Algorithms

Announcement

Objective

A new editorial department called "Algorithms" has been added to the *Communications*. This department was established to publish algorithms consisting of "procedures" and programs in the ALGOL language. Contributed items may take one of three possible forms: An ALGORITHM, CERTIFICATION of a previously published algorithm, REMARK on a previously published algorithm. The section will always be headed by a disclaimer and a statement permitting reproduction. Contributions should be sent to

J. H. WEGSTEIN
Computation Laboratory
National Bureau of Standards
Washington 25, D.C.

It should be noted that algorithms in ALGOL may continue to appear in any other department of the *Communications* when these altorithms occur naturally within an exposition of a subject oriented towards the activities of that department.

Algorithm

A contributed Algorithm should be in the following form:

SERIAL NUMBER* ALGORITHM NAME

AUTHOR'S NAME

AUTHOR'S EMPLOYING ORGANIZATION AND ADDRESS

ABSTRACT: This is a brief description of the algorithm, and must be less than 200 words in length. The purpose of the algorithm should be stated, giving accuracy, ranges, derivation or references, tabular space requirements, and any other information that might help a user. The status and method of checkout might also be given. This abstract should always be expressed as a **comment** in ALGOL notation.

ALGORITHM: The ALGOL language must be used (see Preliminary Report-International Algebraic Language *Communications, ACM* December, 1958). The author should use the *Publication* form of ALGOL and is requested to write the algorithms in a style patterned after the most recent algorithms appearing in this section of the *Communications*.

* Supplied by Editor.

Certification

The successful use of a published algorithm, either by machine or hand compilation, enhances its value. Therefore, contributions in the form of "Certifications" are also solicited by this department. It is anticipated that periodically, perhaps once a year, an index or catalog of algorithms along with references to all their certifications and remarks will be published. An algorithm in this "library" which includes several certifications assumes a certain stature.

A contributed Certification should be in the following form:

The word "CERTIFICATION".

IDENTIFICATION OF THE ALGORITHM BEING CERTIFIED. This consists of the year and month of the *Communications* and the serial number of the algorithm, its name, and author.

NAME, EMPLOYING ORGANIZATION AND ADDRESS OF THE CONTRIBUTOR OF THIS CERTIFICATION.

CERTIFICATION: This should be a brief statement of how the algorithm was used, the values of quantities used, and results obtained. Information such as the type of computer used and the compiler or compiling method used may also be helpful. A strong form of certification is a test routine or test value generator which tests each possible branch of the algorithm. This test, in ALGOL, may be included in the certification.

Remarks

It is anticipated that a published algorithm may contain an error, may require clarification, or may be modified by someone (e.g., its author). For these purposes, "remarks" are solicited.

A contributed Remark should be in the following form:

The word "REMARK".

IDENTIFICATION OF THE ALGORITHM BEING REMARKED UPON.

NAME, EMPLOYING ORGANIZATION AND ADDRESS OF CONTRIBUTOR OF THIS REMARK:

REMARK: This should be limited to 200 words or less, but, in addition, it might include sections in ALGOL notation intended for addition to or replacement of sections of the algorithm being referred to.

Algorithms

Contributions to this department must be in the form stated in the Algorithms Department policy statement (*Communications*, February, 1960). Contributions should be sent to J. H. Wegstein, Computation Laboratory, National Bureau of Standards, Washington 25, D. C. Algorithms should be in the Publication form of ALGOL and written in a style patterned after the most recent algorithms appearing in this department.

1. QuadI

R. J. Herbold
National Bureau of Standards, Washington 25, D. C.

```
comment     QuadI is useful when integration of several func-
            tions of same limits at same time using same
            point rule is desired. The interval (a,b) is di-
            vided into m equal subintervals for an n-point
            quadrature integration. p is the number of func-
            tions to be integrated. w_k and u_k are normalized
            weights and abscissas respectively, where
            k=1,2,3,···,n. u_k must be in ascending order.
            P(B,j) =: (c) is a procedure which must be sup-
            plied by the programmer. It evaluates (c) the
            function (as indicated by j) for B. I_j is the result
            of integration for function j.;
procedure   QuadI (a,b,m,n,p,w_k,u_k,P(B,j) =: (c)) =: (Ij)
            begin
QuadI:              h := (b-a)/m
            for     j := 1(1)p  ;  I_j := 0
                    A := a-h/2
            for     i := 1(1)m
L1          begin   A := A+h
            for     k := 1(1)n
L2          begin   B := A+(h/2)×u_k
            for     j := 1(1)p
L3:         begin   P(B,j) =: (c)
                    I_j := I_j+w_k×c      end L3  ;  end L2
                      end L1
            for     j := 1(1)p
                    I_j := (h/2)×I_j
            return
            integer (j,k,i)
            end     QuadI
```

2. Rootfinder

J. Wegstein
National Bureau of Standards, Washington 25, I

```
comment     This procedure computes a value of g=x
            fying the equation x=f(x). The procedure c
            statement gives the function, an initial ap
            mation a≠0 to the root, and a tole
            paramater ε for determining the number c
            nificant figures in the solution. This accele
            iteration or secant method is described b
            author in Communications, June, 1958.;
procedure   Root(f( ), a, ε) =: (g)
begin
Root        b := a  ;  c := f(b)  ;  g := c
            if (c=a)  ;  return
            d := a  ;  b := c  ;  e := c
Hob:        c := f(b)
            g := (d×c-b×e)/(c-e-b+d)
            if (abs((g-b)/g)≦ε)  ;  return
            e := c  ;  d := b  ;  b := g  ;  go to Hob
end
```

3. Solution of Polynomial Equation by Bairs Hitchcock Method

A. A. Grau
Oak Ridge National Laboratory, Oak Ridge, Tenn

```
procedure
BAIRSTOW    (n, a[ ], eps0, eps1, eps2, eps3, K) =:
            (m, x[ ], y[ ], nat[ ], ex[ ]);
comment     The Bairstow-Hitchcock iteration is used t
            successively pairs of roots of a polyr
            equation of degree n with coefficien
            (i = 0, 1, ···, n) where a_n is the constant ter
            exit from the procedure, m is the number o
            of roots found, x[i] and y[i]  (i = 1, ···, m
            pair of real roots if nat[i]=1, the real and
            nary parts of a complex pair if nat[i]= —
            ex[i] indicates which of the following conc
```

was met to exit from the iteration loop in finding this pair:

1. Remainders, r1, r0, become absolutely less than eps1.
2. Corrections, incrp, incrq, become absolutely less than eps2.
3. The ratios, incrp/p, incrq/q, become absolutely less than eps3.
4. The number of iterations becomes K.

In the last case, the pair of roots found is not reliable and no further effort to find additional roots is made. The quantity eps0 is used as a lower bound for the denominator in the expressions from which incrp and incrq are found.;

```
begin
integer    (i, j, k, n1, n2, m1)   ;
array      (b, c[0 : n+1])   ;
BAIRSTOW   for i := 0(1)n   ;   b_i := a_i
           b_{n+1} := 0   ;   n2 := entire((n+1)/2)
             n1 := 2×n2
           for m1 := 1(1)n2   ;   begin p := 0   ;   q := 0
           for k := 1(1)K   ;   begin
step:      for i := 0(1)n1   ;   c_i := b_i
           for j := n1−2, n1−4   ;   begin
           for i := 0(1)j   ;   begin
           c_{i+1} := c_{i+1} − p × c_i
           c_{i+2} := c_{i+2} − q × c_i end end
           r0 := c_{n1}   ;   r1 := c_{n1−1}
           s0 := c_{n1−2}   ;   s1 := c_{n1−3}
           v0 := −q × s1   ;   v1 := s0 − s1 × p
           det0 := v1 × s0 − v0 × s1
           if (abs(det0)<eps0)   ;   begin
           p := p+1   ;   q := q+1   ;   go to step end
           det1 := s0 × r1 − s1 × r0
           det2 := r0 × v1 − v0 × s1
           incrp := det1/det0   ;   incrq := det2/det0
           p := p + incrp   ;   q := q + incrq
           if (abs (r0) < eps1)   ;   begin
           if (abs (r1) < eps1)   ;   begin
           ex_{m1} := 1   ;   go to next end end
           if (abs (incrp) < eps2)   ;   begin
           if (abs (incrq) < eps2)   ;   begin
           ex_{m1} := 2   ;   go to next end end
           if (abs (incrp/p) < eps3)   ;   begin
           if (abs (incrq/q) < eps3)   ;   begin
           ex_{m1} := 3   ;   go to next end end end
           ex_{m1} := 4
next:      S := p/2   ;   T := S² − q
           if (T ≧ 0)   ;   begin T := sqrt (T)
           nat_{m1} := 1   ;   x_{m1} := S + T
             y_{m1} := S − T end
           if (T < 0)   ;   begin nat_{m1} := −1   ;   x_{m1} := S
             y_{m1} := sqrt(−T) end
           if (ex_{m1} := 4)   ;   go to out
           for j := 0(1) (n1−2)   ;   begin
           b_{j+1} := b_{j+1} − p × b_j
           b_{j+2} := b_{j+2} − q × b_j   ;   end
           n1 := n1 − 2   ;   if   (n1 < 1)
out:       begin m := m1   ;   return end
           if (n1 < 3)   ;   begin
           m1 := m1 + 1   ;   ex_{m1} := 1
           p := b_1/b_0   ;   q := b_2/b_0
           go to next end
           end end
```

Corrigendum

In the paper "A Technique for Handling Macro Instructions" by Irwin D. Greenwald, *Communications ACM 2*, No. 11 (Nov. 1959):

(1) Use the term SCAT (instead of SOS) in referring to the SHARE assembler. SOS is too comprehensive a term. (p. 21, 2d par.)

(2) Label the example following (2) as example (2a), which reads:

ALPHA   M1   Q + 20, GAM, RHO, Z, OP6   (2a)

(p. 21, 2d col., line 12).

(3) Then on p. 22, 1st col., 10th line from bottom, read: "Thus, for example (2a), . . .".