ALGOL 60 PROCEDURES
IN NUMERICAL ALGEBRA

PART 1

T. J. DEKKER

MATHEMATICAL CENTRE TRACTS 22
MATHEMATICAL CENTRE TRACTS
22

ALGOL 60 PROCEDURES
IN NUMERICAL ALGEBRA

PART 1

BY

T.J. DEKKER

2nd Edition

MATHEMATISCH CENTRUM AMSTERDAM
PREFACE

We here present a set of ALGOL 60 procedures for solving systems of linear equations, for inverting matrices and for solving linear least-squares problems. The procedures use single-length scalar-product procedures and no iterations are applied for improving the solutions. In the future, we plan to present a corresponding set using double-length scalar-product procedures and applying iterations for improving the solutions, and a set for calculating eigenvalues and eigenvectors of matrices.

The procedures have been tested on an Electrologica X3 computer by means of the "MC ALGOL 60 system for the X3" of the Mathematical Centre, Amsterdam, written by F. E. J. Kruseman Aretz. Only the procedures mca 2000 to 2005 of section 200 are available as dL X3-machine-code procedures.

The texts of the procedures have been edited by an ALGOL editing program written by H. L. Oudshoorn, H. N. Glorie and G. C. J. M. Hogeredel[12].

In the second edition some minor errors have been corrected. Three of these corrections concern errors in the texts of mncelm (mca 2110), detsolbd (mca 2120) and detsolbd (mca 2122) which were related to the handling of the row norms in the pivot selection.

ACKNOWLEDGEMENTS

The author is indebted to Prof. Dr. Ir. A. van Wijngaarden and Prof. Dr. F. E. J. Kruseman Aretz for invaluable suggestions and criticisms, to B. J. Mailloux M. Sc. for carefully reading the manuscript and suggesting several improvements, to P. Beertens, F. J. van de Bosch, W. Hoffmann, R. van der Horst, D. P. de Jong, L. Meertens, R. P. Nederpelt Lazarom, J. van der Welden and R. Wiggers for their help in writing the ALGOL texts and in testing the procedures, to Mrs. E. Binnenmarsch for typing the manuscript, and to D. Zwart and J. Suiker for the printing and the binding of this booklet.
CONTENTS

PREFACE
ACKNOWLEDGEMENTS
CONTENTS
NOTATIONS
DEFINITIONS
INTRODUCTION

CHAPTER 20 VECTOR OPERATIONS

<table>
<thead>
<tr>
<th>Section 200 Scalar products</th>
<th>7 - 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2000 vecvec</td>
<td>8 - 11</td>
</tr>
<tr>
<td>mca 2001 matvec</td>
<td>8 - 9</td>
</tr>
<tr>
<td>mca 2002 temvec</td>
<td></td>
</tr>
<tr>
<td>mca 2003 matmat</td>
<td></td>
</tr>
<tr>
<td>mca 2004 temmat</td>
<td></td>
</tr>
<tr>
<td>mca 2005 mattam</td>
<td></td>
</tr>
<tr>
<td>mca 2006 seqvec</td>
<td>10 - 11</td>
</tr>
<tr>
<td>mca 2008 scaprd1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 201 Elimination</th>
<th>12 - 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2010 elmvec</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2011 elmvec col</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2012 elmcolvec</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2013 elmcov</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2014 elmrow</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2019 maxelmrow</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 202 Interchanging</th>
<th>14 - 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2020 ichvec</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2021 ichcol</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2022 ichrow</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2023 ichrow col</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2024 ichseqvec</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2025 ichseq</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 203 Rotation</th>
<th>16 - 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2031 rotecol</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2032 rotrow</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

CHAPTER 21 LINEAR SYSTEMS AND MATRIX INVERSION

<table>
<thead>
<tr>
<th>Section 210 Triangular decomposition with partial pivoting</th>
<th>18 - 39</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2100 det</td>
<td>19 - 23</td>
</tr>
<tr>
<td>mca 2101 sol</td>
<td>20 - 21</td>
</tr>
<tr>
<td>mca 2102 detsol</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2103 inv</td>
<td>&quot;</td>
</tr>
<tr>
<td>mca 2104 detsinv</td>
<td>22 - 23</td>
</tr>
</tbody>
</table>
Section 211 Elimination with complete pivoting

- mca 2110 rnkelm
- mca 2111 solelm
- mca 2112 rnksolelm
- mca 2113 solhom
- mca 2114 invelm

Section 212 Band matrices

- mca 2120 detbnd
- mca 2121 solbnd
- mca 2122 detsolbnd

CHAPTER 22 POSITIVE DEFINITE SYMMETRIC LINEAR SYSTEMS AND MATRIX INVERSION

Section 220 Cholesky decomposition without pivoting

- mca 2200 detsym2
- mca 2201 solsym2
- mca 2202 detsolsym2
- mca 2203 invsym2
- mca 2204 detsinvsym2
- mca 2205 detsym1
- mca 2206 solsym1
- mca 2207 detsolsym1
- mca 2208 invsym1
- mca 2209 detsinvsym1

Section 221 Cholesky decomposition with pivoting

- mca 2210 rnkaysym20
- mca 2211 solsym20
- mca 2212 rnksolesym20
- mca 2213 invsym20
- mca 2214 rnkinvsym20
- mca 2215 rnkaysym10
- mca 2216 solsym10
- mca 2217 rnksolesym10
- mca 2218 invsym10
- mca 2219 rnkinvsym10

Section 222 Cholesky decomposition for band matrices

- mca 2220 detsymbnd
- mca 2221 solsymbnd
- mca 2222 detsolsymbnd

Section 224 Least-squares problems

- mca 2240 isqdec
- mca 2241 isqsol
- mca 2242 isqglinv
- mca 2243 isqdecsv

REFERENCES

APPENDIX. TIMES FOR THE MC ALGOL 60 SYSTEM FOR THE X3

70 - 71 - 72
NOTATIONS

References are given between the square brackets "[" and "]". 
" : " denotes the integer division symbol " : " [1,3.3.4.2]. 
"goto" denotes the same symbol as "go to", [1,4.3]. 
"O" denotes a null vector; the number of elements will be clear from 
the context. 
" min " (" max ") denotes the function whose value is the minimum 
(maximum) value of its two operands. 
The prime symbol " ' " denotes transposition of a matrix. 
" M " denotes the matrix considered and " n " denotes its order, 
unless stated otherwise.

DEFINITIONS

The "dimension" of an array is the number of its subscripts (see also 
[1,5.2.3.2]). Thus we speak about "one-dimensional" and "two-
dimensional" arrays. The first subscript of a two-dimensional array is 
called the "row index" and the second the "column index". 
The i-th "row" ("column") of a two-dimensional array is the set of its 
elements for which the row (column) index equals i. 
The "upper triangle" of a matrix or of a two-dimensional array is the 
set of its elements for which the first subscript does not exceed the 
second. 
A "unit triangular" matrix is a triangular matrix whose diagonal 
elements are equal to 1. 
We use the following vector norms [2, p.80] [3, p.55] : the "one-
norm", i.e. the sum of the absolute values of the elements of the 
vector; the "Euclidean norm", i.e. the square root of the sum of their 
squares; and the "infinity-norm", i.e. their maximum absolute value. 
We use the following matrix norms : the "infinity-norm", i.e. the 
maximum one-norm of its rows; the "maximum-norm", i.e. the maximum 
absolute value of its elements. The maximum-norm of a positive 
semidefinite symmetric matrix obviously equals the maximum of its 
diagonal elements.

The "machine precision" is the largest number, $p$, for which $1 + p = 1$ 
on the computer (about $10^{-12}$ for the x8). 
A "relative tolerance" is a tolerance relative to some vector or 
matrix norm. Relative tolerances must be chosen smaller than one, and 
should be chosen not smaller than the machine precision.
INTRODUCTION

Chapter 20 contains a set of procedures for vector operations. Most of these are used in the subsequent chapters; some procedures of section 201 and 203 will be used in procedures for calculating eigenvalues and eigenvectors (to be published). A vector is given either in a one-dimensional array or in a row or column of a two-dimensional array. In the former case, we often use the same name for the vector as for the array if the whole array is used for the vector.

Chapter 21 contains a set of procedures for solving linear systems and for inverting matrices. The matrix is given in a two-dimensional array, the columns and rows of which correspond with the columns and rows of the matrix. A band matrix, however, is given in a one-dimensional array. For details, see sections 212 and (for the positive-definite symmetric case) 222.

Chapter 22 deals with the special case of positive-definite symmetric matrices and, moreover, contains a section for solving linear least-squares problems. Of symmetric matrices and upper-triangular matrices only the upper triangle will be given or delivered, either in the upper triangle of a two-dimensional array (in which case the remaining part of the array is not used) or in a one-dimensional array [8]. In the latter case, the (i, j)−th element of the matrix is, for $1 \leq j \leq 2 + i$. Thus, the memory space occupied by the matrix is cut nearly in half. A drawback is that the elements in a row of the upper triangle are not linear functions of the running subscript, so that special procedures for the vector operations are needed.

In each chapter, we give a survey of its contents and some numerical considerations and comparisons. The chapters are subdivided into sections in each of which we give a more detailed survey of its contents and explain the numerical methods used. Each section contains one or more procedures. For each procedure, we give a description in which the required data, the delivered results and the (directly or indirectly) used nonlocal procedures are mentioned. The data of the procedures are given by actual parameters whose corresponding formal parameters are either specified real or integer and called by value, or specified array or integer array and called by name. The results of the procedures are delivered either as the value assigned to the procedure identifier (of type real or integer), or in arrays corresponding to formal parameters specified array or integer array. Some arrays are used as well for data as for results.

For each formal parameter specified array or integer array, we give the "minimal declaration", i.e. a declaration with the appropriate number of bound pairs, where each pair indicates the range which is actually used by the procedure. Of course, the declaration of the corresponding actual parameter may contain smaller lower bounds and
greater upper bounds. (The descriptions of the procedures of sections 2.02 and 2.03 contain two minimal declarations of the same parameter with the meaning that the declaration of the corresponding actual parameter has to include both of them.) Sometimes not all elements of the array indicated by the minimal declaration are used. In the descriptions, we always mention which part is used for the data and which part for the results, so that it is always clear which elements are not used at all and which elements are left unchanged. Only in some cases shall we explicitly state that some elements are left intact, if it is important for the applications.
CHAPTER 20

VECTOR OPERATIONS

This chapter contains procedures for calculating scalar products, for adding a multiple of a vector to another vector, for interchanging two vectors and for rotating two vectors.

Each of the vectors is given by means of a pair of subscript bounds and either a two-dimensional array identifier with a row or column number or a one-dimensional array identifier. The procedures which use only one-dimensional arrays, have some extra facilities. Some of these procedures handle rows of upper-triangular or symmetric matrices given in a one-dimensional array, which are represented in a special way, viz. with linearly increasing spacing of the successive elements.

Compared with the technique of defining the vectors by means of subscripted variables explicitly depending on a bound variable, as is common for scalar product procedures [1, 4.7.2, and 5.4.2] [5] [6] [10], our technique is less flexible, but more efficient (at least in the MACL 60 system for the X8); moreover, our procedures may well be written as machine-code procedures in which the elements of the vectors are selected more efficiently on account of their equidistant (or linearly increasing) spacing in the memory. As to the lesser flexibility, instead of one procedure we need a set of procedures for the most important applications. As to the machine-code procedures, if explicit bound variables were used, the more efficient element selection would be possible only if the subscripts were linear (or, in the case of linearly increasing spacing, quadratic) functions of the bound variable, but this requirement is easily violated and difficult to check.

Our procedures avoid this difficulty.
comment mca 2000;
real procedure vecvec(l, u, shift, a, b); value l, u, shift;
integer l, u, shift; array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[k] \times b[shift + k] + s;
vecvec:= s
end vecvec;

comment mca 2001;
real procedure matvec(l, u, i, a, b); value l, u, i; integer l, u, i;
array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[i,k] \times b[k] + s; matvec:= s
end matvec;

comment mca 2002;
real procedure tamvec(l, u, i, a, b); value l, u, i; integer l, u, i;
array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[k,i] \times b[k] + s; tamvec:= s
end tamvec;

comment mca 2003;
real procedure matmat(l, u, i, j, a, b); value l, u, i, j;
integer l, u, i, j; array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[i,k] \times b[k,j] + s; matmat:= s
end matmat;

comment mca 2004;
real procedure tammat(l, u, i, j, a, b); value l, u, i, j;
integer l, u, i, j; array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[k,i] \times b[k,j] + s; tammat:= s
end tammat;

comment mca 2005;
real procedure mattam(l, u, i, j, a, b); value l, u, i, j;
integer l, u, i, j; array a, b;
begin integer k; real s;
  s := 0;
  for k:= 1 step 1 until u do s := a[i,k] \times b[j,k] + s; mattam:= s
end mattam;
Section 200 Scalar products

The procedures of this section calculate the scalar product of two vectors, each of which is given either as (a part of) a one-dimensional array or as row or column of a two-dimensional array. If the lower bound of the running subscript is greater than the upper bound, then 0 is delivered as scalar product.
The lower and upper bound of the running subscript are given by two parameters;
vecc and seqvec feature the additional possibility of shifting the range of the running subscript of the second vector; in scprd1,
moreover, the spacings of the vectors are arbitrary constants; in
seqvec the spacing of the successive elements of the first vector
increases linearly. (The latter procedure is used for symmetric or
upper-triangular matrices given in one-dimensional arrays.)
In the NC ALGOL 60 system for the X8 the procedures mca 2000 to 2005
are available as machine-code procedures (which are about 7 times
faster than the corresponding equivalent ALGOL procedures, see
Appendix).

Description mca 2000
vecc: scalar product of the vectors given in array a[l:u] and
array b[shift + 1 : shift + u].

Description mca 2001
matvec: scalar product of the row vector given in array a[i:i, l:u]
and the vector given in array b[l:u].

Description mca 2002
tasvec: scalar product of the column vector given in
array a[l:u, i:i] and the vector given in array b[l:u].

Description mca 2003
matmat: scalar product of the row vector given in array a[i:i, l:u]
and the column vector given in array b[l:u, j:j].

Description mca 2004
tasmat: scalar product of the column vectors given in
array a[l:u, i:i] and array b[l:u, j:j].

Description mca 2005
matmat: scalar product of the row vectors given in array a[i:i, l:u]
and array b[j:j, l:u].
comment mca 2006;
real procedure seqvec(l, u, il, shift, a, b); value l, u, il, shift;
integer l, u, il, shift; array a, b;
begin real s;
    s := 0;
    for l := l step 1 until u do
        begin s := a[Il] x b[l + shift] + s; il := il + 1 end;
    seqvec := s
end seqvec;

comment mca 2008;
real procedure scaprd1(la, sa, lb, sb, n, a, b);
value la, sa, lb, sb, n; integer la, sa, lb, sb, n; array a, b;
begin integer k;
    real s;
    s := 0;
    for k := 1 step 1 until n do
        begin s := a[la] x b[lb] + s; la := la + sa; lb := lb + sb end;
    scaprd1 := s
end scaprd1;
Description mca 2005
\[ \text{seqvec} = \text{scalar product of the vectors given in} \]
\[ \text{array } a[l : il + (u + 1 - 1) \times (u - 1) : 2] \text{ and} \]
\[ \text{array } b[\text{shift} + 1 : \text{shift} + u], \text{ where the elements of the first vector} \]
\[ \text{are } a[il + (j + 1 - 1) \times (j - 1) : 2] \text{ for } j = 1, \ldots, u. \]

Description mca 2008
\[ \text{scaprd} = \text{scalar product of the vectors given in} \]
\[ \text{array } a[\text{min} (la, la + (n - 1) \times sa) : \text{max} (la, la + (n - 1) \times sa)] \]
\[ \text{and array } b[\text{min} (lb, lb + (n - 1) \times sb) : \text{max}(lb, lb + (n - 1) \times sb)], \]
\[ \text{where the elements of the vectors are} \]
\[ a[la + (j - 1) \times sa] \text{ and } b[lb + (j - 1) \times sb] \text{ for } j = 1, \ldots, n. \]
comment mca 2010;
procedure elmvec(l, u, shift, a, b, x); value l, u, shift, x;
integer l, u, shift; real x; array a, b;
for l:= 1 step 1 until u do a[l] := a[l] + b[l + shift] × x;

comment mca 2011;
procedure elmvveccol(l, u, i, a, b, x); value l, u, i, x;
integer l, u, i; real x; array a, b;
for l:= 1 step 1 until u do a[l] := a[l] + b[l, i] × x;

comment mca 2012;
procedure elmcovvec(l, u, i, a, b, x); value l, u, i, x;
integer l, u, i; real x; array a, b;
for l:= 1 step 1 until u do a[l, i] := a[l, i] + b[l, i] × x;

comment mca 2013;
procedure elmcoll(l, u, i, j, a, b, x); value l, u, i, j, x;
integer l, u, i, j; real x; array a, b;
for l:= 1 step 1 until u do a[l, i] := a[l, i] + b[l, j] × x;

comment mca 2014;
procedure elmcollrow(l, u, i, j, a, b, x); value l, u, i, j, x;
integer l, u, i, j; real x; array a, b;
for l:= 1 step 1 until u do a[l, i] := a[l, i] + b[l, j] × x;

comment mca 2019;
inumber procedure maxelmrow(l, u, i, j, a, b, x); value l, u, i, j, x;
inumber l, u, i, j; real x; array a, b;
begin integer k;
  real r, s;
  s := 0;
  for k:= 1 step 1 until u do
    begin r := a[i, k] := a[i, k] + b[j, k] × x; if abs(r) > s then
      begin s := abs(r); 1 := k end
    end;
  maxelmrow := 1
end maxelmrow;
Section 201 Elimination

The procedures of this section perform a Gaussian elimination on a vector. More precisely, a multiple of one vector is added to another vector. Each vector is given either as a one-dimensional array or as row or column of a two-dimensional array. The lower and upper bound of the running subscript are given by two parameters; elmvec features the additional possibility of shifting the range of the running subscript of one vector.

Description mca 2010
elmvec adds x times the vector given in array b[shift + 1 : shift + u]
to the vector given in array a[1:u].

Description mca 2011
elmveccol adds x times the column vector given in array b[1:u, i:i] to
the vector given in array a[1:u].

Description mca 2012
elmcolvec adds x times the column vector given in array b[1:u]
to the column vector given in array a[1:u, 1:1].

Description mca 2013
elmcol adds x times the column vector given in array b[1:u, j:j] to
the column vector given in array a[1:u, 1:1].

Description mca 2014
elmrow adds x times the row vector given in array b[j:j, 1:u] to the
row vector given in array a[1:i, 1:u].

Description mca 2019
maxelmrow adds x times the row vector given in array b[j:j, 1:u] to
the row vector given in array a[1:i, 1:u].
Moreover, maxelmrow := the value of the second subscript of an element
of the new row vector in array a which is of maximum absolute value.
If, however, 1 > u, then maxelmrow := 1.
comment mca 2020;
procedure ichvec(l, u, shift, a); value l, u, shift;
    integer l, u, shift; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[l ]; a[l]:= a[l + shift ]; a[l + shift]:= r end
end ichvec;

comment mca 2021;
procedure ichcol(l, u, i, j, a); value l, u, i, j;
    integer l, u, i, j; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[l,i ]; a[l,i]:= a[l,j ]; a[l,j]:= r end
end ichcol;

comment mca 2022;
procedure ichrow(l, u, i, j, a); value l, u, i, j;
    integer l, u, i, j; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[l,i ]; a[l,i]:= a[l,j ]; a[l,j]:= r end
end ichrow;

comment mca 2023;
procedure ichrowcol(l, u, i, j, a); value l, u, i, j;
    integer l, u, i, j; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[l,i ]; a[l,i]:= a[l,j ]; a[l,j]:= r end
end ichrowcol;

comment mca 2024;
procedure ichseqvec(l, u, il, shift, a); value l, u, il, shift;
    integer l, u, il, shift; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[il ]; a[il]:= a[l + shift ]; a[l + shift]:= r;
            il:= il + 1
        end
end ichseqvec;

comment mca 2025;
procedure ichseq(l, u, il, shift, a); value l, u, il, shift;
    integer l, u, il, shift; array a;
begin real r;
    for l:= 1 step 1 until u do
        begin r:= a[il ]; a[il]:= a[il + shift ]; a[il + shift]:= r;
            il:= il + 1
        end
end ichseq;
Section 202 Interchanging

The procedures of this section interchange the elements of two vectors. Each vector is given either as (a part of) a one-dimensional array or as row or column of a two-dimensional array. The lower and upper bound of the running subscript are given by two parameters; ichvec and ichseqvc feature the additional possibility of shifting the range of the running subscript of the second vector; in ichseqvc and ichsqk the spacing of the successive elements of one or both of the vectors increases linearly. (The latter procedures are used for symmetric matrices given in one-dimensional arrays.)

Description mca 2020
Ichvec interchanges the elements of the vector given in array a[1:u] and array a[shift + 1 : shift + u].

Description mca 2021
Ichvec interchanges the elements of the column vectors given in array a[1:u, i:j] and array a[1:u, j:j].

Description mca 2022
Ichrow interchanges the elements of the row vectors given in array a[i:i, 1:u] and array a[j:j, 1:u].

Description mca 2023
Ichrowcol interchanges the elements of the row vector given in array a[i:i, 1:u] and the column vector given in array a[1:u, j:j].

Description mca 2024
Ichseqvc interchanges the elements of the vectors given in array a[i1 : i1 + (u + l - 1) × (u - 1) / 2] and array a[shift + 1 : shift + u], where the elements of the first vector are a[i1 + (j + l - 1) × (j - 1) / 2] for j = 1, ..., u.

Description mca 2025
Ichseq interchanges the elements of the vectors given in array a[i1 : i1 + (u + l - 1) × (u - 1) / 2] and array a[shift + i1 : shift + i1 + (u + l - 1) × (u - 1) / 2], where the elements of the vectors are a[i1 + (j + l - 1) × (j - 1) / 2] and a[shift + i1 + (j + l - 1) × (j - 1) / 2] for j = 1, ..., u.
procedure rotcol(l, u, i, j, a, c, s); value l, u, i, j, c, s;
integer l, u, i, j; real c, s; array a;
begin real x, y;
  for l:= 1 step 1 until u do
    begin x:= a[l,1]; y:= a[l,j]; a[l,1]:= x * c + y * s;
        a[l,j]:= y * c - x * s
    end
end rotcol;

procedure rotrow(l, u, i, j, a, c, s); value l, u, i, j, c, s;
integer l, u, i, j; real c, s; array a;
begin real x, y;
  for l:= 1 step 1 until u do
    begin x:= a[1,l]; y:= a[j,l]; a[1,l]:= x * c + y * s;
        a[j,l]:= y * c - x * s
    end
end rotrow;
Section 203 Rotation

The procedures of this section perform a rotation on two vectors, x and y (say); i.e. the two vectors are replaced by cx + sy and cy - sx, where c and s are two given real values. Each vector is given as row or column of a two-dimensional array. The lower and upper bound of the running subscript are given by two parameters. (These procedures are to be used in procedures for calculating eigenvalues and eigenvectors (to be published).)

Description mca 2031
rotcol replaces the column vector x given in array a[1:u, i:1] and the column vector y given in array a[1:u, j:j] by the vectors cx + sy and cy - sx.

Description mca 2032
rotrow replaces the row vector x given in array a[i:i, 1:u] and the row vector y given in array a[j:j, 1:u] by the vectors cx + sy and cy - sx.
CHAPTER 21

LINEAR SYSTEMS AND MATRIX INVERSION

This chapter contains procedures for solving systems of linear equations and for matrix inversion. Moreover, section 211 contains a procedure for calculating the rank of a matrix and a procedure for solving a homogeneous linear system.

In section 210 triangular decomposition with partial pivoting is used and in section 211 Gaussian elimination with complete pivoting. The procedures of section 212 solve linear systems whose matrices are in band form, by means of Gaussian elimination with partial pivoting.

In exceptional cases, partial pivoting may yield useless results, even for well-conditioned matrices; this may occur only for large matrices whose order (or in the case of band matrices, band width) is not much smaller than the number of binary digits in the number representation [2, p.97] [3, p.212]. Complete pivoting, however, always yields good results for well-conditioned matrices; a "condition number" (i.e. a norm of the matrix times a norm of its inverse) is a measure of the relative accuracy of the solution [2, p.91]. Moreover, complete pivoting is indispensable for calculating the rank of a matrix and for solving homogeneous systems.

For large order $n$ the computation time for solving linear systems and for matrix inversion is proportional to $n$ cubed.

Complete pivoting requires some extra time for the pivot selection, which, for large $n$, is a nearly constant (small) fraction of the total computation time. In the MC ALGOL 60 system for the X3, the partial pivoting procedures of section 210 are much faster than the complete pivoting procedures of section 211, because the procedures mcs 2000 to 2005 are available in machine-code.

The procedures of section 212, for solving linear systems whose matrices are in band form, save a considerable amount of computation time and memory space, if the band width is much smaller than $n$. For large matrices, the computation time is proportional to $n \times$ band width $\times$ the number of diagonals on or to the left of the main diagonal.
Section 210 Triangular decomposition with partial pivoting

This section contains procedures for solving linear systems and for inverting matrices:
- deteval solves a system of linear equations and calculates the determinant of the system;
- determ calculates the inverse and the determinant of a matrix;
- det calculates the determinant of a matrix;
- sol solves a system of linear equations and inv inverts a matrix, provided the matrix is given in the triangularly decomposed form produced by det. One call of det followed by several calls of sol may be used to solve several linear systems having the same matrix but different right hand sides.

The method used in det is triangular decomposition with stabilizing row interchanges, also called "partial pivoting" [2, p.115] [3, p.201] [4] [5] [6].

The method yields a lower-triangular matrix L and a unit upper-triangular matrix U such that the product LU equals the given matrix M with permuted rows.

The process is performed in n steps. The k-th step, k = 1,..., n, produces the k-th column of L; subsequently, the "pivot" is selected in this column; the pivotal row and the k-th row of M (and thus also of L) are interchanged; finally, the k-th row of U is produced. That element of the k-th column of L is chosen as pivot, whose absolute value divided by the Euclidean norm of the corresponding row of M is maximal. Thus, matrix M is "equilibrated" in this pivoting strategy such that the rows effectively obtain unit Euclidean norm. No test for singularity of M is performed.

The determinant equals the product of the diagonal elements of L or minus this value, if the number of proper interchanges is odd.

After the triangular decomposition, sol obtains the solution x of the linear system Mx = b by first permuting the elements of b in the same way as the rows of M, then calculating y such that Ly equals b with permuted elements (forward substitution), and finally calculating x such that Ux = y (back substitution).

The method used in inv is as follows. The inverse, X, of the product LU is calculated from the conditions that XL be a unit upper-triangular matrix and UX a lower-triangular matrix [4, p.34-38]. Subsequently, in correspondence with the interchanges applied on the rows of M, the same interchanges are carried out in reverse order on the columns of X, in order to obtain the inverse of M.
comment mca 2100;
real procedure det(a, n, p); value n; integer n; array a;
ingteger array p;
begin integer i, k, k1, pk;
real d, r, s;
array v[i:n];
for i := 1 step 1 until n do v[i] := 1 / sqrt(matvec(1, n, i, i, a, a)); d := 1;
for k := 1 step 1 until n do
begin r := -1; k1 := k - 1;
for i := k step 1 until n do
begin a[i,k] := a[i,k] - matmat(1, k1, i, k, a, a);
s := abs(a[i,k]) * v[i]; if s > r then
begin r := s; pk := i end
end lower;
p[k] := pk; v[pk] := v[k]; s := a[pk,k]; d := s * d;
if pk + k then
begin d := -d; ichrow(1, n, k, pk, a) end;
for i := k + 1 step 1 until n do a[k,i] := (a[k,i] - matmat(1, k1, k, i, a, a)) / s
end lu;
det := d
end det;

comment mca 2101;
procedure sol(a, n, p, b); value n; integer n; array a, b;
ingteger array p;
begin integer k, pk;
real r;
for k := 1 step 1 until n do
begin r := b[k]; pk := p[k];
b[k] := (b[pk] - matvec(k, k - 1, k, a, b)) / a[k,k];
if pk + k then b[pk] := r
end;
for k := n step -1 until 1 do b[k] := b[k] - matvec(k + 1, n, k, a, b)
end sol;

comment mca 2102;
real procedure detesol(a, n, b); value n; integer n; array a, b;
ingteger array p[1:n];
detsol := det(a, n, p); sol(a, n, p, b)
end detesol;
Description mca 2100

det := determinant of the n-th order matrix M given in
array a[1:n, 1:n], and the triangular decomposition of M is performed.
The resulting lower-triangular matrix and unit upper-triangular matrix
with its unit diagonal omitted are overwritten on a.
The pivotal indices are delivered in integer array p[1:n].
det uses mttam, matmat and ichrow (chapter 20).

Description mca 2101

sol should be called after det and solves the linear system Mx = b,
where M is the n-th order matrix whose triangularly decomposed form
and pivotal indices, as produced by det, are given in
array a[1:n, 1:n] and integer array p[1:n], and where b is the vector
given as array b[1:n]. The solution vector x is overwritten on b.
sol leaves a and p intact, so that, after one call of det, several
calls of sol may follow for solving several systems having the same
matrix but different right hand sides.
sol uses matvec (mca 2001).

Description mca 2102

detsol := determinant of the n-th order matrix M given in
array a[1:n, 1:n], and the triangular decomposition of M is performed.
Moreover the linear system Mx = b is solved, where vector b is given
as array b[1:n]. The solution vector x is overwritten on b, and the
triangularly decomposed form of M is overwritten on a.
detsol uses det, sol and, indirectly, also mttam, matmat, matvec and
ichrow (chapter 20).
comment mca 2103;
procedure inv(a, n, p); value n; integer n; array a; integer array p;
begin integer j, k, k1;
  real r;
  array v[1:n];
  for k:= n step - 1 until 1 do 
    begin k1:= k + 1;
    for j:= n step - 1 until k1 do 
      begin a[j,k1]:= v[j]; v[j]:= -matmat(k1, n, k, j, a, a) end;
      r:= a[k,k];
    for j:= n step - 1 until k1 do 
      begin a[k,j]:= v[j]; v[j]:= -matmat(k1, n, j, k, a, a) / r end;
      v[k]:= (1 - matmat(k1, n, k, k, a, a)) / r end;
  for j:= n step - 1 until 1 do a[j,1]:= v[j];
  for k:= n - 1 step - 1 until 1 do 
    begin k1:= p[k]; if k1 + k then Inchcol(1, n, k, k1, a) end 
end inv;

comment mca 2104;
real procedure detinv(a, n); value n; integer n; array a;
begin integer array p[1:n];
  detinv:= det(a, n, p); inv(a, n, p)
end detinv;
Description mca 2103
Inv should be called after det and calculates the inverse of the matrix whose triangularly decomposed form and pivotal indices, as produced by det, are given in array a[1:n, 1:n] and integer array p[1:n]. The calculated inverse is overwritten on a.
Inv uses matmat and ichool (chapter 20).

Description mca 2104
detinv:= determinant of the n-th order matrix given in array a[1:n, 1:n]. Moreover, the inverse of this matrix is calculated and overwritten on a.
detinv uses det, inv and, indirectly, also mattam, matmat, ichrow and ichool (chapter 20).
Section 211 Elimination with complete pivoting

This section contains procedures for calculating the rank of a matrix, for solving linear systems and for inverting matrices:
rankelm solves a system of linear equations and calculates the determinant of the system;
invelm calculates the inverse and the determinant of a matrix;
rkelm calculates the rank of a rectangular matrix;
solelm solves a system of linear equations and solhsm calculates a solution of a homogeneous system, provided the matrix is given in the Gaussian-eliminated form produced by rkelm.
One call of rankelm followed by several calls of solelm may be used to solve several linear systems having the same matrix but different right-hand sides. By means of successive calls of solhsm one can obtain a complete linearly independent set of solutions of a homogeneous system.

The method used, in rankelm, is Gaussian elimination with complete pivoting [2, p.97] [3, p.212] which yields a lower-triangular matrix L and a unit upper-triangular matrix U such that the product LU equals the given matrix M with permuted rows and columns. Let M have n rows and m columns. The elimination is performed in at most \( \min(n, m) \) steps. In the \( k \)-th step, \( k = 1, \ldots, \min(n, m) \), a "pivot" is selected from the remaining submatrix having \( n - k + 1 \) rows and \( m - k + 1 \) columns; then the pivotal row is interchanged with the \( k \)-th row and the pivotal column with the \( k \)-th column; subsequently, the \( k \)-th "unknown" is eliminated in the last \( n - k \) rows. The pivot is selected in such a way that its absolute value divided by the one-norm of the corresponding row of \( M \) is maximal. Thus, matrix \( M \) is "equilibrated" in this pivoting strategy such that the rows effectively obtain unit one-norm. If all elements of the remaining submatrix are smaller in absolute value than a given relative tolerance times the one-norm of the corresponding row of \( M \), then the elimination is discontinued and the previous step number is delivered as the rank of \( M \).

After the Gaussian elimination, solelm obtains the solution \( x \) of the linear system \( Mx = b \) by calling sol (mca 2101) and then interchanging the elements of the solution vector produced by sol in "reverse correspondence" with the interchanges of the columns of \( M \); i.e. the same interchanges are carried out in reverse order.
In solution, a solution $x$ of the homogeneous system $Mx = 0$, where $M$ is an $n \times m$ matrix of rank $r$, is obtained as follows. Let $V$ be the $r$-th order upper-triangular matrix consisting of the first $r$ columns of the matrix $U$ produced by rank-reduced $U$ the $r \times (m - r)$ matrix consisting of the remaining part of the first $r$ rows of $U$ (the other rows of $U$ are negligible). First, the system $Vy = \text{minus the } k$-th column of $W$, where $k$ is a given positive integer $< m - r$, is solved (back substitution). Then the vector $y$ and the $k$-th unit $(m - r)$-vector are combined to form a single $m$-vector; its elements are then interchanged in reverse correspondence with the interchanges of the columns of $M$, in order to obtain a solution vector $x$. The solution vectors thus obtained for $k = 1, \ldots, m - r$ form a complete set of linearly independent solution vectors of the homogeneous system.

The method used in invelm is Gauss–Jordan elimination with complete pivoting. This method introduces the zeros not only below but also above the main diagonal. At each stage, the element of greatest absolute value of the remaining submatrix is chosen as pivot. (Here, the matrix is not equilibrated, because this would not leave the inverse invariant.) After completing the elimination, the rows and columns are interchanged in reverse correspondence with the interchanges of the columns and rows of $M$. 
comment mca 2110;
integer procedure rmkelma(a, n, m, aux, ri, ci); value n, m;
begin integer i, j, p, q, r, rl, jcrit;
array aux; integer array ri, ci;
array norm[1:n];
crit:= 0;
for pi:= 1 step 1 until n do
begin norm:= max:= abs(a[pi, 1]); jcrit:= 1;
for q:= 2 step 1 until m do
begin aid:= abs(a[pi, q]); rnorm:= rnorm + aid;
if aid > max then
begin max:= aid; jcrit:= q end
end;
norm[pi]:= rnorm:= 1 / rnorm; if max x rnorm > crit then
begin crit:= max x rnorm; i:= pi; j:= jcrit end
end;
eps:= aux[0]; det:= 1; min piv:= crit;
for ri:= 1 step 1 until n do
begin if crit < eps then goto rank; rl:= r + 1;
if crit < min piv then min piv:= crit; if i ≠ r then
begin det:= - det; norm[i]:= norm[r]; ichow[1, m, r, i, a] end;
if j ≠ r then
begin det:= - det; ichow[1, n, r, j, a] end;
rl[r]:= i; ci[r]:= j; pivot:= a[r, r]; det:= det x pivot;
crit:= 0; if rl < m then
begin for q:= rl step 1 until m do a[r, q]:= a[r, q] / pivot;
for pi:= rl step 1 until n do
begin jcrit:= max elrov[rl, m, p, r, a, a, - a[p, r]];
aid:= abs(a[pi, jcrit]) x norm[p]; if aid > crit then
begin crit:= aid; i:= pi; j:= jcrit end
end
end elimination;
r:= n + 1;
rank:= rmkelma = r - 1; aux[1]:= 1 / min piv; aux[2]:= crit;
aux[3]:= det
end rmkelma;
Description mca 2110

rnkelm:= rank, r, of the n x m matrix M given in array a[1:n, 1:m].
In array aux[0:3] one must give a relative tolerance, aux[0].
The Gaussian-eliminated form of M is overwritten on a, and the pivotal row and column indices are delivered in integer array ri, ci[1:r].
Moreover,
aux[1]:= reciprocal of the minimum absolute value of "pivot / one-norm of the corresponding row of M";
aux[2]:= maximum absolute value of "element of the remaining (n - r) x (m - r) submatrix / one-norm of the corresponding row of M"
if \( r < \min(n, m) \), and otherwise 0;
aux[3]:= determinant of the principal submatrix of order r.
rnkelm uses maxelmrow, ichrow and ichcol (chapter 20).
comment mca 2111;
procedure solelm(a, n, r1, ci, b); value n; integer n; array a, b;
integer array r1, ci;
begin integer r, cir;
    real w;
    sol(a, n, r1, b);
    for r := n step -1 until 1 do
        cir := ci[r];
        if cir + r then
            begin w := b[r]; b[r] := b[cir]; b[cir] := w
                end
    end solelm;
end mca 2112;
integer procedure rnksolelm(a, n, aux, b); value n; integer n;
array a, aux, b;
begin integer rank;
    integer array r1, ci[1:n];
    rank := rnksolelm(a, n, aux, r1, ci);
    if rank = n then solelm(a, n, r1, ci, b)
end rnksolelm;
end mca 2113);
procedure solhom(a, rank, m, k, ci, x); value rank, m, k;
integer rank, m, k; array a, x; integer array ci;
begin integer r, rk;
    real w;
    rk := rank + k;
    for r := rank step -1 until 1 do x[r] := -(matvec(r + 1, rank, r, a, x) + a[ryrk]);
    for r := rank + 1 step 1 until m do x[r] := 0; x[rk] := 1;
    for r := rank step -1 until 1 do
        if cir + r then
            begin w := x[r]; x[r] := x[k]; x[k] := w
                end
end solhom;
Description mca 2111
solem should be called after rnkelm (but only if the rank delivered equals n), and solves the linear system \( Mx = b \), where \( M \) is the \( n \)-th order matrix whose Gaussian-eliminated form and pivotal row and column indices, as produced by rnkelm, are given in array \( a[1:n, 1:n] \) and integer array \( ri, ci[1:n] \), and where \( b \) is the vector given as array \( b[1:n] \).

The solution vector \( x \) is overwritten on \( b \), and sol leaves the other data invariant, so that, after one call of rnkelm, several calls of solelm may follow for solving several systems having the same matrix but different right-hand sides.
solem uses sol (mca 2101) and, indirectly, also matvec (mca 2001).

Description mca 2112
rnksolelm := rank, \( r \), of the \( n \)-th order matrix \( M \) given in array \( a[1:n, 1:n] \).
In array \( aux[0:3] \) one must give a tolerance, \( aux[0] \).
If \( r = n \), the linear system \( Mx = b \) is solved, where \( b \) is the vector given as array \( b[1:n] \), and the solution vector \( x \) is overwritten on \( b \).
The Gaussian-eliminated form of \( M \) is overwritten on \( a \).
Moreover,
aux[1] := reciprocal of the minimum absolute value of "pivot / one-norm of the corresponding row of \( M \)";
aux[2] := 0;
aux[3] := determinant of \( M \).
If, however, \( r < n \), then no solution is calculated and the effect of rnksolelm is the same as that of rnkelm.
rnksolelm uses rnkelm, solelm and, indirectly, also sol (mca 2101), matvec, maxelmrow, ichrow and ichcol (chapter 20).

Description mca 2113
solhom should be called after rnkelm and calculates the \( k \)-th solution vector of the homogeneous system \( Mx = 0 \), where \( M \) is the matrix whose Gaussian-eliminated form (or rather its first rank rows) and pivotal column indices, as produced by rnkelm, are given in array \( a[1: \text{rank}, 1:m] \) and integer array \( ci[1: \text{rank}] \), rank being the rank of \( M \) delivered by rnkelm. The given integer \( k \) must satisfy \( 1 \leq k \leq m - \text{rank} \).
The solution vector is delivered as array \( x[1:m] \).
Calling solhom consecutively with \( k = 1, \ldots, m - \text{rank} \), one obtains a complete set of linearly independent solution vectors of the homogeneous system.
solhom uses matvec (mca 2001).
comment mca 2114;
real procedure invelm(a, n, aux); value n; integer n; array a, aux;
begin integer p, q, r, i, j;
real t, v, det, pivot, co, tol, max;
integer array ri, ci[1:n];
i:= j:= 1; pivot:= abs(a[1,1]);
for p:= 1 step 1 until n do
for q:= 1 step 1 until n do if abs(a[p,q]) > pivot then
begin i:= p; j:= q; pivot:= abs(a[p,q]) end;
max:= pivot; det:= 1; co:= 0; tol:= aux[0] × max;
for r:= 1 step 1 until n do
begin if pivot < tol then
begin det:= 0; aux[1]:= −r + 1; goto exit end;
if i ≠ r then
begin det:= −det; ichrow(1, n, r, i, a) end;
if j ≠ r then
begin det:= −det; ichcol(1, n, r, j, a) end;
ri[r]:= i; ci[r]:= j; w:= a[r,r]; det:= det × w;
a[r,r]:= 1 / w;
for q:= n step −1 until r + 1, r −1 step −1 until 1 do
a[r,q]:= a[r,q] / w; pivot:= 0;
for p:= 1 step 1 until r −1 do
begin t:= −a[p,r]; a[p,r]:= 0; elmrow(1, n, p, r, a, a, t) end;
for p:= r + 1 step 1 until n do
begin t:= −a[p,r]; a[p,r]:= 0; elmrow(1, n, p, r, a, a, t); q:= maxelmrow(r + 1, n, p, r, a, s, t);
if abs(a[p,q]) > pivot then
begin i:= p; j:= q; pivot:= abs(a[p,q]) end
end elimination;
for p:= n step −1 until 1 do
begin for q:= 1 step 1 until n do if abs(a[p,q]) > co then co:=
abs(a[p,q]); r:= ci[p]; if r ≠ p then ichrow(1, n, p, r, a) end;
for q:= n step −1 until 1 do
begin r:= ri[q]; if r ≠ q then ichcol(1, n, q, r, a) end;
aux[1]:= max × co;
exit: invelm:= det
end invelm;
Description mca 2114
Invebm: = determinant of the n-th order matrix M given in
array a[1:n, 1:n]. In array aux[0:1] one must give a relative
 tolerance, aux[0].
The inverse of M is calculated and overwritten on a, and aux[1]: = the
product of the maximum-norm of M and that of its calculated inverse,
this being a condition number of M.
If, however, M is singular (more precisely, if, at some stage, the
absolute value of the pivot is smaller than aux[0] x the maximum-norm
of M), then the calculation is discontinued, invebm:= 0, and aux[1]:= 
minus the rank of M.
Invebm uses elmvrow, maxelmrw, ichrow and ichcol (chapter 20).
Section 212 Band matrices

The procedures of this section solve a system of linear equations whose matrix is in band form and / or calculate the determinant of a band matrix:

detsolbnd solves a system of linear equations and calculates the determinant of the system;
solbnd solves a system of linear equations whose matrix is given in the Gaussian-eliminated form produced by detbnd.

One call of detbnd followed by several calls of solbnd may be used to solve several linear systems having the same matrix, but different right-hand sides.

The method used is Gaussian elimination with stabilizing row interchanges (partial pivoting) [2, p.94] [3, p.204] [7]. Complete pivoting is superfluous if the band width is small (certainly if it is much smaller than the number of binary digits in the number representation).

If the given matrix \( M \) has \( lw \) nonzero codiagonals to the left and \( rw \) to the right of the main diagonal, then the Gaussian elimination yields a unit lower-triangular band matrix \( L \) of Gaussian multipliers having \( lw \) nonzero codiagonals, and an upper-triangular band matrix \( U \) of the resulting equivalent system having \( lw + rw \) nonzero codiagonals.

The Gaussian elimination is performed in \( n \) steps. In the \( k \)-th step, \( k = 1, \ldots, n \), a "pivot" is selected in the \( k \)-th column of the remaining submatrix of order \( n - k + 1 \) (which column contains at most \( lw + 1 \) nonzero elements); then the pivotal row is interchanged with the \( k \)-th row; subsequently, the \( k \)-th "unknown" is eliminated in the last \( n - k \) rows (at most the first \( lw \) rows of which are involved).

The pivot is selected in such a way that its absolute value divided by the Euclidean norm of the corresponding row of \( M \) is maximal. Thus, matrix \( M \) is "equilibrated" in this pivoting strategy such that the rows effectively obtain unit Euclidean norm.

If \( M \) is singular (i.e. if, in some step, the absolute value of the pivot is smaller than a given relative tolerance times the Euclidean norm of the corresponding row of \( M \)), then the elimination is discontinued and 0 is delivered as the determinant value.

The solution \( x \) of the linear system \( Mx = b \) is obtained by carrying out the corresponding eliminations on \( b \), thereby yielding a vector \( y \) (say), and then solving the system \( Ux = y \) (back substitution).
comment mca 2120;
real  procedure detbdn(a, n, lw, rw, aux, m, p); value n, lw, rw;
integer n, lw, rw; integer array p; array a, m, aux;
begin
integer i, j, k, kk, kkl, pk, mk, lk, lw1, f, q, w, w1, w2, iw,
nrw;
real r, s, norm, eps, min, det;
array v[1:n];
if lw; det:= 1; w1:= lw + rw; w:= w1 + 1; w2:= w - 2; iw:= 0;
nrw:= n - rw; lw1:= lw + 1; q:= lw - 1;
for i:= 2 step 1 until lw do
begin
q:= q - 1; lw:= lw + w1;
for j:= iw = q step 1 until iw do a[j]:= 0
end;
norm:= 0; iw:= - w2; q:= nrw + w - 1; j:= rw - 1;
for i:= 1 step 1 until n do
begin
iw:= iw + w1; if i < lw1 then iw:= iw - 1;
ri:= v[i];:= sqrt(vecvec(iw, lw + (if i < lw then j + 1 else
if i > nrw then q - 1 else w1), 0, a, s));
if r > norm then norm:= r
end;
eps:= aux[0]; min:= 1; kk:= - w1; mk:= - lw;
if f > nrw then w2:= w2 + nrw - f;
for k:= 1 step 1 until n do
begin
if f < n then f:= f + 1; ik:= kk:= kk + w; mk:= mk + lw;
s:= abs(a[kk]) / v[k]; pk:= k; kk:= kk + 1;
for i:= k + 1 step 1 until f do
begin
kkl:= ik + w1; mlmk + 1 - k]:= r:= a[ik]; a[ik]:= 0;
ri:= abs(r) / v[i];
if r > s then
begin s:= r; pk:= i end
end;
if s < min then min:= s; if s < eps then
begin detbdn:= 0; aux[1]:= 1 - k; aux[2]:= s; goto end end;
if f > nrw then w2:= w2 - 1; p[k]:= pk; if pk + k then
begin
v[pk]:= v[k]; pk:= pk - k;
ichvec(kkl, kk1 + w2, pk + w1, a); det:= - det;
ri:= mlmk + pk; mlmk + pk:= a[kk]; a[kk]:= r
end
else ri:= a[kk]; det:= r * det; iw:= kk1; lw1:= f - k + mk;
for i:= mk + 1 step 1 until lw1 do
begin
mlmk + 1 := mlmk + 1 / ri
iw:= iw + w1;
elmvec(iv, iw + w2, kk1 - iw, a, a, - s)
end
end;
aux[1]:= 1 / min; detbdn:= det; aux[2]:= min;
end;
detbdn;
detbdn := determinant of the n-th order band matrix M having lw
codiagonals to the left and rv to the right of the main diagonal, and
which is given in array a[1: (lw + rv) x (n - 1) + n] in such a way
that the (i,j)-th element of M is a[(lw + rv) x (i - 1) + j] for
i = 1,..., n and j = max(i, i - lw),..., min(n, i + rv). The values of
the remaining elements of a are irrelevant. In array aux[0:2], one
must give a relative tolerance, aux[0].
The upper-triangular band matrix U resulting from the Gaussian
elimination is delivered in a such that the (i, j)-th element of U is
a[(lw + rv) x (i - 1) + j] for i = 1,..., n and
j = i,..., min(n, i + lw + rv); the matrix L, of Gaussian
multipliers, is delivered in array m[1 : lw x (n - 2) + 1] such that
the i-th multiplier of the j-th step is m[lw x (j - 1) + i - j]
for j = 1,..., n - 1 and i = j + 1,..., min(n, j + lw);
the pivotal indices are delivered in integer array p[1:n].
Moreover,
aux[2] := minimum absolute value of "pivot / Euclidean norm of the
corresponding row of M";
If, however, M is singular, then the Gaussian elimination is
discontinued, detbdn := 0, aux[1] := minus the previous step number,
and aux[2] := absolute value of the last (rejected) pivot / Euclidean
norm of the corresponding row of M.
detbdn uses vecvec, elmvect and ichvec (chapter 20).
comment mca 2121;
procedure solhnd(a, n, lw, rw, m, p, b); value n, lw, rw;
integer n, lw, rw; integer array p; array a, b, m;
begin integer f, i, k, kk, w, w1, w2, shift;
    real s;
    f:= lw; shift:=- lw; w1:= lw - 1;
for k:= 1 step 1 until n do
    begin if f < n then f:= f + 1; shift:= shift + w1; i:= p[k];
        s:= b[i]; if i + k then
            begin b[i]:= b[k]; b[k]:= s end;
        elmvec(k + 1, f, shift, b, m, - s)
    end;
    w1:= lw + rw; w:= w1 + 1; kk:= (n + 1) * w - w1; w2:= - 1;
    shift:= n * w1;
for k:= n step - 1 until 1 do
    begin kk:= kk - w; shift:= shift - w1;
        if w2 < w then w2:= w2 + 1;
        b[k]:= (b[k] - vecvec(k + 1, k + w2, shift, b, a)) / a[kk]
    end
end solhnd;
Description mca 2121
solbnd should be called after detbnd (but only if the determinant is
not zero), and solves the linear system Mx = b, where M is the n-th
order band matrix having lw codiagonals to the left and rw to the
right of the main diagonal, and whose Gaussian-eliminated form and
pivot row indices, as produced by detbnd, are given in
array a[1 : (lw + rw) × (n - 1) + n], m[1 : lw × (n - 2) + 1] and
integer array p[1:n], and where b is the vector given as array b[1:n].
The solution vector x is overwritten on b.
solbnd leaves a, m and p invariant, so that, after one call of detbnd,
several calls of solbnd may follow for solving several systems having
the same band matrix but different right-hand sides.
solbnd uses vecvec and elmvec (chapter 20).
comment mca 2122;
real procedure detolsbnd(a, n, lw, rw, aux, b); value n, lw, rw;
integer n, lw, rw; array a, b, aux;
begins integer i, j, k, kk, kk1, pk, ik, lw1, f, q, w, w1, w2, lw, nrw, shift;
real r, s, norm, eps, min, det; array m(0:1w), v(1:n);
for f:= lw; det:= 1; v:= lw + rw; \( w:= w + 1 \); w2:= w - 2; lw:= 0;
for nrw:= n - lw; lw:= lw + 1; q:= lw - 1; for i:= 2 step 1 until lw do
begins q:= \( q - 1 \); lw:= lw + 1;
for j:= lw - q step 1 until lw do a(j):= 0
end;
for i:= 1 step 1 until n do
begins lw:= lw + 1; if i < lw then lw:= lw - 1;
\( r:= v[1] := \sqrt{v[1]/v[1]} \); if i < lw then j + 1 else
\( r:= v[1]/v[1] \); if i > nrw then \( q - 1 \) else \( w1 \), 0, a, a); if \( r > \) norm then norm:= r
end;
end;
for k:= 1 step 1 until n do
begins if f > nrw then w2:= w2 + nrw - f;
for i:= 1 step 1 until f do
begins if f < n then f:= f + 1; ik:= kk:= kk + w;
\( s:= \text{abs}(a[kk]) / v[k] \); pk:= k; kk1:= kk + 1;
if i:= k + 1 step 1 until f do
begins ik:= ik + 1; \( m[i]:= r:= a[i] \); a[i] := 0;
end;
end;
end;
end;
end;
end detolsbnd;
end detolsbnd;
detsolnbd := determinant of the n-th order band matrix M having lw
codiagonals to the left and rw to the right of the main diagonal, and
which is given in array a[1: (lw + rw) x (n - 1) + n] in such a way
that the (i, j)-th element of M is a[1 + lw + rw x (i - 1) + j] for
i = 1, ..., n and j = max(1, i - lw), ..., min(n, i + rw). The values of
the remaining elements of a are irrelevant. In array aux[0:2], one
must give a relative tolerance, aux[0].
The solution vector x of the linear system Mx = b, where b is the
vector given as array b[1:n], is calculated and overwritten on b.
The upper-triangular band matrix U resulting from the Gaussian
elimination is overwritten on a (in the same way as in detbnd).
Moreover,
aux[2] := minimum absolute value of "pivot / Euclidean norm of the
corresponding row of M";
If, however, M is singular, then the Gaussian elimination is
discontinued, detbnd := 0, aux[1] := minus the previous step number,
aux[2] := absolute value of the last (rejected) pivot / Euclidean norm
of the corresponding row of M, and no solution vector is calculated.
detsolnbd uses vecvec, elmvect and ichvec (chapter 20).
CHAPTER 22

POSITIVE DEFINITE SYMMETRIC LINEAR SYSTEMS AND MATRIX INVERSION.

This chapter contains procedures for solving systems of linear equations and for matrix inversion, provided the matrix is positive definite symmetric. Moreover, section 221 contains procedures for calculating the rank and solving a homogeneous system whose matrix is positive definite symmetric, and section 224 contains procedures for solving linear least squares problems.

In sections 220 and 222 (the latter section deals with band matrices) the ordinary Cholesky method is used and in section 221 Cholesky with pivoting along the main diagonal. The latter method is indispensable for determining the rank of singular positive semidefinite symmetric matrices and for solving homogeneous systems.

Section 224 uses Householder transformations with pivoting [10].

For large order \( n \), the computation time for solving linear systems and for matrix inversion is proportional to \( n^3 \) cubed, and about one half of the time required for the general case (see chapter 21).

Pivoting along the main diagonal requires extra computation time which is proportional to \( n^2 \) squared, and, thus, small with respect to the total time for (very) large \( n \).

The procedures exist in two versions; one version uses the upper triangle of a two-dimensional array for the matrix and the other a one-dimensional array, so that, in the latter case, the memory space occupied by the matrix is cut nearly in half [8]. In the one-dimensional array, the elements of the upper triangle of the matrix are equidistant in the columns but not in the rows, so that special procedures for handling these rows are needed.

In the MC ALGOL 60 system for the X8, a large positive definite symmetric matrix given in a two-dimensional array is inverted faster than one given in a one-dimensional array, because the procedures mca 2000 to 2005 are available in machine-code, but mca 2006 is not. A similar statement holds for solving many (at least \( n/2 \), say) linear systems having the same positive definite symmetric matrix, but different right-hand sides.

The procedures of section 222, for solving linear systems with positive definite symmetric band matrices, save a considerable amount of computation time and memory space, if the band width is much smaller than \( n \).

For large matrices, the computation time is proportional to \( n \times \) the square of the band width.
Section 280 Cholesky decomposition without pivoting

This section contains procedures for solving linear systems and for inverting matrices, provided the matrices are positive definite symmetric:
detsolym2 and detolsym1 solve a system of linear equations and calculate the determinant of the system;
detinvym2 and detinvym1 calculate the determinant and inverse of a matrix;
detsym2 and detsym1 calculate the determinant of a matrix.
The other procedures of this section are to be used in combination with detsym2 or detsym1 for solving a linear system (or several systems having the same matrix but different right-hand sides) or for inverting a matrix.

The method used is Cholesky's square-root method without pivoting [2, p.117] [3, p.229] [4] [5] [6]. If the given symmetric matrix M is positive definite, then the method yields an upper-triangular matrix U, the "Cholesky matrix" of M, such that U'U equals M; moreover, the determinant of M is delivered, calculated as the product of the squares of the diagonal elements of U (and, thus, always positive). The process is completed in n stages, each stage producing a row of U. However, the process is discontinued if at some stage, k, the k-th diagonal element of M minus the sum of the squared elements of the k-th column of U (the sqrt of this quantity being the k-th diagonal element of U) is not positive, meaning that M, perhaps modified by rounding errors, is not positive definite. In that case, instead of the determinant, minus the last stage number k is delivered.

The solution of the linear system U'Ux = b is obtained by solving U'y = b (forward substitution) and Ux = y (back substitution).

The inverse, X, of U'U is obtained from the condition that UX be a lower-triangular matrix whose main diagonal elements are the reciprocals of the diagonal elements of U [4, p. 34-38].

The procedures mca 2200 – 2204 use the upper triangle of a two-dimensional array a[1:n, 1:n] in which the upper triangle of M or U must be given and the upper triangle of U or X is delivered. Thus, a[i, j] is the (i, j)-th element of the matrix only for i ≤ j. The elements a[i, j] for i > j are neither used nor changed.

The procedures mca 2205 – 2209 use a one-dimensional array a[1: (n + 1) x n + 2] in which the upper triangle of M or U must be given and the upper triangle of U or X is delivered in such a way that the (i, j)-th element of the matrix is a[(j - 1) x j + 2 + i] for 1 ≤ i ≤ j ≤ n.
comment mca 2200;
real procedure detsolve2(a, n); value n; integer n; array a;
begin integer k, j; real r, d;
d := 1;
for k := 1 step 1 until n do
begin r := a[k, k] - tamm(1, k - 1, k, k, a, a);
if r <= 0 then
begin detsolve2 := -k; goto end end;
d := r * d; a[k, k] := r := sqrt(r);
for j := k + 1 step 1 until n do a[k, j] := (a[k, j] - tamm(1, k - 1, j, k, a, a)) / r
end;
detsolve2 := d;
end detsolve2;
end mca 2200;

comment mca 2201;
procedure solvem2(a, n, b); value n; integer n; array a, b;
begin integer i;
for i := 1 step 1 until n do b[i] := (b[i] - tamm(1, i - 1, i, a, b)) / a[i, i];
for i := n step -1 until 1 do b[i] := (b[i] - tamm(i + 1, n, i, a, b)) / a[i, i];
end solvem2;
end mca 2201;

comment mca 2202;
real procedure detsolve2(a, n, b); value n; integer n; array a, b;
begin real det;
detsolve2 := det := detsolve2(a, n);
if det > 0 then solvem2(a, n, b)
end detsolve2;
end mca 2202;

comment mca 2203;
procedure invvem2(a, n); value n; integer n; array a;
begin real r; integer i, j, ii; array u[1:n];
for i := n step -1 until 1 do
begin
r := 1 / a[i, i]; ii := i + 1;
for j := ii step 1 until n do u[j] := a[i, j];
for j := n step -1 until ii do a[i, j] := -((tamm(ii, j, i, a, u)) + tamm(ii, i, j, a, u)) * r;
end;
a[i, i] := (r - tamm(ii, n, i, a, u)) * r
end invvem2;
end mca 2203;

comment mca 2204;
real procedure detsolve2(a, n); value n; integer n; array a;
begin real det;
detsolve2 := det := detsolve2(a, n);
if det > 0 then invvem2(a, n)
end detsolve2;
end mca 2204;
**Description mca 2200**

detasm2:= determinant of the n-th order positive definite symmetric matrix M whose upper triangle is given in array a[1:n, 1:n].

Moreover, the Cholesky matrix of M is calculated and overwritten on the upper triangle of M.

If, however, M is not positive definite, the Cholesky decomposition is discontinued and detasm2:= minus the last stage number.

detasm2 uses tammat (mca 2004).

**Description mca 2201**

solsym2 solves the n-th order linear system \( U^T X = b \), where \( U \) is the upper-triangular matrix, given in the upper triangle of array a[1:n, 1:n], and \( b \) is the vector given as array b[1:n].

The solution vector \( x \) is overwritten on b.

If \( U \) is the Cholesky matrix of a positive definite symmetric matrix M, as produced by detasm2, then the calculated solution vector \( x \) is the solution of the linear system \( Mx = b \).

solsym2 leaves the elements of a invariant, so that after one call of detasm2 several calls of solsym2 may follow for solving several linear systems having the same matrix but different right-hand sides.

solsym2 uses matvec and tammat (section 200).

**Description mca 2202**

detdolsym2:= determinant of the n-th order positive definite symmetric matrix M whose upper triangle is given in array a[1:n, 1:n].

Moreover, detdolsym2 solves the linear system \( Mx = b \), where the vector \( b \) is given as array b[1:n]. The solution vector \( x \) is overwritten on b and the Cholesky matrix of M is overwritten on the upper triangle of M. If, however, M is not positive definite, then the Cholesky decomposition is discontinued, no solution is calculated, and detdolsym2:= minus the last stage number.

detdolsym2 uses detasm2, solsym2 and, indirectly, also matvec, tamvec and tammat (section 200).

**Description mca 2203**

invsym2 calculates the inverse, \( X \), of the matrix \( U^T U \), where \( U \) is the upper-triangular matrix given in the upper triangle of array a[1:n, 1:n]. The upper triangle of \( X \) is overwritten on a.

invsym2 uses matvec and tammat (section 200).

**Description mca 2204**

detinvsym2:= determinant of the n-th order positive definite symmetric matrix M whose upper triangle is given in array a[1:n, 1:n].

Moreover, the upper triangle of the inverse of M is calculated and overwritten on a.

If, however, M is not positive definite, the Cholesky decomposition is discontinued and detinvsym2:= minus the last stage number.

detinvsym2 uses detasm2, invsym2 and, indirectly, also matvec, tamvec and tammat (section 200).
comment moa 2205;
real procedure detsym1(a, n); value n; integer n; array a;
begin integer i, j, k, kk, kj, low, up;
real d, r;
d := 1; kk := 0;
for k := 1 step 1 until n do
begin
kk := kk + k; low := kk - k + 1; up := kk - 1;
r := a[kk] - vecvec(low, up, 0, a, a); if r <= 0 then
begin
detsym1 := -k; goto end end;
da := d * r; a[kk] := r := sqrt(r); kj := kk + k;
for j := k + 1 step 1 until n do
begin
a[kj] := (a[kj] - vecvec(low, up, kj - kk, a, a)) / r;
kj := kj + j
end
end;
detsym1 := d;
end;
detsym1;

comment moa 2206;
procedure solsym1(a, n, b); value n; integer n; array a, b;
begin integer i, ii;
ii := 0;
for i := 1 step 1 until n do
begin
ii := ii + 1;
b[i] := (b[i] - vecvec(1, i - 1, ii - i, b, a)) / a[ii]
end;
for i := n step -1 until 1 do
begin
b[i] := (b[i] - vecvec(i + 1, n, ii + i, 0, a, b)) / a[ii];
ii := ii - 1
end
endsolsym1;

comment moa 2207;
real procedure detolsym1(a, n, b); value n; integer n; array a, b;
begin real det;
detsolsym1 := det := detsym1(a, n);
if det > 0 then solsym1(a, n, b)
end detsolsym1;
Description mca 2205
detsyml := determinant of the n-th order positive definite symmetric matrix M whose upper triangle is given in array a[1 : (n + 1) × n : 2].
Moreover, the Cholesky matrix of M is calculated and overwritten on a.
If, however, M is not positive definite, the Cholesky decomposition is discontinued and detsyml := minus the last stage number.
detsyml uses vecvec (mca 2000).

Description mca 2206
solsyml solves the n-th order linear system U'Ux = b, where U is an upper-triangular matrix, given in array a[1 : (n + 1) × n : 2] and b is given as array b[1:n].
The solution vector x is overwritten on b.
If U is the Cholesky matrix of a positive definite symmetric matrix M, as produced by detsyml, then the calculated solution vector x is the solution of the linear system Mx = b.
solsyml leaves the elements of a invariant, so that after one call of detsyml several calls of solsyml may follow for solving several linear systems having the same matrix but different right-hand sides.
solsyml uses vecvec and seqvec (section 200).

Description mca 2207
detsolsyml := determinant of the n-th order positive definite symmetric matrix M whose upper triangle is given in array a[1 : (n + 1) × n : 2].
Moreover, detsolsyml solves the linear system Mx = b, where the vector b is given as array b[1:n].
The solution vector x is overwritten on b and the Cholesky matrix of M is overwritten on a.
If, however, M is not positive definite, then the Cholesky decomposition is discontinued, no solution is calculated, and detsolsyml := minus the last stage number.
detsolsyml uses detsyml, solsyml and, indirectly, also vecvec and seqvec (section 200).
comment mca 2208;
procedure invsym1(a, n); value n; integer n; array a;
begin integer i, ii, il, j, ij, jj;
    real r;
    array u[1:n];
    ii := (n + 1) * n : 2;
    for i := n step - 1 until 1 do
        begin r := 1 / a[ii]; ii := i + i; ij := ii + i;
            for j := i step 1 until n do
                begin u[j] := a[ij]; ij := ij + j end;
            for j := n step - 1 until 1 do
                begin ij := ij - i; ij := ij - j;
                    a[ij] := - (vecvec(ii, i, jj - j, u, a) + seqvec(j + 1, n, jj + j, 0, a, u)) * r
                end;
        a[ii] := (r - seqvec(ii, n, ii + i, 0, a, u)) * r; ii := ii - i
    end invsym1;

comment mca 2209;
real procedure detinsym1(a, n); value n; integer n; array a;
begin real det;
    detinsym1 := det = detsym1(a, n); if det >= 0 then invsym1(a, n)
end detinsym1;
**Description mca 2208**

`invaym1` calculates the inverse, $X$, of the matrix $U'U$, where $U$ is an upper-triangular matrix, given in array $a[1 : (n + 1) \times n : 2]$. The upper triangle of $X$ is overwritten on $a$.

`invaym1` uses vecvec and seqvec (section 200).

**Description mca 2209**

`detinvaym1` is the determinant of the $n$-th order positive definite symmetric matrix $M$ whose upper triangle is given in array $a[1 : (n + 1) \times n : 2]$. Moreover, the upper triangle of the inverse of $M$ is calculated and overwritten on $a$.

If, however, $M$ is not positive definite, then the Cholesky decomposition is discontinued and `detinvaym1` is minus the last stage number.

`detinvaym1` uses `detsym1`, `invaym1` and, indirectly, also `vecvec` and `seqvec` (section 200).
Section 221 Cholesky decomposition with pivoting

This section contains procedures for calculating the rank of matrices, for solving linear systems and for inverting matrices, provided the matrices are positive definite symmetric: rnkaym20 and rnkaym10 calculate the rank of a matrix; rnkaslaym20 and rnkaslaym10 moreover solve a linear system, and rnkinvym20 and rnkinvym10 invert a matrix; solaymmon solves a system of homogeneous linear equations.

The other procedures of this section are to be used in combination with rnkaym20 or rnkaym10 for solving a linear system (or several linear systems having the same matrix but different right-hand sides) or for inverting a matrix.

The method used is Cholesky's square-root method (see section 220) with pivoting along the main diagonal. If the given symmetric matrix M is positive semidefinite, then the method yields an upper-triangular matrix U, the "pivoting-Cholesky matrix" of M, such that the product U'U equals M with permuted rows and columns. If the rank, r, of M is smaller than the order n, then the last n - r rows of U (nearly) vanish.

The process is performed in at most n stages. At the k-th stage, the k-th row and column are interchanged with the p[k]-th row and column (thus preserving the symmetry), the k-th "pivotal index" p[k] being chosen in such a way that the diagonal elements of U turn out to be monotonically nonincreasing, and then the k-th row of U is produced. The process is terminated if at some stage, k, the k-th pivot (i.e. the maximum diagonal element of the remaining submatrix of order n - k + 1, the sqrt of this quantity being the k-th diagonal element of U) is negative or smaller than some tolerance, viz. a given relative tolerance times the maximum diagonal element of M (the maximum diagonal element being equal to the maximum-norm of M, if M is positive semidefinite). If no such k exists, then n is delivered as the rank of M. Otherwise, the maximum absolute value of the elements of the remaining submatrix of order n - k + 1 is calculated. If this is smaller than twice the tolerance, then k - 1 is delivered as the rank of the positive semidefinite matrix M; otherwise, M is apparently not positive semidefinite, and, instead of the rank, the value - k is delivered.

The solution of the linear system Mx = b is obtained by first interchanging the elements of b in the same way as the rows (and columns) of M, subsequently performing forward and back substitution (see section 220) and finally carrying out the same interchanges in reverse order ("reverse correspondence") on the elements of the solution vector.

The inverse of U'U is obtained by calling invaym2 or invaym1 (section 220). Then, the inverse of M is obtained by interchanging the rows and columns in reverse correspondence with the interchanges of the rows and columns of M.
A homogeneous linear system $Mx = 0$, where $M$ is an $n$-th order positive semidefinite symmetric matrix of rank $r$, is obtained as follows. Let $V$ be the $r$-th order upper-triangular matrix consisting of the first $r$ columns of the pivot-Cholesky matrix $U$ of $M$ and $W$ the $r \times (n - r)$ matrix consisting of the remaining part of the first $r$ rows of $U$ (the other rows of $U$ are negligible). First, the system $VY = -W$ is solved (back substitution). Then $Y$ and the $(n - r)$-th order identity matrix are combined to form a single $r \times n$ matrix; its rows are then interchanged in reverse correspondence with the interchanges of the rows and columns of $M$. The columns of the resulting matrix form a complete linearly independent set of solution vectors of the homogeneous system.

The procedures mca 2210 - 2214 and 221a use the upper triangle of a two-dimensional array $a[1:n, 1:n]$ in which the upper triangle of $M$ or $U$ must be given and the upper triangle of $U$ or $X$ is delivered. Thus $a[i, j]$ is the $(i, j)$-th element of the matrix only for $i \leq j$. The elements $a[i, j]$ for $i > j$ are neither used nor changed. (Only mca 221a delivers a rectangular matrix involving some elements in the lower triangle of $a$ as well.)

The procedures mca 2215 - 2219 use a one-dimensional array $a[1 : (n + 1) \times n ]$ in which the upper triangle of $M$ or $U$ must be given and the upper triangle of $U$ or $X$ is delivered, in such a way that the $(i, j)$-th element of the matrix is $a[(j - 1) \times j + 2 + i]$ for $1 \leq i \leq j \leq n$. 
comment mca 2210;
integer procedure mkeys20(a, n, p, aux); value n; integer n;
begin integer k, i, j, pk; real w, max, m, t, r, d, norm, epsnorm;
d:= 1; norm:= 0;
for i:= 1 step 1 until n do if a[i,i] > norm then norm:= a[i,i];
epsnorm:= aux[0] X norm; aux[1]:= norm; m:= 0;
for k:= 1 step 1 until n do
begin max:= epsnorm;
for j:= k step 1 until n do if a[j,j] > max then
begin max:= a[j,j]; pk:= j end;
if max < epsnorm then
begin for i:= k step 1 until n do
begin t:= abs(a[i,k]); if t > m then m:= t;
for j:= 1 + 1 step 1 until n do
begin t:= a[i,j];= a[i,j] - tanmat(1, k - 1, i, j, a, a); t:= abs(t); if t > m then m:= t
end;
end;
goto end end;
p[k]:= pk; d:= d X max; if pk + 1 then
begin ichool(1, k - 1, k, pk, a);
ichrowcol(k + 1, pk - 1, k, pk, a);
ichrow(pk + 1, n, k, pk, a); a[pk,pk]:= a[k,k] end;
a[k,k]:= r:= sqrt(max);
for j:= k + 1 step 1 until n do
begin w:= a[k,j];= (a[k,j] - tanmat(1, k - 1, j, a, a))/
r; a[j,j]:= a[j,j] - v X w
end;
k:= n + 1;
end; aux[2]:= m; aux[3]:= d;
mkeys20:= if m <= 2 X epsnorm then k - 1 else - k
end mkeys20;

comment mca 2211;
procedure solsym20(a, n, p, b); value n; integer n; integer array p;
array a, b;
begin integer i, pi; real r;
for i:= 1 step 1 until n do
begin r:= b[i]; pi:= p[i];
b[i]:= (b[pi] - tanvec(1, i - 1, i, a, b))/a[i,i];
if pi + 1 then b[pi]:= r end;
for i:= n step - 1 until 1 do
begin pi:= p[i]; if pi + 1 then
begin r:= b[i]; b[i]:= b[pi]; b[pi]:= r end
end;
solsym20;
**Description mca 2210**

\texttt{rnkaym20:=} \texttt{rank, r, of the n-th order positive semi-definite symmetric matrix M whose upper triangle is given in array a[1:n, 1:n]. In array aux[0:3], one must give a relative tolerance, aux[0].}

The pivot-Cholesky matrix of M is overwritten on the upper triangle of a, and the pivotal indices are delivered in integer array \( p[1:r] \). Moreover,

\begin{align*}
\text{aux[1]} & := \text{the maximum diagonal element of M;} \\
\text{aux[2]} & := \text{the maximum absolute value of the elements of the remaining submatrix of order } n - r \text{ if } r < n, \text{ and otherwise 0;} \\
\text{aux[3]} & := \text{determinant of the principal submatrix of order } r.
\end{align*}

However, if M is not positive semidefinite, then \texttt{rnkaym20:=} minus the last stage number.

\texttt{rnkaym20 uses tammat, ichool, ichrow and ichrovcol (chapter 20).}

**Description mca 2211**

\texttt{solasym20} should be called after \texttt{rnkaym20} (but only if the rank equals \( n \)), and solves the n-th order linear system \( Mx = b \), where \( M \) is the positive definite symmetric matrix whose pivot-Cholesky matrix and pivotal indices, as produced by \texttt{rnkaym20}, are given in the upper triangle of array \( a[1:n, 1:n] \) and in integer array \( p[1:n] \), and where \( b \) is the vector given as array \( b[1:n] \).

The solution vector \( x \) is overwritten on \( b \).

\texttt{solasym20} leaves \( a \) and \( p \) invariant, so that after one call of \texttt{rnkaym20} several calls of \texttt{solasym20} may follow for solving several linear systems having the same matrix but different right-hand sides.

\texttt{solasym20 uses matvec and tamvec (section 200).}
comment mca 2212;
integer procedure mksolaym20(a, n, b, aux); value n; integer n;
array a, b, aux;
begin integer rank;
    integer array p[1:n];
    mksolaym20 := rank := mksolaym20(a, n, p, aux);
    if rank = n then solaym20(a, n, p, b)
end mksolaym20;

comment mca 2213;
procedure invaym20(a, n, p); value n; integer n; integer array p;
array a;
begin integer i, j, pi;
    real r;
    invaym20(a, n);
    for i := n step - 1 until 1 do
        begin pi := p[1];
            if pi + 1 then
                begin ichcol(1, i - 1, i, pi, a);
                    ichrowcol(i + 1, pi - 1, i, pi, a);
                    ichrow(pi + 1, n, i, pi, a); r := a[i, i];
                    a[i, i] := a[pi, pi]; a[pi, pi] := r
                end
        end
end invaym20;

comment mca 2214;
integer procedure mkinvaym20(a, n, aux); value n; integer n;
array a, aux;
begin integer rank;
    integer array p[1:n];
    mkinvaym20 := rank := mkinvaym20(a, n, p, aux);
    if rank = n then invaym20(a, n, p)
end mkinvaym20;
Description mca 2212

\texttt{rnksolym20}:= \texttt{rank}, \texttt{r}, of the \texttt{n}-th order positive semidefinite symmetric matrix \texttt{M} whose upper triangle is given in array \texttt{a[1:n, 1:n]}. In array \texttt{aux[0:3]} one must give a relative tolerance, \texttt{aux[0]}. If \texttt{r} = \texttt{n}, the linear system \texttt{Mx} = \texttt{b} is solved, where \texttt{b} is the vector given as array \texttt{b[1:n]}, and the solution vector \texttt{x} is overwritten on \texttt{b}. The pivot-Cholesky matrix of \texttt{M} is overwritten on the upper triangle of \texttt{a}. Moreover, \texttt{aux[1]}:= the maximum diagonal element of \texttt{M}; \texttt{aux[2]}:= 0; \texttt{aux[3]}:= determinant of \texttt{M}.

However, if \texttt{1} \leq \texttt{r} \leq \texttt{n} (\texttt{M} positive semidefinite) or \texttt{r} \leq \texttt{0} (\texttt{M} not positive semidefinite), then no solution vector is calculated and the results of \texttt{rnksolym20} are the same as those of \texttt{rnksym20}. \texttt{rnksolym20} uses \texttt{rnksym20}, \texttt{solym20} and, indirectly, also \texttt{matvec}, \texttt{tamvec}, \texttt{tammat}, \texttt{ichol}, \texttt{ichrow} and \texttt{ichrowcol} (chapter 20).

Description mca 2213

\texttt{invym20} should be called after \texttt{rnksym20} (but only if the rank equals \texttt{n}), and calculates the inverse, \texttt{X}, of the \texttt{n}-th order positive definite symmetric matrix \texttt{M} whose pivot-Cholesky matrix and pivotal indices, as produced by \texttt{rnksym20}, are given in the upper triangle of array \texttt{a[1:n, 1:n]} and in integer array \texttt{p[1:n]}. The upper triangle of \texttt{X} is overwritten on \texttt{a}.

\texttt{invym20} uses \texttt{invym2} (mca 2203), \texttt{ichol}, \texttt{ichrow}, \texttt{ichrowcol} (section 202) and, indirectly, also \texttt{matvec} and \texttt{tamvec} (section 200).

Description mca 2214

\texttt{rnkinsym20}:= \texttt{rank}, \texttt{r}, of the \texttt{n}-th order positive semi-definite symmetric matrix \texttt{M} whose upper triangle is given in array \texttt{a[1:n, 1:n]}. In array \texttt{aux[0:3]} one must give a relative tolerance, \texttt{aux[0]}. If \texttt{r} = \texttt{n}, the upper triangle of the inverse of \texttt{M} is calculated and overwritten on \texttt{a}.

Moreover, \texttt{aux[1]}:= the maximum diagonal element of \texttt{M}; \texttt{aux[2]}:= 0; \texttt{aux[3]}:= determinant of \texttt{M}.

However, if \texttt{1} \leq \texttt{r} \leq \texttt{n} (\texttt{M} positive semidefinite) or \texttt{r} \leq \texttt{0} (\texttt{M} not positive semidefinite), then no inverse is calculated and the results of \texttt{rnkinsym20} are the same as those of \texttt{rnksym20}. \texttt{rnkinsym20} uses \texttt{rnksym20}, \texttt{invym20} and, indirectly, also \texttt{invym2} (mca 2203), \texttt{matvec}, \texttt{tamvec}, \texttt{tammat}, \texttt{ichol}, \texttt{ichrow} and \texttt{ichrowcol} (chapter 20).
comment mos 221a;
integer procedure solayhom20(a, n, aux); value n; integer n;
array a, aux;
begin integer i, p, j, rank;
real r;
integer array p[1:n];
solayhom20:= rank:= rankhom20(a, n, p, aux); if rank > 0 then
begin for i:= rank + 1 step 1 until n do
  for j:= rank + 1 step 1 until n do a[i,j]:= if i = j then 1
  else 0;
  for i:= rank step - 1 until 1 do begin r:= - a[i,i];
    for j:= rank + 1 step 1 until n do a[i,j]:= (a[i,j] +
    matmat(i + 1, n, i, j, a, p)) / r
    end;
  for i:= rank step - 1 until 1 do begin p[j]:= p[j]; if p[j] ≠ j then ichrow(rank + 1, n, j, p, a)
    end
end solayhom20;
Description mca 221a
solsymphom20 solves the homogeneous linear system whose \( n \)–th order positive semidefinite symmetric matrix \( M \) is given in the upper triangle of array \( a[1:n, 1:n] \).
In array \( aux[0:3] \) one must give a relative tolerance, \( aux[0] \).
\( solsymphom20 : = \) rank, \( r \), of \( M \), calculated by means of \( rnksym20 \).
Subsequently, a complete set of \( n - r \) linearly independent solution vectors of the homogeneous linear system is calculated and delivered in the last \( n - r \) columns of \( a \). The first \( r \) columns of the pivot-
Cholesky matrix of \( M \) are delivered in the first \( r \) columns of \( a \).
Moreover, the same results are delivered in \( aux \) as by \( rnksym20 \).
However, if \( rnksym20 \) delivers a negative (integral) value, indicating that \( M \) is not positive semidefinite, no solution of the homogeneous system is calculated and only the results of \( rnksym20 \) are delivered.
\( solsymphom20 \) uses \( rnksym20 \), \( matmat \), \( ichrow \) and, indirectly, also \( tammat \), \( ichcol \) and \( ichrowcol \) (chapter 20).
\begin{verbatim}
comment mca 2215;
integer procedure rnsyml0(a, n, p, aux); value n; integer n;
integer array p; array a, aux;
begin integer k, pk, kk, kj, pp, i, j, jj, t, low, up;
    real norm, epanorm, m, max, d, r, w;
    d := 1; norm := 0; kk := 0;
    for k := 1 step 1 until n do
        begin
            kk := kk + kj;
            if a[kk] > norm then norm := a[kk];
            epanorm := aux[0] * norm;
            aux[1] := norm;
            m := 0; kk := 0;
        end;
        for k := 1 step 1 until n do
            begin
                max := epanorm;
                t := kk;
                for j := k step 1 until n do
                    begin
                        t := t + j;
                        if a[t] > max then
                            begin
                                max := a[t];
                                pk := j;
                                pp := t;
                            end;
                    end;
            end;
        if max < epanorm then
            begin
                for i := 1 step 1 until n do
                    begin
                        kk := kk + i;
                        low := kk - i + 1;
                        up := low + k - 2;
                        r := abs(a[kk]);
                        if r > m then
                            begin
                                m := r;
                                kj := kk + i;
                            end;
                        end;
        end;
        goto end;
    end;
end;
k := kk + k;
low := kk - k + 1;
up := kk - 1;
p[k] := pk;
begin
    d := d * max;
    if pk = k then
        begin
            ichvec(low, up, pp - pk - kk + k, a);
            ichseqvec(k + 1, pk - 1, kk + k, pp - pk, a);
            ichseq(pk + 1, n, pp + k, pk - k, a);
            a[pp] := a[kk];
        end;

    a[kkk] := r := sqrt(max);
    kj := kk + k;
    for j := k + 1 step 1 until n do
        begin
            w := a[jj] := (a[jj] - vecvec(low, up, kj - kk, a, a)) / r;
            jj := jj + j;
            a[jj] := a[jj] - w * w;
            kj := kj + j;
        end;
    k := n + 1;
end;
aux[1] := m;
rnsyml0 := if m < 2 * epanorm then k - 1 else k;
end rnsyml0;
\end{verbatim}
Description vec 2215

rnksym10:= rank, r, of the n-th order positive semidefinite symmetric matrix M whose upper triangle is given in array a[l : ( n + 1) x n ; 2]. In array aux[0:3] one must give a relative tolerance, aux[0].

The pivot-Cholesky matrix of M is overwritten on a and the pivotal indices are delivered in integer array p[1:r]. Moreover,

aux[1]:= the maximum diagonal element of M;
aux[2]:= the maximum absolute value of the elements of the remaining submatrix of order n - r if r < n, and otherwise 0;
aux[3]:= determinant of the principal submatrix of order r.

However, if M is not positive semidefinite, then rnksym10:= minus the last stage number.

rnksym10 uses vecve, ichve, ichseve and ichseq (chapter 20).
comment mca 2216;
procedure solsym10(a, n, p, b); value n; integer n; array a, b;
  integer array p;
begin integer i, ii, pi; real s;
  ii := 0;
  for i := 1 step 1 until n do
    begin a := b[i]; pi := p[i]; ii := ii + i;
      b[i] := (b[pi] - vecvec(1, i - 1, ii - 1, b, a)) / a[ii];
      if pi + 1 then b[pi] := s
    end;
  for i := n step -1 until 1 do
    begin b[i] := (b[i] - vecvec(1 + 1, n, ii + 1, 0, a, b)) / a[ii];
      ii := ii - 1
    end;
end solsym10;

comment mca 2217;
integer procedure rnsolsym10(a, n, b, aux); value n; integer n;
array a, b, aux;
begin integer rank; integer array p[1:n];
  rnsolsym10 := rank := rnsym10(a, n, p, aux);
  if rank = n then solsym10(a, n, p, b)
end rnsolsym10;

comment mca 2218;
procedure invsym10(a, n, p); value n; integer n; integer array p;
array a;
begin integer i, ii, pi, pp; real r;
  invsym10(a, n); ii := (n + 1) * n : 2;
  for i := n step -1 until 1 do
    begin pi := p[i]; if pi + 1 then
      begin pp := (pi + 1) * pi : 2;
        ichvec(ii - i + 1, ii - 1, pp - pi - ii + 1, a);
        ichseq(ii + 1, ii - 1, pp - pi, a);
        ichseq(pi + 1, n, pp + 1, pi - i, a);
        r := a[ii];
        a[ii] := a[pp]; a[pp] := r
      end;
      ii := ii - 1
    end
end invsym10;

comment mca 2219;
integer procedure rnkinvsym10(a, n, aux); value n; integer n;
array a, aux;
begin integer rank; integer array p[1:n];
  rnkinvsym10 := rank := rnkkey10(a, n, p, aux);
  if rank = n then invsym10(a, n, p)
end rnkinvsym10;
Description mca 2216

solsym10 should be called after rnkasym10 (but only if the rank equals n), and solves the n-th order linear system \( Mx = b \), where \( M \) is the positive definite symmetric matrix whose pivot-Cholesky matrix and pivotal indices, as produced by rnkasym10, are given in array \( a[1 : (n + 1) \times n : 2] \) and in integer array \( p[1:n] \), and where \( b \) is the vector given as array \( b[1:n] \).

The solution vector \( x \) is overwritten on \( b \), solsym10 leaves \( a \) and \( p \) invariant, so that after one call of rnkasym10 several calls of solsym10 may follow for solving several linear systems having the same matrix but different right-hand sides. solsym10 uses vecvec and seqvec (section 200).

Description mca 2217

rnksolsym10 := rank, \( r \), of the n-th order positive semidefinite symmetric matrix \( M \) whose upper triangle is given in array \( a[1 : (n + 1) \times n : 2] \).

In array aux[0:3] one must give a relative tolerance, aux[0].

If \( r = n \), the linear system \( Mx = b \) is solved, where \( b \) is the vector given as array \( b[1:n] \), and the solution vector \( x \) is overwritten on \( b \).

The pivot-Cholesky matrix of \( M \) is overwritten on \( a \).


aux[3] := determinant of \( M \).

However, if \( 1 \leq r < n \) \( (M \) positive semidefinite) or \( r \leq 0 \) \( (M \) not positive semidefinite), then no solution vector is calculated and the results of rnksolsym10 are the same as those of rnkasym10.

rnksolsym10 uses rnkasym10, solsym10 and, indirectly, also vecvec, seqvec, ichvec, ichseqvec and ichseq (chapter 20).

Description mca 2218

invasym10 should be called after rnkasym10 (but only if the rank equals n), and then calculates the inverse, \( X \), of the n-th order positive definite symmetric matrix \( M \) whose pivot-Cholesky matrix and pivotal indices, as produced by rnkasym10, are given in array \( a[1 : (n + 1) \times n : 2] \) and in integer array \( p[1:n] \).

The upper triangle of \( X \) is overwritten on \( a \).

invasym10 uses invasym1 (mca 2208), ichvec, ichseqvec, ichseq (section 202) and, indirectly, also vecvec and seqvec (section 200).

Description mca 2219

rnkinvasym10 := rank, \( r \), of the n-th order positive semidefinite symmetric matrix \( M \) whose upper triangle is given in array \( a[1 : (n + 1) \times n : 2] \).

In array aux[0:3] one must give a relative tolerance, aux[0].

If \( r = n \), the upper triangle of the inverse of \( M \) is calculated and overwritten on \( a \).


aux[3] := determinant of \( M \).

However, if \( 1 \leq r < n \) \( (M \) positive semidefinite) or \( r \leq 0 \) \( (M \) not positive semidefinite), then no inverse is calculated and the results of rnkinvasym10 are the same as those of rnkasym10.

rnkinvasym10 uses rnkasym10, invasym1 and, indirectly, also invasym1 (mca 2208), vecvec, seqvec, ichvec, ichseqvec and ichseq (chapter 20).
Section 222 Cholesky decomposition for band matrices

The procedures of this section solve a system of linear equations and/or calculate the determinant of a matrix, provided the matrix is a positive definite symmetric band matrix:

detbsymbnd solves a system of linear equations and calculates the determinant of the system;
debsymbnd calculates the determinant of a matrix;
solbsymbnd solves a linear system whose matrix is given in the Cholesky-decomposed form produced by detbsymbnd.

One call of detbsymbnd followed by several calls of solbsymbnd may be used to solve several linear systems having the same matrix but different right-hand sides.

The method used is Cholesky's square-root method without pivoting (see section 220 and [9]). If the given symmetric band matrix $M$ is positive definite, then the method yields an upper-triangular band matrix $U$, the "Cholesky matrix" of $M$, such that $U^TU$ equals $M$; moreover, the determinant of $M$ is delivered, calculated as the product of the squares of the diagonal elements of $U$ (and, thus, always positive).

The number of nonzero diagonals of $U$ is the same as that of the upper triangle of $M$. The process is completed in $n$ stages, each stage producing a row of $U$. However, the process is discontinued if at some stage, $k$, the $k$-th diagonal element of $M$ minus the sum of the squared elements of the $k$-th column of $U$ (the sqrt of this quantity being the $k$-th diagonal element of $U$) is not positive, meaning that $M$, perhaps modified by rounding errors, is not positive definite. In that case, instead of the determinant, minus the last stage number $k$ is delivered.

The solution of the linear system $U'Ux = b$ is obtained by solving $U'y = b$ (forward substitution) and $Ux = y$ (back substitution).

The procedures of this section use a one-dimensional array $a[1 : (n - 1) \times w + n]$ for the upper triangle of $M$ and $U$, where $n$ is the order of the matrix and $w$ the number of non-zero codiagonals above the main diagonal; the $(i, j)$-th element of matrix $M$ or $U$ is $a[(j - 1) \times w + i]$ for $j = 1, \ldots, n$ and $i = \max(1, j - w), \ldots, j$; the other elements of $a$ are neither used nor changed.
comment mca 2220;
real procedure detsymbnd(a, n, w); value n, w; integer n, w; array a;
begin integer j, k, jmax, kk, kj, w1, start;
real r, det;
det:= 1; jmax:= w; w1:= w + 1; kk:= -w;
for k:= 1 step 1 until n do
begin if k + w > n then jmax:= jmax - 1; kk:= kk + w1;
start:= kk - k + 1;
  r:= a[kk] - vecvec(if k < w1 then start else kk - w, kk - 1, 0, a, a); if r <= 0 then
begin detsymbnd:= -k; goto end end;
det:= r * det; a[kk]:= r:= sqrt(r); kj:= kk;
for j:= 1 step 1 until jmax do
begin kj:= kj + w1;
a[kj]:= (a[kj] - vecvec(if k + j < w1 then start else kk - w + j, kk - 1, kj - kk, a, a)) / r
end
end;
detsymbnd:= det;
end detsymbnd;

comment mca 2221;
procedure solsymbnd(a, n, w, b); value n, w; integer n, w; array a, b;
begin integer i, k, imax, kk, w1;
kki:= w; w1:= w + 1;
for k:= 1 step 1 until n do
begin kk:= kk + w1;
b[k]:= (b[k] - vecvec(if k < w1 then 1 else k - w, k - 1, kk - k, b, a)) / a[kk]
end;
imax:= -1;
for k:= n step -1 until 1 do
begin if imax < w then imax:= imax + 1;
b[k]:= (b[k] - scaprd(kk + w, w, k + 1, 1, imax, a, b)) / a[kk]; kk:= kk - w1
end
endsolsymbnd;

comment mca 2222;
real procedure detsolsymbnd(a, n, w, b); value n, w; integer n, w; array a, b;
begin real det;
detsolsymbnd:= det:= detsymbnd(a, n, w);
if det > 0 then solsymbnd(a, n, w, b)
end detsolsymbnd;
Detsymbnd: = determinant of the n-th order positive definite symmetric band matrix M having w codiagonals on each side of the main diagonal, and whose upper triangle is given in array \texttt{a[1 : (n - 1) \times w + n]}. Moreover, the Cholesky matrix of M is calculated and delivered in \texttt{a}. If, however, M is not positive definite, then the Cholesky decomposition is discontinued, and detsymbnd: = minus the last stage number. detsymbnd uses vecvec (mca 2000).

Solsymbnd solves the n-th order linear system \texttt{U'Ux = b}, where \texttt{b} is the vector given as \texttt{array b[1 : n]}, and \texttt{U} is the upper-triangular band matrix having w codiagonals, and which is given in array \texttt{a[1 : (n - 1) \times w + n]}. The solution vector \texttt{x} is overwritten on \texttt{b}.

If \texttt{U} is the Cholesky matrix of a positive definite symmetric band matrix \texttt{M}, as produced by detsymbnd, then the calculated solution vector \texttt{x} is the solution of the linear system \texttt{Mx = b}. Solsymbnd leaves the elements of \texttt{a} invariant, so that after one call of detsymbnd several calls of solsymbnd may follow for solving several linear systems having the same matrix but different right-hand sides. solsymbnd uses vecvec and scaprd1 (section 200).

Detacolsymbnd: = determinant of the n-th order positive definite symmetric band matrix M having w codiagonals on each side of the main diagonal, and whose upper triangle is given in array \texttt{a[1 : (n - 1) \times w + n]}. Moreover, the solution vector \texttt{x} of the linear system \texttt{Mx = b}, where \texttt{b} is the vector given as \texttt{array b[1:n]}, is calculated and overwritten on \texttt{b}, and the Cholesky matrix of \texttt{M} is delivered in \texttt{a}. If, however, \texttt{M} is not positive definite, then the Cholesky decomposition is discontinued, no solution is calculated, and detacolsymbnd: = minus the last stage number. detacolsymbnd uses detsymbnd, solsymbnd and, indirectly, also vecvec and scaprd1 (section 200).
Section 224 Least-squares problems

This section contains procedures for solving linear least-squares problems:

- **linalg**: calculates the solution \( x \) of a least-squares problem \( Mx = b \) and, moreover, the main diagonal of the inverse of the product \( MM^T \.

- **linalg**: performs the Householder triangularization of \( M \) and calculates its rank.

- **linalg** and **linalg**: are to be used in combination with **linalg** for solving a linear least-squares problem (or several problems having the same matrix \( M \) but different right-hand sides) or for calculating the main diagonal of the inverse of \( MM^T \).

Apart from some changes and adaptations to our vector procedures, **linalg** and **linalg** have been derived from [10]. However, our procedures do not perform iterations for improving the solution; these iterations would be of limited value, as is pointed out in [11].

The method is Householder triangularization with column interchanges. Let \( M \) have \( n \) rows and \( m \) columns; **linalg** produces an \( n \times n \) orthogonal matrix \( Q \) and an \( n \times m \) upper-triangular matrix \( R \) such that \( R = QM \) with permuted columns. Matrix \( Q \) is the product of at most \( m \)

orthogonal symmetric \( n \times n \) matrix "Householder matrices", which are of the form \( I - sw^T \), where \( I \) is the identity matrix, \( w \) a column vector and \( s \) a scalar. Matrix \( M \) is reduced to \( R \) in (at most) \( m \) stages. In the \( k \)th stage, the desired zeroes are introduced in the \( k \)th column of the matrix as follows: first the "pivotal" column, i.e. the column having maximum Euclidean norm, is selected from the remaining \( (n - k + 1) \times (m - k + 1) \) submatrix, and the pivotal and the \( k \)th columns are interchanged; then the \( k \)th Householder matrix is calculated and postmultiplied by the remaining submatrix.

The \( k \)th Householder matrix is chosen such that this postmultiplication introduces the desired zeroes in the \( k \)th column, and the first \( k - 1 \) elements of \( w \) are zero.

If at some stage \( k \) the Euclidean norm of the pivotal column is smaller than some tolerance, viz. a given relative tolerance times the maximum of the Euclidean norms of the columns of \( M \), then the process is discontinued, and \( k - 1 \) is delivered as the rank of \( M \); otherwise, the rank equals \( m \).

In **linalg**, the least-squares solution \( x \) of the problem \( Mx = b \) is obtained by first calculating \( y = Q^T \), then solving the triangular system consisting of the first \( m \) equations of \( Rx = y \) (back substitution), and finally interchanging the elements of \( x \) in "reverse correspondence" with the interchanges of the columns of \( M \), i.e. the same interchanges are carried out in reverse order. As by-product the last \( n - m \) elements of \( y \) are delivered; the sum of the squares of these elements is approximately equal to the square of the Euclidean norm of the residue vector \( Mx - b \).

In **linalg**, the main diagonal of \( MM^T \) is obtained by calculating the inverse of \( R \), from this the main diagonal of the inverse of \( R^T R \), and then interchanging the calculated diagonal elements in reverse correspondence with the interchanges of the columns of \( M \).
INTEGER procedure laqdec(a, n, m, aux, aid, ci); value n, m;
begin integer J, k, k piv;
      real beta, sigmat, norm, w, eps, akk, aidk;
array sum[1:m];
norm:= 0; laqdec:= m;
for k:= 1 step 1 until m do
begin w:= sum[k]:= tamm(k, n, k, a, a);
   if v > norm then norm:= w
end;
end:= aux[1]:= sqrt(norm); eps:= aux[0] * w;
for k:= 1 step 1 until m do
begin sigma:= sum[k]; k piv:= k;
   for j:= k + 1 step 1 until m do if sum[j] > sigma then
      begin sigma:= sum[j]; k piv:= j end;
   if k piv = k then
begin sum[k piv]:= sum[k]; ichool(1, n, k, k piv, a) end;
   ci[k]:= k piv; akk:= a[k,k]; sigmat:= tamm(k, n, k, k, a, a);
   w:= sqrt(sigma); aidk:= aid[k]:= if akk < 0 then w else - w;
   if v < eps then
begin laqdec:= k - 1; goto end dec end;
   beta:= 1 / (sigma - akk * aidk); ak[k,k]:= akk - aidk;
   for j:= k + 1 step 1 until m do
begin elmcol(k, n, j, k, a, a, - beta * tamm(k, n, k, j, a, a)); sum[j]:= sum[j] - a[k,j] * 2
end;
end dec:= aux[2]:= w
end laqdec;
Description mca 2240

lsqdec := rank, r, of the n x m matrix M given in array a[1:n, 1:m]. In array aux[0:2] one must give a relative tolerance, aux[0]. The pivotal column indices are delivered in integer array ci[1 : r], the (r first) diagonal elements of the upper-triangular matrix R in array aid[1 : r], and the other elements of the upper triangle of R in array a, together with the vectors w of the Householder matrices. Moreover, aux[1] := the maximum Euclidean norm of the columns of M, aux[2] := the absolute value of the r-th diagonal element of R.

lsqdec uses tammat, elmool and ichool (chapter 20).
comment mca 2241;
procedure laqso1(a, n, m, aid, ci, b); value n, m; integer n, m;
array a, aid, b; integer array ci;
begin integer k, cik;

real w;

for k := 1 step 1 until m do elmvecol(k, n, k, b, a, tamvec(k, 
 n, k, a, b)) / (aid[k] + a[k,k]);
for k := m step -1 until 1 do b[k] := (b[k] - matvec(k + 1, m, k, 
 a, b)) / aid[k];
for k := m step -1 until 1 do
begin cik := ci[k]; if cik = k then
    begin w := b[k]; b[k] := b[cik]; b[cik] := w end
end
end laqso1;

comment mca 2242;
procedure laqdglnv(a, m, aid, ci, diag); value m; integer m;
array a, aid, diag; integer array ci;
begin integer j, k, cik;
real w;

for k := 1 step 1 until m do
begin diag[k] := 1 / aid[k];
    for j := k + 1 step 1 until m do diag[j] := - tamvec(k, j - 1, 
 j, a, diag) / aid[j]; diag[k] := vecvec(k, m, 0, diag, diag)
end;

for k := m step -1 until 1 do
begin cik := ci[k]; if cik = k then
    begin w := diag[k]; diag[k] := diag[cik]; diag[cik] := w end
end
end laqdglnv;

comment mca 2243;
integer procedure laqdescsl(a, n, m, aux, diag, b); value n, m;
integer n, m array a, aux, diag, b;
begin integer rank;
array aid[1:m];
integer array ci[1:m];
rank := laqdescsl(a, n, m, aux, aid, ci);
if rank = m then
begin laqdglnv(a, m, aid, ci, diag); laqso1(a, n, m, aid, ci, b)
end
end laqdescsl;
Description mca 2241

lqsdol should be called after lqsdolc (but only if the rank equals m), and calculates the least-squares solution x of Mx - b, where b is the vector given as array b[1:n], and M is the n x m matrix whose Householder-triangularised form R, with the vectors w of the Householder matrices and the pivotal indices, as produced by lqsdolc, are given in array a[1:n, 1:m], aid[1:m] and integer array ci[1:m]. The solution vector x is overwritten on the first m elements of b, and the last n - m elements of y are overwritten on the last n - m elements of b.

lqsdol leaves the elements of a, aid and ci intact, so that, after one call of lqsdolc, several calls of lqsdol may follow for solving several least-squares problems having the same matrix M but different right-hand sides b.

lqsdol uses matvec, tamvec and elmveccol (chapter 20).

Description mca 2242

lqsdolinv should be called after lqsdolc (but only if the rank equals m), and calculates the main diagonal of the inverse of M'M, where M is the matrix whose Householder-triangularised form R with the pivotal indices, as produced by lqsdolc, are given in array a[1:m, 1:m], aid[1:m] and integer array ci[1:m]. The calculated main diagonal is delivered in array diag[1:m]; the elements of a, aid and ci are left intact.

lqsdolinv uses vecvec and tamvec (section 200).

Description mca 2243

lqsdolinv = rank, r, of the n x m matrix M given in array a[1:n, 1:m]. If r = m, then the least-squares solution x of Mx - b, where b is the vector given as array b[1:n], is calculated and overwritten on the first m elements of b; the last n - m elements of y are overwritten on the last n - m elements of b; moreover, the main diagonal of the inverse of M'M is delivered in array diag[1:m]. However, if r < m, then no solution and main diagonal are calculated, and the elements of b and diag are left unchanged.

In either case,

aux[1]: the maximum Euclidean norm of the columns of M,
aux[2]: the absolute value of the r-th diagonal element of R, and the elements of a are altered.

lqsdolinv uses lqsdolc, lqsdol, lqsdolinv and, indirectly also vecvec, matvec, tamvec, tammat, elmveccol, elmcol and ichecol (chapter 20).
REFERENCES


APPENDIX

TIMES FOR THE MC ALOQL 60 SYSTEM FOR THE X8.

In this appendix we give practical formulas for the computation times in milliseconds of the procedures published above. The coefficients of these formulas have been obtained from tests on an Electrologica X8 computer using the MC ALOQL 60 system for the X8, in which system the procedures mca 2000 to 2005 are available as machine-code procedures. For comparison we moreover give the formulas for the computation times of the nonmachine-code ALOQL 60 procedures mca 2000 to 2005 and of the procedures of section 210 using them. The coefficients of the time formulas have a relative precision of at most one or two digits.

CHAPTER 20 VECTOR OPERATIONS

Here $n$ is the number of elements used in each vector, thus, $n = u - l + 1$, except for scoapr1.

<table>
<thead>
<tr>
<th>Section 200</th>
<th>Scalar products</th>
<th>machine-code</th>
<th>ALOQL 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2000 vecvec</td>
<td>0.085 $n + 1.1$</td>
<td>0.46 $n + 0.9$</td>
<td></td>
</tr>
<tr>
<td>mca 2001 matvec</td>
<td>0.085 $n + 1.2$</td>
<td>0.54 $n + 0.9$</td>
<td></td>
</tr>
<tr>
<td>mca 2002 tempvec</td>
<td>0.085 $n + 1.2$</td>
<td>0.54 $n + 0.9$</td>
<td></td>
</tr>
<tr>
<td>mca 2003 matmat</td>
<td>0.085 $n + 1.4$</td>
<td>0.63 $n + 1.0$</td>
<td></td>
</tr>
<tr>
<td>mca 2004 tempmat</td>
<td>0.085 $n + 1.4$</td>
<td>0.63 $n + 1.0$</td>
<td></td>
</tr>
<tr>
<td>mca 2005 matmat</td>
<td>0.085 $n + 1.4$</td>
<td>0.63 $n + 1.0$</td>
<td></td>
</tr>
<tr>
<td>mca 2006 seqvec</td>
<td>0.50 $n + 1.0$</td>
<td>0.50 $n + 1.0$</td>
<td></td>
</tr>
<tr>
<td>mca 2008 scoapr1</td>
<td>0.53 $n + 1.1$</td>
<td>0.53 $n + 1.1$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 201</th>
<th>Elimination</th>
<th>machine-code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2010 elmvec</td>
<td>0.61 $n + 1.1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2011 elmveccool</td>
<td>0.60 $n + 1.1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2012 elmsolvec</td>
<td>0.78 $n + 1.1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2013 elmcoll</td>
<td>0.87 $n + 1.2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2014 elmrrow</td>
<td>0.87 $n + 1.2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2019 maxelmrow</td>
<td>0.94 $n + 1.3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 202</th>
<th>Interchanging</th>
<th>machine-code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2020 ichvec</td>
<td>0.81 $n + 0.7$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2021 ichcol</td>
<td>1.14 $n + 0.8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2022 ichrow</td>
<td>1.14 $n + 0.8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2023 ichrowcol</td>
<td>1.14 $n + 0.8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2024 ichseqvec</td>
<td>0.84 $n + 0.8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2025 ichseq</td>
<td>0.84 $n + 0.8$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 203</th>
<th>Rotation</th>
<th>machine-code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mca 2031 rotcol</td>
<td>1.40 $n + 1.3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mca 2032 rotrow</td>
<td>1.40 $n + 1.3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 21 LINEAR SYSTEMS AND MATRIX INVERSION

The formulas for this chapter hold for nonsingular matrices, unless stated otherwise.

Section 210 Triangular decomposition with partial pivoting

<table>
<thead>
<tr>
<th>mca 2000 to 2005</th>
<th>mca 2000 to 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>in machine-code</td>
<td>in ALOOL 60</td>
</tr>
<tr>
<td>mca 2100 det</td>
<td>((.033 \times n + 3.4) \times n \leq 2)</td>
</tr>
<tr>
<td>mca 2101 sol</td>
<td>((.094 \times n + 4.5) \times n)</td>
</tr>
<tr>
<td>mca 2102 detsol</td>
<td>((.033 \times n + 3.5) \times n \leq 2)</td>
</tr>
<tr>
<td>mca 2103 inv</td>
<td>((.081 \times n + 3.4) \times n \leq 2)</td>
</tr>
<tr>
<td>mca 2104 detinv</td>
<td>((.094 \times n + 6.8) \times n \leq 2)</td>
</tr>
</tbody>
</table>

Section 211 Elimination with complete pivoting

The formula for rnkelm holds, if n and m are nearly equal and the rank equals \(\min(n, m)\); the formula for solhom holds, provided the rank is not much smaller than m.

| mca 2110 rnkelm                          | \((.51 \times n + 4.2) \times n \times (m - n/3)\) |
| mca 2111 solelm                          | \((.100 \times n + 4.8) \times n\)            |
| mca 2112 rnksolelm                       | \((.34 \times n + 2.9) \times n \leq 2\)     |
| mca 2113 solhom                          | \((.046 \times \text{rank} + 2.9) \times \text{rank}\) |
| mca 2114 invelm                          | \((.95 \times n + 5.7) \times n \leq 2\)   |

Section 212 Band matrices

Here \(w\) is the band width, thus, \(w = lw + rw + 1\). The formulas for this section hold only if \(w\) is much smaller than \(n\).

| mca 2120 detbnd                          | \((.56 \times lw + 2.3) \times w \times n\) |
| mca 2121 solbnd                          | \((.7 \times lw + .1 \times rw + 4.2) \times n\) |
| mca 2122 detsolbnd                       | \((.55 \times lw + 2.9) \times w \times n\) |
CHAPTER 22 POSITIVE DEFINITE SYMMETRIC LINEAR SYSTEMS AND MATRIX INVERSION

The formulas for this chapter hold for nonsingular matrices, unless stated otherwise.

Section 220 Cholesky decomposition without pivoting

| mca 2200 detsym2 | 
| (0.16 \times n + 1.2) \times n \times 2 |
| mca 2201 solsym2 | 
| (0.02 \times n + 4.1) \times n |
| mca 2202 detolsym2 | 
| (0.16 \times n + 1.3) \times n \times 2 |
| mca 2203 invsym2 | 
| (0.032 \times n + 1.7) \times n \times 2 |
| mca 2204 detinvsym2 | 
| (0.048 \times n + 2.9) \times n \times 2 |
| mca 2205 detsym1 | 
| (0.16 \times n + 1.0) \times n \times 2 |
| mca 2206 solsym1 | 
| (0.30 \times n + 3.3) \times n |
| mca 2207 detolsym1 | 
| (0.16 \times n + 1.3) \times n \times 2 |
| mca 2208 invsym1 | 
| (0.104 \times n + 1.4) \times n \times 2 |
| mca 2209 detinvsym1 | 
| (0.120 \times n + 2.4) \times n \times 2 |

Section 221 Cholesky decomposition with pivoting

The formula for solsym20 holds, if the rank equals n - 1.

| mca 2210 rnsym20 | 
| (0.016 \times n + 2.6) \times n \times 2 |
| mca 2211 solsym20 | 
| (0.032 \times n + 5.6) \times n |
| mca 2212 rnsolsym20 | 
| (0.016 \times n + 2.7) \times n \times 2 |
| mca 2213 invsym20 | 
| (0.032 \times n + 2.8) \times n \times 2 |
| mca 2214 rnkinvsym20 | 
| (0.048 \times n + 5.4) \times n \times 2 |
| mca 2215 rnsym10 | 
| (0.016 \times n + 2.0) \times n \times 2 |
| mca 2216 solsym10 | 
| (0.30 \times n + 4.8) \times n |
| mca 2217 rnsolsym10 | 
| (0.016 \times n + 2.3) \times n \times 2 |
| mca 2218 invsym10 | 
| (0.104 \times n + 2.2) \times n \times 2 |
| mca 2219 rnkinvsym10 | 
| (0.120 \times n + 4.2) \times n \times 2 |

Section 222 Cholesky decomposition for band matrices

The formulas for this section hold only if w is much smaller than n.

| mca 2220 detsymbd | 
| (0.036 \times w + 2.4) \times w \times n |
| mca 2221 solsymbd | 
| (0.67 \times w + 4.8) \times n |
| mca 2222 detolsymbd | 
| (0.036 \times w + 3.1) \times w \times n |

Section 224 Least-squares problems

The formulas for this section hold, if n \geq m and the rank equals m.

| mca 2240 lsgdec | 
| (0.50 \times m + 2.2) \times m \times (n - m/3) |
| mca 2241 lsgsol | 
| (0.75 \times m + 1.0) \times n |
| mca 2242 lsgdlinv | 
| (0.016 \times m + 1.1) \times m \times 2 |
| mca 2243 lsgdecxol | 
| (0.50 \times m + 4.1) \times m \times (n - m/3) |