Raabo.

A/S REGNECENTRALEN

SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

SYSTEM LIBRARY

RCSL NO: 53-M5

TYPE : Algol 5 procedure

AUTHOR : P. Mondrup

EDITION: November 1969 (E)

RC 4000 SOFTWARE

MATHEMATICAL PROCEDURE LIBRARY

invertsym

ABSTRACT:

This boolean procedure inverts a symmetrical matrix. Only the lower half of the matrix has to be stored. The procedure will give a result even if the matrix is singular.

INFORMATION DEPARTMENT

DK-2500 VALBY · BJERREGAARDSVEJ 5 · PHONE: (01) 46 08 88 · TELEX: 64 64 rcinf dk · CABLES: INFOCENTRALEN

boolean procedure invertsym(n, A);

1. Function and parameters.

```
boolean procedure invertsym(n, A);
value n integer n; array A;
```

Function

The procedure inverts a symmetrical $n \times n$ matrix M(1:n, 1:n) of which the lower part is stored as a one-dimensional array A(1:n×(n+1)//2) so that

$$M(r, s) = M(s, r) = A(r\times(r-1)//2 + s) \quad \text{for} \quad 1 \leq s \leq r \leq n.$$

On return the inverse of M is found stored in A and the procedure is given the value true. This is only in case the call of the procedure has been a success. If it is a failure (i.e. if M is singular) the procedure has the value false, but even in this case the result M' found in A is with meaning, since M' will have the property that M' × B is a solution of the matrix equation M × X = B whenever this equation has a solution. Moreover, the degenerate elements may be found as those diagonal elements for which the corresponding rows and columns are identically zero.

Parameters

call parameter:

n	integer. The order of M

call and return parameter:

A(1:n×(n+1)//2)	array. Must on entry contain the lower half of M, so that M(r, s) = M(s, r) = A(r×(r+1)//2+S). At return A will contain the inverse of M stored in the same way

return parameter:

invertsym	boolean procedure. It is false if M is singular else true.

2. Mathematical Method.

The method is by Gauss-Jordan elimination using pivoting n times. In each step there are 3 cases.

Case 1: There is an index r, which has not been used as pivot index in an earlier step and for which the diagonal element M(r, r) is $\neq 0$. Let E be the set of all such indices. A new pivot index is selected from E in the following way: For each r in E the quantity

m(r) = max abs M(r, s)/abs M(r, r)   r in E

(maximum over s in E, s $\neq$ r)

is computed, and the pivot index r is chosen arbitrarily among those indices which make m(r) attain its minimum. A pivoting is carried out with M(r, r) as pivot element, and in a boolean array B(1:n) the r'th element is set to false to indicate that this index cannot be used in later steps.

The pivoting means that the elements M(i, k) are replaced by

$$\begin{array}{ll} M(i,k) - M(i,r)\times M(r,i)/M(r,r) & \text{for } i \neq r \wedge k \neq r \\ M(r,k)/M(r,r) & \text{for } i \neq r \wedge k = r \\ -M(i,r)/M(r,r) & \text{for } i = r \wedge k \neq r \\ 1/M(r,r) & \text{for } i = r \wedge k = r \end{array}$$

The result of this transformation is not a symmetrical matrix but

$$M(r,s) = \begin{cases} -M(s,r) & \text{if r has been pivot index, and s has not} \\ & \text{(i.e. } B(r) = \text{false, } B(s) = \text{true)} \\ M(s,r) & \text{in all other cases.} \end{cases}$$

Only the lower part of M is stored in A, since the upper part may be reestablished by means of B.

Case 2: M(r, r) = 0 for all r not used as pivot indices before, but there are elements M(r, s) $\neq$ 0 outside the diagonal (i.e. for r $\neq$ s) for some r and s not used as pivot indices before. In this case two new pivot indices r and s have to be chosen. First s is chosen arbitrarily among such possible indices. Next to choose r, let E be the set of the indices r $\neq$ s not used before as pivot indices and for which M(r, s) $\neq$ 0. For each r in E the quantity

m(r) = max abs M(r, k)/abs M(r, s)

where k runs over all indices $\neq$ r and $\neq$ s not used as pivot indices. Now r is chosen such that m(r) attains its minimum (which possibly is zero). In the boolean array B the r'th and s'th element are set to false to indicate that these indices may not be used in the following steps. Now a pivoting is carried out with M(r, s) and M(s, r) as pivot elements. This means that the matrix elements M(i, k) are replaced by

$$
\begin{array}{ll}
M(i, k)-M(i, r)\times M(s, k)/M(r, s)-M(i, s)\times M(r, k)/M(r, s) & \text{for } i\neq r\wedge i\neq s\wedge k\neq r\wedge k\neq s \\
M(i, r)/M(r, s) & \text{for } i\neq r\wedge k=s \\
-M(r, k)/M(r, s) & \text{for } i=s\wedge k\neq r \\
M(i, s)/M(r, s) & \text{for } k=r\wedge i\neq s \\
-M(s, k)/M(r, s) & \text{for } i=r\wedge k\neq s \\
1/M(r, r) & \text{for } i=r\wedge k=s
\end{array}
$$

As in case 1 the result is not a symmetrical matrix, but the upper part may be reestablished in the same manner from the lower part.
Case 3: There are no matrix elements $M(r, s) \neq 0$, where r and s have not been pivot indices. In this case the submatrix of M obtained by taking only the indices not used as pivot indices is identical zero. This means that M is singular. The value of the procedure is then set to false and the remaining rows and columns are set to zero, so that the result delivered in A may have the property mentioned in the section above.

If it is possible to do the pivoting n times without ever entering case 3 then M is nonsingular. So the value of the procedure is set to true, and the result of the algorithm delivered in A is the inverse of M.

3. Accuracy, time and storage requirement

Accuracy

In practice the relative error measured as $||A\times X - B||/||X||$ has been found to be about $_{10}$-10. This is not an exact error bound. Theoretical error bounds are discussed in detail in literature, see e.g. Forsythe and Moler (ref).

Time: .14×(n+1)××3 mS

Storage requirement

Program length: 6 segments

variables: 23 + 2.5×n words in stack.

Typographical length: 145 lines, 6 segments.

4. Test and discussion

The procedure is intended for use in such cases where the total matrix M is too big for the available store. A program using decompose and solve will be faster than a program using invert_sym even if the program must generate the matrix M from the half matrix A.

The procedure has been tested by some random matrices and by a representative set of singular matrices.

The following program will read n, A and write out the inverse of A:

```
Program to read a symmetrical matrix and output its inverse.
begin integer n, i, j, k, l;
  read(in, n);
  begin array A(1:(n×(n+1)) shift (-1));
    read(in, A);
    if ¬, invertsym(n, A) then write(out, <:<10> A is singular:>);
    write(out, <:<10>:>);
    for i:= 1 step 5 until n do
    begin
      j:= if i + 4 < n then i + 4 else n;
      for k:= i step 1 until j do write(out, <<__________ddd>, k);
      for k:= 1 step 1 until n do
      begin
        write(out, <:<10>:>, <<ddd>, k);
        j:= if i + 4 < k then i + 4 else k;
        for l:= i step 1 until j do
          write(out, <<_-d.dddddd₁₀-dd>, A((k×(k-1)) shift (-1) + l));
      end k;
      write(out, <:<12><10>:>)
    end i
  end A
end program;
```

5. Reference

Georg Forsythe and Cleve B. Moler: Computer solution of Linear Algebraic Systems, Prentice-Hall, Inc. (1967).

6. Algorithm

```
invertsym = set 6
invertsym = algol
external

boolean procedure invert_sym(n,A);
message invert sym, 13 11 69, RCSL 53-M5;
      value n; integer n; array A;
begin integer i,j,k,r,s,t,r1,s1,p;
      real m, aj,ak,ar,aj1,mp;
      boolean bj,mf;
      array M(1:n); boolean array B(1:n);

  i:=0;
  for p:= 1 step 1 until n do
  begin
    m:=0;
    for k:=p-1 step -1 until 1 do
    begin
      if abs A(i+k)> m then m:= abs A(i+k);
      if abs A(i+k)>M(k) then M(k):=abs A(i+k)
    end k;
    M(p):= m; B(p):= true; i:=i+p
  end p;
  t:=n; mp:=-1; mf:=true;
  for j:=n step -1 until 1 do
  begin
    if mf then
    begin
```

```
        if abs A(i)>M(j)×mp then
        begin
          if M(j)=0 then mf:=false else mp:=abs A(i)/M(j); p:=j
        end abs A(i)>M(j)×mp
      end mf;
      M(j):=0;  i:=i-j
    end j;
  next_pivot:
    s:=p; r:=(s×(s-1))shift(-1);
    if mp>0 | -,mf then
    begin comment this is the normal case where
          there has been found a pivot-element
          in the diagonal;
      B(s):=false; t:=t-1; ar:=A(r+s):=1/A(r+s); mp:=-1; mf:=true;
      for j:=n step -1 until 1 do if j<>s then
      begin
        i:=(j×(j-1))shift(-1);  bj:=B(j);  m:=M(j);
        aj:=if s<j then A(i+s)×ar else
          (if bj then ar else -ar)×A(r+j);
        for k:= 1 step 1 until j do if k<>s then
        begin
          ak:=A(k+i):=A(k+i)-(if k<s then A(k+r)×aj else
            (if B(k) then aj else -aj)×A((k×(k-1))shift(-1)+s));
          if bj then begin if mf then begin if k<j then
          begin
            if abs ak>M(k) then M(k):= abs ak;
            if abs ak>m then begin if B(k) then m:=abs ak end;
          end end end bj
        end k;
        if s<j then A(i+s):=aj else A(r+j):=if bj then -aj else aj;
        if bj then
        begin
          if mf then
          begin
            if abs ak>m×mp then
            begin
```

```
          if m=0 then mf:=false else mp:=abs ak/m;  p:=j
         end abs ak>m×mp
        end mf;
        M(j):=0
       end bj
      end j;
      goto next_pivot
     end mp>0 | -,mf;
     if mp=0 then
     begin comment this is the exceptional case where
           all diagonal-elements are zero;
      B(s):=false;  m:=0;
      for j:=s-1 step -1 until 1 do if B(j) then
      begin
       i:=(j×(j-1))shift(-1); ak:=0;
       for k:= s-1 step -1 until 1 do if B(k) then
       begin
        if abs A(if k<j then k+i else j+(k×(k-1))shift(-1))>ak then
         ak:=abs A(if k<j then k+i else j+(k×(k-1))shift(-1))
       end k;
       if abs A(r+j)>m×ak then
       begin
        s1:=j;
        if ak=0 then goto L;
        m:=abs A(r+j)/ak
       end
      end j;
L:    t:=t-2; r1:=(s1×(s1-1))shift(-1);
      ar:=A(r+s1):=1/A(r+s1); B(s1):=false; mp:=-1; mf:=true;
      for j:=n step -1 until 1 do if j<>s∧j<>s1 then
      begin
       i:=(j×(j-1))shift(-1); bj:=B(j);  m:=M(j);
       aj:=if s<j then A(i+s)×ar else
        (if bj then ar else -ar)×A(r+j);
       aj1:=if s1<j then A(i+s1)×ar else
        (if bj then ar else -ar)×A(r1+j);
```

```
      for k:=1 step 1 until j do if k<>s ∧ k<>s1 then
      begin
        ak:=A(i+k):=A(i+k)-(if k<s then A(r+k)×aj1 else
          (if B(k) then aj1 else -aj1)×A((k×(k-1))shift(-1)+s))
          -(if k<s1 then A(r1+k)×aj else
          (if B(k) then aj else -aj)×A((k×(k-1))shift(-1)+s1));
        if bj then begin if mf then begin if k<j then
        begin
          if abs ak>m then begin if B(k) then m:=abs ak end;
          if abs ak>M(k) then M(k):= abs ak
        end end end bj
      end k;
      if s<j then A(i+s):=aj1 else
        A(r+j):=if bj then -aj1 else aj1;
      if s1<j then A(i+s1):=aj else
        A(r1+j):=if bj then -aj else aj;
      if bj then
      begin
        if mf then
        begin
          if abs ak>m×mp then begin
            if m=0 then mf:=false else mp:=abs ak/m; p:=j
          end abs ak>mp
        end mf;
        M(j):=0
      end bj
    end j;
    goto next_pivot
  end m=0;
  invert_sym:= t=0;
  if t<>0 then
  begin
    i:=0;
    for j:=1 step 1 until n do
    begin
      for k:=1 step 1 until j do if B(j) | B(k) then A(i+k):=0;
      i:=i+j
    end j
  end t<>0
end invert_sym;
```

comment

Parameters

call parameter:

n integer. The order of M

call and return parameter:

A(1:n×(n+1)//2) array. Must on entry contain the lower half of M, so that $M(r, s) = M(s, r) = A(r\times(r+1)//2+S)$. At return A will contain the inverse of M stored in the same way

return parameter:

invertsym boolean procedure. It is false if M is singular else true;