A numerically stable implementation of the algorithm of Descloux

Citation for published version (APA):
A numerically stable implementation of the algorithm of Deseloux.

by

L.S. de Jong

T.H.-Report 72-WSK-05

October 1972
In this report we shall discuss a numerically stable implementation of the algorithm of Descloux [5], which is capable of finding a solution for the problem:

\[ \text{find } x \in \mathbb{R}^n \text{ so that} \]
\[
\begin{align*}
A_1 x &= b_1 , \\
\| A_2 x - b_2 \|_{\infty} \text{ is minimal ,}
\end{align*}
\]

where \( A_1 \in \mathcal{M}_{r,n}^\mathbb{R}, \ b_1 \in \mathbb{R}^r, \ (A_2) \in \mathcal{M}_{m,n}^\mathbb{R}, \ (b_2) \in \mathbb{R}^m \) and \( r < n < m \). (0)

This problem is amongst other things analyzed in [6].
The notions and definitions of that report will be used.
0. For reasons of completeness, let us recall some notions and give once more an outline of the algorithm.

If \( S = \{i_1, \ldots, i_s\} \) is a subset of the index set \( \{1, \ldots, m\} \) and

\[
A = \begin{bmatrix}
    a_1^T \\
    \vdots \\
    a_m^T
\end{bmatrix},
\]

then

\[
A(S) = \begin{bmatrix}
    a_{i_1}^T \\
    \vdots \\
    a_{i_s}^T
\end{bmatrix}.
\]

The minimax problem is:

\[
\begin{aligned}
\text{find } x & \in \mathbb{R}^n \text{ so that } \\
\begin{bmatrix}
A(E)x = b(E) \\
\|A(L\setminus E)x - b(L\setminus E)\|_\infty
\end{bmatrix} \text{ is minimal,}
\end{aligned}
\]

with \( L = \{1, \ldots, m\}, E = \{1, \ldots, r\}, A(E) \in \mathbb{M}_{r,n}, A \in \mathbb{M}_{m,n} \) and \( r < n < m \).

A solution for this problem is called a minimax solution.

The value \( \rho \) of the minimum is called the deviation.

The vector \( r(x) = Ax - b \) is called the residual vector to \( x \).

If \( I \) is an index set so that \( E \subset I \subset L \) and \( A(I) \in \mathbb{M}_{n+1,n} \), then \( A(I) \) is called a reference.

An outline of the algorithm:

(0) \( j := 0 \),

(1) choose \( I \in \mathcal{R}(A) \), (e.i., \( A(I) \) is a reference)

(2) determine \( \lambda \neq 0 \) so that

\[
\sum_{i \in I} \lambda_i a_i = 0,
\]

and compute

\[
\rho := (- \sum_{i \in I} \lambda_i b_i) \left( \sum_{i \in I \setminus E} |\lambda_i| \right)^{-1}.
\]

after which, if necessary, \( \lambda \) is adapted so that \( \rho \geq 0 \),
(3) determine the cadre of $I$,

$$K := \{i \mid i \in I, \lambda_i \neq 0\}$$

and choose for $i \in I \setminus (K \cup E)$ scalars $\gamma_i$ that are unequal to zero.

(4) determine

for $i \in E$, $r_i := 0$,

for $i \in K \setminus E$, $r_i := \rho \text{ sign}(\lambda_i)$,

for $i \in I \setminus (K \cup E)$, $r_i := \rho \text{ sign}(\gamma_i)$,

(5) compute the solution $x$ of

$$A(I)x = b(I) + r(I),$$

(6) determine $\ell \in L \setminus I$ so that $|r_\ell(x)|$ is maximal, if $|r_\ell(x)| \leq \rho$ then $x$ is a solution, otherwise,

(7) compute $v$ so that

$$\varphi_\ell = \sum_{i \in I} v_i \varphi_i,$$

(8) determine $s \in K \setminus E$ and $M$ so that

$$M := r_\ell(x) \frac{v_s}{\lambda_s} = \max_{i \in K \setminus E} \left( r_\ell(x) \frac{v_i}{\lambda_i} \right),$$

determine $t \in I \setminus (K \cup E)$ and $N$ so that

$$N := r_\ell(x) \frac{v_t}{\gamma_t} = \max_{i \in I \setminus (K \cup E)} \left( r_\ell(x) \frac{v_i}{\gamma_i} \right),$$

if $K = I$, then $N := 0$,

(9) if $N \leq 0$, then $I := (I \setminus \{s\}) \cup \{\ell\}$, $j := j + 1$, and return to (2),

(10) if $N > 0$, then $I := (I \setminus \{t\}) \cup \{\ell\}$, $j := j + 1$, for $i \in (I \setminus (K \cup E)) \setminus \{t\}$:

if $\gamma_i - \frac{v_t}{\gamma_t} \gamma_i \neq 0$, then $\gamma_i := \gamma_i - \frac{v_t}{\gamma_t} v_i$, $\gamma_t := \frac{v_t}{\gamma_t}$.

(11) return to (4).

Remark. Any $A(I)$ occurring in the sequence of iterations contains $A(E)$. 
1.1. Introduction

Of the implementation of the algorithm of Descloux, we demand that it is efficient and numerically stable. Since the algorithm is an iterative one, the demand of efficiency implies in the first place that in any iteration the amount of work is a minimum. The most time-consuming action consists of the computation of the vectors \( \lambda, x \) and \( y \) as solutions of three sets of equations. Since two of these systems have the same matrix of coefficients and the third one has the transposed of this matrix as its matrix of coefficients, the main effort is to find some decomposition of this matrix. As any new reference differs from the previous one in only one row, it is possible to obtain the new decomposition by only partly adapting the old decomposition. However, the efficiency of this adaptation depends on the choice of the decomposition. It is therefore important that it is so chosen that it itself can be computed and adapted in an efficient way and, secondly, it enables an efficient computation of \( \lambda, x \) and \( y \). On the other hand, the implementation should not only be efficient, but also numerically stable, which means that the calculations in an iteration - such as the computation of \( \lambda, x \) and \( y \) - have to be performed in a stable way, and, secondly, that in adapting the decomposition no disastrous growth of inherited errors occurs. In fact, we shall demand that it is possible to give an a priori upper-bound for these errors.

The decomposition and the way it is adapted, have the major effect on the efficiency and stability of the implementation and, therefore, it will be our first concern. Furthermore, issues of less importance for the efficiency and stability will be discussed, such as, how is the first reference determined and how is a cadre recognized.

Finally, the Algol-text of the implementation of the algorithm on the EL-X8 of the Technological University of Eindhoven is given.

1.2. The decomposition

Let us assume that in some iteration the systems of equations

\[
\begin{align*}
\lambda^T B &= \theta^T, & B x &= r \\
y^T B &= b^T, & \text{with } B \in \mathcal{M}_{n,n}^n
\end{align*}
\]

occur and that in the next iteration we have the systems

\[
\begin{align*}
\lambda^T B' &= \theta^T, & B' x &= r' \\
y^T B' &= b^T, & \text{with } B' \in \mathcal{M}_{n,n}^n
\end{align*}
\]
We know that $B'$ is equal to $B$ except for one row. Consequently, we may assume that $B'$ is constructed by replacing the $s$-th row of $B$ by a new row, $c^T$. In order to solve the systems (1), a suitable decomposition of $B$ (or $B^{-1}$) has to be known. Our problem is to find the decomposition of $B'$ (or $(B')^{-1}$) in an efficient and numerically stable way, using the decomposition of $B$ (or $B^{-1}$). First, let us investigate how $B^{-1}$ can be adapted so that $(B')^{-1}$ is obtained.

(i) Since

$$B' = B + e_s c^T - e_s b_s^T = (I + e_s c^T B^{-1} - e_s e_s^T)B,$$

we have

$$(B')^{-1} = B^{-1} (I - \frac{e_s (c^T B^{-1} - e_s^T)}{c^T B^{-1} e_s}) .$$

Thus, it is simple to compute $(B')^{-1}$, once $B^{-1}$ is known, using $O(n^2)$ operations. However, this adaptation (which is essentially Jordan elimination and which is also used by most of the implementations of the Simplex Algorithm in Linear Programming), is numerically unstable. The reason for this unstability is that $c^T B^{-1} e_s$ may be arbitrarily small - the exchange rules only guarantee that $c^T B^{-1} e_s \neq 0$, not that it is reasonably bounded away from zero.

(ii) Secondly, let us consider a triangular decomposition of $B$:

$$PB = LU .$$

(2)

$P$ is a permutation matrix, $L$ is of unit lower-triangular form and $U$ is of upper-triangular form.

If this decomposition is obtained by Gaussian elimination with row pivoting, it is well known that a priori upper-bounds can be given for $\|L\|\infty$ and $\|U\|\infty$ and that the solutions of the systems (1) can be computed in a numerically stable way [8]. Let us suppose that $PB$ and $PB'$ differ in their $t$-th row. Applying Gaussian elimination to $B'$ with the restriction that in the first $(t-1)$ iterations the choice of the pivot is the same as when decomposing $B$, results in

$$P'B' = L'U' ,$$

where $L'$ is equal to $L$ except for the $t$-th row and its last $(n-t)$ columns and $U'$ is equal to $U$ except for its last $(n-t)$ rows. Therefore, we can find $L'$ and $U'$ by only partly adapting $L$ and $U$. This method is unstable, since the $t$-th row of $L'$ may contain arbitrarily large elements as a result of the restriction with which the decomposition is made.
(iii) Let us suppose that we have a decomposition of the form

\[ BQ = LU, \tag{3} \]

where \( Q \) is a permutation matrix, \( L \) is of lower-triangular form and \( U \) is of unit upper-triangular form.

Gaussian elimination applied to \( B' \) with column pivoting, supplies a decomposition

\[ B'Q' = L'U', \]

where \( Q' \) is equal to \( Q \) in the first \((s - 1)\) columns, \( L' \) is equal to \( L \) except for the \( s \)-th row and its last \((n - s)\) columns and \( U' \) is equal to \( U \) except for \( s \) its last \((n - s)\) rows. \( L' \) and \( U' \) may be computed by only partly adapting \( L \) and \( U \). This method of adaptation is numerically stable since a priori upper-bounds exist for \( \|L'\|_\infty \) and \( \|U'\|_\infty \) in terms of \( B \) and \( c \). It is easy to verify that the adaptation requires \( O(n^3) \) operations. This method is used in [2] and [3] and discussed in [1].

(iv) Let us, again, consider the decomposition of \( B \) fixed by (3). We have for \( B' \)

\[ B'Q = \tilde{L}U, \]

where \( \tilde{L} \) is equal to \( L \) except for its \( s \)-th row, which is \( c^T U^{-1} \). A permutation matrix \( \tilde{P} \) exists so that \( \tilde{P}L \) is of lower Hessenberg form. \( \tilde{P}L \) can be decomposed into \( L'F \), where \( L' \) is triangular and \( F \) the product of permutation matrices and stabilized elementary matrices.

Consequently, for \( \tilde{P}B'Q \) we have the decomposition

\[ \tilde{P}B'Q = L'U', \]

where \( U' = FU \).

However, \( U' \) is, generally, not of triangular form.

Therefore, considering that Gaussian elimination is equivalent with multiplication of the columns of \( B \) by the inverse of \( L \) or with multiplication of the rows by \( U^{-1} \), we shall investigate a decomposition for \( B \) of the form:

\[ BM = L \tag{4} \]

where \( L \) is of lower-triangular form. We shall disregard the other possibility \( M'B = U \), since of the matrices \( B \) the rows are of primary importance. The error analysis of the construction of (4) will be given in 1.3. Provisionally, let us assume that a priori upper-bounds for \( M \) and \( R \) exist.

We have
\[ B'M = \tilde{L} \]

where \( \tilde{L} \) equals \( L \) except for the \( s \)-th row, which is \( c^T M \). Thus, \( \tilde{L} \) is of the form

\[
\tilde{L} = \begin{bmatrix}
* & 0 & . & . & . & 0 \\
* & * & 0 & 0 & 0 & 0 \\
. & * & * & * & * & + (s) \\
. & . & * & 0 & 0 & . \\
. & . & . & * & 0 & . \\
* & . & * & * & * & .
\end{bmatrix}
\]

A permutation matrix \( \tilde{P} \) exists so that \( \tilde{P}L \) is of lower Hessenberg form:

\[
\tilde{P}L = \begin{bmatrix}
* & 0 & . & . & . & 0 \\
* & * & 0 & 0 & 0 & 0 \\
. & * & * & * & 0 & 0 \\
. & . & * & 0 & 0 & . \\
. & . & . & * & 0 & . \\
* & . & * & * & * & .
\end{bmatrix}
\]

\( \tilde{P}L \) is almost of lower-triangular form. Its non-zero super diagonal elements can be annihilated by Gaussian elimination with column pivoting. This is equivalent with post multiplication of \( PL \) by permutation matrices \( Q_i \) and stabilized elementary matrices \( F_i \) and we have

\[ L' = \tilde{P}LQ_i F_i \ldots Q_{n-1} F_{n-1}, M' = MQ_i F_i \ldots Q_{n-1} F_{n-1} \]

\( Q_i \) is either the identity matrix or a permutation matrix interchanging column \( i \) and \( i+1 \). \( F_i \) has the form

\[
F_i = \begin{bmatrix}
1 & & & & & \\
. & 1 & & & & \\
. & . & 1 & & & \\
. & . & . & \ddots & & \\
. & . & . & & 1 & \\
0 & . & . & \ldots & 1 & \\
\end{bmatrix} + (i),
\]

with \( |m_i| \leq 1 \).
If we, for a moment, disregard the permutation matrices $Q_i$, we have

$$F := F_s, \ldots, F_{n-1} = \begin{bmatrix} 1 & 0 & 0 \\ \cdot & \cdot & \cdot \\ 0 & m_s & 1 \\ 1 & m_{s+1} & 1 \\ \cdot & \cdot & \cdot \\ 0 & 1 \end{bmatrix}$$

and, thus, $\|F\|_\infty \leq 2$.

Consequently, a priori upper-bounds exists for $\|R'\|_\infty$ and $\|M'\|_\infty$. If, initially, $\|M\|_\infty \leq 1$, then we can only prove that after $k$ of these adaptations $\|M\|_\infty \leq 2^k$, but, in practice, one will find that $\|M\|_\infty \approx 1$. Anyway, unlimited growth of inherited errors is excluded, which by definition implies that this adaptation is numerically stable. It is easy to verify that the required number of operations is $O(n^2)$. The method is also used in [4] and in a slightly different form discussed in [1]. One should be aware that $M'$ is generally not of triangular form if $M$ initially was. Indeed, the fourth possibility is used in the implementation of the algorithm of Descloux, since it is stable and efficient.

Remark. In 1.2 we supposed that in an iteration systems of equations with matrix of coefficients $B \in \mathbb{R}^{n \times n}$ have to be solved. In the outline of the algorithm as given in 0., this matrix is $A(I)$ with $A(I) \in \mathbb{R}^{n \times n}$. Therefore, a regular $n \times n$ submatrix of $A(I)$ should be determined, but we should be aware that the condition number of the submatrix is not larger than necessary. How this is achieved will be discussed in 1.4.

1.3. The initial reference

Before the sequence of iterations can start, an initial reference has to be determined. We want to combine this with the construction of a decomposition of the form (4). This is achieved in the following way. Let Gaussian elimination be applied to $A$ with complete pivoting, and, doing so, let the following modifications be introduced:
(i) if, normally, one has $PAQ = LU$, with $L \times n$ unit lower trapezoidal, we now arrange the calculations so that we obtain $PAQM = L$, with $M$ unit triangular.

(ii) in the first $r$ iterations of the elimination the pivot is allowed only to be found in the first $r$ rows of $A$.

It is obvious that the first $(n + 1)$ rows of $PA$, $PA(I)$, make up a reference for which, simultaneously, a decomposition of the form (4) is known. The effect of the second modification is that $PA(I)$ contains the rows corresponding with the equations, which have to be satisfied exactly. That $U^{-1}$ ($= M$) instead of $U$ is constructed is explained by the fact that, after subsequent adaptations of the decomposition as described in 1.2 (iv), $U$ is not anymore of triangular form. That $M$ should be unit and that complete pivoting is used, becomes clear from the error analysis of this modified Gaussian elimination. However, it is not essential that $M$ should be unit. We shall only need that $M$ has the property $|M_{i,j}| \leq |M_{ii}|, j \geq i$. Let us, for a moment, assume that $L$ and $A$ are $n \times n$ matrices. Disregarding row and column interchanges the elimination process may be described as follows:

$$
\begin{align*}
L_0 & := A \\
L_1 & := L_0 F_1 + E_1 \\
\vdots & \quad \vdots \\
L_n & := L_{n-1} F_{n-1} + E_{n-1} \\
M_0 & := I \\
M_1 & := M_0 F_1 + G_1 \\
\vdots & \quad \vdots \\
M_n & := M_{n-1} F_{n-1} + G_{n-1}.
\end{align*}
$$

The $F_i$ are elimination matrices of the form

$$
F_i = \begin{bmatrix}
1 & 0 \\
\vdots & \ddots \\
& \ddots & m_{ii+1} m_{in} \\
0 & \cdots & 1
\end{bmatrix},
$$

with $|m_{ij}| \leq 1$.

The $E_i$ and $G_i$ are error matrices.
Let \( U := F_{n-1}^{-1} \ldots F_1^{-1} \). We have
\[
U = \begin{bmatrix}
1 & -m_{12} & \ldots & -m_{1n} \\
0 & 1 & -m_{23} & \ldots & -m_{2n} \\
& & \ddots & \ddots & \ddots \\
& & & 1 & -m_{n-1,n} \\
0 & & & 0 & 1
\end{bmatrix}
\]

It is well known from the error analysis of the common LU decomposition that
\[
LU = A + E_1,
\]
with \(|E_1| \leq n \cdot 2^{-t} |L||U|\).

Let us define \( E_2 \) by
\[
MU = I + E_2.
\]

We have
\[
i_{ij} + (E_2)_{ij} = \sum_{k=1}^{j-1} M_{ik} U_{kj} + M_{ij} = -\sum_{k=1}^{j-1} M_{ik} m_{kj} + M_{ij},
\]

or
\[
(-E_2)_{ij} = i_{ij} + \sum_{k=1}^{j-1} M_{ik} m_{kj} - M_{ij}.
\]

Considering that
\[
(G_1)_{ij} + (M_2)_{ij} = i_{ij} + M_{ij} m_{ij},
\]
and, generally,
\[
(G_k)_{ij} + (M_k)_{ij} = (M_{k-1})_{ik} m_{k-1,j}, \quad (k \leq j),
\]
we obtain, since \( M_{i\ell} = (M_t)_{i\ell} \) for \( t \leq \ell \),
\[
(E_2)_{ij} = (G_1)_{ij} + \ldots + (G_j)_{ij},
\]
and
\[
(E_2)_{ij} = f_{i\ell}(i_{ij} + \sum_{k=1}^{j-1} M_{ik} m_{kj}) - (i_{ij} + \sum_{k=1}^{j-1} M_{ik} m_{kj}).
\]
After some manipulation it follows that

\[ |E_2|_{ij} \leq j.2^{-t} \sum_{k=1}^{i} |M_{ik}| |U_{kj}|, \]

or,

\[ |E_2| \leq n.2^{-t} |M| |U|. \]

The backward error in \( A \) is \( \Delta A = LM^{-1} - A = E_1 - AE_2 \). We have:

\[ \frac{\| \Delta A \|_\infty}{\| A \|_\infty} \leq n.2^{-t} \frac{\| L \| |U| \|_\infty}{\| A \|_\infty} + n.2^{-t} \| |M| |U| \|_\infty \leq 2n.2^{-t} |M| |M^{-1}| \|_\infty, \]

or,

\[ \frac{\| \Delta A \|_\infty}{\| A \|_\infty} \leq 2n.2^{-t} c_r(M). \]  

(\( c_r(M) \) is called right-hand side condition number.)

Since \( M \) is unit and, thanks to the complete pivoting, its elements in absolute value are less than one we have \( c_r(M) \leq 2^{n-1}. [7]. \) However, it is well known that for a large class of matrices this upper bound for \( c_r(M) \) is very pessimistic. In practice, we commonly have \( c_r(M) \approx 0(n) \). Next, let us discuss how the system \( Ax = b \), where \( (A + \Delta A)M = L \), is solved and give an error analysis.

First, we solve \( Lz = b \). The answer \( z' \), satisfies a system \( (L + \Delta L)z = b \) exactly, where \( |\Delta L| \leq n.2^{-t} |L| \).

Secondly, \( x \) is obtained from \( x := My \) so that \( x \) satisfies \( x = (M + \Delta M)z' \), where \( |\Delta M| \leq n.2^{-t} |M| \).

Consequently, \( x \) is the exact solution of \( (A + E)x = b \), where \( E = \Delta A - \Delta A M M^{-1} + \Delta L M^{-1} \).

Hence,

\[ \frac{\| E \|_\infty}{\| A \|_\infty} \leq 2n.2^{-t} |M| |M^{-1}| \|_\infty + n.2^{-t} |M| |M^{-1}| \|_\infty + n.2^{-t} |M| |M^{-1}| \|_\infty, \]

or,

\[ \frac{\| E \|_\infty}{\| A \|_\infty} \leq 4n.2^{-t} c_r(M). \]
The forward error in $\mathbf{x}$ satisfies

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq 4n \cdot 2^{-\varepsilon} c(A) c_r(M).$$

(6)

Here, again, it should be pointed out that, usually, $c_r(M) \approx O(n)$.

Results similar to (5) and (6) are valid after a number of adaptations of the decomposition. In fact we have for the backward error in $A$ after $k$ (say) adaptations:

$$\frac{\|\Delta^{(k)} A\|_\infty}{\|A\|_\infty} \leq 2^{-\varepsilon} (n c_r(M) + \|M(2)\| \cdot \|U(2)\| \cdot \|M(1)^{-1}\|) + \ldots$$

$$\ldots + \|M(k+1)\| \cdot \|U(k)\| \cdot \|M(k)^{-1}\|),$$

where $M(\ell) \approx M(\ell+1) U(\ell)$.

We may expect the total number of iterations in the algorithm of Descloux to decrease, if the first reference is so chosen that the corresponding deviation increases. Therefore, the question arises whether the initial reference, within the frame of the proposed elimination, can be so chosen that the corresponding deviation is as great as possible. We know that at the end of the elimination process the first $n$ rows of $PA$ are linearly independent. Consequently, any other row of $PA$ can be chosen to form, together with the first $n$ rows of $PA$, the initial reference $A(II)$. If the Gaussian eliminations are extended to the right hand side vector $\mathbf{b}$, we have

$$(A \mid b) \begin{bmatrix} m \\ n^T \\ 1 \end{bmatrix} = (L \mid \xi),$$

where $(L \mid \xi)$ is lower trapezoidal.

If the first reference is chosen to consist of the first $(n+1)$ rows of $PA$ we obtain for the first deviation

$$\rho = \lambda_{n+1} |\xi|_{n+1} = (L \mid \xi)^{-1} \xi_{n+1}.$$

$\xi_{n+1}$ denotes the $(n+1)$-th component of $\xi$. We may expect $\rho$ to be greater if the $(n+1)$-th row is so chosen that $|\xi_{n+1}|$ is maximal. This, indeed, is what is done by the implementation.
1.4. The calculation of $\lambda$, $x$ and $v$

Let $A(I)$ denote the reference current in some iteration. The systems

$$\lambda^T A(I) = \bar{\theta}^T, \quad A(I)x = b(I) + r(I), \quad \text{and} \quad \nu^T A(I) = a_k^T,$$

have to be solved. Each of these systems is compatible. There are two possibilities.

(i) Either we solve

$$\lambda^T (A(I), b(I)) = (\bar{\theta}^T, 1), \quad (A(I) | b(I))(x) = r(I),$$

$$\nu^T (A(I) | b(I)) = (a_k^T, 1);$$

and, thus, a decomposition of the $(n+1) \times (n+1)$ matrix $(A(I) | b(I))$ should be available,

(ii) or we try to find a regular $n \times n$ submatrix of $A(I)$, so that a decomposition of the submatrix should be known.

For two reasons the second possibility is preferred. The first one is that we also in the case that the deviation equals zero - which means that $(A(I) | b(I))$ is singular - want to find a solution. The second reason is that we want to compute iterative refinements to an obtained extremal minimax solution $x$ and the corresponding deviation, without constructing a new decomposition. Let us suppose that, approximately,

$$(A(I) | b(I)) \begin{bmatrix} x_0 \\ 0 \end{bmatrix} \approx \rho_0 g,$$

where $s_i = 0$ for $i \in E = \{1, \ldots, r\}$ and $|s_i| = 1$ for $i \in I \setminus E$.

We want iterative refinements for $x_0$ and $\rho_0$. So the system

$$(A(I) | b(I)) \begin{bmatrix} x_0 \\ 0 \end{bmatrix} = \delta x + r,$$

has to be solved, where $r_1$ is a residual (calculated in double-length). Consequently, of $(A(I) | g)$ a decomposition should be known, but is is not possible to obtain this using the decomposition of $(A(I) | b(I))$ since the two matrices differ from one another in a column.
In any iteration we have - now considering the second possibility -

\[ A(I)M = L, \]

where \( A(I) \) is the current reference and \( L \) is \((n + 1) \times n\) lower trapezoidal. The systems

\[
\begin{align*}
\tilde{A}^T L &= \tilde{b}^T, \\
Ly &= \tilde{b}(I) + \tilde{r}(I), \\
\tilde{v}^T L &= \tilde{a}_x^T M \\
x &= \tilde{M}y
\end{align*}
\]

have to be solved. If the first \( n \) rows of \( L \) are linearly independent, then the first and third system may be solved by setting \( \lambda_{n+1} = -1 \), respectively \( \nu_{n+1} = 0 \), while the second system may be solved by omitting the last equation.

Are the first \( n \) rows of \( L \) linearly independent?

In the first iteration \( L \) (and \( M \)) result from a Gaussian elimination process applied to \( A \ (1.3) \). Since \( A \) has full rank and complete pivoting is employed, the first \( n \) rows of the first \( L \) are linearly independent.

Also in the following iterations it can be arranged that \( L \) has this property. Let us assume that in the decomposition \( A(I)M = L \), the first \( n \) rows of \( L \) are linearly independent, and that the new reference \( A(I') \) is obtained from \( A(I) \) by exchanging the \( s \)-th row for \( c \) (say). We have

\[ A(I')M = \tilde{L}. \]

\( \tilde{L} \) is equal to \( L \) except for its \( s \)-th row, which is \( c^T M \).

The decomposition is updated as follows: by row interchanges \( \tilde{L} \) transforms to \( \tilde{P}L \),

\[
\begin{align*}
\tilde{P}L &= \begin{bmatrix}
* & 0 & . & . & 0 \\
* & * & 0 & 0 & 0 \\
. & * & * & 0 & 0 \\
. & . & . & * & * \\
. & . & * & * & * \\
. & . & * & * & * \\
* & . & . & * & * \\
* & . & . & * & * \\
\end{bmatrix} \rightarrow (s) \\
\rightarrow (n - 1) \\
\rightarrow (n)
\end{align*}
\]

(the \((n + 1)\)-th row of \( \tilde{P}L \) is equal to the \((n + 1)\)-th row of \( \tilde{L} \)), after which, by Gaussian elimination with column pivoting, \( \tilde{P}L \) is reduced to \( L' \), which has lower trapezoidal form.
In this process it is left possible that either the \((n - 1)\)-th or the \(n\)-th row of \( \vec{P}L \) is equal to \( c^T M \). In both cases we have that the first \((n - 1)\) rows of \( \vec{P}L \) are linearly independent. (In case the \((n - 1)\)-th row equals \( c^T M \), this is a consequence of the exchange rules, otherwise the first \((n - 1)\) rows of \( \vec{P}L \) are a subset of the first \( n \) rows of \( L \) and, consequently, linearly independent.) Hence, it can be arranged — since the exchange rules guarantee that \( \vec{P}L \) has rank \( n \) — that the first \( n \) rows of \( L' \) are linearly independent by comparing the \( n\)-th components of the last two rows and, if necessary, exchanging these two rows.

The foregoing suggests that it makes no difference whether the \((n - 1)\)th or the \(n\)-th row of \( \vec{P}L \) equals \( c^T M \).

However, if we demand that any reference of \( A \) contains an \( n \times n \) submatrix with a modest condition number (which is a minimal precondition in order that the algorithm is stable — we can ensure that — in case the \( n\)-th row of \( \vec{P}L \) equals \( c^T M \) — the first \( n \) rows of \( L' \) make up a matrix with a modest condition number.

Also this is demonstrated by an inductive argument. Let \( G \) be the matrix made up by the first \( n \) rows of \( L \) and let us assume that \( G \) has a modest condition number (this is certainly true for the first \( L \)).

Let the \( n\)-th row of \( \vec{P}L \) be equal to \( c^T M \). In this case the first \((n - 1)\) rows of \( L' \), \( \{v_1, \ldots, v_{n-1}\} \) are transformed (in a stable way) rows of \( G \). Since the \( n\)-th components of these vectors are equal to zero, we have that the \((n - 1) \times (n - 1)\) principal submatrix of \( L' \) has a modest condition number. \((7)\)

The matrix \( G' \), consisting of the first \( n \) rows of \( L' \), is either equal to

\[ G'_1 = (v_1, \ldots, v_{n-1}, v_n)^T \]

or to

\[ G'_2 = (v_1, \ldots, v_{n-1}, v_{n+1})^T , \]

where \( v_n \) is the transformed \( c^T M \) and \( v_{n+1} \) is the transformed \((n + 1)\)-th row of \( \vec{P}L \). Let us suppose that \( G'_1 \) as well as \( G'_2 \) has a large condition number.

This implies (using \((7)\)) that the \( n\)-th component of \( v_n \) as well as the \( n\)-th component of \( v_{n+1} \) is small compared to \( \|A(I)\| \). But this implies since the \( n\)-th components of \( v_1, \ldots, v_{n-1} \) are zero, that \( L' \) does not contain an \( n \times n \) submatrix with a small condition number, which is contrary to the precondition.
Therefore, at least one of the two, \( G_1' \) or \( G_2' \), has a modest condition number. The implementation, indeed, places \( c^T \mathbf{M} \) in the \( n \)-th row of \( \mathbf{P} \).

1.5. The sequence of references

In 1.4 we assumed that the matrix \( A \) of (0) has the property that any of its references has an \( n \times n \) submatrix with a modest condition number. (8)

The consequence of this is that in any iteration well-conditioned systems of equations occur, since in any iteration reference systems occur [6].

However, this is only true when exact arithmetic is employed. With non-exact arithmetic, it is first of all very likely that the "machine-\( A \)" has more references than the exact \( A \), (some of which do not have the quality mentioned above). Secondly, the non-exact representation of \( A \) in the machine combined with the effect of round-off errors may have as a result that in some iteration an ill-conditioned reference systems occurs. We shall show that this "failure" is a real possibility, but that under certain conditions for \( A \), a remedy exists.

With non-exact arithmetic it is almost certain that in any iteration the computed \( \lambda_i \) as well as \( v_i \) are all unequal to zero so that, generally, the row to be exchanged is determined by the index \( s \) satisfying

\[
\frac{v_s}{\lambda_s} r_\ell(x) = \max \left( \frac{v_i}{\lambda_i} r_\ell(x) \right).
\]

It is possible that in some iteration \( v_s \) as well as \( \lambda_s \) are nearly equal to zero: In that case the new reference is ill-conditioned:

\[
a_\ell = \sum_{i \in I} v_i a_i \text{ enters the reference, } a_s \text{ leaves the reference},
\]

since \( \lambda_s \approx 0, \)

\[
\left\| \sum_{i \in I \setminus \{s\}} \lambda_i a_i \right\| \ll \|A(I')\|
\]

since \( v_s \approx 0, \)

\[
\left\| \sum_{i \in I \setminus \{s\}} v_i a_i - a_\ell \right\| \ll \|A(I')\|
\]

which implies that \( A(I') \) does not contain an \( n \times n \) submatrix with a modest condition number.
The obvious remedy is to set relatively small $\lambda_i$ as well as $v_i$ equal to zero, but this can only work if one can distinguish between $\lambda_i$ (and $v_i$) that should be zero (but are not) and $\lambda_i$ (and $v_i$) that should not be zero.

Let us suppose that in some iteration the reference

$$ R = \begin{pmatrix} A_k \\ \frac{T}{a_k} \end{pmatrix} $$

occurs, where $A_k \in \mathbb{R}_{n,n}$ and let

$$ \lambda = \begin{pmatrix} \lambda_k \\ \frac{\lambda_k'}{\lambda_k} \end{pmatrix} \neq 0 $$

so that

$$ A_k^T \lambda_k = a_k \lambda_k' = 0. $$

If $\lambda_k' = 0$, then $A_k \in \mathbb{R}^{n-1}_{n,n}$; if $\lambda_k' \neq 0$, then $A_k \in \mathbb{R}^n_{n,n}$.

We have, if $\lambda_k' \neq 0$,

$$ |\lambda_k'| = \frac{\| \lambda_k \|_\infty}{\| A_k^T \|_\infty} \geq \frac{\| \lambda_k \|_\infty}{\| A_k^T s_k \|_\infty} \geq \frac{\| \lambda_k \|_\infty}{\| A_k^T \|_\infty \| R \|_\infty} =: \frac{\| \lambda_k \|_\infty}{g_k} . $$

Since $g_k \geq 1$ we also have

$$ |\lambda_k'| \geq \frac{\| \lambda \|_\infty}{g_k} . $$

Hence, either $\lambda_j = 0$ or

$$ |\lambda_j| \geq \frac{\| \lambda \|_\infty}{\max_k g_k} =: \frac{\| \lambda \|_\infty}{g_R} . $$

For the computed $\lambda_j$ we have the accuracy

$$ \| \delta \lambda \|_\infty \leq f(n) 2^{-t} c_R \| \lambda \|_\infty , $$

where $c_R$ is the condition number of the matrix with which the computations are made.
Consequently, we can distinguish between the $\lambda_j$ that should be zero and the other components if

$$f(n)2^{-t}c_R < \frac{1}{g_R} - f(n)2^{-t}c_R,$$

or,

$$g_R < \frac{2^{t-1}}{f(n)c_R}.$$

The condition for $A$ in order that this technique may work is that for all references $R$, $g_R$ is bounded by

$$\frac{2^{t-1}}{n^*c_R}, \ldots. \tag{9}$$

The device is employed by the algorithm and ensures that only well-conditioned subsystems are considered in the sequence of iterations.

Remark. When this device is employed, it is still possible, even though (9) is true, that, due to under flow, the wrong row is exchanged. This happens when $N$ becomes nearly zero due to a very large value of one of the scalars $\gamma_i$.

1.6. Let the implementation be applied to (0). In [6] it is proved that for any two consecutive iterations holds that for

$$i \in I_{j+1}, r_i(x_j)r_i(x_{j+1}) \geq 0, \tag{10}$$

where $I_{j+1}$ determines the $(j + 1)$-th reference and $x_j$ and $x_{j+1}$ are two consecutive minimax solutions supplied in the $j$-th respectively $(j + 1)$-th iteration.

(10) was necessary to prove the finiteness of an étappe (the deviation remains the same during a number of iterations). Also when non-exact arithmetic is used one should have that for

$$i \in I_{j+1}, r_i(x_j)r_i(x_{j+1}) \geq 0, \tag{11}$$

where $I_j$ and $I_{j+1}$ belong to one étappe, in order to ensure the finiteness of an étappe (with exact arithmetic (10) also holds when $I_j$ and $I_{j+1}$ do not belong to one étappe; in 1.7 we shall see that then (10) is not vital in order to ensure the finiteness).
It is necessary and sufficient for (11) that for

\[ i \in I_{j+1} | (K_j \cup E), \gamma_i^{(j)} \gamma_i^{(j+1)} > 0 \]

or, equivalently, for

\[ i \in I_{j+1} | (K_j \cup E), \gamma_i^{(j)} (\gamma_i^{(j)} - \frac{\gamma_t^{(j)}}{\nu_t^{(j)}}) > 0 \]

from which it follows that for

\[ i \in I_{j+1} | (K_j \cup E), |\gamma_i^{(j)}| > |\frac{\gamma_t^{(j)}}{\nu_t^{(j)}} \gamma_i^{(j)}| . \quad (12) \]

Here \( K_j \) denotes the cadre of the étappe, while the scalars \( \gamma \) and \( \nu \) have the same meaning as in 0. It is assumed that the \( t \)-th row of \( A(I_j) \) does not belong to \( A(I_{j+1}) \).

Since

\[ \frac{\nu_t^{(j)}}{\gamma_t^{(j)}} r_g(x_j) = \max \frac{\nu_t^{(j)}}{\gamma_t^{(j)}} r_g(x_j) , \]

(12) is true with exact arithmetic, but in the non-exact case it is possible that (12) is violated.

The remedy is to change step (10) of the algorithm into

(10) ...........

if \( |\gamma_i^{(j)} - \frac{\gamma_t^{(j)}}{\nu_t^{(j)}} \nu_i^{(j)}| > \text{tolerance} \), then \( \gamma_i^{(j+1)} := \gamma_i^{(j)} - \frac{\gamma_t^{(j)}}{\nu_t^{(j)}} \nu_i^{(j)} , .... \)

Here tolerance should be a realistic estimate for the error in \( \frac{\gamma_t^{(j)}}{\nu_t^{(j)}} \nu_i^{(j)} \).

This device is employed by the implementation.
1.7. The finiteness of the implementation

Let the implementation be applied to (0) and let a sequence of iterations be supplied fixed by

\[(I_1, \tilde{\rho}_1, \tilde{x}_1), (I_2, \tilde{\rho}_2, \tilde{x}_2), \ldots \]

where \( I_j \) determines the reference current in the \( j \)-th iteration, \( \tilde{\rho}_j \) is the computed deviation and \( \tilde{x}_j \) is the computed minimax solution of

\[ A(I_j)x = b(I_j) + \tilde{\rho}_j \mathbf{e}. \]

Let the matrix \( A \) satisfy the conditions (8) and (9).

We shall show that the implementation in this case is finite. Let us consider the transition from the \( j \)-th iteration to the \((j+1)\)-th. The row entering \( A(I_j) \) is determined by the index \( \ell \) so that

\[ r_\ell(\tilde{x}_j) = \max_{i \in L \setminus I_j} |r_i(\tilde{x}_j)|. \]

The implementation compares \( |r_\ell(\tilde{x}_j)| \) with \( \tilde{\rho}_j + \text{tolerance} \), where the tolerance is a realistic estimate (which indeed a priori exists) for the summed errors in \( \tilde{x}_j \) and \( \tilde{\rho}_j \).

If \( |r_\ell(\tilde{x}_j)| \leq \tilde{\rho}_j + \text{tolerance} \), then the sequence of iterations is ended with an \( \tilde{x}_j \) for which holds

for \( i \in E \), \( |r_i(\tilde{x}_j)| << 1 \),

for \( i \in I_j \setminus E \), \( |r_i(\tilde{x}_j) - s_i \tilde{\rho}_j| << 1 \),

for \( i \in L \setminus I_j \), \( |r_i(\tilde{x}_j)| \leq \tilde{\rho}_j + \text{tolerance} \),

\[ |\tilde{\rho}_j - \rho_j| << 1. \]

(S) is true, since, firstly, in \( S_i \) no error is made thanks to 1.5 and (9), and, secondly, the inherited errors in \( \tilde{x}_j \) and \( \tilde{\rho}_j \) are a priori bounded and, in practice, do not grow significantly ((8), 1.3, 1.4).

Otherwise, if \( |r_\ell(\tilde{x}_j)| > \tilde{\rho}_j + \text{tolerance} \), then we have that \( |r_\ell(\tilde{x}_j)| \) is reasonably bounded away from \( \rho_j \).

Next, let us consider which row enters the reference.

Due to the device of 1.5 and condition (9) no mistake is made when deciding whether \( N > 0 \) or \( N \leq 0 \).
Let us suppose that during a number of iterations \((j,j+1,\ldots)\) we have that \(N > 0\). The deviation remains the same during these iterations.

Since the scalars \(\gamma_i^{(k)}\) itself correspond with an artificial perturbation \([6]\), the influence of errors on these scalars may be neglected provided that \(\gamma_i^{(k)}\gamma_i^{(k+1)} > 0\), \(k = j,j+1,\ldots\), but this is ensured by 1.6.

The device of 1.5 is used so that for

\[
|\lambda^{(k)}_{t_k}| << 1,
\]

where \(t_k\) determines the row leaving \(A(I_k)\).

This implies together with (9) that non-cadre rows are exchanged for rows for which, by (13), holds that for

\[
k = j,j+1,\ldots \quad |r_{ik}^{(k)}(x_k)| > \rho_j,
\]

By 1.6 we have that for

\[
k = j,j+1,\ldots \quad r_i^{(k)}(\tilde{x}_k)r_i^{(k)}(\tilde{x}_{k+1}) \geq 0, \quad i \in I_{k+1},
\]

and, since in the signs of the residuals thanks to 1.5 no error is made, also for

\[
k = j,j+1,\ldots \quad r_i^{(k)}(x_k)r_i^{(k+1)} \geq 0, \quad i \in I_{k+1}.
\]

Consequently, the exact algorithm applied to (0) with starting reference \(I_j\) and corresponding minimax solution \(x_j\) may supply the same \(\text{étape} I_jI_{j+1},\ldots,\) provided that it is applied with the proviso that the residual of the row entering the reference should be greater than the current deviation but need not be the maximal residual. In \([6]\) it is proved that such an \(\text{étape}\) is finite so that also the \(\text{étape}\) supplied by the implementation is finite.

On the other hand, if \(N \leq 0\), then a row is exchanged for which holds that \(|\lambda_s|\) is reasonably bounded away from zero.

In this case we have for the new deviation:

\[
\rho_{j+1} \approx \frac{\left| \sum_{i \in I \setminus \{s\}} \lambda_i^{\frac{v_s}{\lambda_i}} - v_i \right| r_i(x_j) + \sum_{i \in I \setminus \{s\}} v_i r_i(x_j) - r_i(x_j)}{\left| \lambda_i \right|} \quad (\text{see} \ [6]).
\]

\[
\rho_{j+1} \approx \frac{\sum_{i \in I \setminus \{s\} \cup E} \left| \lambda_i^{\frac{v_s}{\lambda_i}} - v_i \right| + 1}{\left| \lambda_i \right|}.
\]
Since

$$N = \frac{v_s}{\lambda_s} r_s(x) = \max_{i \in \mathbb{I} \setminus \{s\}} \frac{v_i}{\lambda_i} r_i(x)$$

and $N \leq 0$, all terms in the numerator have the same sign so that $\rho_{j+1} > \rho_j$ and by (13) $|\rho_{j+1} - \rho_j|$ is reasonably bounded away from zero.

Remark. If $\frac{v_s}{\lambda_s} \rightarrow \frac{v_r}{\lambda_r}$ then still holds that $\rho_{j+1} > \rho_j$ by (13).

The consequence of the foregoing is that the implementation is finite. Since at the end an $x_j$ is supplied for which (S) holds, we may call the implementation global stable.

2. ALGOL-text
procedure MINIMAX APPROXIMATION(ml, m, n, A, b, x, residuals, rho, q, macheps, singular);
value ml, m, n, macheps, singular; real array A, b, x, residuals; integer array q; integer ml, m, n; real rho, macheps; label singular;

comment

MINIMAX APPROXIMATION supplies a solution for the problem:

find x so that

for i = 1, ..., ml, \[ A[i,1] \times x[1] + A[i,2] \times x[2] + \ldots + A[i,n] \times x[n] = b[i], \]

max(abs(A[i,1] \times x[1] + A[i,2] \times x[2] + \ldots + A[i,n] \times x[n]), i = ml + 1, ..., m) is minimal.

It is assumed that ml < n < m, that the first ml rows of A make up a matrix of full rank, and that A itself has full rank. A may or may not satisfy the Haar condition. In case ml = 0, a solution of the ordinary minimax problem is supplied. The parameters to the procedure are:

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>TYPE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ml</td>
<td>integer</td>
<td>the first ml equations have to be satisfied exactly.</td>
</tr>
<tr>
<td>m</td>
<td>integer</td>
<td>number of equations.</td>
</tr>
<tr>
<td>n</td>
<td>integer</td>
<td>number of unknowns.</td>
</tr>
<tr>
<td>A</td>
<td>real array</td>
<td>matrix of coefficients, array bounds [1:m,1:n].</td>
</tr>
<tr>
<td>b</td>
<td>real array</td>
<td>right-handside vector, array bounds [1:m].</td>
</tr>
<tr>
<td>x</td>
<td>real array</td>
<td>solution-vector, array bounds [1:n].</td>
</tr>
<tr>
<td>residuals</td>
<td>real array</td>
<td>contains the residuals to the solution-vector, array bounds [1:m].</td>
</tr>
<tr>
<td>rho</td>
<td>real</td>
<td>contains the deviation, rho = maximum(residuals).</td>
</tr>
<tr>
<td>q</td>
<td>integer array</td>
<td>contains a permutation of (1,...,m). The first n + 1 elements determine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the reference corresponding with the solution-vector.</td>
</tr>
<tr>
<td>macheps</td>
<td>real</td>
<td>if during the decomposition of a reference it appears that a pivot is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>less than a tolerance depending on macheps, the exit singular is taken.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A recommended value for macheps is the relative machine-precision,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which is defined as the smallest real so that in the real arithmetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the machine : 1 + macheps &gt; 1.</td>
</tr>
<tr>
<td>singular</td>
<td>label</td>
<td></td>
</tr>
</tbody>
</table>
At the end of the procedure, \( x \) contains an extremal minimax solution, \( \rho \) contains the corresponding deviation, residuals contains the residuals to \( x \) and the first \( n + 1 \) elements of \( q \) determine a maximal reference.

The array \( A \) is not changed by MINIMAX APPROXIMATION. The procedure uses the non-local procedures INPROD and DLINPROD, which compute the value of an inner-product in single- respectively double-precision.

For details concerning notation and method of solution the user is referred to:
L.S.de Jong, -Discrete Linear Chebychev Approximation-, (THE-report 72-wsk-02, 1972);

begin procedure \text{EXCHANGE}(i, lb, ub, ai, bi);
    value lb, ub; integer i, lb, ub; real ai, bi;
    comment The values of \( ai \) and \( bi \) are exchanged;
    begin real term;
        for \( i := lb \) step 1 until \( ub \) do begin term := ai; ai := bi; bi := term end
    end \text{EXCHANGE};

procedure \text{SOLUTION}(n, U, x, r);
    value n; real array U, x, r; integer n;
    comment Supplies the solution of \( U \times x = r \), where \( U \) is of upper-triangular form. The array \( r \) is not changed by \text{SOLUTION};
    begin integer \( j, k \);
        for \( k := n \) step -1 until 1 do \( x[k] := \text{INPROD}(j, k + 1, n, -U[k,j], x[j], r[k])/U[k,k] \)
    end \text{SOLUTION};

procedure \text{TSOLUTION}(n, M, U, x, r);
    value n; real array M, U, x, r; integer n;
    comment Supplies the solution of \( B \times x = r \), where \( B \) is determined by \( M \) and \( U \) in the following way: \( M \times BT = U \). \( BT \) denotes the transposed of \( B \). \( U \) is of upper-triangular form. Contrary to the procedure \text{SOLUTION} the array \( r \) is changed by \text{TSOLUTION}, but a call of the form \text{TSOLUTION}(n, M, U, r, r) is possible;
begin integer i, k; real array y[1:n];
    for k := 1 step 1 until n do r[k] := INPROD(i, 1, k - 1, -U[i,k], r[i], r[k])/U[k,k];
    for k := 1 step 1 until n do y[k] := INPROD(i, 1, n, M[i,k], r[i], 0);
    for k := 1 step 1 until n do x[k] := y[k]
end TSOLUTION;

real procedure NORM(lb, ub, a);
value lb, ub; integer lb, ub; array a;
comment The L-1 norm of the vector (a[lb],...,a[ub]) is computed;
begin integer i; real sum;
    sum := 0; for i := lb step 1 until ub do sum := sum + abs(a[i]); NORM := sum
end NORM;

procedure UPDATE(n, M, q, U, s, l, r, eps, singular);
value n, s, l, eps; real array M, U, r; integer array q; real eps; integer n, s, l; label singular;
comment Let $M \times AT \times Q(I) = U(I)$, where $AT$ stands for the transposed of $A$, $Q$ is the permutation-matrix determined by $q$, $U(I)$ is of upper-triangular form and $(I)$ means that the first $n + 1$ columns are considered. The procedure updates $M$, $q$ and $U$, if the $s$-th column of $AT \times Q(I)$ ($s < n + 1$) is exchanged for the $l$-th column ($l > n + 1$). At the call, the array $r$ should contain $M$ multiplicated with the $l$-th column;
begin integer i, j, k, n1, n2; real term, max;
    n1 := n + 1; n2 := n - 1;
    k := q[s]; q[s] := q[l]; q[l] := k;
    for i := 1 step 1 until n do U[i,s] := r[i];
    if s < n then
        begin k := q[s]; for j := s + 1 step 1 until n do q[j - 1] := q[j]; q[n] := k;
            for i := 1 step 1 until n do
                begin term := U[i,s]; for j := s + 1 step 1 until n do U[i,j - 1] := U[i,j]; U[i,n] := term end;
        end
end UPDATE;
for k := s step 1 until n2 do 
begin max := abs(U[k,k]); term := abs(U[k + 1,k]);
if max > term then
begin if max < eps then goto singular end else
begin if term < eps then goto singular;
   EXCHANGE(j, k, n1, U[k,j], U[k + 1,j]); EXCHANGE(j, 1, n, M[k,j], M[k + 1,j])
end;
term := U[k + 1,k]/U[k,k];
for j := k + 1 step 1 until n1 do U[k + 1,j] := U[k + 1,j] - U[k,j] × term;
for j := 1 step 1 until n do M[k + 1,j] := M[k + 1,j] - M[k,j] × term
end
end UPDATE;

procedure ITERATIVE REFINEMENT(n, M, U, q, x, rho, s);
value n; integer n; real array M, U, x, s; integer array q; real rho;
comment If x is an approximate minimax solution and rho is the approximate deviation for a maximal reference,
s is the vector that determines the signs of the residuals to x, then iterative refinements for x and
rho are computed;
begin real ss, t; integer i, j, k, n1; real array r[1:n + 1];
n1 := n + 1;
for j := 1 step 1 until n1 do 
begin k := q[j];
   r[j] := DLINPROD(i, 1, n1, if i = n1 then b[k] else -A[k,i], if i = n1 then 1 else x[i], rho × s[j])
end;
TSOLUTION(n, M, U, x, r); TSOLUTION(n, M, U, s, s); k := q[n1];
ss := INPROD(i, 1, n, A[k,i], r[i], 0); t := INPROD(i, 1, n, A[k,i], s[i], 0);
t := -(r[n1] - ss)/(s[n1] - t);
for i := 1 step 1 until n do x[i] := x[i] + (r[i] + s[i] × t); rho := rho + t
end ITERATIVE REFINEMENT;
real max, term, sum, rl, resid, Mk, Nr, coef, toll, tol2;
integer i, j, m2, n1, n2, k, s, t, l, sk, sn, etappe, part;
boolean oldcadre, newcadre;

n1 := n + 1; n2 := n - 1; m2 := m1 + 1;

begin real array M[1:n,1:n], U[1:n,1:m], r[1:m], labda, gamma, nu, signum[1:n1];

comment STEP 1.
A starting reference containing the rows corresponding with the equations, which have to be satisfied
exactly, is obtained by applying the Gauss-algorithm to the transposed of A : AT, allowing row- and
column-interchanges. A matrix M is found so that M × AT × Q = U, where U is of upper-triangular form
and Q is a permutation-matrix stored in an one-dimensional array. No scaling is applied.
The first n + 1 columns of AT × Q make up an initial reference : AT × Q (I) for which the decomposition
M × AT × Q (I) = U(I) applies;
for i := 1 step 1 until n do
begin for j := 1 step 1 until n do begin M[i,j] := if i = j then 1 else 0; U[i,j] := A[j,i] end;
for j := n1 step 1 until m do U[i,j] := A[j,i] end;
end;

for j := 1 step 1 until m do begin q[j] := j; r[j] := b[j] end;
for k := 1 step 1 until n do
begin max := -1; sn := if k < m2 then m1 else m;
for i := k step 1 until j := k step 1 until sn do
begin term := abs(U[i,j]); if term > max then begin max := term; s := i; t := j end end;
if k = 1 then toll := macheps × max else if max < toll then goto singular;
if s ≠ k then begin EXCHANGE(j, 1, m, U[k,j], U[s,j]); EXCHANGE(j, 1, n, M[k,j], M[s,j]) end;
if t ≠ k then
begin EXCHANGE(i, 1, n, U[i,k], U[i,t]); s := q[k]; q[k] := q[t]; q[t] := s;
term := r[k]; r[k] := r[t]; r[t] := term
end;
max := U[k,k];
for i := k + 1 step 1 until n do
begin term := U[i,k]/max;
    for j := k + 1 step 1 until m do U[i,j] := U[i,j] - U[k,j] \times term;
    for j := 1 step 1 until n do M[i,j] := M[i,j] - M[k,j] \times term
end;
term := r[k]/max; for j := k + 1 step 1 until m do r[j] := r[j] - U[k,j] \times term
end;
t := n1; max := -1;
for j := n1 step 1 until m do begin term := abs(r[j]); if max < term then begin max := term; t := j end end;
if t ≠ n1 then begin EXCHANGE(i, 1, n, U[i,t], U[i,n1]); s := q[n1]; q[n1] := q[t]; q[t] := s end;
for etappe := 1, etappe + 1 while newcadre do
    begin comment STEP 2.
The lambda-vector and the deviation to the current reference are computed;
    for i := 1 step 1 until n do r[i] := -U[i,n1];
    SOLUTION(n, U, labia, r); labda[n1] := 1;
    sum := NORM(m2, n1, labda); rho := -INPROD(i, 1, n1, labda[i], b[q[i]], 0)/sum;
    tol2 := sum \times \sqrt{macheps};
    if rho < 0 then
        begin for i := 1 step 1 until n1 do labda[i] := -labda[i]; rho := -rho end;
        comment STEP 3.
For the rows of the reference not belonging to a cadre of the reference and not corresponding with the
equations, which have to be satisfied exactly, the corresponding components of the gamma-vector are made
equal to 1, the other components are made equal to zero;
    for i := m2 step 1 until n1 do
        if abs(labda[i]) < tol2 then gamma[i] := 1 else gamma[i] := 0;
    for part := 1, part + 1 while oldcadre do
        begin
            comment
        end;
begin comment STEP 4.
An extremal minimax solution $x$ for the current reference is computed. For the equations of the reference, which have to be satisfied exactly, the residual is equal to zero, while for the other equations we have that for non-cadre equations the residual of the minimax solution is equal to $\text{sign}(\gamma[i]) \times \rho$ and for cadre equations the residual is equal to $\text{sign}(\lambda[i])$;

for $i := 1$ step 1 until $m$ do begin $r[i] := b[q[i]]$; $\text{signum}[i] := 0$ end;
for $i := m2$ step 1 until $n$ do
begin $s := \text{sign}(\lambda[i] + \gamma[i])$; $\text{signum}[i] := s$; $r[i] := s \times \rho + b[q[i]]$ end;
TSOLUTION($n$, $M$, $U$, $x$, $r$);

comment STEP 5.
For the equations not belonging to the current reference the maximal residual is computed.
If this maximum is less than the current deviation, the extremal minimax solution for the entire system has been found, else the nu-vector is computed;

$\rho_l := 0$; $l := n1 + 1$;
for $i := 1$ step 1 until $n$ do
begin $k := q[i]$; $\text{resid} := \text{INPROD}(j, 1, n, A[k,j], x[j], -b[k])$;
if $\text{abs}(\rho_l) < \text{abs}(\text{resid})$ then begin $\rho_l := \text{resid}$; $l := i$ end
end;
$\text{oldcadre} := \text{newcadre} := \text{false}$;
$\text{sum} := \text{NORM}(1, n, x)$; $\text{tol2} := n1 \times t12 \times \text{sum}$;
if $\text{abs}(\rho_l) > \rho + \text{tol2}$ then
begin $k := q[i]$;
for $i := 1$ step 1 until $n$ do $r[i] := \text{INPROD}(j, 1, n, M[i,j], A[k,j], 0)$;
SOLUTION($n$, $U$, $\nu$, $r$);
$\nu[n1] := 0$; $\text{sum} := \text{NORM}(1, n, \nu)$; $\text{tol2} := \text{sum} \times \text{sqrt(macheps)}$;
for $i := 1$ step 1 until $n$ do if $\text{abs}(\nu[i]) < \text{tol2}$ then $\nu[i] := 0$;
The equation to be exchanged is computed. If the deviation strictly increases, the process returns to STEP 2, else the gamma-vector is adapted and the process returns to STEP 4;

\( M_k := N_r := 0; \) \( r_l := \text{sign}(r_l); \)

for \( i := m_2 \) step 1 until \( n_1 \) do

begin if \( \gamma[i] = 0 \) then

begin \( \text{term} := r_l \times \nu[i]/\lambda[i]; \) if \( M_k < \text{term} \) then begin \( M_k := \text{term}; \) \( \sk := i \) end end else

begin \( \text{term} := r_l \times \nu[i]/\gamma[i]; \) if \( N_r < \text{term} \) then begin \( N_r := \text{term}; \) \( \sn := i \) end end;

if \( N_r = 0 \) then

begin UPDATE(\( n, M, q, U, \sk, l, r, \text{tol}, \text{singular}));

if abs(\( U[n,n] \)) < abs(\( U[n,n_1] \)) then

begin EXCHANGE(\( i, 1, n, U[i,n], U[i,n_1] \)); \( s := q[n]; \) \( q[n] := q[n_1]; \) \( q[n_1] := s \) end;

\( \text{newcadre} := \text{true} \);

\ndraw line\n
else

begin UPDATE(\( n, M, q, U, \sn, l, r, \text{tol}, \text{singular})); \( \text{coef} := \gamma[\sn]/\nu[\sn]; \)

for \( i := m_2 \) step 1 until \( n_1 \) do

begin if \( \gamma[i] \neq 0 \) then

begin \( \text{term} := \gamma[i] - \text{coef} \times \nu[i]; \)

if abs(\( \text{term} \)) > 3 \times \text{macheps} then \( \gamma[i] := \text{term} \)

end;

\( \gamma[\sn] := \text{coef}; \)

if \( \sn < n_2 \) then

begin \( \text{term} := \gamma[\sn]; \) for \( j := \sn + 1 \) step 1 until \( n_2 \) do \( \gamma[j-1] := \gamma[j]; \)

\( \gamma[n_2] := \text{term}; \) \( \text{term} := \lambda[\sn]; \)

for \( j := \sn + 1 \) step 1 until \( n_2 \) do \( \lambda[\j-1] := \lambda[j]; \) \( \lambda[n_2] := \text{term} \)

end;
if \( \text{abs}(U[n,n]) < \text{abs}(U[n,n1]) \) then

\[
\text{begin EXCHANGE}(i, 1, n, U[i,n], U[i,n1]);
\]
\[
s := q[n]; q[n] := q[n1]; q[n1] := s;
\]
\[
term := \gamma[n]; \gamma[n] := \gamma[n1]; \gamma[n1] := \text{term};
\]
\[
term := \lambda[n]; \lambda[n] := \lambda[n1]; \lambda[n1] := \text{term}
\]

\end;

\]

\text{oldcadre} := \text{true}
\]

\end

\end

\text{else}

\begin comment STEP 7.

An iterative refinement is made for the obtained extremal minimax solution and the obtained deviation. The residuals to the corrected solution are computed;

\text{ITERATIVE REFINEMENT}(n, M, U, q, x, \rho, \text{signum});

\end

\text{for } i := 1 \text{ step } 1 \text{ until } m \text{ do residuals}[i] := \text{DLINPROD}(j, 1, n, A[i,j], x[j], -b[i]);

\end

\text{end etappe;}

\end

\text{MINIMAX APPROXIMATION;}

\]
3. References


Errata

page 6: \( c^T U^{-1} \) in rule 18 should be \( c^T Q U^{-1} \).

page 7: PL in rule 19 should be \( \tilde{P} \).

page 8: The first paragraph should be replaced by:

Let \( F := Q_s F_s Q_{s+1} F_{s+1} \ldots Q_{n-1} F_{n-1} \).

\( F \) has the form:

\[
F = \begin{pmatrix}
I_{s-1} & 0 \\
\frac{1}{C_{n-s+1}} & 0 \\
0 & C_{n-s+1}
\end{pmatrix}.
\]

\( I_{s-1} \) denotes the unit-matrix or order \((s - 1)\).

It is easy to prove that all elements of \( C_{n-s+1} \) are, in absolute value, equal to or less than one, and therefore \( \|F\| \leq n - s + 1 \).

page 8: \( 2^k \) in rule 13 should be \( n^k \).

page 9: \( M \) in rule 15 should be replaced by \( M^{-1} \).

page 11: \( M \) in rule 10 should be \( M^{-1} \).

page 12: the last formula should be:

\[
\rho = \left| -\lambda_{n+1} \lambda_{n+1} | \left( \sum_{i} \lambda_i \right)^{-1} \right|.
\]