Algorithms

J. G. HERRIOT, Editor

ALGORITHM 287
MATRIX TRIANGULATION WITH INTEGER ARITHMETIC [F1]

W. A. BLANKINSHIP

(Recd. 19 May 1965 and 17 Sept. 1965)

National Security Agency, Ft. Geo. G. Meade, Md.

```
integer procedure INTRANK (mat, m, n, e); value m, n, e;
  integer m, n, e; integer array mat;
comment This procedure operates on an m by n+e matrix whose
  name is mat and whose elements are integers. If mat is considered
  as composed of two submatrices U and V, where U comprises
  the first n columns of mat and V comprises the last e columns,
  then the effect of the procedure is as follows:
    (1) The rank of the submatrix U is returned as the value of
  INTRANK (designated by r in the following discussion).
    (2) mat is transformed by a sequence of elementary row opera-
  tions in such a manner that U is reduced to triangular form.
  Triangular form means that the leading, or first nonzero, ele-
  ment of each row appears to the right of the leading element
  of the preceding row.
    (3) It is easy to deduce from the proof in [1, p. 72, Th. 12]
  that for any set of k columns of mat, the greatest common divisor
  of all kth order minors selected from those columns is preserved.
  In particular, the product of all leading elements in U (final)
  (which are preserved as the first r elements of the local array a)
  will be equal to the gcd of all nth order minors of U.
    (4) It is also easy to show, by the methods of [2] that if mat
  contains an m × m identity matrix, I, then I ends up as a record
  of the row operations actually performed, specifically:

        mat (final) = I (final) × mat (initial)

    (5) Since (3) implies that the rank of U is preserved, and the
  rank of U (final) is obviously equal to the number of nonzero
  rows that it contains, this number, r, is returned as the value of
  INTRANK.
    (6) Under the conditions of (4), it follows that the last
  m−r rows of I (final) comprise a complete, linearly independent
  set of left-annihilators (row-dependences) of the matrix U.
    The preceding properties are the basis of the claims for the
  procedure SOLVEINTEGER [Algorithm 288, Comm. ACM 9
  (July 1966), 514] which calls this procedure.
    INTRANK is designed to minimize the likelihood of overflow,
  the detection of which is left to the user. The best method is to
  include an identity matrix in mat and check the relation de-
  scribed in 4 (above). In many instances overflow doesn't matter.
  In particular, if (a) the machine-compiler combination does
  integer addition, subtraction and multiplication modulo 2i+1
  where i is the maximum integer representable in the machine,
  (b) division is done by the usual long-division algorithm, and
  (c) the answers sought are either known to be less than i in
  absolute value, or only desired modulo 2i+1, then, short of
  interference by an over-zealous monitor, the procedure will
  produce satisfactory results in spite of overflow. (Although the
  CDC 1604 does not satisfy (a), the same effect can be achieved
  by using a suitable subroutine in place of the multiplication
  sign in the procedure REDUCE.)
    Overflow is generally dependent upon the magnitude of the
  greatest common divisor of all r × r minors contained in U, as
  this number, or a large divisor of it will appear in the rth row
  of mat (final) and as a[r]. Thus if U is a square matrix whose
  determinant is a prime greater than the capacity of the machine,
  there is obviously no way to avoid overflow. Even if the deter-
  minant is composite, it is most likely that only small factors
  will be left on the diagonal and overflow will still occur. When
  elements of U are chosen from a flat-random population of
  integers in the closed interval [−13, +13] it has been found
  empirically that overflow almost never occurs for m=n=11
  when run on the CDC 1604 where i = 2^46−1. See also the dis-
  cussions on overflow in the procedure SOLVEINTEGER;
begin integer i, j, k, Q, T, topel, nextel, itop, inext;
  integer array a [1:m];
  procedure FINDNEXT;
  begin nextel := 0;
    for k := i step 1 until m do
      if a[k]>nextel∧k ≠ itop then
      begin nextel := a[k] ; inext := k
      end
  end;
  procedure SWAPROWS;
  begin for k := j step 1 until T do
    begin Q := − mat[i,k];
      mat [i,k] := mat [itop, k];
      mat [itop, k] := Q
    end;
    a[i] := a [itop];
    comment The last statement is a luxury which ensures that,
      at the end of the algorithm, a will contain the leading ele-
      ments of the first INTRANK rows of mat;
  end;
  procedure REDUCE;
  begin Q := mat [itop,j] ÷ mat [inext, j];
    for k := j step 1 until T do
      mat [itop,k] := mat [itop,k] −Q × mat [inext,k];
    a [itop] := if mat [itop,j] < 0 then − mat [itop,j] else
      mat [itop,j];
  end;
  i := j:= itop := 0; T := n+e;
NEXTROW: if itop ≠ i then SWAPROWS;
  i := i+1; if i > m then go to OUT;
NEXTCOL: j := j+1; if j > n then go to OUT;
  for k := i step 1 until m do
    a[k] := if mat [k,j] < 0 then − mat [k,j] else mat [k,j];
  comment Find the value and location of the largest element at
    or below position (i,j) of mat.;
  itop := i−1; FINDNEXT;
  if nextel = 0 then go to NEXTCOL;
CONTINUE: itop := inext; topel := nextel;
  comment Find the value and location of the next largest
    element at or below position (i,j);
  FINDNEXT;
  if nextel = 0 then go to NEXTROW;
  comment Subtract row containing next highest element from
    that containing highest element. Repeat until highest element
    no longer ranks highest;
  REDUCE;
  go to CONTINUE;
OUT: INTRANK := i−1;
end
```

REFERENCES:

1. ALBERT, A. A. *Fundamental Concepts of Higher Algebra*. U. of Chicago Press., Chicago, Ill., 1956.
2. BLANKINSHIP, W. A. A new version of the Euclidean algorithm. *Amer. Math. Month. 70* (1963), 742–745.

ALGORITHM 288
SOLUTION OF SIMULTANEOUS LINEAR DIOPHANTINE EQUATIONS [F4]

W. A. Blankinship
(Recd. 19 May 1965 and 17 Sept. 1965)
National Security Agency, Ft. Geo. G. Meade, Md.

```
Boolean procedure SOLVEINTEGER (A) times: (x) equals the vector: (b) times a least integer: (d) where A is a matrix of
  dimension one to: (m) by one to: (n) Also find: (k) linearly independent auxiliary solutions and store in the matrix: (Y);
  value m, n;
  integer m, n, d, k;
  integer array A, x, b, y;
comment Seeks the smallest positive integer, d, for which an integer solution to the equation Ax = bd exists.
    If no solution exists then SOLVEINTEGER is returned as false. Otherwise SOLVEINTEGER is returned as true and the values of d and the solution vector x are returned.
    If more than one solution exists then auxiliary solutions are returned in the matrix Y. The additional solutions are obtained by adding any linear combination of the first k rows of Y to the solution x.
    It is assumed that
                A is dimensioned [1:m,1:n],
                x is dimensioned [1:n],
                b is dimensioned [1:m],
                Y is dimensioned [1:n,1:n].
    Note that a diophantine solution exists if and only if d is returned as 1 and SOLVEINTEGER is returned as true.
    The procedure relies entirely on the action of the procedure INTRANK [Algorithm 287, Comm. ACM 9 (July 1966), 513]. In particular, a matrix, mat, is formed by adjoining -b to the transpose of A, and then adjoining an (n + 1)th order identity matrix as follows:
                mat = ( -b   )
                      ( A^T  I )
    INTRANK is then called upon to triangularize the first m+1 columns of mat (reaching into the first column of I). The value of INTRANK will be returned as an integer r which is 1 greater than the rank of A. Furthermore, as a consequence of properties (4) and (6) claimed under INTRANK, the last n-r+1 rows of I (final) will comprise a complete set of left annihilators of the matrix ( -b  ). Since only the first of these rows (if any) will have
                ( A^T )
    a nonzero element in the first column, it follows that this first row expresses the value d and the desired solution (if d ≠ 0), and the succeeding n-r rows constitute solutions to the homogeneous equation. If any linear combination of these last n-r+1 rows were to yield a vector whose elements have a greatest common divisor not equal to 1, this would imply that det (I (final)) = det (I (initial)) ≠ 1, which is false. This ensures that d is the smallest value, as claimed.
    Overflow cannot occur in this procedure except as inherited from the procedure INTRANK. Overflow seems to be no problem when solutions (x,d) exist which are within the machine's capacity to verify. I am unable to fully explain this but numerous cases have been run on the CDC-1604 (47-bit integers plus sign bit) with elements of A chosen randomly between -13 and +13 inclusive and for m=n=5 through 20 (10 or more cases each). Only a single failure (in the case m=n=20) occurred. These cases were devised by preassigning integer values to x, calculating b and then calling SOLVEINTEGER. It is difficult to devise significant test cases where det(A) ≠ d ≫ 1 as this involves assigning values of x satisfying Ax=0 (mod d). This implies d must be a divisor of det (A) which must therefore be precalculated. But det (A) may overflow even though there may be a d for which solution is possible. When m=n the values of x and d will usually be, according to Cramer's rule, nth order determinants, or high divisors thereof, which may exceed machine capacity. When the elements of both b and A are chosen equiprobably between -α and +α, inclusive, it can be shown that the standard deviation of such a determinant is (n!α^n (α+1)^n/3)^½. Since this is an upper bound for the expected absolute value of such a determinant, it may be used as a rule of thumb to predict overflow. If α=13, then for n=11 this value is 10^13.6 and for n=12 it is 10^15.0. 1604 capacity is 10^14.1. In test cases, the procedure invariably succeeded for n=11 and invariably failed for n=12. (Remember, we are referring to cases where b is chosen randomly so that an integer solution will hardly ever exist.)
    Note that if m=1, this algorithm solves the gcd problem in much the same way as Algorithm 237 [J. E. L. Peck, Comm. ACM 8 (Aug. 1964), 481];
begin integer i, j, rank, s;
  integer array mat [1 : n+1, 1 : m+n+1];
  for j := 1 step 1 until m do
  begin mat [1,j] := -b [j];
    for i := 1 step 1 until n do mat [i+1, j] := A [j,i]
  end;
  for j := 1 step 1 until n+1 do
  for i := 1 step 1 until n+1 do
    mat [i, j+m] := if i = j then 1 else 0;
  rank := INTRANK (mat, n+1, m+1, n);
  d := mat [rank, m+1];
  if d = 0 then begin SOLVEINTEGER := false; go to OUT
                end;
  for i := rank step 1 until m do
    if mat [rank, i] ≠ 0 then
    begin SOLVEINTEGER := false; go to OUT
    end;
SOLVEINTEGER := true;
  s := if d < 0 then -1 else 1; d := s × d;
  k := n - rank + 1;
  for i := 1 step 1 until n do
  begin x[i] := mat [rank, m+i+1] × s;
    for j := 1 step 1 until k do
      Y [j,i] := mat [rank+j, m+i+1]
  end;
OUT:
end of procedure SOLVEINTEGER
```

ALGORITHM 289
CONFIDENCE INTERVAL FOR A RATIO [G1]

I. D. Hill and M. C. Pike (Recd. 8 Oct. 1965)
Statistical Research Unit, Medical Research Council, London, England

```
procedure Fieller (y, x, Vyy, Vxy, Vxx, t, r1, r2, inclusive);
  value y, x, Vyy, Vxy, Vxx, t;
  real y, x, Vyy, Vxy, Vxx, t, r1, r2;
  Boolean inclusive;
comment This procedure finds the (1-2×α) confidence limits for θ/φ where y and x are estimates of θ and φ respectively, subject to random errors 'normally' distributed with zero means, variance estimates Vyy and Vxx, and covariance estimate Vxy, each based on f degrees of freedom, and t is the upper (100×α) percent point of the t distribution on f degrees of freedom.
    At exit, if inclusive is true then the confidence interval includes all values such that r1 ≦ value ≦ r2. Otherwise the
```

Continued on page 518.