(‹x›) else PUTEXT1(‹y›) end
end end end;
real procedure INFORM FUNC(a,i,j); value i,j; real a; integer i,j;
begin integer fk; Boolean PU; switch SS:= SIN,COS,EXP,LN,SQRT,ARCTAN;
  procedure A(f,st,psi); integer psi; real procedure f; string st;
  begin integer n; if i = 0 then begin fk:= fk + 1; PSI[fk,1]:= psi; PSI[fk,0]:= 1 end
    else if PU then begin PUTEXT1(st); goto END end else
```

```
begin comment Test program for the Cauchy problem R1050 RPR 310565/39492;
 integer kmax,kcmax; kmax:= XEEN(1023); kcmax:= XEEN(1023 × 1024): 1024;
 begin integer k,kc,ka,kf,Xf; integer array PSI[1:6,0:2],PHI[1:2],H[0:kmax,1:3];
  array HC[0:kcmax],fac[0:20];
  integer procedure STORE(i,l,j); value i,j; integer i,l,j;
  begin STORE:= k:= k + 1; if k > kmax then begin PUTEXT1(⟨k too large⟩); stop end;
    H[k,1]:= i; H[k,2]:= l; H[k,3]:= j
  end;
  integer procedure S(i,j); value i,j; integer i,j; S:=
  if i = 0 then j else if j = 0 then i else STORE(i,1,j);
  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; P:=
  if i = 0 ∨ j = 0 then 0 else if i = 1 then j else
  if j = 1 then i else STORE(i,2,j);
  integer procedure Q(i,j); integer i,j; Q:= STORE(i,3,j);
  integer procedure SIN(i); integer i; SIN:= STORE(1,0,i);
  integer procedure COS(i); integer i; COS:= STORE(2,0,i);
  integer procedure EXP(i); integer i; EXP:= STORE(3,0,i);
  integer procedure LN(i); integer i; LN:= STORE(4,0,i);
  integer procedure SQRT(i); integer i; SQRT:= STORE(5,0,i);
  integer procedure ARCTAN(i); integer i; ARCTAN:= STORE(6,0,i);
  integer procedure NUMBER(c); real c;
  begin kc:= kc + 1; if kc > kcmax then begin PUTEXT1(⟨kc too large⟩); stop end;
    HC[kc]:= c; NUMBER:= STORE(-1,0,kc)
  end;
  real procedure VALUE(i); value i; integer i; VALUE:=
  if H[i,2] > 0 then VADIPUCOMPFORM(i,0,1) else
  if H[i,1] > 0 then VADIPUFUNC(i,0,1) else
  if H[i,1] = -1 then HC[H[i,3]] else VADIPUST(i,0,1);
  integer procedure DIFF(i,n); value i,n; integer i,n;
  begin integer j; j:= 2; if i = n then DIFF:= 1 else if H[i,1] = -1 then DIFF:= 0 else
    begin if H[i,2] > 0 then VADIPUCOMPFORM(i,n,j) else
      if H[i,1] > 0 then VADIPUFUNC(i,n,j) else VADIPUST(i,n,j); DIFF:= j
  end end;
  procedure OUTPUT(i); value i; integer i;
  if H[i,2] > 0 then VADIPUCOMPFORM(i,0,3) else
```

```
S2: IP:= SUM(i,0,n[1],SUM(j,1,n[2],j × a[rp,c[i,j]] × a[lp,c[n[1]-i,n[2]-j]]));
END: a[lp,c[n[1],n[2]]]:= alp; a[rp,c[n[1],n[2]]]:= arp
end;
integer procedure CALC ARRAY(i); value i; integer i;
if H[i,1] = -1 then begin CALC ARRAY:= ka:= ka + 1; a[ka,0]:= HC[H[i,3]] end
else if H[i,1] = -2 ∨ H[i,1] = -3 then
begin integer p; p:= -H[i,1] - 1; CALC ARRAY:= ka:= ka + 1; a[ka,c[n[1],n[2]]]:=
  if n[p] > H[i,3] ∨ n[3 - p] ≠ 0 then 0 else if H[i,3] = n[p] then 1 else FAC(H[i,3])
  /(FAC(n[p]) × FAC(H[i,3] - n[p])) × (if p = 1 then x0 else y0)↑(H[i,3] - n[p])
end else if H[i,1] = - 10 then
begin integer p; p:= H[i,3]; CALC ARRAY:= ka:= ka + 1; if type[p] = -H[i,2] ∧
  ¬(n[1] = 0 ∧ n[2] = 0) then f[ka]:= unknown[p] else a[ka,c[n[1],n[2]]]:= (if H[i,2] =
  0 then u[p,n[1],n[2]] else if H[i,2] = -1 then u[p,n[1]+1,n[2]]×(1 + n[1]) else
  if H[i,2] = -2 then u[p,n[1],n[2]+1]×(1 + n[2]) else 1)
end else if H[i,2] > 0 then
begin integer m1,m2; switch SS:= sum,product,quotient;
  procedure A(a1,b,c1); real a1,b,c1;
  begin switch SP:= S0,S1,S2,S3; goto SP[(sign(f[m1]+.1)+1): 2 + sign(f[m2]+.1)+2];
  S0: a[ka,c[n[1],n[2]]]:= a1 + b × a[m1,c[n[1],n[2]]] + c1 × a[m2,c[n[1],n[2]]];
      goto END;
  S1: f[ka]:= S(NUMBER(a1 + c1 × a[m2,c[n[1],n[2]]]),P(NUMBER(b),f[m1]));
      goto END;
  S2: f[ka]:= S(NUMBER(a1 + b  × a[m1,c[n[1],n[2]]]),P(NUMBER(c1),f[m2]));
      goto END;
  S3: f[ka]:= S(NUMBER(a1),S(P(NUMBER(b),f[m1]),P(NUMBER(c1),f[m2])));
      goto END
  end; m1:= CALC ARRAY(H[i,1]); m2:= CALC ARRAY(H[i,3]);
  CALC ARRAY:= ka:= ka + 1; goto SS[H[i,2]];
sum: A(0,1,1);
product: A(IP(m1,m2,0),a[m2,0],(if n[1] = 0 ∧ n[2] = 0 then 0 else a[m1,0]));
quotient: A(-IP(m2,ka,0)/a[m2,0],1/a[m2,0],-a[ka,0]/a[m2,0]);
END: end else if H[i,1] > 0 then
begin integer p,j,m1; m1:= CALC ARRAY(H[i,3]); kpsi:= kpsi + 1; psi[kpsi,-1]:= i;
  psi[kpsi,0]:= m1; for j:= 1 step 1 until PSI[H[i,1],0] do phi[kpsi,j]:= ka:= ka + 1;
  CALC ARRAY:= phi[kpsi,1]; p:= if n[1] = 0 then 2 else 1;
  for j:= 1 step 1 until PSI[H[i,1],0] do
```

```
    begin INFORM FUNC:= f(a); goto END end
  end; INFORM FUNC:= 0; PU:= i < 0; i:= abs(i); if i = 0 then
  begin fk:= 0; goto SIN end else goto SS[i];
SIN: if j = 2 then goto COSS; A(sin,⟨sin⟩,PHI[2]);
  COSS: A(cos,⟨⟩,P(NUMBER(-1),PHI[1])); PSI[1,2]:= PSI[fk,1]; fk:= 1; PSI[1,0]:= 2;
COS: if j = 2 then goto SINN; A(cos,⟨cos⟩,P(NUMBER(-1),PHI[2]));
  SINN: A(sin,⟨⟩,PHI[1]); PSI[2,2]:= PSI[fk,1]; fk:= 2; PSI[2,0]:= 2;
EXP: A(exp,⟨exp⟩,PHI[1]);
LN: A(ln,⟨ln⟩,Q(1,Xf));
SQRT: A(sqrt,⟨sqrt⟩,Q(NUMBER(.5),PHI[1]));
ARCTAN: A(arctan,⟨arctan⟩,Q(1,S(1,P(Xf,Xf))));
END: end;
procedure TAYLOR(dimension,M,F,N,x0,y0,u); value dimension,M,F,N,x0,y0;
integer dimension,M,N; real x0,y0; array u; integer array F;
begin integer nu,m,ka,kpsi,K,KC,AB,l,j; integer array type[1:M];
  procedure INITIALIZE(i); value i; integer i;
  begin AB:= AB + 1; if H[i,2] > 0 then
    begin INITIALIZE(H[i,1]); INITIALIZE(H[i,3]) end else if H[i,1] > 0 then
    begin integer k,l; kpsi:= kpsi + 1; INITIALIZE(H[i,3]); AB:= AB - 1;
      k:= PSI[H[i,1],0]; nu:= if k > nu then k else nu; for l:= 1 step 1 until k do
      begin AB:= AB + 1; INITIALIZE(PSI[H[i,1],l]) end
    end else if H[i,1] = - 4 then AB:= AB - 1 else if H[i,1] = -10 then
    begin if H[i,2] = -2 then type[H[i,3]]:= 2 else if H[i,2] = -1 ∧ type[H[i,3]] ≠ 2
      then type[H[i,3]]:= 1
  end end;
  for l:= 1 step 1 until M do type[l]:= 0; AB:= 0; kpsi:= nu:= 0;
  for l:= 1 step 1 until M do INITIALIZE(F[l]);
  begin array a[1:AB,0:(if dimension = 2 then ((N-1)×(N+2)): 2 else N-1)];
    integer array c[0:N-1,0:N-1],tp[1:M,1:2],unknown,formula[1:M],n[1:2],f[1:AB],
    phi[0:kpsi,0:nu],psi[0:kpsi,-1:nu];
    real procedure IP(lp,rp,diff); integer lp,rp,diff;
    begin integer i,j; real alp,arp; switch S:= S0,S1,S2;
      alp:= a[lp,c[n[1],n[2]]]; arp:= a[rp,c[n[1],n[2]]];
      a[lp,c[n[1],n[2]]]:= a[rp,c[n[1],n[2]]]:= 0; goto S[diff+1];
    S0: IP:= SUM(i,0,n[1],SUM(j,0,n[2],a[lp,c[i,j]] × a[rp,c[n[1]-i,n[2]-j]])); goto END;
    S1: IP:= SUM(i,1,n[1],SUM(j,0,n[2],i × a[rp,c[i,j]] × a[lp,c[n[1]-i,n[2]-j]])); goto END;
```

```
      N do begin PUSPACE(8); ABSFIXP(1,0,j) end; PUNLCR; PUSPACE(16);
      PUTEXT1(‹u[›); ABSFIXP(1,0,1); PUTEXT1(‹,i,j]›)
    end; PUNLCR
  end
end TAYLOR;
real procedure FAC(n); value n; integer n;
begin integer i; A: if n ≤ kf then FAC:= fac[n] else
  begin for i:= 1 step 1 until 5 do fac[kf+i]:= fac[kf+i-1] × (kf+i);
    kf:= kf + 5; goto A
end end;
procedure CHANGE(i,j); value i,j; integer i,j;
begin H[i,1]:= H[j,1]; H[i,2]:= H[j,2]; H[i,3]:= H[j,3] end;
procedure CALCULATE(n,i,unknown); value n; integer n; integer array i,unknown;
begin integer j,m,K; array c[0:n]; K:= k; m:= n - 1;
A: for j:= 1 step 1 until n do c[j]:= VALUE(DIFF(i[n],unknown[j]));
  c[0]:= - VALUE(i[n]); if abs(c[n]) < 10-10 then
  begin if m = 0 then begin PUTEXT1(‹system is unsolvable›); stop end;
    k:= K; j:= i[m]; i[m]:= i[n]; i[n]:= j; m:= m - 1; goto A
  end; j:= NUMBER(c[0]/c[n]); for m:= 1 step 1 until n - 1 do
  j:= if abs(c[m]) < 10-10 then j else S(j,P(NUMBER(-c[m]/c[n]),unknown[m]));
  CHANGE(unknown[n],j); if n > 1 then
  begin CALCULATE(n-1,i,unknown); CHANGE(unknown[n],NUMBER(VALUE(unknown[n])))
end end;

kc:= k:= -1; NUMBER(0); NUMBER(1); kf:= 0; fac[0]:= 1; Xf:= STORE(-4,0,0);
PHI[1]:= STORE(-4,0,0); PHI[2]:= STORE(-4,0,0); INFORM FUNC(0,0,0);
```

```
    begin if f[m1] < 0 then a[phi[kpsi,j],c[n[1],n[2]]]:= (if n[1] = 0 ∧ n[2] = 0 then
      INFORM FUNC(a[m1,0],H[i,1],j) else IP(psi[kpsi,j],m1,p)/n[p] +
      a[psi[kpsi,j],0] × a[m1,c[n[1],n[2]]])
      else f[phi[kpsi,j]]:= S(NUMBER(IP(psi[kpsi,j],m1,p)/n[p]),
      P(NUMBER(a[psi[kpsi,j],0]),f[m1]))
  end end else if H[i,1] = -4 then CALC ARRAY:= H[i,3];

  comment Continuation of TAYLOR; K:= k; KC:= kc; n[1]:= 0; n[2]:= 0;
  for k:= 1 step 1 until AB do
  begin f[k]:= -1; for j:= 0 step 1 until (if dimension = 2 then ((N-1)×(N+2)): 2
    else N-1) do a[k,j]:= 0
  end; for l:= 0 step 1 until N-1 do for j:= 0 step 1 until N-1-l do
  c[l,j]:= (j × (2 × N + 1 - j)): 2 + 1; for l:= 1 step 1 until M do
  begin tp[l,1]:= type[l] - 2 × (type[l]: 2); tp[l,2]:= type[l]: 2 end;
AA: k:= K; kc:= KC; for l:= 1 step 1 until M do unknown[l]:= NUMBER(0);
  kpsi:= ka:= 0; for l:= 1 step 1 until M do formula[l]:= f[CALC ARRAY(F[l])];
  if n[1] = 0 ∧ n[2] = 0 then goto CALC PSI; CALCULATE(M,formula,unknown);
  for l:= 1 step 1 until M do u[l,n[1]+tp[l,1],n[2]+tp[l,2]]:= VALUE(unknown[l])/(1 +
  (if type[l] = 0 then 0 else n[type[l]])); for j:= 1 step 1 until ka do
  begin if f[j] ≥ 0 then
    begin a[j,c[n[1],n[2]]]:= VALUE(f[j]); CHANGE(f[j],NUMBER(a[j,c[n[1],n[2]]]))
  end end;
CALC PSI: for j:= 1 step 1 until kpsi do
  begin l:= H[psi[j,-1],1]; H[Xf,3]:= psi[j,0]; f[psi[j,0]]:= -1; for m:= 1 step 1
    until PSI[1,0] do begin H[PHI[m],3]:= phi[j,m]; f[phi[j,m]]:= -1 end; for m:= 1
    step 1 until PSI[1,0] do psi[j,m]:= CALC ARRAY(PSI[l,m])
  end; if dimension = 2 then
  begin n[1]:= n[1] - 1; n[2]:= n[2] + 1; if n[1] = -1 then
    begin n[1]:= n[2]; n[2]:= 0 end; if n[1] ≠ N then goto AA end else
  begin n[1]:= n[1] + 1; if n[1] ≠ N then goto AA end;
  k:= K; kc:= KC; for l:= 1 step 1 until M do
  begin PUNLCR; for K:= (if dimension = 1 then 0 else if type[l] = 0 then N-1
    else N) step -1 until 0 do
    begin PUNLCR; ABSFIXP(1,0,K); for j:= 0 step 1 until N - K - (if type[l] = 0
      then 1 else 0) do FLOP(5,1,u[l,j,K])
    end; PUNLCR; PUTEXT1(≠ j,i ≠); ABSFIXP(1,0,0); for j:= 1 step 1 until
```

```
PUTEXT1(‹
Results of test program for the Cauchy problem RPR 310565/39492›);
begin integer K,KC,n,un,x,y,xy; integer array F,U,dUdx,dUdy,xttp,yttp[1:2];
  array u[1:2,0:4,0:4];
  procedure A(d,M,F1,F2,x0,y0); integer d,M,F1,F2; real x0,y0;
  begin k:= K; kc:= KC; F[1]:= F1; F[2]:= F2; PUNLCR;
    PUTEXT1(‹The differential equation(s)›); PUNLCR; OUTPUT(F[1]);
    PUTEXT1(‹ = 0›); if M = 2 then
    begin PUNLCR; OUTPUT(F[2]); PUTEXT1(‹ = 0›) end;
    PUNLCR; PUTEXT1(‹dimension      M      x0      (y0)›); PUNLCR;
    PUSPACE(2); ABSFIXP(1,0,d); PUSPACE(10); ABSFIXP(1,0,M); PUSPACE(6);
    FLOP(2,1,x0); if d = 2 then FLOP(2,1,y0); TAYLOR(d,M,F,4,x0,y0,u)
  end; for n:= 1,2 do
  begin U[n]:= STORE(-10,0,n); dUdx[n]:= STORE(-10,-1,n); dUdy[n]:= STORE(-10,-2,n);
    xttp[n]:= STORE(-2,0,n); yttp[n]:= STORE(-3,0,n)
  end; x:= xttp[1]; y:= yttp[1]; xy:= P(x,y); K:= k; KC:= kc;
  n:= -1; for un:= 1,1,0,0,0 do begin n:= n + 1; u[1,n,0]:= un end; u[1,0,1]:= 0;
  A(2,1,S(P(U[1],dUdy[1]),x),0,0,0);
  u[1,0,0]:= 1;
  A(2,1,S(LN(U[1]),y),0,0,0);
  u[1,0,0]:= 1;
  A(1,1,D(ARCTAN(Q(SIN(U[1]),COS(U[1]))),EXP(xttp[2])),0,0,0);
  u[1,0,0]:= u[2,1,1]:= 0; u[1,1,0]:= u[2,0,0]:= 1;
  A(1,2,D(dUdx[1],U[2]),  S(dUdx[2],U[1]),0,0);
  u[1,0,0]:= 0; u[2,0,0]:= 1;
  A(2,2,D(U[1],LN(xy)),  D(U[2],SQRT(xy)),1,1);
  u[1,0,0]:= u[2,0,0]:= -1;
  A(2,2,S(Q(1,U[1]),Q(COS(x),S(xy,1))),  S(S(xy,1),P(U[2],COS(x))),0,0);
  for n:= 0,1,2,3,4 do u[1,0,n]:= u[2,n,0]:= 1/FAC(n); u[1,1,0]:= u[2,0,1]:= 0;
  A(2,2,D(LN(S(dUdx[1],S(dUdy[2],1))),LN(S(xy,1))),
    D(SQRT(S(D(dUdx[1],dUdy[2]),1)),SQRT(S(xy,1))),0,0)
 end
end
end
```

Results of test program for the Cauchy problem RPR 310565/39492

The differential equation(s)

((U[1]×dU[1]/dy)+x↑(1)) = 0

dimension	M	x0	(y0)
2	1	+.00	+.00

4	-.00000				
3	-.00000	-.00000			
2	-.00000	+.00000	$-.50000_{10}+0$		
1	+.00000	$-.10000_{10}+1$	$+.10000_{10}+1$	$-.10000_{10}+1$	
0	$+.10000_{10}+1$	$+.10000_{10}+1$	+.00000	+.00000	+.00000
j,i	0	1	2	3	4
		u[1 ,i,j]			

The differential equation(s)

(ln(U[1])+y↑(1)) = 0

dimension	M	x0	(y0)
2	1	+.00	+.00

3	$-.16667_{10}+0$				
2	$+.50000_{10}+0$	-.00000			
1	$-.10000_{10}+1$	-.00000	-.00000		
0	$+.10000_{10}+1$	-.00000	-.00000	-.00000	
j,i	0	1	2	3	4
		u[1 ,i,j]			

The differential equation(s)

(arctan((sin(U[1])/cos(U[1]))+((-$.100_{10}+1$)×exp(x↑(2)))) = 0

dimension	M	x0	(y0)
1	1	+.00	

0	$+.10000_{10}+1$	-.00000	$+.10000_{10}+1$	+.00000	
j,i	0	1	2	3	4
		u[1 ,i,j]			

The differential equation(s)
(dU[1]/dx+((-.100_{10}+1)×U[2])) = 0
(dU[2]/dx+U[1]) = 0
dimension M x0 (y0)
1 2 +.00

0 +.00000 +.10000_{10}+1 +.00000 -.16667_{10}+0 -.00000
j,i 0 1 2 3 4
u[1 ,i,j]

0 +.10000_{10}+1 +.00000 -.50000_{10}+0 -.00000 +.41667_{10}-1
j,i 0 1 2 3 4
u[2 ,i,j]

The differential equation(s)
(U[1]+((-.100_{10}+1)×ln((x∧(1)×y∧(1))))) = 0
(U[2]+((-.100_{10}+1)×sqrt((x∧(1)×y∧(1))))) = 0
dimension M x0 (y0)
2 2 +.10_{10}+1 +.10_{10}+1

3 +.33333_{10}+0
2 -.50000_{10}+0 +.00000
1 +.10000_{10}+1 -.00000 +.00000
0 +.00000 +.10000_{10}+1 -.50000_{10}+0 +.33333_{10}+0
j,i 0 1 2 3 4
u[1 ,i,j]

3 +.62500_{10}-1
2 -.12500_{10}+0 -.62500_{10}-1
1 +.50000_{10}+0 +.25000_{10}+0 -.62500_{10}-1
0 +.10000_{10}+1 +.50000_{10}+0 -.12500_{10}+0 +.62500_{10}-1
j,i 0 1 2 3 4
u[2 ,i,j]

The differential equation(s)

$(((+.100_{10}+1)/U[1])+(\cos(x\wedge(1))/((x\wedge(1))\times y\wedge(1))+(+.100_{10}+1)))) = 0$

$(((x\wedge(1))\times y\wedge(1))+(+.100_{10}+1))+(U[2]\times\cos(x\wedge(1)))) = 0$

dimension	M	x0	(y0)
2	2	+.00	+.00

3	-.00000				
2	-.00000	-.00000			
1	-.00000	$-.10000_{10}+1$	-.00000		
0	$-.10000_{10}+1$	-.00000	$-.50000_{10}+0$	-.00000	
j,i	0	1	2	3	4

u[1 ,i,j]

3	-.00000				
2	-.00000	-.00000			
1	-.00000	$-.10000_{10}+1$	-.00000		
0	$-.10000_{10}+1$	-.00000	$-.50000_{10}+0$	-.00000	
j,i	0	1	2	3	4

u[2 ,i,j]

The differential equation(s)

(ln((dU[1]/dx+(dU[2]/dy+(+.100_{10}+1))))+
((-.100_{10}+1)×ln(((x^(1)×y^(1))+(+.100_{10}+1))))) = 0

(sqrt(((dU[1]/dx+((-.100_{10}+1)×dU[2]/dy))+(+.100_{10}+1)))+
((-.100_{10}+1)×sqrt(((x^(1)×y^(1))+(+.100_{10}+1))))) = 0

dimension	M	x0	(y0)
2	2	+.00	+.00

u[1 ,i,j]

j,i	0	1	2	3	4
4	+.41667_{10}-1				
3	+.16667_{10}+0	-.00000			
2	+.50000_{10}+0	-.00000	-.00000		
1	+.10000_{10}+1	-.00000	+.50000_{10}+0	-.00000	
0	+.10000_{10}+1	+.00000	-.00000	-.00000	-.00000

u[2 ,i,j]

j,i	0	1	2	3	4
4	+.00000				
3	+.00000	+.00000			
2	+.00000	+.00000	+.00000		
1	+.00000	+.00000	+.00000	+.00000	
0	+.10000_{10}+1	+.10000_{10}+1	+.50000_{10}+0	+.16667_{10}+0	+.41667_{10}-1

Table of contents

1. Introduction

Many physical problems lead to the so-called Cauchy problem (see ref [2] p. 39), i.e. the determination of functions $U_k(x,y)$ satisfying the partial differential equations

$$F_l\left(\frac{\partial U_k}{\partial x}, \frac{\partial U_k}{\partial y}, U_k, x, y\right) = 0 \qquad (1.1)$$

for $k = 1, \ldots, M$ and $l = 1, \ldots, M$
and the initial conditions

$$U_k(x,0) = G_k(x) \quad \text{for } k = 1, \ldots, M. \qquad (1.2)$$

Sometimes it is possible to express the functions U_k in the form of Taylor series

$$U_k(x,y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} u_{k,i,j}\, x^i y^j. \qquad (1.3)$$

In this report we will be concerned with the calculation of the Taylor coefficients $u_{k,i,j}$.
In general this is an elementary but tedious calculation, we made therefore an ALGOL 60 procedure (called TAYLOR), by means of which a digital computer can calculate these coefficients.

Extensive use is made in this procedure of a set of ALGOL 60 procedures by which one can do algebra with the computer. These procedures will be described in section 4, they enable the user to store, differentiate, evaluate, or output a rather general type of formulae. We remark that a similar set of procedures was used by the author in ref [1], to calculate a second-order approximation of the solution of a complicated non-linear problem.

We require in the sequel that the Cauchy problem (1.1), (1.2) satisfies the following conditions

1. The differential equations (1.1) and initial functions (1.2) are analytic in their variables in a nabourhood of the point $x = 0$, $y = 0$

2. $$\det \left\{\frac{\partial F_l}{\partial Q_k}\right\} \neq 0 \text{ in } x=0,\ y=0 \qquad (1.4)$$

with $$Q_k = \frac{\partial U_k}{\partial y}$$

3. The M equations

$$F_l\left(\frac{\partial U_k}{\partial x}\Big|_{x=0,y=0}, \frac{\partial U_k}{\partial y}\Big|_{x=0,y=0}, U_k(0,0),0,0\right) = 0 \qquad (1.5)$$

are explicitely solvable for the quantities $\frac{\partial U_k}{\partial y}\Big|_{x=0,y=0}$.

We require that these quantities are given as part of the initial conditions.

There are two cases for which condition 2 is not satisfied, but for which the Taylor coefficients may still be calculatable.

1. For a certain k, the variable $\frac{\partial U_k}{\partial y}$, does not occur in the equations (1.1), but condition (1.4) still holds if Q_k is replaced by $\frac{\partial U_k}{\partial x}$.

   The initial conditions should then be replaced by $U_k(0,y) = G_k(y)$, and $\frac{\partial U_k}{\partial x}\Big|_{x=0,y=0}$ should be given beforehand.

2. For a certain k neither one of the variables $\frac{\partial U_k}{\partial x}$ or $\frac{\partial U_k}{\partial y}$ enter in equations (1.1), but condition (1.4) still holds if Q_k is replaced by U_k.

   In this case one does not need an initial condition for U_k.

   We require however (condition 3) that $U_k(0,0)$ is given.

Important examples of these cases are

1. ordinary differential equations: $F_l\left(\frac{\partial U_k}{\partial x}, U_k, x\right) = 0$
2. implicit equations $F_l(U_k, x, y) = 0$.

In section 2 we will define the process by which the Taylor coefficients can be calculated.

In section 3 a motivation of the general form of the Cauchy problem will be given, together with a discussion of some literature.

Section 4 is devoted to the ALGOL 60 procedures for doing algebra on the computer.

The procedure TAYLOR will be discussed in section 5 and in section 6 we reproduce the program by which some test cases were calculated.
We remark that the procedure TAYLOR is very inefficient with respect to memory space and calculation time.
In a forth-coming report however, we will describe a procedure which generates an ALGOL program.
The generated ALGOL program is then intended for the actual calculation of the Taylor coefficients, and it is made as efficient as is reasonably possible with respect to memory space and calculation time.

2.1. The Cauchy problem

Consider the differential equations (1.1) and differentiate the F_1 n times with respect to x and m times with respect to y, obtaining for $n+m > 0$

$$\sum_{k=1}^{M} \left(\frac{\partial F_1}{\partial P_k} \frac{\partial^{n+m} P_k}{\partial x^n \partial y^m} + \frac{\partial F_1}{\partial Q_k} \frac{\partial^{n+m} Q_k}{\partial x^n \partial y^m} \right) + \ldots = 0 \qquad (2.1)$$

in which $P_k = \frac{\partial U_k}{\partial x}$ and $Q_k = \frac{\partial U_k}{\partial y}$.
Terms involving derivatives of P_k, Q_k and U_k

$$\left.\begin{array}{l} \dfrac{\partial^{i+j} P_k}{\partial x^i \partial y^j} \\ \dfrac{\partial^{i+j} Q_k}{\partial x^i \partial y^j} \end{array}\right\} \quad \text{with } i \leq n,\ j \leq m \text{ and } i+j \neq n+m$$

and $$\frac{\partial^{i+j} U_k}{\partial x^i \partial y^j} \quad \text{with } i \leq n,\ j \leq m,$$

and terms involving partial derivatives of F_1 with respect to x and y are represented by the dots in (2.1).

The Taylor series for U_k is

$$U_k = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} u_{k,n,m}\, x^n y^m. \qquad (1.3)$$

(Here and in the sequel we denote the function by a capital letter and its Taylor coefficients by a small letter).

Since

$$\left.\begin{aligned} \frac{\partial^{n+m} P_k}{\partial x^n \partial y^m} &= (n+1)!\, m!\, u_{k,n+1,m} \\ \text{and} \\ \frac{\partial^{n+m} Q_k}{\partial x^n \partial y^m} &= n!\, (m+1)!\, u_{k,n,m+1}, \end{aligned}\right\} \qquad (2.2)$$

relation (2.1) is a linear relation between the coefficients $u_{k,n,m+1}$ and $u_{k,n+1,m}$.

We may therefore $u_{k,n,m+1}$ express as a function of the $u_{l,i,j}$, with $l = 1, \ldots, M$ and $(i,j) \in V_{n,m}$ (see fig. 1).

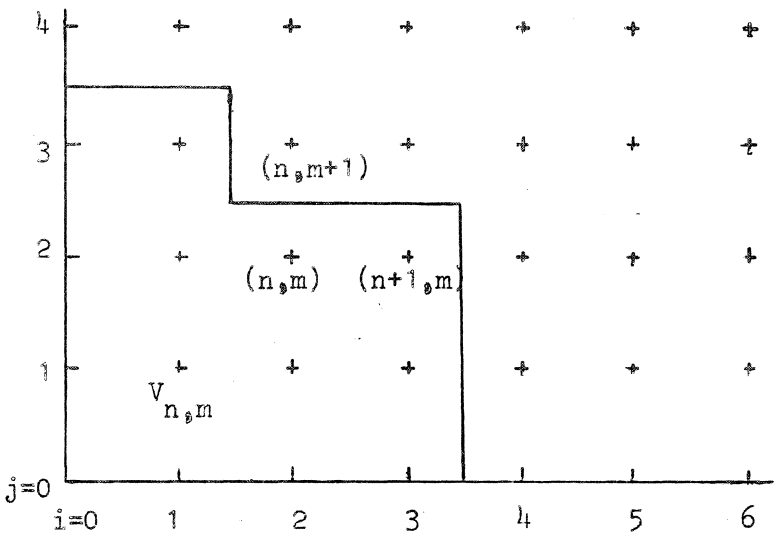

fig. 1

fig. 1 represents a grid of points (i,j) corresponding to the index pairs i,j.

Thus $$u_{k,n,m+1} = A(u_{l,i,j}) \qquad (2.3)$$

with $$(i,j) \in V_{n,m} \,. \qquad (2.4)$$

From the initial conditions and condition (1.5), the $u_{k,i,0}$, for $i = 0, 1, 2, \ldots$ and $u_{k,0,1}$ can be obtained.
We are thus able to calculate consecutively the $u_{k,n,m+1}$ for the points

$$(n,m) = (1,0), (0,1), (2,0), (1,1), (0,2), (3,0), (2,1), (1,2), (0,3), (4,0), (3,1), (2,2), \ldots \,.$$

In the sequel we will use the ordering of the grid points (i,j) just as is indicated by the above sequence.
This ordering is precisely the order of the grid points lying on the broken line in fig. 2, drawn from $(0,0)$ to (n,m).
To each grid point we assign an index $S(n,m)$, equal to the number of points preceding and including the point (n,m) on this broken line.
Then

$$S(n,m) = \frac{1}{2}(n + m + 1)(n + m) + m + 1. \qquad (2.5)$$

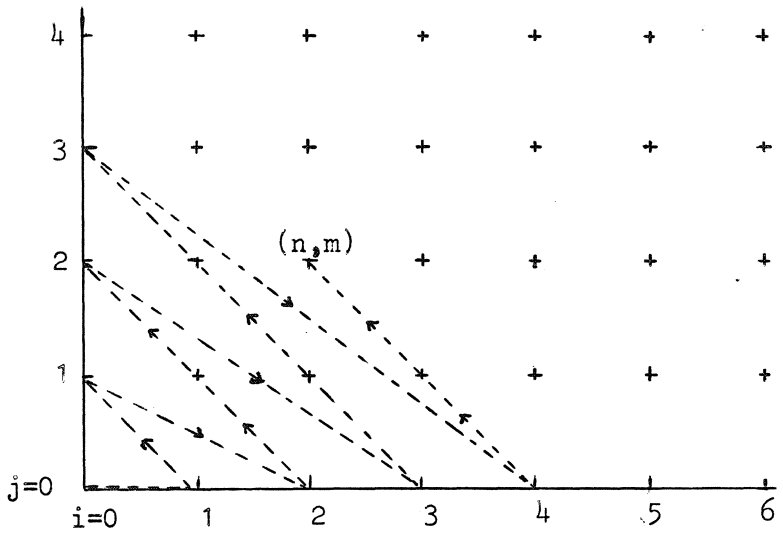

fig. 2

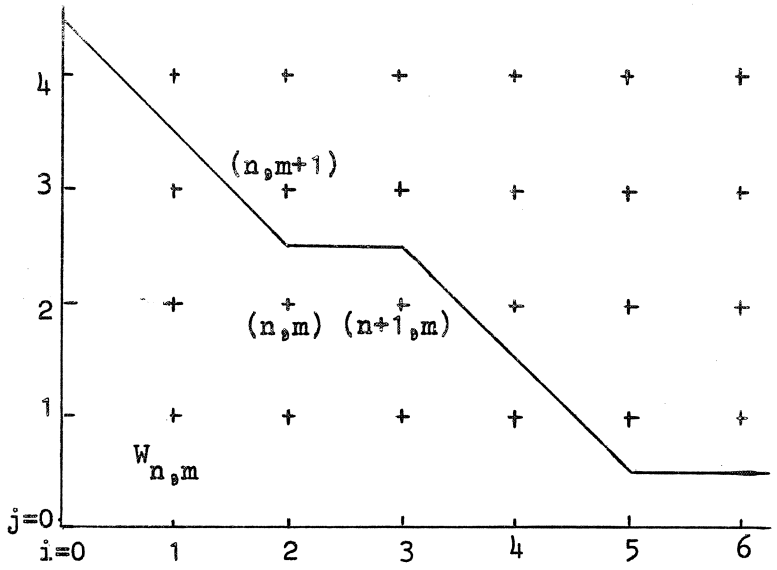

fig. 3

The order in which the $u_{k,n,m}$ are calculated, is such that $S(n,m)$ increases.

In fig. 3 we illustrate which $u_{k,i,j}$ are already calculated or known, if it is the turn to $u_{k,n,m+1}$ to be calculated, these index pairs (i,j) lie in $W_{n,m}$.

Above we sketched how the Taylor coefficients can be calculated for the case that $\frac{\partial U_k}{\partial y}$ enters in the differential equations.

It is obvious that the whole procedure can also be derived in the cases that $\frac{\partial U_k}{\partial y}$ or both $\frac{\partial U_k}{\partial y}$ and $\frac{\partial U_k}{\partial x}$ are lacking in the differential equations.

The method just described can be laid down in an ALGOL program. This program has to differentiate the differential equations and construct from the derivative, the function A in equ. (2.3), by which $u_{k,n,m+1}$ can be calculated. This is already done by Perlis, Iturriagia and Standish [8].

There is however a major disadvantage attached to this approach. This disadvantage is that the derivatives of the differential equations, on the form of which we do not want to impose heavy restrictions, will in general become very lengthy so that the memory capacity of the computer will soon be exhausted.

We chose therefore an alternative method, which is more arduous to describe but which has the advantages that, firstly the memory capacity will not as soon be exhausted and secondly it bears the possibility to separate the algebraic part of the calculation from the numerical part of the calculation in two different programs (which will be the subject of a next report).

2.2. The structure of the differential equations

Before going on, we have to define the structure of the differential equations.
In section 4 this structure will be defined by means of Backus notation, we will define it now in a less stringent way.

Roughly speaking, the differential equations are formulae in which the only operations are the algebraic operations: sum, difference, product and quotient.
These operations will be assumed to be binary.
The other constituants of the formulae are: real numbers, the variables $\frac{\partial U_k}{\partial x}$, $\frac{\partial U_k}{\partial y}$, U_k, x, y and functions of a single parameter (the parameter may be a formula itself).

Since the difference of two formulae: A - B, can be replaced by A + (-1).B, we will assume that the difference operation does not occur in the differential equations.

Putting it more precisely, the differential equations may have the form of the formula F, which is recursively defined in the table below in which A and B are also of the form F. We require that A and B are less "complicated", or contain less algebraic operations and function symbols than F. In other words, F should be built up with a finite number of symbols.

case	F	
1	A+B	
2	A.B	
3	A/B	
4	c	c is some real number
5	U_k	k is some positive integer
6	$\frac{\partial U_k}{\partial x}$	" "
7	$\frac{\partial U_k}{\partial y}$	" "
8	x^p	p is some positive integer
9	y^p	" "
10	$\Phi(A)$	Φ is some function symbol

table 1

If F is of the form 1, 2 or 3 we call F a compound formula, in the case 4, F is called a number, in the cases 5 until 9, F is called a simple term and in the case 10, F is called a function.

For the function Φ we require firstly, that there exists a set of first order ordinary differential equations

$$\frac{d\Phi_k}{dX} = \Psi_k(X, \Phi_1, \ldots, \Phi_\nu) \quad \text{for } k=1,\ldots,\nu \qquad (2.6)$$

of which $\Phi = \Phi_1(X)$ is a solution, and secondly that the value of the functions $\Phi_\nu(X)$ is well-defined for every real X.

For the form of the functions Ψ_k we require that it is the same form as the form of the function Ψ, defined in table 2, in which A and B are also of the form Ψ.

Also for Ψ require that it has a finite structure.

case	Ψ	
1	A+B	
2	A.B	
3	A/B	
4	c	c is some real number
5	Φ_σ	σ is some positive integer
6	X	

table 2

The functions used in this report are the standard functions of the ALGOL 60 report (ref [4]) (excluding abs and sign).

The differential equations of these functions are listed in table 3.

Φ_1	ν	Ψ
sin	2	$\Psi_1 = \Phi_2$; $\Psi_2 = -\Phi_1$; Φ_2 = cos
cos	2	$\Psi_1 = -\Phi_2$; $\Psi_2 = \Phi_1$; Φ_2 = sin
exp	1	$\Psi_1 = \Phi_1$
ln	1	$\Psi_1 = 1/X$
sqrt	1	$\Psi_1 = .5/\Phi_1$
arctan	1	$\Psi_1 = 1/(1 + X.X)$

table 3

Let us illustrate the form of F by means of the following examples, which will also be used further on.

Example 1. The differential equation $F_1 = U_1 \frac{\partial U_1}{\partial y} + x = 0$ can be brought in the form of a formula by defining the $A_1, \ldots, A_5$

$$A_1 = U_1$$

$$A_2 = \frac{\partial U_1}{\partial y}$$

$$A_3 = A_1 \circ A_2$$

$$A_4 = x$$

$$F_1 = A_5 = A_3 + A_4$$

Example 2. The implicit equation $F_1 = \ln U_1 + y = 0$

$$A_1 = U_1$$

$$A_2 = \ln(A_1) = \Phi_1(A_1)$$

$$A_3 = y$$

$$F_1 = A_4 = A_2 + A_3$$

$$A_5 = 1$$

$$A_6 = X$$

$$\frac{d\Phi_1}{dX} = \Psi_1 = A_7 = A_5/A_6$$

We will now describe the recursive process for calculating the Taylor coefficients $f_{n,m}$ of F, if it is assumed that the $u_{k,i,j}$ are known for all (i,j).
In section 2.2.5 we will clarify this odd assumption.

2.2.1 The zeroth order Taylor coefficients

To start the calculation process we have to define $f_{0,0}$.
If F is of the form 4 until 9 in table 1, then $f_{0,0}$ follows from table 5 at the next page.

Let F be of the form 1, 2, 3 or 10 in table 1.
Assume $a_{0,0}$ and $b_{0,0}$ are already calculated then $f_{0,0}$ follows from

case	$f_{0,0}$
1	$a_{0,0} + b_{0,0}$
2	$a_{0,0} \circ b_{0,0}$
3	$a_{0,0}/b_{0,0}$
10	$\Phi(a_{0,0})$

table 4

Using the fact that A and B are defined in the same way as F but are "less complicated", it follows that all zero$^{\underline{th}}$ order Taylor coefficients of the formulae constituting the differential equations (1.1) are defined.

2.2.2 The Taylor coefficients of a number or a simple term

If F is of the form 4 until 9 in table 1, then $f_{n,m}$ can be calculated by means of

case	$f_{n,m}$
4	$c \circ \delta_{n,0} \circ \delta_{m,0}$ $\quad \delta_{i,j}$ is the Kronecker δ
5	$u_{k,n,m}$
6	$(n + 1)u_{k,n+1,m}$
7	$(m + 1)u_{k,n,m+1}$
8	$\delta_{n,p}\ \delta_{m,0}$
9	$\delta_{n,0}\ \delta_{m,p}$

table 5

2.2.3 The Taylor coefficients of a compound formula

Let F be of the form 1, 2 or 3 in table 1.

We assume:

1. all $f_{i,j}$ are already calculated for $S(i,j) \leq S(n,m) - 1$ in which S is defined in (2.5)

2. all $a_{i,j}$ and $b_{i,j}$ are already calculated for $S(i,j) \leq S(n,m)$

Then $f_{n,m}$ follows from the following table

case	$f_{n,m}$
1	$a_{n,m} + b_{n,m}$
2	$\sum_{i=0}^{n} \sum_{j=0}^{m} a_{i,j}\, b_{n-i,m-j}$
3	$\frac{1}{b_{0,0}} \{a_{n,m} - f_{0,0}\, b_{n,m} - \sum_{i=0}^{n} \sum_{j=0}^{m}{}' f_{i,j} b_{n-i,m-j}\}$ (The symbol $\sum\sum'$ is defined by $\sum_{i=0}^{n} \sum_{j=0}^{m}{}' c_{i,j} = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} - c_{0,0} - c_{n,m}$)

table 6

By means of this table $f_{n,m}$ is recursively defined, firstly with respect to its indexes and secondly with respect to the way in which F is defined.

If the original differential equations (1.1) did not contain function symbols, then the calculation process, concerning $f_{n,m}$ would be completely defined.

Let us illustrate therefore this case by means of example 1.

If the coefficients $u_{1,i,j}$ were known then each right-hand side of the equations below is a well-defined number.

Example 1. (continued) $n = 0 \quad m = 0$

$$a_{1,0,0} = u_{1,0,0}$$

$$a_{2,0,0} = u_{1,0,1}$$

$$a_{3,0,0} = a_{1,0,0}\, a_{2,0,0}$$

$$a_{4,0,0} = 0$$

$$f_{1,0,0} = a_{5,0,0} = a_{3,0,0} + a_{4,0,0}$$

$n = 1 \quad m = 0$

$$a_{1,1,0} = u_{1,1,0}$$

$$a_{2,1,0} = u_{1,1,1}$$

$$a_{3,1,0} = a_{1,0,0}\, a_{2,1,0} + a_{1,1,0}\, a_{2,0,0}$$

$$a_{4,1,0} = 1$$

$$f_{1,1,0} = a_{5,1,0} = a_{3,1,0} + a_{4,1,0}$$

etc.

2.2.4 The Taylor coefficients of a function

Let F be of the form 10 in table 1.

Assume:

1) The Taylor coefficients $a_{i,j}$ are already calculated for $S(i,j) \leq S(n,m)$

2) The Taylor coefficients $\phi_{1,i,j}, \ldots, \phi_{\nu,i,j}$ are already calculated for $S(i,j) \leq S(n,m) - 1$

3) The Taylor coefficients $\psi_{1,i,j}, \ldots, \psi_{\nu,i,j}$ are already calculated for $S(i,j) \leq S(n,m) - 1$

We remark that the zero-th coefficients $\phi_{k,0,0}$ and $\psi_{k,0,0}$ easily follow from

$$\phi_{k,0,0} = \Phi_k(a_{0,0}) \tag{2.8}$$

and

$$\psi_{k,0,0} = \Psi_k(a_{0,0}, \phi_{1,0,0}, \ldots, \phi_{\nu,0,0}) \tag{2.9}$$

Differentiating the functions $\Phi_k(A)$ with respect to x, we obtain

$$\frac{\partial \Phi_k}{\partial x} = \Psi_k(A, \Phi_1(A), \ldots, \Phi_\nu(A)) \circ \frac{\partial A}{\partial x}, \quad k=1,\ldots,\nu. \tag{2.10}$$

Hence we have for $n \neq 0$

$$\phi_{k,n,m} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=0}^{m} \psi_{k,n-i,m-j}\ i\ a_{i,j}, \tag{2.11}$$

from which $\phi_{k,n,m}$ and thus $f_{n,m} = \phi_{1,n,m}$ can be calculated.

If $n = 0$ then we may differentiate $\Phi_k(A)$ with respect to y and we get

$$\phi_{k,n,m} = \frac{1}{m} \sum_{i=0}^{n} \sum_{j=1}^{m} \psi_{k,n-i,m-j}\ j\ a_{i,j}. \tag{2.12}$$

The $\phi_{k,n,m}$ now being defined, we will determine $\psi_{k,n,m}$.
Comparing table 2 and table 1, we observe that if F does not contain function symbols, then Ψ_k would be an F_σ but for the variables X and Φ_σ.
The Taylor series for these variables are however known, therefore the $\psi_{k,n,m}$ can be calculated in the same way as $f_{n,m}$ in section 2.2.3 is calculated.

We have completely defined now the calculation process of $f_{n,m}$.
We close this section with the continuation of example 2.

Example 2. (continued) $n = 0 \quad m = 0$

$$a_{1,0,0} = u_{1,0,0}$$

$$\phi_{1,0,0} = a_{2,0,0} = \ln(a_{1,0,0})$$

$$a_{3,0,0} = 0$$

$$f_{1,0,0} = a_{4,0,0} = a_{2,0,0} + a_{3,0,0}$$

$$a_{5,0,0} = 1$$

$$a_{6,0,0} = a_{1,0,0}$$

$$\psi_{1,0,0} = a_{7,0,0} = a_{5,0,0}/a_{6,0,0}$$

$$\underline{n = 1 \quad m = 0}$$

$$a_{1,1,0} = u_{1,1,0}$$

$$\phi_{1,1,0} = a_{2,1,0} = a_{7,0,0}\, a_{1,1,0}$$

$$a_{3,1,0} = 0$$

$$f_{1,1,0} = a_{4,1,0} = a_{2,1,0} + a_{3,1,0}$$

$$a_{5,1,0} = 0$$

$$a_{6,1,0} = a_{1,1,0}$$

$$\psi_{1,1,0} = a_{7,1,0} = \frac{1}{a_{6,0,0}} \{a_{5,1,0} - a_{7,0,0}\, a_{6,1,0}\}$$

etc.

2.2.5 The Taylor coefficients of the unknown functions

At the end of section 2.2 we assumed that $u_{k,i,j}$ were known for all (i,j).
We shall now treat the more realistic case that the $u_{k,i,j}$ with $(i,j) \in W_{n,m}$ (fig. 3) are known and the $u_{k,n,m+1}$ are unknown.
Replace these $u_{k,n,m+1}$ by the quantities X_k.
Calculate now in an algebraic way, all Taylor coefficients $f_{n,m}$ according to the remarks in the sections above.
In particular the Taylor coefficients $f_{l,n,m}$ of the differential equations F_l themselves can be calculated.
The $f_{l,n,m}$ will be functions of the X_k.
From table 6 and formulae (2.11) and (2.12) it follows that these functions are linear in the X_k.

From equation (1.1) it follows that $f_{1,n,m} = 0$.
We have therefore M linear equations in the M unknown quantities X_k.
The condition (1.4) guarantees that the quantities X_k can be calculated.
Thus the $u_{k,n,m+1}$ can also be calculated. (We remark that in the program the quantities $(m+1)u_{k,n,m+1}$ are replaced by the quantities X_k, since then the matrix of the set of equations $f_{1,n,m} = 0$ is independent of n and m).

Repeating now the same arguments as in section 2.1 we see that all the coefficients $u_{k,n,m}$ are calculatable.

For the examples 1 and 2 we have

Example 1. (continued) Let us choose $u_{1,0,0} = 1$, $u_{1,1,0} = 1$.
Then $u_{1,0,1} = 0$ (here we use the requirement (1.5)).

$a_{1,0,0} = 1$; $a_{2,0,0} = 0$; $a_{3,0,0} = 0$; $a_{4,0,0} = 0$;
$f_{1,0,0} = a_{5,0,0} = 0$ (as it should be).

$a_{1,1,0} = 1$; $a_{2,1,0} = X_1$; $a_{3,1,0} = X_1$; $a_{4,1,0} = 1$;
$f_{1,1,0} = a_{5,1,0} = X_1 + 1 = 0$ thus $X_1 = -1$ and $u_{1,1,1} = -1$.
$a_{2,1,0} = -1$; $a_{3,1,0} = -1$.

Example 2. (continued) Requirement (1.5) gives $u_{1,0,0} = 1$.

$a_{1,0,0} = 1$; $\phi_{1,0,0} = a_{2,0,0} = 0$; $a_{3,0,0} = 0$;
$f_{1,0,0} = a_{4,0,0} = 0$ (as it should be);
$a_{5,0,0} = 1$; $a_{6,0,0} = 1$; $\psi_{1,0,0} = a_{7,0,0} = 1$.

$a_{1,1,0} = X_1$; $\phi_{1,1,0} = a_{2,1,0} = X_1$; $a_{3,1,0} = 0$
$f_{1,1,0} = a_{4,1,0} = X_1 = 0$, thus $X_1 = 0$ and $u_{1,1,0} = 0$.
$a_{1,1,0} = 0$; $\phi_{1,1,0} = a_{2,1,0} = 0$;
$a_{5,1,0} = 0$; $a_{6,1,0} = 0$; $\psi_{1,1,0} = a_{7,1,0} = 0$.

3. A less general Cauchy problem

Gibbons [3], Richtmyer [5] and Moore [6] constructed (not in ALGOL) programs for the calculation of Taylor coefficients.

The Cauchy problem for which these programs are constructed have the form

$$\frac{\partial U_l}{\partial y} = H_l\left(\frac{\partial U_k}{\partial x}, U_k, x, y\right) \quad \text{with } l=1,\ldots,M \text{ and } k=1,\ldots,M \tag{3.1}$$

together with initial conditions of the form (1.2).

This is a less general type than equations (1.1).
If the functions H_l are of the form of a formula as defined in this section, then we may in the way as described in section 2.2, calculate the Taylor coefficients $h_{l,n,m}$.
These coefficients are independent of the $u_{k,n,m+1}$ of course, we can therefore calculate immediately the $u_{k,n,m+1}$ from

$$u_{k,n,m+1} = \frac{1}{(m+1)}\, h_{k,n,m}$$

and we do not have to manipulate algebraically with unknown quantities X_k. This simplifies the program considerably, it is not even necessary to use the procedures for differentiating and evaluating a formula in the computer.

A peculiar fact is that the general form (1.1) can be brought into the above form (3.1).
Introduce namely M new functions $V_k = \frac{\partial U_k}{\partial y}$, differentiate (1.1) with respect to y, obtaining

$$\sum_{k=1}^{M} \left\{\frac{\partial F_l}{\partial P_k}\frac{\partial V_k}{\partial x} + \frac{\partial F_l}{\partial Q_k}\frac{\partial V_k}{\partial y} + \frac{\partial F_l}{\partial U_k}\cdot V_k\right\} + \frac{\partial F_l}{\partial y} = 0. \tag{3.2}$$

Let the inverted matrix of $\left\{\frac{\partial F_l}{\partial Q_k}\right\}$ be $\{a_{i,j}\}$, then we get the system

$$\left.\begin{aligned} \frac{\partial V_j}{\partial y} &= -\sum_{l=1}^{M} a_{j,l}\left\{\sum_{k=1}^{M}\left(\frac{\partial F_l}{\partial P_k}\frac{\partial V_k}{\partial x}+\frac{\partial F_l}{\partial U_k}V_k\right)+\frac{\partial F_l}{\partial y}\right\} \\ \frac{\partial U_k}{\partial y} &= V_k \end{aligned}\right\} \qquad (3.3)$$

which is of the desired form.

Thus we could have based ourselves upon the form (3.1).

There are however, three disadvantages connected with the translation of the equations (1.1) into (3.3).

1. One has to do the differentiating and inversion (of a matrix of functions) oneself, with the risk of making errors.
2. The formulae at the right-hand side of (3.3) will in general be far more lengthy than the original formulae F_l in (1.1)
3. The initial conditions for the functions V_k should be calculated beforehand, which is in general not easy.

These three disadvantages show clearly why we chose in favour of the more general form (1.1), though the program becomes more complicated.

The difficult calculations however, are completely overtaken by the program.

As for the more complicated form of the program, we remark that the procedures for doing algebra within the computer, enable us to construct still a rather lucid program.

In the articles of Richtmyer and Moore, the attention is drawn to an important feature, namely that the calculated Taylor coefficients of higher order, may become inaccurate, due to cancellation of significant digits in additioning and subtracting.

They recommend a method of calculation, in which the accuracy is also calculated.

In this report we do not enter upon this question, since we are mainly concerned with the general method of calculating implicitly defined Taylor coefficients.

We remark however, that it is possible to construct a program (see the end of section 1) which generates a machine code program in which this method of calculation is built in.

We close this section by remarking that the conditions 1 and 2 of the introduction are sufficient for convergence of the Taylor series for U_k in a nabourhood of the point x=0, y=0.
This follows at once from the Cauchy-Kowalewski theorem (see Courant Hilbert ref [2] p. 39).
We can in fact apply this theorem for the Cauchy problem (3.3), which, as was shown, is equivalent to the Cauchy problem (1.1).
We do not enter upon questions of rate of convergence or region of convergence.

4. Doing algebra with the computer

We define a formula by means of Backus notation (see the ALGOL 60 report ref [4]),

<p> ::= <unsigned integer> [1])
<simple term> ::= U[<p>] | dUdx[<p>] | dUdy[<p>] | xttp[<p>] | yttp[<p>]
<function identifier> ::= sin | cos | exp | ln | sqrt | arctan
<function> ::= <function identifier> (<formula>)
<operator> ::= + | - | * | /
<compound formula> ::= (<formula><operator><formula>)
<formula> ::= <simple term>|<number>|<function>|<compound formula>

For the definition of number and unsigned integer see the ALGOL 60 report.

Each formula is represented within the computer by a non-negative number, which we will call the index of the formula.
The representation is such that different formulae have different indexes, we may therefore identify the formula with its index.

[1]) We assume that p is unequal to zero.

4.1 Storing a formula

The set of procedures for storing a formula consists of the integer procedures S, D, P, Q, SIN, COS, EXP, LN, SQRT, ARCTAN and NUMBER. The ALGOL 60 definition is given in section 6.
The first four procedures store a compound formula, the next six procedures store a function and the procedure NUMBER stores a number.

The word "store" must be understood as to mean assigning to each formula an index and storing relevant information in the arrays

integer array H[0 : kmax, 1 : 3]
and the real array HC[0 : kcmax].

The integers kmax and kcmax should be chosen large enough.
As pointers for the arrays H and HC, the non-local integers k and kc are used.

Let the index of the stored formula be f, the indexes of the formulae a and b in the table below be a and b and c a real number, then we see from this table the ALGOL statement to be used and the information stored in the arrays H and HC.

f =	ALGOL statement	H[f,1] =	H[f,2] =	H[f,3] =	HC[kc] =
a + b	f := S(a,b)	a	1	b	
a - b	f := D(a,b)	a-b is converted by D into a+(-1)*b			
a * b	f := P(a,b)	a	2	b	
a/b	f := Q(a,b)	a	3	b	
sin(a)	f := SIN(a)	1	0	a	
cos(a)	f := COS(a)	2	0	a	
exp(a)	f := EXP(a)	3	0	a	
ln(a)	f := LN(a)	4	0	a	
sqrt(a)	f := SQRT(a)	5	0	a	
arctan(a)	f := ARCTAN(a)	6	0	a	
c	f := NUMBER(c)	-1	0	kc	c

table 7

Example 3. The formula f = arctan $(\frac{\sin(3.14)}{\cos(3.14)})$ = 3.14 is stored by means of the statement

f := D(ARCTAN(Q(SIN(NUMBER(3.14)),COS(NUMBER(3.14)))),NUMBER(3.14))

The effect of this statement can be seen from the following table for H and HC

		k	H[k,1]	H[k,2]	H[k,3]	kc	HC[kc]
	NUMBER	2	-1	0	2	2	3.14
	SIN	3	1	0	2		
	NUMBER	4	-1	0	3	3	3.14
	COS	5	2	0	4		
	Q	6	3	3	5		
	ARCTAN	7	6	0	6		
D	NUMBER	8	-1	0	4	4	-1
D	NUMBER	9	-1	0	5	5	3.14
D	P	10	8	2	9		
D	S	11	7	1	10		

table 8

The ultimate effect is that f gets the value 11.

We use in the program two formulae with fixed indexes 0 and 1. These formulae correspond to the numbers 0 and 1 respectively. They may be considered as the zero and the unit element in the set of formulae.

The procedures S and P are constructed such that the formulae

$0 + a,\ a + 0,\ 1 * a,\ a * 1,\ 0 * a,\ a * 0$ are replaced by

$a\ ,\ a\ ,\ a\ ,\ a\ ,\ 0\ ,\ 0$

Use of this zero and unit element is made within the procedure DIFF (for DIFFerentiating).

Concerning the simple terms we remark that within the program we declare

integer array U, dUdx, dUdy[1:M], xttp, yttp[1:POWER],

in which M and POWER are integers ≥ 1 (see equation (1.1)).
Before the differential equations are stored within the computer, these array elements get the values of the indexes corresponding to the simple terms.
The representation in the array H and the ordinary meaning of these simple terms follow from the table below, in which p is some positive integer.

ordinary notation	k	H[k,1]	H[k,2]	H[k,3]
U_p	U[p]	-10	0	p
$\frac{\partial U_p}{\partial x}$	dUdx[p]	-10	-1	p
$\frac{\partial U_p}{\partial y}$	dUdy[p]	-10	-2	p
x^p	xttp[p]	-2	0	p
y^p	yttp[p]	-3	0	p

table 9

Example 1 (continued). $F_1 = U_1 \frac{\partial U_1}{\partial y} + x$ is stored by the statement

F[1] := S(P(U[1], dUdy[1]), xttp[1])

Example 2 (continued). $F_1 = \ln(U_1) + y$ is stored by the statement

F[1] := S(LN(U[1]), yttp[1]).

4.2 Operations with a formula

In this section we describe procedures for the following operations

1. Evaluating a formula i (which can only be done if the formula has a numerical value, such as formula f in example 3 p. 21).
   This operation is carried out by the real procedure VALUE(i).
   The calculated value is assigned to the procedure identifier itself.

2. Differentiating a formula i, with respect to some other formula n.
   This is done by the integer procedure DIFF(i,n).
   The index of the stored derivative is assigned to the procedure identifier itself.

3. Definition of the output form of a formula i on output paper tape.
   This is the task of the procedure OUTPUT(i).
   OUTPUT is used here only for checking if the stored formula is correctly stored.
   It will however, be used in a next paper for generating an ALGOL 60 program.

All these procedures use the real procedure VADIPUCOMP FORM, VADIPU FUNC and VADIPUST.
The procedure VADIPUCOMP FORM determines for a compound formula the value , the derivative and the output.
The procedure VADIPU FUNC determines for a function only the output (use is made of the procedure INFORM FUNC).
The procedure VADIPUST determines only the output for a simple term.

One may regard the procedures VALUE, DIFF, OUTPUT and VADICOMP FORM as to be rather basic, they may be used for other problems.
The procedures VADIPU FUNC and VADIPUST, however, are special purpose procedures, their definition depends strongly on the particular problem under consideration.

4.3 The procedure INFORM FUNC

Within the real procedure INFORM FUNC(a,i,j) is defined all necessary information for the functions used in this report.

1. If i > 0 then the value $\Phi(a)$ is assigned to the procedure identifier.
2. If i < 0 then relevant output is punched on the output paper tape.
3. If i = 0 then the functions Ψ_k in table 3 are stored in the computer, therefore a call INFORM FUNC (0,0,0) should precede the actual computation.

It should be remarked that the functions Ψ_k do not have the form of a formula as defined in section 4.
We define therefore a psi formula by means of

<p> ::= <unsigned integer>
<psi simple term> ::= PHI[<p>] | Xf
<psi formula> ::= <psi simple term>|<number>|
(<psi formula><operator><psi formula>)

and require that Ψ_k has the form of a psi formula.
In the same way as in section 4.1 we can store a psi formula and assign an index to it, by means of the procedures S, D, P, Q, and NUMBER.
The internal representation of the psi simple terms is listed below, use is made of the non-local integer Xf and the non-local integer array PHI[1 : 2]

ordinary notation	k	H[k,1]	H[k,2]	H[k,3]
Φ_p	PHI[p]	-4	0	
X	Xf	-4	0	

table 10

We remark that the array elements H[k,3] get values later on, in TAYLOR.

The indexes of the psi formulae are stored in the non-local integer array PSI[1 : 6, 0 : 2].

In this report we use six different functions, the number ν for each function symbol is stored in PSI[j,0],
the indexes are stored in PSI[j,1] and in PSI[j,2]
(the PSI[j,2] is only used for a sine or a cosine function, see table 3).

5. The procedure TAYLOR

The heading of TAYLOR reads: procedure TAYLOR (dimension,M,F,N,x0,y0,u).
The differential equations (1.1) can be stored in the computer by means of the procedures of section 4.1.
There are M differential equations, the indexes of the left-hand sides of these differential equations are stored in the integer array F.
Although we treated in sections 1.2 and 3 the case that the point x=0, y=0 is the point in which the Taylor series are expanded, we shall assume here that the point where the Taylor series will be expanded is x=x0, y=y0.
In principle we want to calculate an infinite number of Taylor coefficients, in practice we can only calculate a finite number of them, namely e.g. those $u_{k,n,m}$ for which $n+m \leq N$, for a certain positive integer N.
The parameters M, F, N, x0 and y0 are now introduced, the parameter dimension equals 1 for one-dimensional problems and 2 for 2-dimensional problems. The parameter u is a real array, the array elements u[k,n,m] should become equal to the $u_{k,n,m}$ (equ.(1.3)); moreover the initial conditions are stored in u.

Each differential equation is built up by means of several formulae.
For each of these formulae the Taylor coefficients will be calculated.
We have to know therefore the number AB of these formulae.
It is necessary to know the type of the Cauchy problem with respect to a certain unknown function U_k (see section 1). We therefore introduce the integer array type[1 : M].

It is the task of the procedure INITIALIZE(i) to calculate

1. the number of formulae with which i is built up
2. to define the array elements type[k] according to the following table

$\frac{\partial U_k}{\partial x}$	$\frac{\partial U_k}{\partial y}$	U_k	type [k]
-	-	+	0
+	-	?	1
?	+	?	2

The symbols +, ? and - mean:

+ the quantity occurs in i

? the quantity may occur in i

- the quantity does not occur in i.

table 11

After the statement

for l := 1 step 1 until M do INITIALIZE(F[l])

the number AB and the array elements are known.
Moreover, the number of function symbols in the differential equations and the maximum of the ν's of these function symbols (see table 3) are calculated and are assigned to the integers kpsi and nu resp. [1])

Since we are interested in the $u_{k,n,m}$ for which $n+m \leq N$, it follows that the Taylor coefficients $f_{n,m}$ of the formulae which build up the differential equations are needed for $n+m \leq N-1$.
There are AB formulae we could therefore store the Taylor coefficients in the array a[1 : AB, 0 : N-1, 0 : N-1], however, about half the number of array elements would then be unused.
Therefore we store these Taylor coefficients in the two-dimensional array a[1 : AB, 0 : if dimension = 2 then ((N-1) × (N+2)) ÷ 2 else N-1], in which the number 1 + ((N-1) × (N+2)) ÷ 2 is just the sum of all Taylor coefficients $f_{n,m}$ with $n+m \leq N-1$.
The administration in the array a is governed by the integer array c[0 : N-1, 0 : N-1], whose values are equal to

$$c[1,j] = (j \times (2 \times N+1-j)) \div 2 + 1.$$

[1]) Note that TAYLOR is independent of the special choice of the functions.

In this way the array element a[k,c[1,j]], represents the Taylor coefficient with indexes 1 and j, of a certain formula; we call k the Taylor index of this formula.

We note that this is the only time that we think in terms of efficiency. The fact that we use for a number, or a simple term corresponding to the variable x^p, e.g., a full array, almost completely filled with zeros, is very inefficient but this will be of no interest in this report. It is the purpose of a next report to take away these inefficiencies.

As is already said the initial conditions should be stored in the array u, corresponding of course to the type of the Cauchy problem. We remark that instead of equations (1.2) we require that the Taylor coefficients $u_{k,j,0}$ should be given, for $j = 0, \ldots, N$.
One has to calculate therefore the Taylor coefficients of $G_k(x)$, beforehand, possibly by means of the procedure TAYLOR.
It is further required that $u_{k,0,1}$ is given, which follows from the solvability condition (1.5).
It does not need saying that the above remarks concern the case that type[k] = 2, in the other cases we need similar requirements for the $u_{k,n,m}$.
It should be remarked that the integer array elements n[1] and n[2], serve as the current indexes n and m respectively.

The piece of the procedure body of TAYLOR from the procedure declarations until the label AA is now described.

5.1 Outline of the calculation process

We shall briefly repeat the calculation process just as it is defined in section 2.

1. All zero-th order Taylor coefficients for n[1] = 0, n[2] = 0 are calculated.
   This is possible, due to the initial conditions.
   It is done after a call for the procedure CALC ARRAY.

2. For a certain n[1] and n[2], the calculation of the Taylor coefficients a[1,c[n[1], n[2]]], should be performed.
The unknown quantities (n[2]+1)u[k,n[1],n[2]+1] are beforehand replaced by the formulae with indexes unknown [k], for k = 1, ..., M. (The unknown [k] plays the role of the X_k in section 2.2.5).
Some coefficients can then actually be calculated by means of the remarks in section 2 (e.g., the coefficient corresponding to a number or a power of x).
Other coefficients may not be calculatable, in the place of them however, formulae in the unknown's are stored in the integer array f[1 : AB].
This is done again by the procedure CALC ARRAY.

3. Let the indexes of the formulae in the unknown's for the Taylor coefficients of the differential equations themselves, be stored in the array elements formula [l] for l = 1, ..., M.
These formulae are linear in the quantities unknown [k] with k = 1, ..., M.
We may therefore calculate the unknown quantities.
This is done after a call for the procedure CALCULATE.

4. The unknown quantities being known, we can calculate firstly the u[k,n[1], n[2]+1] and secondly the until yet uncalculated coefficients a[l,c[n[1], n[2]]], for which in f[l] a formula was stored.

5. The eventually used Taylor coefficients for the functions Ψ in section 2.2.4 can be calculated, again by means of CALC ARRAY.

6. The n[1] and n[2] are changed such that S(n[1], n[2]) in equ.(2.5) is augmented by one, and the process is repeated until n[1] becomes N.

5.2 The procedure CALC ARRAY

The integer procedure CALC ARRAY(i) performs the following operations

1. By means of the non-local integer ka (non-local with respect to CALC ARRAY), CALC ARRAY determines the Taylor index ka of the formula i. The number ka is assigned to the procedure identifier itself.

2.1 If i is a number or a simple term of the form x^p or y^p then a[ka,c[n[1], n[2]]] is calculated according to table 5.

2.2 If i is a simple term of the form U_k, $\frac{\partial U_k}{\partial x}$ or $\frac{\partial U_k}{\partial y}$ then it may be that a[ka,c[n[1], n[2]]] can be calculated by aid of the array elements u[k,l,j].
If however, the corresponding u[k,l,j] is not known, then the array element f[ka] is set equal to the index unknown [k] of the formula corresponding to the X_k in section 2.2.5.
Note that all f[j] are set beforehand equal to -1 in TAYLOR.
Therefore if f[j] ≥ 0, it is necessarily equal to the index of a formula.

2.3 If i is a compound formula say A * B, then CALC ARRAY determines firstly the Taylor indexes m1 and m2 of A and B resp.

2.3.1 We first treat the case that both f[m1] and f[m2] are negative, then a[m1,c[n[1], n[2]]] and a[m2,c[n[1], n[2]]] are already calculated and a[ka,c[n[1], n[2]]] can be calculated by the statement

$$a[ka,c[n[1], n[2]]] := a1 + b * a[m1,c[n[1], n[2]]] + c1 * a[m2,c[n[1], n[2]]] \qquad (5.1)$$

in which

$$a1 = \sum_{i=0}^{n[1]} \sum_{j=0}^{n[2]}{}' a[m1,c[i,j]] * a[m2,c[n[1]-i, n[2]-j]]$$

$$= IP(m1, m2, 0) \quad \text{(see section (5.4))}.$$

(The symbol $\sum\sum'$ is defined in table 6).

b = a[m2,0] = a[m2,c[0,0]]

and

c1 = a[m1,0] = a[m1,c[0,0]]

2.3.2 Secondly we treat the case that one or both f[m1] and f[m2] are non-negative, e.g. f[m1] ≥ 0 and f[m2] < 0.
Then a[m1,c[n[1], n[2]]] is not yet calculated, instead of it, a formula is stored with index f[m1].
In this case a[ka,c[n[1], n[2]]] can neither be calculated and we have to store a formula which has in ordinary notation the form (5.1).
The following statement stores this formula

```
f[ka] := S(NUMBER(a1 + c1 * a[m2,c[n[1], n[2]]]),
           P(NUMBER(b), f[m1]))
```

Note that the actual parameters of NUMBER are ordinary real numbers.

The other cases for which f[m2] ≥ 0 are treated similarly.

If i is of the form A+B or A/B then we proceed in the same way as for A * B.

2.4 Let i be a function, say Φ(A).
The Taylor index m1 of A is calculated, the integer kpsi is augmented by one (in TAYLOR, kpsi gets beforehand the value zero).
The index i and the Taylor index m1 are stored in the non-local array elements psi[kpsi, -1] and psi[kpsi, 0] resp.
The Taylor indexes of the functions Φ_1 (see equation (2.6)) are determined and stored in the non-local array elements phi[kpsi, j].
If f[m1] < 0, then a[m1,c[n[1], n[2]]] is already calculated and the a[phi[kpsi,j], c[n[1], n[2]]] can be calculated by the formula (2.9) or (2.10).
If f[m1] ≥ 0, the Taylor coefficients for the functions Φ_1 can not be calculated and formulae will be stored instead of them.
For another time use is made of the real procedure IP.

2.5 The case that i is a psi simple term (H[i,1] = -4) will be treated in section 5.8.

5.3 The procedure IP

The real procedure IP(lp, rp, diff) becomes equal to
if diff = 0 then

$$\sum_{i=0}^{n[1]} {\sum_{j=0}^{n[2]}}' a[lp,c[i,j]] \circ a[rp,c[n[1]-i,n[2]-j]]$$

if diff = 1 then

$$\sum_{i=1}^{n[1]} {\sum_{j=0}^{n[2]}}' a[lp,c[n[1]-i,n[2]-j]] \circ i \circ a[rp,c[i,j]]$$

if diff = 2 then

$$\sum_{i=0}^{n[1]} {\sum_{j=1}^{n[2]}}' a[lp,c[n[1]-i,n[2]-j]] \circ j \circ a[rp,c[i,j]]$$

5.4 The procedure CHANGE

The procedure CHANGE(i,j), changes the formula i into j.

5.5 The procedure FAC

The real procedure FAC(n) becomes equal to n!.
Use is made of the non-local integer kf and the non-local array fac[0 : 20]. kf and fac[0] get the values 0 and 1 beforehand.

5.6 The procedure CALCULATE

The procedure CALCULATE(n, i, unknown) calculates in a recursive way the n unknown quantities X_k, whose indexes are stored in the integer array unknown, from the n linear equations symbolically written as

$$"i[j]" = 0 \quad j = 1, \circ\circ\circ, n \qquad (5.2)$$

in which the i[j] are the indexes of the formulae in X_k.

The indexes unknown [j] are actually the indexes of numbers zero (not the zero element 0).

By means of the statement

for j := 1 step 1 until n do

c[j] := VALUE(DIFF(i[n], unknown[j]))

The real array elements c[j] get the values of the coefficients of the n unknown quantities X_k.
In the following statement use is made of the fact that the integers unknown [j] are indexes of numbers zero.

c[0] := -VALUE(i[n])

The equation (5.2) with index n can now be written as

$$\sum_{j=1}^{n} c[j]X_j = c[0]. \qquad (5.3)$$

If c[n] ≠ 0 then this equation may be solved for X_n, and we have

$$X_n = \frac{1}{c[n]} \{c[0] - \sum_{j=1}^{n-1} c[j]X_j\} \qquad (5.4)$$

The array element unknown [n], is now set equal to the index of a stored formula corresponding to the right-hand side of (5.4), and the number of equations and unknown's is reduced by 1.
By the procedure call (in the case n > 1)

CALCULATE(n-1, i, unknown)

the unknown [j] become indexes which correspond to numbers whose values are precisely the values of the solution vector X_j (j = 1, ..., n-1).
By means of the statement

CHANGE(unknown[n], NUMBER(VALUE(unknown[n]))),

the unknown [n] becomes equal to the index of the number whose value is the value of X_n.

If n = 1, the right-hand side of (5.4) was already a number, and the unknown [1] was already set equal to the index of the number whose value is X_1.

If c[n] = 0 then the formula with index i[n-1] is interchanged with the formula with index i[n]. And the process is repeated, if again c[n] turns out to be zero then i[n-2] is interchanged with i[n], etc., until a formula is found for which c[n] ≠ 0 or until the set of formulae is exhausted.
In the latter case the n equations (5.2) are singular.
The following sentence is then punched on the output paper tape: "system is insolvable".

We remark that a direct matrix inversion method would certainly lead to a faster calculation process of the unknown X_j.
The compact and charming procedure which results from using the procedures for doing algebra in the computer, urged us to publish it in this report and to use it in the program of section 6 which after all, we do not recommend for general use.

5.7 Filling up the open places

We now come to stage 4 of the calculation process of section 5.1.
By means of CALC ARRAY some Taylor coefficients a[j,c[n[1], n[2]]] are calculated, and some are not.
For the last ones, a formula is stored with index f[j].
The formulae with indexes unknown [k], (k = 1, ..., M), correspond now to the known numbers X_k of section (2.2.5).
We may therefore write down the statement

a[j,c[n[1], n[2]]] := VALUE(f[j])

and the uncalculated Taylor coefficient gets its value.
In the same way we can calculate the unknown Taylor coefficients u[1,i,j]. This is done with the auxiliary integer array elements tp[1,j] having the values

type[1]	tp[1,1]	tp[1,2]
0	0	0
1	1	0
2	0	1

table 12

We write

u[1,n[1] + tp[1,1], n[2] + tp[1,2]] :=
VALUE(unknown[1])/(1 + (if type[1]=0 then 0 else n[type[1]]))

5.8 Calculating the Taylor coefficients of Ψ

Recalling section 4.3 and section 5.2 point 2.4, we know that in all the differential equations (1.1) the number of times a function symbol occurs is kpsi.

Let us describe the calculation of the $\psi_{k,n[1],n[2]}$ for the function i = Φ(A) which occurred the j^{th} time.

In CALC ARRAY the index i of Φ(A) and the Taylor index m1 of A are stored in psi[j,-1] and psi[j,0] ; moreover in phi[j,p], with p = 1, ..., ν, the Taylor indexes of the functions $\Phi_p(A)$ are stored. (Note that ν = PSI[1,0] and 1 = H[i,1]). We now define the H[Xf,3] and H[PHI[p], 3] as to be equal to m1 and phi[j,p] respectively.

Looking now at the last line of CALC ARRAY which reads

if H[i,1] = -4 then CALC ARRAY := H[i,3]

it becomes obvious that CALC ARRAY interprets the psi simple term Xf as the formula A with Taylor index m1 and the psi simple terms PHI[p] as formulae with Taylor indexes phi[j,p].

Finally the Taylor indexes psi[j,p] and the Taylor coefficients a[psi[j,p], c[n[1], n[2]]] of the formulae $\psi_p(A, \Phi, \ldots, \Phi_\nu)$ are determined by the statement

for p := 1 step 1 until PSI[1,0] do psi[j,p] :=
CALC ARRAY(PSI[1,p])

6. The ALGOL 60 program

The test cases worked out by the program are chosen rather arbitrarily. The results are reproduced after the program.

The following problems were calculated:

1. $U_1 \frac{\partial U_1}{\partial y} + x = 0$

   with initial conditions $U_1(x,0) = 1 + x$.

   The solution of this problem is obviously

$$U_1 = \sqrt{(1+x)^2 - 2xy} = 1 + x - xy + x^2y - \tfrac{1}{2}x^2y^2 - x^3y + \tfrac{3}{2}x^3y^2 - \tfrac{1}{2}x^3y^3 + \ldots$$

2. $\ln U_1 + y = 0$

   from which we have

$$U_1 = \exp(-y) = 1 - y + \tfrac{1}{2}y^2 - \tfrac{1}{6}y^3 + \tfrac{1}{24}y^4 - \ldots$$

3. $\arctan\left(\frac{\sin U_1}{\cos U_1}\right) - \exp(x^2) = 0$

   thus

$$U_1 = \exp(x^2) = 1 + x^2 + \tfrac{1}{2}x^4 + \ldots$$

4. $\frac{\partial U_1}{\partial x} - U_2 = 0$

$$\frac{\partial U_2}{\partial x} + U_1 = 0$$

   with initial conditions $U_1(0) = 0$ and $U_2(0) = 1$

   it is easily seen that

$$U_1 = \sin(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + \ldots$$

   and

$$U_2 = \cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \ldots$$

5. $U_1 - \ln(xy) = 0$

$U_2 - \sqrt{xy} = 0$

Expansion in the point x=1 and y=1 gives

$U_1 = \ln(1 + (x-1)) \cdot (1 + (y-1)) =$

$(x-1) - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \ldots +$

$(y-1) - \frac{1}{2}(y-1)^2 + \frac{1}{3}(y-1)^3 - \ldots$

$U_2 = \sqrt{(1 + (x-1))(1 + (y-1))} =$

$\{1 + \frac{1}{2}(x-1) - \frac{1}{8}(x-1)^2 + \frac{1}{16}(x-1)^3 + \ldots\}\{1 + \frac{1}{2}(y-1) - \frac{1}{8}(y-1)^2 +$

$\frac{1}{16}(y-1)^3 + \ldots\} =$

$= 1 + \frac{1}{2}(x-1) + \frac{1}{2}(y-1) - \frac{1}{8}(x-1)^2 + \frac{1}{4}(x-1)(y-1) - \frac{1}{8}(y-1)^2 +$

$+ \frac{1}{16}(x-1)^3 - \frac{1}{16}(x-1)^2(y-1) - \frac{1}{16}(x-1)(y-1)^2 + \frac{1}{16}(y-1)^3 + \ldots$

6. $\frac{1}{U_1} + \frac{\cos x}{xy + 1} = 0$

$xy + 1 + U_2 \cos x = 0$

from which it follows that U_1 and U_2 are equal.

$U_1 = U_2 = -\frac{xy + 1}{\cos x} = -1 - \frac{1}{2}x^2 - xy - \frac{5}{24}x^4 - \frac{1}{2}x^3y + \ldots$

7. $\ln\left(\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial y} + 1\right) = \ln(xy + 1) = 0$

$\sqrt{\left(\frac{\partial U_1}{\partial x} - \frac{\partial U_2}{\partial y} + 1\right)} - \sqrt{xy + 1} = 0$

with initial conditions

$U_1(0,y) = e^y$ and $U_2(x,0) = e^x$

Evidently $\frac{\partial U_1}{\partial x} = xy$ thus $U_1 = \frac{1}{2} x^2 y + e^y$

and $\frac{\partial U_2}{\partial y} = 0$ thus $U_2 = e^x$.

6.1 Description of standard procedures

As input procedure we use the integer procedure XEEN(n).
XEEN can be given an integer value by means of the console.
The procedure stop, stops the calculation of the machine.
As output procedures we use the procedures:
PUTEXT1(st) , punching the string st on the output paper tape.
(Note that as string quotes the symbols ↓ and ↑ are used).
PUNLCR, punching a "new line carriage return" symbol.
PUSPACE(n), punching n space symbols.
FLOP(i,j,a), punching the real number a in floating point notation, i decimals of the mantissa and j decimals of the exponent.
ABSFIXP(i,j,a), punching the absolute value of the real number a in fixed point notation, i decimals before and j decimals behind the decimal point.

The real procedure SUM(i,n,m,ei) becomes equal to

$\sum_{i=n}^{m} ei$ if $m \geq n$, if $m < n$ then it becomes equal to zero.

For the integer division symbol $\div$, the symbol $\overset{\circ}{\div}$ is used.

Running the program on the Electrologica X1 computer of the Mathematical Centre lasted about half an hour.
The integers kmax and kcmax got the values 320 and 64 resp. with the aid of XEEN.

Note that the pages of the program are ordered in such a way that one can see every two consecutive pages in one view (except pages 6 and 7).

7. References

[1] R.P. van de Riet : Algebraic operations in ALGOL 60, a second order program.
Report T.W. 96, Mathematical Centre (1965).

[2] R. Courant, D. Hilbert: Methods of Mathematical Physics II.
Interscience publishers, New York, London (1962).

[3] A. Gibbons : A program for the automatic integration of differential equations, using the method of Taylor series.
The Computer Journal 3 (1960) pp 108-111.

[4] J.W. Backus et.al. : Revised report on the Algorithmic Language ALGOL 60.
Regnecentralen, Copenhagen (1962).

[5] R.D. Richtmyer : Power series solution, by machine, of a non-linear problem in two-dimensional fluid flow.
Annals of the New York Acadamy of Sciences.

[6] R.E. Moore : The automatic analysis and control of error in digital computation based on the use of interval numbers.
published in "Error in Digital Computation" Vol. 1, edited by Louis B. Rall.
John Wiley & Sons, Inc. New York, 1964.

[7] Arnold Lapidus and Max Goldstein: Some experiments in Algebraic Manipulation by Computer.
Communications of the A.C.M. 8 number 8, 1965.

[8] Alan J. Perlis, Renato Iturriagia and Thomas A. Standish:
A preliminary sketch of formula ALGOL.
Carnegie Institute of Technology, April 1965.