Raabo.

A/S REGNECENTRALEN
SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

SYSTEM LIBRARY

RCSL NO: 55-D60

TYPE : Algol 5 Procedures

AUTHOR : Peter Fleron

EDITION: November 1969 (E)

RC 4000 SOFTWARE

MATHEMATICAL PROCEDURE LIBRARY

decompose

solve

ABSTRACT

The procedure decompose performs a triangular decomposition of an arbitrary non-singular matrix. One set of equations can then be solved by the procedure solve.

1. Function and Parameters.

1.1 Decompose:

Decompose calculates upper and lower triangular matrices u and l such that l×u=a, where a is a given n×n square matrix. With the additional requirement u(i,i)=1, the decomposition is unique if a is non-singular. In order to ensure numerical stability, row-exchanges are performed (explicitly) and information about these exchanges is stored for further use in subsequent procedures handling the decomposed matrix.

Implied procedure head:

```
boolean procedure decompose(a,p,mode);
value mode;
array a;
integer array p;
integer mode;
```

Call parameter:

mode : (integer or real). This parameter governs the precision in the calculation of the inner-products in the algorithm:

- mode=0 : The inner-products are calculated in normal floating point mode.
- mode=1 : The inner-products are calculated by means of intermediate variables of 45 bits mantissa and 24 bits exponent.

Call and Return Parameter:

a : (real array or zone record with n×n elements). Contains at entry the square matrix to be decomposed. On exit, each element of a is replaced by the corresponding element of u or l. (The diagonal of u is not stored). In case of a one-dimensional array or a record, the elements of a must be stored row-wise.

Return Parameters:

decompose : (boolean). True if the matrix a is non-singular, otherwise false.

p : (integer array with n elements). Contains information about the row-exchanges. (see section 2.Method).

1.2 Solve:

Solve calculates the solution-vector x to the system of equations a×x=b, where a is a n×n square matrix, decomposed by a previous call of decompose, and where b is a column vector containing the given righthand side. Thus, the solution of several systems of equations with the same matrix of coefficients requires one call of decompose followed by a number of calls of solve.

Implied procedure head:

```
procedure solve(a,p,mode,b);
value mode;
array a,b;
integer array p;
integer mode;
```

Call Parameters:

mode : (real or integer). cf. decompose.

a : (real array or zone record with n×n elements). Contains the decomposed coefficient-matrix as produced by decompose.

p : (integer array with n elements). Contains information on the rowexchanges of the matrices held in a.

Call and Return Parameter:

b : (real array or zone record with n elements). Contains on entry the given right-hand side. On exit, the corresponding solutions are stored in b.

1.3 Parameter-check.

In case of wrong parameters the run is terminated with an error message on current output consisting of the procedure name (decomp or solve) and a number, indicating the wrong parameter as follows:

1: The number of elements of a is different from n××2 (n being the number of elements of p).

2: Wrong content of p (solve only). Indicates an impossible row-exchange or an attempt to solve a singular system of equations.

3: mode<0 or mode>1.

4: The number of elements of b is different from n (solve only).

2. Method.

Decompose produces the triangular matrices l and u in n steps, in the k-th of which the k-th column of l and the k-th row of u ($0 <= k <= n-1$) are calculated by

$$(2.1)\qquad l: \quad a(j,k):=a(j,k)-\sum_{i=0}^{k-1} a(j,i)\times a(i,k) \quad ; \quad j:=k,k+1,\ldots,n-1$$

$$(2.2)\qquad u: \quad a(k,j):=(a(k,j)-\sum_{i=0}^{k-1} a(i,j)\times a(k,i))/a(k,k); \quad j:=k+1,k+2,\ldots,n-1$$

During the calculation of the elements of l, the k-th pivotal index, piv, is found using the criterion

$$\text{abs } a(j,k)/2\times\times ex(j) \; = \; \text{maximum with respect to } j$$

where ex(j) is the initial maximum exponent of the numbers constituting the j-th row.

This pivotal strategy is chosen on two counts: It is simple, and none is known to be universally better (cf. [1]).

If all elements of the column of l turns out to be (exactly) zero, p(1) is set equal to 2048, and the procedure exits with the value false. Otherwise, p(k) is set to the pivotal index, piv, and if piv is greater than k, ex(piv) is set to ex(k) and the k-th and the piv-th row of a are exchanged before the elements of u are calculated.

Solve proceeds in two steps: First, the equations

$$l\times y=b$$

is solved for y, exchanging the elements of b as described by p, after which the final solution x is found by solving

$$u\times x=y$$

Here, b is successively replaced by y and x. The formulae used are analogous to those of (2.1)-(2.2):

$$(2.3)\qquad l: \quad b(k):=(b(k)-\sum_{i=0}^{k-1} b(i)\times a(k,i))/a(k,k) \quad ; \quad k:=0,1,\ldots,n-1$$

$$(2.4)\qquad u: \quad b(k):=b(k)-\sum_{i=k+1}^{n-1} b(i)\times a(k,i) \quad ; \quad k:=n-2,n-3,\ldots,1,0$$

During the first step, it is checked that $n > p(k) >= k$. If this check fails, the run is terminated as described in section 1.

If the value of the parameter mode is 1, the inner-products of (2.1)-(2.4), i.e. expressions of the form

$$-(\text{sum } r(i)\times s(i)+r(k)\times(-1))$$

are calculated by retaining 45 bits of each product and adding this to a sum of 45 significant bits. (The exponent is kept in 24 bits). Thus, instead of the rounding errors in each multiplication and addition, introduced by the normal floating-point operations, an error is introduced only in the final rounding of the sum to a floating-point number. However, it should be noted that only to a certain extent this procedure can cope with a severe cancellation of significant bits that may arise when a product is added to the sum.

The following peculiarities, due to the fact that the procedures are written in the assembler language SLANG 3, should be mentioned:

a) The error message constituents lin.eq.1 and lin.eq.2 occur in these messages instead of ext<line number>. The possibilities are:

   lin.eq.1 : Overflow/underflow in calculations outside the inner-product procedure.

   lin.eq.2 : a) Overflow/underflow in the inner-product procedure. (If mode=1, this can happen only in the final rounding to a floating-point number).

   b) The parameter errors as described in section 1.3

   Some examples are shown in section 4.

b) The formal parameter p contains as explained the pivotal indices; however, the k-th index is not found in p(k) (i.e. the word number k of p), but in the k-th byte of p. A possible way of unpacking these indices is shown in the program in section 4.
   The remaining bytes of p are used for the exponents ex(k).

c) As stated implicitly in section 1, the index bounds and the number of indices of the actual array-parameters are irrelevant. Only the number of elements in the declaration is taken into consideration.

3. Accuracy, Time and Storage Requirements.

Accuracy: Depends on the problem and on the choice of the parameter mode.

The table below shows the median-error (in units of $_{10}-10$) of 11 sets of equations, consisting of equally distributed random numbers $(-_{10}20|_{10}20)$. The error is expressed as the residual norm relative to the norm of the right-hand side. (The Euclidian norm is used).

order	error mode=0	error mode=1
10	2.6	1.9
20	3.9	2.3
30	7.4	4.0
40	9.5	5.0
50	27	8.3
60	21	7.6
70	33	8.7

Time: Based on recorded solution-times for the systems mentioned above, the following execution-times in msec., expressed as functions of the order, holds within ± 10 pct. for orders between 50 and 100:

	mode=0	mode=1
decompose	0.02×(1+10/n)×n××3	0.08×(1+5/n)×n××3
solve	0.07×n××2	0.3×n××2

Storage Requirements: 2 segments of program

0 variables.

4. Test and Discussion.

As may be expected, the results obtained for mode=1 are significantly better than those for mode=0 only if n is sufficiently large. On the other hand, if the system is ill-conditioned, the results can be widely different even for small n. As an example, the system

$$a: \begin{matrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{matrix} \qquad b: \begin{matrix} 32 \\ 23 \\ 33 \\ 31 \end{matrix}$$

the exact solution of which is (1,1,1,1), yields the results:

mode=0:

decomposed matrix:

$7.0000000000_{10}\ 0$	$7.1428571432_{10}\ -1$	$8.5714285716_{10}\ -1$	$7.1428571432_{10}\ -1$
$8.0000000000_{10}\ 0$	$2.8571428570_{10}\ -1$	$1.1000000002_{10}\ 1$	$1.1500000002_{10}\ 1$
$7.0000000000_{10}\ 0$	$0.0000000000_{10}\ 0$	$3.0000000001_{10}\ 0$	$1.6666666667_{10}\ 0$
$1.0000000000_{10}\ 1$	$-1.4285714296_{10}\ -1$	$1.0000000014_{10}\ 0$	$-1.6666666791_{10}\ -1$

piv	ex	solutions
1	4	$1.0000000459_{10}\ 0$
2	4	$9.9999992456_{10}\ -1$
3	4	$1.0000000186_{10}\ 0$
3	4	$9.9999998884_{10}\ -1$

mode=1:

decomposed matrix:

$7.0000000000_{10}\ 0$	$7.1428571432_{10}\ -1$	$8.5714285716_{10}\ -1$	$7.1428571432_{10}\ -1$
$8.0000000000_{10}\ 0$	$2.8571428570_{10}\ -1$	$1.1000000002_{10}\ 1$	$1.1500000002_{10}\ 1$
$7.0000000000_{10}\ 0$	$-2.9103830457_{10}-11$	$3.0000000006_{10}\ 0$	$1.6666666665_{10}\ 0$
$1.0000000000_{10}\ 1$	$-1.4285714290_{10}\ -1$	$1.0000000009_{10}\ 0$	$-1.6666666733_{10}\ -1$

piv	ex	solutions
1	4	$1.0000000082_{10}\ 0$
2	4	$9.9999998644_{10}\ -1$
3	4	$1.0000000034_{10}\ 0$
3	4	$9.9999999800_{10}\ -1$

The Euclidian error-norm is $9.1_{10}-8$, $1.6_{10}-8$ respectively.

The following program was used

```
lin.eq. test parameter error etc.
begin integer d1,d2,d3,d4,mode;
  underflows:=-1;
  read(in,d1,d2,d3,d4,mode);
  begin array a(1:d1), b(d2:d3);
    integer array p(1:d4), piv(1:2×d4);
    integer i,j,k;
    read(in,a,b);
    if -,decompose(a,p,mode) then write(out,<:<10>sing:>);
    write(out,<:<10>decomposed matrix::>);
    k:=1;
    for i:=1 step 1 until d4 do
    begin write(out,<:<10>:>);
      for j:=1 step 1 until d4 do
      begin write(out,<< -d.dddddddddd_10-dd>,a(k));
        k:=k+1
      end;
      j:=p(i);
      piv(2×i-1):=j shift (-12) extract 12;
      piv(2×i):=j extract 12;
    end;
    solve(a,p,mode,b);
    write(out,<:<10><10>    piv    ex    solutions:>);
    for i:=1 step 1 until d4 do
      write(out,<:<10>:>,<< ddddd>,piv(i),piv(i+d4),
                 << -d.dddddddddd_10-dd>,b(i+d2-1))
  end block
end
```

This program produces the error-messages shown below when the input is

4, 1, 2, 2, 0, a, 1, 1,

where a means the four elements of a 2X2 matrix:

I) a: 1, 2, 1, 2

```
solve        2  lin. eq. 2
called from     line  21-22
```

II) a: $_{10}400$, $_{10}-400$, $_{10}400$, $_{10}-400$

```
real            lin. eq. 1
called from     line   8-8
```

III) a: 1, $8_{10}615$, 0.5, $-8_{10}615$

```
real            lin. eq. 1
called from     line  21-22
```

IV) a: 1, $12_{10}615$, 0.5, $-12_{10}615$

```
real            lin. eq. 2
called from     line   8-8
```

5. References

[1] Forsythe, G and Moler, C.B.: Computer Solution of Linear Algebraic Systems. Prentice-Hall. 1967.

6. Algorithms.

Since the procedures are written in SLANG, the algorithms will not be given.