ALGORITHM 16
CROUT WITH PIVOTING
GEORGE E. FORSYTHE
Stanford University, Stanford, California

real procedure INNERPRODUCT(u, v) index : (k) start : (s)
  finish : (f);
value s, f; integer k, s, f; real u, v;
comment INNERPRODUCT forms the sum of u(k) × v(k) for k = s, s+1, . . . , f.
  If s > f, the value of INNERPRODUCT is zero. The substitution of
  a very accurate inner product procedure would make CROUT more
  accurate;

begin
  real h;
  h := 0; for k := s step 1 until f do h := h + u × v;
  INNERPRODUCT := h
end INNERPRODUCT;

procedure CROUT (A, b, n, y, pivot, INNERPRODUCT);
value n; array A, b, x, pivot; integer n, pivot;
real procedure INNERPRODUCT;
comment This is Cour's method with row interchanges, as
  formulated in reference [1], for solving Ax = b
  and transforming the augmented matrix [A b]
  into its triangular decomposition L U with all
  L[k, k] = 1. If A is singular we exit to 'singular,' a
  non-local label. pivot[k] becomes the current
  row index of the pivot element in the k-th
  column. Thus enough information is preserved
  for the procedure SOLVE to process a new
  right-hand side without repeating CROUT.
  The accuracy obtainable from CROUT would
  be much increased by calling CROUT with a
  more accurate inner product procedure than
  INNERPRODUCT;

begin
  integer k, i, j, imax, p; real TEMP, quot;
  for k := 1 step 1 until n do
    begin
      TEMP := 0;
      for i := k step 1 until n do
        begin
            p, 1, k - 1);
          if abs(A[i, k]) > TEMP then
            begin
              TEMP := abs(A[i, k]); imax := i
              end 3;
            end 2;
          pivot[k] := imax;
        comment We have found that A[imax, k] is the largest
          pivot in column k. Now we interexchange rows k and imax
          if imax ≠ k then
          if imax ≠ k then
            begin
              end 5;
            TEMP := b[k]; b[k] := b[imax]; b[imax] := TEMP
          end 4;
        comment The row interchange is done. We proceed to the
          elimination;
          if A[k, k] = 0 then go to singular;
          for i := k+1 step 1 until n do
            begin
              quot := 1.0/A[k, k]; A[i, k] := quot × A[i, k]
              end;
            for j := k+1 step 1 until n do
                A[p, j], p, 1, k - 1);
              b[k] := b[k] - INNERPRODUCT(A[k, p], b[p], p, 1, k - 1)
            end 1;
        comment The triangular decomposition is now finished,
          and we do the back substitution;
          for k := n step -1 until 1 do
            begin
              y[k] := (b[k] - INNERPRODUCT(A[k, p], y[p], p, k + 1, n))/A[k, k]
              end 4 CROUT;
          procedure SOLVE (B, c, n, z, pivot, INNERPRODUCT);
value n; array B, c, z, pivot; integer n, pivot;
real procedure INNERPRODUCT;
comment SOLVE assumes that a matrix A has already been
  transformed into B by CROUT, but that a new
  column c has not been processed. SOLVE solves the
  system Az = c, and the output z of SOLVE is
  precisely the same as the output y of the procedure
  statement CROUT (A, c, n, y, pivot, INNER-
  PRODUCT). However, SOLVE is faster, because it
does not repeat the triangularization of A;

begin
  integer k; real TEMP;
  for k := 1 step 1 until n do
    begin
      TEMP := c[pivot[k]]; c[pivot[k]] := c[k]; c[k] :=
        TEMP; c[k] := c[k] - INNERPRODUCT(B[k, p],
          c[p], p, 1, k - 1);
      end;
    for k := n step -1 until 1 do
      begin
        z[k] := (c[k] - INNERPRODUCT(B[k, p], z[p], p, k + 1, n))/B[k, k]
        end SOLVE

REFERENCE
  43–100 of John W. Carr III (editor), Application of Advanced
  Numerical Analysis to Digital Computers, (Lectures given at
  the University of Michigan, Summer 1968, College of
  Engineering, Engineering Summer Conferences, Ann Arbor,
  Michigan [1969]).
REMARK ON ALGORITHM 16
CROUT WITH PIVOTING (G. Forsythe, Communications ACM, September, 1960)
GEORGE E. FORSYTHE
Stanford University, Stanford, California

QUERY
Perhaps the most basic procedure for an ALGOL library of matrix programs is an inner product procedure. The procedure Innerproduct given on page 311 of [1] is fairly difficult to comprehend, and probably poses great difficulties for most translating routines. I merely copied its form in writing a modified inner product routine for [2]. My query is: How should one write an inner product procedure in ALGOL?

REFERENCES

REMARK ON ALGORITHM 16
CROUT WITH PIVOTING (G. E. Forsythe, Comm. ACM, 3 (Sept. 1960), 507-8.)
HENRY C. THACHER, JR.,* Argonne National Laboratory, Argonne, Illinois

This procedure contains the following errors:
a. In SOLVE, the expression
\[ c[k] := c[k] - \text{INNERPRODUCT} \]
\[ (B[k, p], c[p], p \neq 1, k - 1) \]
should read:
\[ c[k] := c[k] - \text{INNERPRODUCT} \]
\[ (B[k, p], c[p], p \neq 1, k - 1) \]
b. In CROUT, the specification part should read:
\text{array } A, b, y ; \text{ integer } n ; \text{ integer array } pivot ;
c. In SOLVE, the specification part should read:
\text{array } B, c, z ; \text{ integer } n ; \text{ integer array } pivot ;
The efficiency of the algorithm will be improved by the following changes:
a. In the elimination phase of CROUT, replace
\begin{verbatim}
for i := k + 1 step 1 until n do
begin
   quote := 1.0/A[i, k] ;
   A[i, k] := quot X A[i, k]
end ;
\end{verbatim}
by
\begin{verbatim}
for i := k + 1 step 1 until n do
   quote := 1.0/A[i, k] ;
   for i := k + 1 step 1 until n do
\end{verbatim}
b. Omit INNERPRODUCT from the formal parameter list in both CROUT and SOLVE, and declare INNERPRODUCT either locally, or globally. This avoids any reference to INNERPRODUCT in the calling sequence produced by a compiler.
It is also to be noted that a minor modification of CROUT allows it to be used to evaluate the determinant of \( A \).
All of these suggestions are included in a later algorithm.

* Work supported by the U. S. Atomic Energy Commission.