STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING TOEGEPASTE WISKUNDE

Technical Note TN 27

Three ALGOL - 60 programs

Complex Arithmetic,
Difference and Derivative procedures,
Determination of the point of Toricelli

by

R.P. van de Riet

November 1962

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.

In this note some peculiar ALGOL - 60 procedures are described. The first gives a method to build up arbitrary arithmetic expressions with complex variables.
The other describes a method for calculating forward differences and derivatives from a given set of function-values with equally spaced parameter-values. It makes use of strong recursive procedures.
We do not claim that this method is handsome for numerical use, on the contrary, as we observed with the X1 computer of the Mathematical Centre, it works of course, but wastes awfully much time. We find it however a very beautiful and instructive illustration of the capability of ALGOL - 60.

We now describe the first procedures.

```
procedure Complex Arithmetic (a, b, R, r); array a, b, R, r;
begin   integer i, k; array H [1:4, 1:2];
        integer procedure P (i, j); value i, j; integer i, j;
        begin real a; k:= k - 1; a:= H [i, 1] × H [j, 1] - H [i, 2] ×
              H [j, 2]; H [k, 2]:= H [i, 1] × H [j, 2] + H [i, 2] ×
              H [j, 1]; H [k ,1]:= a; P:= k
        end;
        integer procedure Q (i, j); value i, j; integer i, j;
        begin real a, b; k:= k - 1; b:= H [j, 1]/2 + H [j, 2]/2;
              a:= (H [i, 1] × H [j, 1] + H [i, 2] × H [j, 2])/b;
              H [k, 2]:= (H [i, 2] × H [j, 1] - H [i, 1] × H [j, 2])/b;
              H [k, 1]:= a; Q:= k
        end;
        integer procedure S (i, j); value i, j; integer i, j;
        begin k:= k - 1; H [k, 1]:= H [i, 1] + H [j, 1]; H [k, 2]:=
              H [i, 2] + H [j, 2]; S:= k
        end;
        integer procedure T (a); array a;
        begin k:= k + 1; H [k, 1]:= a [1]; H [k, 2]:= a [2]; T:= k end;
        integer procedure J (i, expi); integer i; real expi;
        begin k:= k + 1; i:= 1; H [k, 1]:= expi; i:= 2; H [k, 2]:= expi;
              J:= k
        end;
        procedure U (i, R); value i; integer i; array R;
        begin R [1]:= H [i, 1]; R [2]:= H [i, 2]; k:= 0 end;

        k:= 0;
        U (S (P (T (a))times:(T (a)))plus:(P (T (b))times:(T (b))), R);
        comment (   a    ×       a ) +      (   b    ×       b )=: R ;

        U (Q (S (T (a))plus:(P (J (i,1-1))times:(T (b))))divided by:
             (S (T (a))plus:(P (J (i,1-i))times:(T (b)))), r);
        comment (   a   +       (    i       ×       b )) /
                (   a   +       (   -i       ×       b ))=: r
end
```

This procedure assigns the value a/2 + b/2 to R and the value (a + ib)/(a - ib) to r, where a, b, R and r are complex numbers.
We focus attention to the value call of the parameters of the procedures, this is essential.
Furthermore we notice that the depth or height of the accumulator H is the maximum number of right-handed brackets placed one after another, not calculating the brackets which occur in parameter-delimiters.

The following program calculates the third derivative of exp (x) with a 6-point difference scheme, the error is of the order h↑3, where h is the steplength.

```
begin   real h; integer i, k; array A [1:50];
        real procedure SUM(i,h,k,ti); value k; integer i,k,h; real ti;
        begin real s; s:= 0; for i:= h step 1 until k do s:= s + ti;
              SUM:= s
        end;
        real procedure DELTA (N,k,k0,fk); value N,k0; real fk;
        integer N,k,k0;
        comment N is the order of the forward difference which is
        calculated from a set of function-values with equidistant
        parameter-values, in formula:

                         DELTA = Δ^N f (k0) ;
        begin   integer i;
                DELTA:= if N = 1
                then SUM (k,k0,k0+1,(-1)↑(k+1-k0) × fk)
                else DELTA (1,i,k0,DELTA (N-1,k,i,fk))
        end;
        real procedure DER (OR,N,h,k,k0,fk); value OR,N,h,k0; real fk,h;
        integer OR,N,k,k0;
        comment OR is the order of the derivative, calculated from a given
        set of function-values f(k), with equidistant parameter-values,
        the error is of the order h↑(N+1-OR), where h is the steplength,
        k0 is the point where the derivative is calculated, in formula:
```

$$DER := h\uparrow(-OR) \sum_{i_1,\ldots,i_{OR}=1}^{N+1-OR} (-\Delta)\uparrow(i_1+\ldots+i_{OR})/(i_1\times\ldots\times i_{OR}) \times f(k0);$$

```
        begin   integer i;
                DER:= if OR = 1
                then SUM (i,1,N,DELTA (i,k,k0,fk) × (-1)↑(i+1)/i)/h
                else DER (1,N+1-OR,h,i,k0,DER (OR-1,N-1,h,k,i,fk))
        end;

        for i:= 1 step 1 until 50 do A[i]:= exp (i/50);
        for i:= 1 step 1 until 25 do A[i]:= DER (3,6,1/50,k,i,A[k])
end
```

At the end of this program the third derivatives of exp (x) in the points i = 1/50, 2/50, ..., 25/50 are assigned to the array A[i].

Next follows a program which calculates from a given set of points (x [i], y[i]), i = 1, 2,...,n, the point (x0, y0) which has a minimal distance to all these points, called the point of Toricelli. Starting with some point (x0, y0) the program seeks automatically the best direction φ, where a better point will be found. Then (x0, y0) is replaced by (x0 + r cos(φ), y0 + r sin(φ)), where r is the steplength. Until the steplength r is smaller then some number eps the program will continue.

On a read-band the following numbers must be punched:
The number of points,
the first point (x0, y0)(in this sequence),
the points (x [i], y [i])(in this sequence),
the first steplength,
the number eps.

```
begin   real x0, y0, r, dx, dy, eps, S, C, RHO1, RHO2; integer i, k, n;
        n:= read; k:= 0;
        begin   array x1, x2, y1, y2, A, B [1:n];
                x0:= read; y0:= read; NLCR; PRINTTEXT (‹Het startpunt is:›);
                print(x0); print (y0); NLCR;
                PRINTTEXT (‹De hoekpunten zijn:›); NLCR; for i:= 1 step 1
                until n do
                begin x1 [i]:= read; print (x1 [i]); y1 [i]:= read; print
                      (y1 [i]); x1 [i]:= x1 [i] - x0; y1 [i]:= y1 [i] - y0;
                      A [i]:= sqrt (x1 [i]↑2 + y1 [i]↑2); NLCR
                end;
                r:= read; NLCR; PRINTTEXT (‹De eerste staplengte is:›);
                print (r); NLCR; eps:= read; print (eps); NLCR; PRINTTEXT
(‹       Het minimum punt is:    De afstand is:  Som sin is:     Som cos is:›);
                NLCR; RHO1:= SUM (i, 1, n, A [i]);
OPNIEUW: S:= SUM (i, 1, n, y1 [i]/A [i]); C:= SUM (i, 1, n, x1 [i]/
                A [i]); if S = 0 ∧ C = 0 then goto EIND;
                dx:= sqrt (S↑2 + C↑2); dy:= S × r/dx; dx:= C × r/dx;
                goto LOOP DOOR;
BEGIN:  dx:= .1 × dx; dy:= .1 × dy; r:= .1 × r;
LOOP DOOR: for i:= 1 step 1 until n do
                begin x2 [i]:= x1 [i] - dx; y2 [i]:= y1 [i] - dy; B [i]:=
                sqrt (x2 [i]↑2 + y2 [i]↑2) end;
                k:= k + 1; RHO2:= SUM (i, 1, n, B [i]); if r < eps then
                goto EIND; if RHO2 ≥ RHO1 then goto BEGIN else
                begin   for i:= 1 step 1 until n do
                        begin x1 [i]:= x2 [i]; y1 [i]:= y2 [i];
                              A [i]:= B [i]
                        end; x0:= x0 + dx; y0:= y0 + dy; RHO1:= RHO2;
                        print (k); print (x0); print (y0); print (RHO1);
                        print (S); print (C); NLCR; goto OPNIEUW
                end;
EIND:   end
end
```