L\[infinity sign]\]- norm problem and mid-range polish

Elsayed A. M. Eltelt

Iowa State University
INFORMATION TO USERS

This reproduction was made from a copy of a manuscript sent to us for publication and microfilming. While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. Pages in any manuscript may have indistinct print. In all cases the best available copy has been filmed.

The following explanation of techniques is provided to help clarify notations which may appear on this reproduction.

1. Manuscripts may not always be complete. When it is not possible to obtain missing pages, a note appears to indicate this.

2. When copyrighted materials are removed from the manuscript, a note appears to indicate this.

3. Oversize materials (maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or in black and white paper format.*

4. Most photographs reproduce acceptably on positive microfilm or microfiche but lack clarity on xerographic copies made from the microfilm. For an additional charge, all photographs are available in black and white standard 35mm slide format.*

*For more information about black and white slides or enlarged paper reproductions, please contact the Dissertations Customer Services Department.
Eltelt, Elsayed A. M.

L(\infty)-NORM PROBLEM AND MID-RANGE POLISH

Iowa State University

Ph.D. 1986

University Microfilms International

300 N. Zeeb Road, Ann Arbor, MI 48106
L∞- Norm Problem and Mid-Range Polish

by

Elsayed A. M. Eltelt

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major: Mathematics

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1986
# TABLE OF CONTENTS

1. INTRODUCTION .................................................. 1

2. STATEMENT OF PROBLEM ........................................ 11
   2.1. Two-Way Tables with Single Observations ............. 11
   2.2. General Two-Way Table .................................. 19
   2.3. General Problem on a Discrete Set .................... 21

3. MINIMIZING THE MAXIMUM ABSOLUTE RESIDUAL AND MID-RANGE POLISH ..... 23
   3.1. Mid-Range Polish ....................................... 23
   3.2. Properties of Mid-Range Polish Operator .............. 28
   3.3. Convergence of Mid-Range Polish Algorithm ........... 35

4. FINITE ALGORITHM FOR THE L∞-PROBLEM ..................... 55
   4.1. Introduction ............................................ 55
   4.2. Phase I Algorithm ...................................... 58
   4.3. Phase II Algorithm ..................................... 68
   4.4. Justification of the Algorithm ......................... 78

5. ANOTHER ALGORITHM FOR THE L∞-PROBLEM FOR COMMENSURABLE DATA .... 82
   5.1. Introduction ............................................ 82
   5.2. Phase II Algorithm ..................................... 85
   5.3. Justification of the Algorithm ......................... 107
   5.4. How We Can Fix the Algorithm for General Data ....... 112

6. REFERENCES .................................................. 123

7. KNOWLEDGEMENTS .............................................. 123

8. APPENDIX ..................................................... 127
1. INTRODUCTION

The $L_\infty$-norm has been widely studied as a criterion of curve fitting problems. We are interested in the best $L_\infty$-approximation to a given finite array of numbers $A = (a_{ij})_{m \times n}$, i.e.

$$\min_{r,s} \max_{i,j} |a_{ij} - r_i - s_j|.$$

A natural iterated polishing (Mid-Range Polish) algorithm is shown, and its convergence in the $L_\infty$-norm is proved. Since the convergence of the Mid-Range Polish algorithm may take infinitely many iterations, we developed a new algorithm which converges in a finite number of steps to an optimal matrix of residual $A^* = (a_{ij}^* - r_i^* - s_j^*)_{m \times n}$, whose $L_\infty$-norm $u$ is the minimum one.

Several conditions are obtained, each of which is necessary and sufficient for a given matrix of residual to be optimal. For instance, a matrix of residual is optimal if and only if the set of entries, which equal to the $L_\infty$-norm of the matrix $u$ in absolute value, contain a loop $L$ alternating the value of $u$ and $-u$. This criterion leads to an elegant and efficient finite algorithm for calculating the best $L_\infty$-approximation. Examples and results of the computational experience with a computer code version of some of the algorithms are presented.

Since Least Square Fitting may be quite inappropriate when the data are uniformly distributed, the Chebychev, or $L_\infty$-norm, presented here, is maximum likelihood for the uniform distribution. Some applications, where the data follow a uniform or near uniform distribution, include: rounding
errors (e.g., in numerical analysis) follow a uniform distribution; some truncated normal distribution (e.g., in standardized exams such as the SAT, where low scores are truncated); and analysis of some worst case circumstances, where the goal is to insure that the largest error is made as small as possible.

The best $L_p$-approximation, $\sum a_i \phi_i(x)$ to a given finite array of numbers $a(x)$, $(x \in X)$, has many interesting applications. For the case $p = 2$, the Least Square Fitting is widely used. For the case $p = 1$, the Least Absolute Deviation (LAD), the $L_1$-norm, for one dimensional case, (one-way table) has been studied by Abdelmalak (1971, 1974, 1975), Anderson and Steiger (1982), Armstrong and Godfrey (1979), Barrodale and Roberts (1973, 1974), Bartels and Conn (1977), Bartels, Conn and Sinclair (1978), Bloomfield (1982), Osborne (1971), and Roberts and Ben-Israel (1969).

The two dimensional case, (two-way table), was considered by Armstrong and Frome (1979) and Armstrong, Elam and Hultz (1977). In both cases they solve the $L_1$-problem as a linear programming problem and use a modified simplex algorithm.

Another approach, for two dimensional case, by using the median polish, was introduced by Tukey (1970), McNeil (1977), Mosteller and Tukey (1977), Velleman and Hoaglin (1981), Anscombe (1981), Siegel (1983), and Kemperman (1984) who uses both approaches and prove also the convergence only in case $1 < p < \infty$.

For the case $p = \infty$, the Chebyshev, $L_\infty$-norm, the only work represented, the one dimensional case (one-way table): Stiefel (1960),
Barrodale and Young (1965), Barrodale and Phillips (1975), Armstrong and Kung (1980), and Sklar and Armstrong (1983). They solve the $L^\infty$-problem as a linear programming problem using a modified simplex algorithm. This tends to be inefficient because a lot of time is spent in degenerate steps.

In this work we solve the two-dimensional case (two-way table), first by using The Mid-Range Polish instead of the median polish introduced by Tukey. In Chapter 3 we give an elegant optimality criterion for the minimum $L^\infty$-norm matrix. The Mid-Range Polish Algorithm is a very fast algorithm for any $M \times N$ dimensional matrix, and we give a mathematical proof for the convergence in the $L^\infty$-norm. We also give the fortran code for this algorithm. In Chapter 4 we introduce a new Finite Algorithm using the advantage of the optimality criterion of the Mid-Range Polish Algorithm and insure the convergence in a finite number of steps by implementing the linear programming theorems. The algorithm gives very good results when compared to any simplex algorithm version. In Chapter 5 we give another Finite Algorithm which is very easy to manipulate either by hand or by computer, and the proof of the convergence in a finite number of steps for both rational and real data.

Consider the following linear programming problems, called the primal problem and the dual problem, respectively,
The matrix $A$ is $m \times n$ with columns $a_j$ for $j = 1,2,\ldots,n$, the vector $c$ is $n \times 1$, and the vector $b$ is $m \times 1$. Define the sets

$$X = \{x | Ax > b, \ x > 0\}$$

$$U = \{u | uA < c, \ u > 0\} \quad (1.1)$$

1.1. **Lemma** (Linear Programming Weak Duality).

Suppose $\tilde{x} \in X$ and $\tilde{u} \in U$. Then

$$c\tilde{x} > \tilde{u}b \quad \blacksquare$$

1.1. **Corollary**

If $\ z^* \in X$ and $\ u^* \in U$ satisfy $c\ z^* = u^*b$, then $\ z^*$ is optimal in (P) and $\ u^*$ is optimal in (D). $\blacksquare$
The following theorem says that when feasible solutions exist for both the primal and the dual, then the conditions of Corollary 1.1 obtain, namely, the optimal objective function values are equal.

1.1. **Theorem** (Linear Programming Strong Duality).

Suppose \( X \neq \emptyset \) and \( U \neq \emptyset \). Then there exists an \( x^* \) optimal in \( (P) \) and a \( u^* \) optimal in \( (D) \) and \( cx^* = u^* b \). □

When \( X \neq \emptyset \) and \( U = \emptyset \), then Theorem 1.1 tells us that \( (P) \) has no optimal solution because otherwise \( U \) would be nonempty. By our convention on unbounded and infeasible linear programming problems, we still have equality of primal and dual objective function values, namely, \( w = z = -\infty \). The same argument applies when \( U \neq \emptyset \) and \( X = \emptyset \), in which case we must have that \( (D) \) has no optimal solution and \( z = w = +\infty \). Finally, it is possible to construct a linear programming problem \( (P) \) such that \( X = \emptyset \) and its dual \( (D) \) is such that \( U = \emptyset \).

**Example** \( \min z = x_1 - 2x_2 \)

such that

\[
\begin{align*}
x_1 - x_2 & \geq 2 \\
-x_1 + x_2 & \geq -1 \\
x_1 & > 0, \ x_2 > 0
\end{align*}
\]

The consequences of duality theory to the construction of simplex and non-simplex algorithms for linear programming is best summarized by the following corollary to Lemma 1.1 and Theorem 1.1.
1.2. Corollary

The solutions \( \tilde{x} \in X \) and \( \tilde{u} \in U \) are optimal in (P) and (D), respectively, if and only if

\[
\tilde{u}(\tilde{A}\tilde{x} - b) = 0
\]

and

\[
(c - \tilde{u}A)\tilde{x} = 0
\]

(1.2)

Proof. Since \( \tilde{x} \in X, \tilde{u} \in U \), we have \( \tilde{A}\tilde{x} > b \) and \( \tilde{u} > 0 \) implying

\( \tilde{u}\tilde{A}\tilde{x} > \tilde{u}b \).

Similarly, we have \( \tilde{u}\tilde{A} < c \) and \( \tilde{x} > 0 \) implying \( \tilde{u}\tilde{A}\tilde{x} < \tilde{c}\tilde{x} \). Thus,

\( \tilde{u}b < \tilde{u}\tilde{A} \tilde{x} < \tilde{c}\tilde{x} \). If \( \tilde{u}(\tilde{A}\tilde{x} - b) = 0 \) and \( (c - \tilde{u}A)\tilde{x} = 0 \), we have \( \tilde{u}b = \tilde{c}\tilde{x} \) and \( \tilde{x} \) and \( \tilde{u} \) are optimal in (P) and (D) by Corollary 1.1.

If one of the conditions (1.2) does not hold, then we must have

\( \tilde{u}b < \tilde{c}\tilde{x} \) and at least one of the solutions \( \tilde{x}, \tilde{u} \) is not optimal in its respective problem.

Conditions (1.2) are called complementary slackness conditions and they state that a primal (dual) variable can be positive only if the slack (surplus) variable in the corresponding dual (primal) constraint is zero.

Thus, there are three sets of conditions that must be met in order to conclude that a solution \( x \) is optimal in a given (primal) linear programming problem. These are:

1. Primal feasibility \( (x \in X) \)
2. Dual feasibility \( (u \in U) \) and
3. Complementary slackness

\[
u(Ax - b) = 0, \quad (c - uA)x = 0.
\]
The simplex algorithm takes implicitly at each iteration the \( m \)-vector of shadow prices as trial dual variables to complement the primal basic feasible solution. This choice of a primal-dual pair ensures that the algorithm maintains at each iteration \textit{primal feasibility} and \textit{complementary slackness}.

The algorithm terminates with optimal solutions to the primal and the dual when dual feasibility of the shadow prices is also attained. Variants of the simplex method are based on maintaining some of these conditions and performing simplex iterations until all of them are satisfied. For example, the dual simplex algorithm maintains dual feasibility and complementary slackness until primal feasibility is achieved.

The primal-dual simplex algorithm maintains dual feasibility until primal feasibility and complementary slackness are simultaneously obtained.

A \textit{loop in a tableau} is a sequence of cells in the tableau that satisfies the following criteria:

1. The sequence consists of horizontal and vertical segments arranged so that the directions of the segments alternate.
2. Each segment joins exactly two cells.
3. The first cell of the sequence is the last, and no other cell is used twice.
Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-6</td>
<td>7</td>
</tr>
<tr>
<td>-1</td>
<td>4</td>
<td>-9</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>-2</td>
<td>2</td>
<td>-5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>-7</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Properties 1 and 2 tell us that if \((i,j)\) is a cell in a loop and if we reached it horizontally (along row \(i\)), then the next cell in the loop must be in column \(j\). Likewise, if we reached cell \((i,j)\) vertically, then the next cell in the loop must be in row \(i\). Consequently, we use the cells in each row two at a time when forming a loop.

A loop must therefore have an **even number** of cells in it.

**Tree:** A tree is a graph characterized by the following equivalent properties:

1. It has \(m\) nodes, \(m-1\) edges, and is connected.
2. It has \(m\) nodes, \(m-1\) edges, and no cycles.
3. There is a unique path from each node to every other node.
4. It has no cycles but exactly one cycle is created by adding an edge.
5. It is connected but ceases to be connected if any edge is removed.
A Spanning Tree  A spanning tree $T$ of a graph $G$ is a spanning subgraph of $G$ and a tree.

Degree of a node is the number of arcs attached to this node.

Leaf: an end of a tree, i.e. a node of degree one.

The figure formed by connecting the basic cell by means of horizontal and vertical lines will be called an espalier, which is a tree growing on a trellis.
The espalier is useful because it is before one's eyes in the tableau which easily accommodates nonbasic variable as well as basic.

1-Tree: A 1-tree defined on a node set $1, 2, \ldots, m$ is a graph consisting of a tree on the nodes $2, 3, \ldots, m$, together with two edges connecting node 1 to the tree. i.e., a connected graph having exactly one cycle. For the one-tree, the number of arcs is equal to the number of nodes.
2. STATEMENT OF PROBLEM

2.1. Two-Way Tables with Single Observations

The Problem:

Suppose that we have a matrix (two way table) of data \( A = (a_{ij})_{m \times n} \), and we wish to find numbers \( r = (r_i)_{m \times 1} \) and \( s = (s_j)_{n \times 1} \) such that \( \max |a_{ij} - r_i - s_j| \) is minimized,

\[
\text{i.e. } \min_{r,s} \max_{i,j} |a_{ij} - r_i - s_j|.
\]

This problem can be transformed to a linear programming problem by putting \( \max |a_{ij} - r_i - s_j| = u \) for any \( r \in \mathbb{R}^m \) and \( s \in \mathbb{R}^n \), this gives the linear programming problem:

\[
\begin{align*}
\min u \quad & \text{when} \quad -u < a_{ij} - r_i - s_j < u \\
\text{or} \quad & u + r_i + s_j > a_{ij} \\
\min u \quad & \text{when} \quad u - r_i - s_j > -a_{ij} \\
& \quad i = 1,2,...,m \quad \quad j = 1,2,...,n.
\end{align*}
\]
Tableau 2.1.1. Simplex Tableau
From the compact (Tableau 2.1.1) the dual of (2.1) is

\[
\max_{x,y} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} (y_{ij} - x_{ij})
\]

when

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij} + x_{ij}) = 1 ;
\]

\[
\sum_{j=1}^{n} (y_{ij} - x_{ij}) = 0 \quad i = 1,2,...,m; \quad (2.2)
\]

\[
\sum_{i=1}^{m} (y_{ij} - x_{ij}) = 0 \quad j = 1,2,...,n;
\]

\[x_{ij} > 0, \quad y_{ij} > 0\]

for all i and j.

**Complementary Slackness:**

The conditions for complementary slackness are

\[y_{ij} (u + r_i + s_j - a_{ij}) = 0, \quad i = 1,2,...,m, \quad j = 1,2,...,n. \quad (2.3.A)\]

\[x_{ij} (u - r_i - s_j + a_{ij}) = 0, \quad i = 1,2,...,m, \quad j = 1,2,...,n. \quad (2.3.B)\]

For any feasible solution for (2.1) and (2.2) put

\[w_{ij} = y_{ij} - x_{ij} \quad \forall (i,j).\]

If \( u = 0 \), we have a trivial problem.
Now for \( u \neq 0 \), if

1. \(-u < a_{ij} - r_i - s_j < u\), then by the right inequality, \( u + r_i + s_j - a_{ij} > 0 \) which implies by (2.3.A) that \( y_{ij} = 0 \). And by the left inequality, \( u - r_i - s_j + a_{ij} > 0 \) which implies by (2.3.B) that \( x_{ij} = 0 \), therefore \( w_{ij} = y_{ij} - x_{ij} = 0 \) for all \((i,j)\) such that \(-u < a_{ij} - r_i - s_j < u\).

2. \( a_{ij} - r_i - s_j = -u > -u\), then \( u - r_i - s_j + a_{ij} > 0 \) implies by (2.3.B) that \( x_{ij} = 0 \), so \( w_{ij} = y_{ij} - x_{ij} = y_{ij} > 0 \) for all \((i,j)\) such that \( a_{ij} - r_i - s_j = u \).

3. \( a_{ij} - r_i - s_j = -u < u\), then \( u + r_i + s_j - a_{ij} < 0 \) and by (2.3.A) \( y_{ij} = 0 \). Thus, \( w_{ij} = y_{ij} - x_{ij} = -x_{ij} < 0 \) for all \((i,j)\) such that \( a_{ij} - r_i - s_j = -u \). Therefore, we can conclude that \( |w_{ij}| = y_{ij} + x_{ij} \) for all \((i,j)\).

Our problem can be written as: The Primal

\[
\text{(Primal)} \quad \max \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} w_{ij} \]

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} |w_{ij}| = 0 \quad (2.4)
\]

\[
\sum_{i=1}^{m} w_{ij} = 0 \quad j = 1, 2, \ldots, n
\]

\[
\sum_{j=1}^{n} w_{ij} = 0 \quad i = 1, 2, \ldots, m
\]
and its dual

\[(\text{Dual}) \quad \min_{r,s,u} \quad u \quad \text{when} \quad -u < a_{ij} - r_i - s_j < u \quad i = 1,2,\ldots,m, \quad j = 1,2,\ldots,n.\]

2.1.1. Lemma

Under the conditions

\[\sum_{i=1}^{m} w_{ij} = 0 \quad j = 1,2,\ldots,n \quad \text{and} \quad \sum_{j=1}^{n} w_{ij} = 0 \quad i = 1,2,\ldots,m,\]

the two problems

\[\max_{w} \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} w_{ij} \quad \text{and} \quad \max_{w} \sum_{j=1}^{n} \sum_{i=1}^{m} \left( a_{ij} - r_i - s_j \right) w_{ij}\]

where \(r_i\)'s and \(s_j\)'s are constants, have the same solution \(w^* = (w^*_{ij})\).

Proof: For any \((w_{ij})\) satisfying the hypotheses, we have

\[\sum_{i=1}^{m} \sum_{j=1}^{n} \left( a_{ij} - r_i - s_j \right) w_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} w_{ij} - \sum_{i=1}^{m} r_i \sum_{j=1}^{n} w_{ij} - \sum_{j=1}^{n} s_j \sum_{i=1}^{m} w_{ij}.\]

But \(\sum_{i=1}^{m} w_{ij} = 0\) and \(\sum_{j=1}^{n} w_{ij} = 0\), therefore

\[\sum_{i=1}^{m} \sum_{j=1}^{n} \left( a_{ij} - r_i - s_j \right) w_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} w_{ij}.\]
2.1.2. Lemma

The primal problem (2.4) and the dual problem (2.5) are always feasible.

Proof: For the primal problem (2.4), take any loop of cells $L$ and put

$$w_{ij} = \begin{cases} \pm \frac{1}{|L|} & \forall (i,j) \in L \\ 0 & \text{otherwise} \end{cases}$$

where $|L|$ is the number of cells in the loop $L$, and let $w_{ij}$ alternate the sign $+$ and $-$ through the cells of $L$.

From the definition of the loop, we have:

$$\sum_{i=1}^{m} w_{ij} = 0 \quad j = 1,2,...,m$$

$$\sum_{j=1}^{n} w_{ij} = 0 \quad i = 1,2,...,m$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} |w_{ij}| = 1 .$$

Therefore, $\{w_{ij}\}$ is a feasible solution for the primal problem (2.4).

For the dual problem (2.5), take

$$u = \max_{i,j} |a_{ij}| \quad \text{and} \quad r_i = s_j = 0 \quad \forall i,j ,$$

implying $\{u, r_i, s_j\}$ is feasible since $-u < a_{ij} - r_i - s_j < u$.  \blacksquare
Now by Theorem (1.1) (strong duality), there exist optimal solutions $w^*$ for the primal problem (2.4) and $u^*$ for its dual (2.5), and moreover
\[ \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} w_{ij}^* = u^* . \]

Define:

**Condition (A):** If $w_{ij} > 0$, then $a_{ij} - r_i - s_j = u > 0$.

**Condition (B):** If $w_{ij} < 0$, then $a_{ij} - r_i - s_j = -u < 0$.

**Condition (C):** If $w_{ij} = 0$, then $-u < a_{ij} - r_i - s_j < u$.

2.1.1. Theorem

The numbers $\{w_{ij}\}$, $i = 1, 2, \ldots, m$, $j = 1, \ldots, n$ are optimal for the primal problem (2.4) and $\{r_i\}_{i=1}^{m}, \{s_j\}_{j=1}^{n}, u$ are optimal for the dual problem (2.5) if and only if Conditions A, B, C and D hold.
Proof: Suppose \( \{w_{ij}\} \) are optimal for the primal problem (2.4) and \( \{r_i\}, \{s_j\}, u \) are optimal for the dual problem (2.5).

Therefore \( \{w_{ij}\} \) is feasible, implying Condition A holds.

Now, if \( w_{ij} > 0 \), \( w_{ij} = y_{ij} - x_{ij} \) implies \( y_{ij} > 0 \), since \( x_{ij} > 0 \). Therefore from (2.3.A), \( u + r_i + s_j - a_{ij} = 0 \), i.e.

\[
a_{ij} - r_i - s_j = u > 0. \text{ Therefore Condition B holds.}
\]

If \( w_{ij} < 0 \), \( w_{ij} = y_{ij} - x_{ij} \) implies \( x_{ij} > 0 \) since \( y_{ij} > 0 \). From (2.3.B) \( u - r_i - s_j + a_{ij} = 0 \). Therefore, \( a_{ij} - r_i - s_j = -u < 0 \). Therefore, Condition C holds.

If \( w_{ij} = 0 \), \( w_{ij} = y_{ij} - x_{ij} \) implies \( x_{ij} = y_{ij} = 0 \). Otherwise, if \( x_{ij} = y_{ij} > 0 \) implying \( u = 0 \) which contradicts our assumption that \( u \neq 0 \) for non-trivial problem. Therefore, \( \{r_i\}, \{s_j\}, u \) feasible implies that Condition D holds.

Conversely, suppose the Conditions A, B, C, and D hold. From Conditions A and D, \( \{w_{ij}\} \) and \( \{r_i\}, \{s_j\}, u \) are feasible for the primal (P) and the dual (D) respectively. Therefore, from Corollary (1.1) it is enough to show that

\[
u = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} w_{ij}.
\]
Let \( S = \{(i,j) \mid w_{ij} \neq 0\} \), from Conditions B and C we have

\[
\sum_{i,j} (a_{ij} - r_i - s_j)w_{ij} = \sum_{(i,j) \in S} |a_{ij} - r_i - s_j|w_{ij} + \sum_{(i,j) \notin S} (a_{ij} - r_i - s_j)w_{ij}
\]

\[
= \sum_{(i,j) \in S} u|w_{ij}|
\]

\[
= u \sum_{(i,j) \in S} |w_{ij}|
\]

\[
= u \sum_{i,j} |w_{ij}|
\]

\[
= u .
\]

But \( \sum_{i,j} a_{ij}w_{ij} = \sum_{i,j} (a_{ij} - r_i - s_j)w_{ij} = u . \)

### 2.2. General Two-Way Table

A two-way \( m \times n \) \((m > 2, n > 2)\) layout with observations \( a_{ijk} \) in cell \((i,j)\) with \( i \in \{1,2,\ldots,m\} \); \( j \in \{1,2,\ldots,n\} \) and \((k = 1,2,\ldots,k_{ij})\) where \( k_{ij} = 0 \) is possible, when one wants to minimize

\[
\max_{i,j,k} |a_{ijk} - r_i - s_j|
\]

This more general case becomes important in application where one has a large number of observations and one likes to keep \( m \) and \( n \) relatively small so as to simplify the calculation. This problem can be transformed
to the linear programming problem:

\[
\min_{r,s,u} u
\]

such that

\[-u + a_{ijk} - r_i - s_j < u \quad i = 1, \ldots, m
\]
\[j = 1, \ldots, n
\]
\[k = 1, \ldots, k_{ij}
\]

whose dual is

\[
\max_{w} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{k_{ij}} a_{ijk} w_{ijk}
\]

such that

\[
\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{k_{ij}} |w_{ijk}| = 1,
\]
\[
\sum_{k=1}^{k_{ij}} w_{ijk} = 0 \quad i = 1, \ldots, m,
\]
\[
\sum_{i=1}^{m} \sum_{k=1}^{k_{ij}} w_{ijk} = 0 \quad j = 1, \ldots, n.
\]
And the corresponding Conditions A, B, C and D are:

**Condition (A):**
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{k_{ij}} |w_{ijk}| = 1, \quad \sum_{j=1}^{n} \sum_{k=1}^{k_{ij}} w_{ijk} = 0, \quad i = 1, 2, \ldots, m,
\]
\[
\quad \text{and} \quad \sum_{i=1}^{m} \sum_{k=1}^{k_{ij}} w_{ijk} = 0 \quad j = 1, 2, \ldots, n.
\]

**Condition (B):**
If \( \sum_{k=1}^{k_{ij}} w_{ijk} > 0 \) then \( a_{ijk} - r_i - s_j = u > 0 \).

**Condition (C):**
If \( \sum_{k=1}^{k_{ij}} w_{ijk} < 0 \) then \( a_{ijk} - r_i - s_j = -u < 0 \).

**Condition (D):**
If \( \sum_{k=1}^{k_{ij}} w_{ijk} = 0 \) then \( -u < a_{ijk} - r_i - s_j < u \).

### 2.3. General Problem on a Discrete Set

Let \( I \) be a fixed index set,

\[
A = \{a(x) \mid x \in I\} \quad \text{a given collection of real numbers or observations.}
\]

\[
\phi_r : I \rightarrow \mathbb{R} \quad (r = 1, 2, \ldots, M) \quad \text{is a given set of linearly independent functions on } I.
\]
The problem is

\[ \min_{\{a_1, a_2, \ldots, a_M\}} \max_{x \in I} \{ w(x) | a(x) - \sum_{r=1}^{M} a_r \phi_r(x) | \} \]

The weights \( w(x) > 0 \) may indicate the multiplicity or importance of the corresponding observations.

Our special case can be derived from this general form by taking

- \( I \) as the set of triplets \((i, j, k)\) with \( i \in \{1, 2, \ldots, m\}; \ j \in \{1, 2, \ldots, n\}\)
- and \( k = 1, 2, \ldots, k_{ij} \)

and \( a(x) = a_{ijk} \) in cell \((i,j)\), while \( w(x) = 1 \).

Further \( M = m + n \) and

\[
\phi_r(x) = \phi_r(i, j, k) = \delta_i^r \text{ for } r = 1, 2, \ldots, m
\]

\[
= \delta_j^{r-m} \text{ for } r = m+1, m+2, \ldots, M
\]

the problem becomes:

\[
\min_{\alpha} \max_{i, j, k} |a_{ijk} - a_i - a_j|
\]
3. MINIMIZING THE MAXIMUM ABSOLUTE RESIDUAL
AND MID-RANGE POLISH

3.1. Mid-Range Polish

If \( \{x_1, x_2, \ldots, x_n\} \) is a set of numbers and
\[ \text{MAR}(t) = \max_{1 \leq i \leq n} \{|x_i - t|\} \]
the maximum absolute residual, it is easy to see that \( \text{MAR}(t) \) is minimized by taking \( t = \frac{\min\{x_i\} + \max\{x_i\}}{2} \). The number \( t \) is called the mid-extreme or mid-range of the set \( \{x_1, x_2, \ldots, x_n\} \). Tukey (1970) developed in detail the idea of calculating a reasonably good additive approximation to a given n-way layout of observations by so-called Median Polish. We will do the polish by using instead "the Mid-Range".

Mid-Range Polish for Two-Way Tables:

For a two-way table \( A = (a_{ij})_{m \times n} \), an additive approximation
\( (r_i + s_j) \) to \( a_{ij} \) is derived as follows: Start with a matrix
\( A^{(0)} = (a_{ij}^{(0)})_{m \times n} \) and apply a Row Mid-Range Polish (RMRP), yielding the matrix \( A^{(1)} = (a_{ij}^{(1)})_{m \times n} \) where
\[ a_{ij}^{(1)} = a_{ij}^{(0)} - a_i^{(1)} \quad \text{for each} \ i = 1, 2, \ldots, m. \]

Here, the adjustment \( a_i^{(1)} \) of the \( i \)th row is taken as the fixed mid-range of the set of numbers \( \{a_{ij}^{(0)}\}_{j=1}^n \) in the \( i \)th row with \( i \) fixed.
Next, apply a Column Mid-Range Polish (CMRP) to matrix $A^{(1)}$ yielding the matrix $A^{(2)} = (a_{ij})_{m \times n}$ where

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \beta_j^{(1)} \quad \text{for each } j = 1, 2, \ldots, n.$$

Here, the adjustment $\beta_j^{(1)}$ of the $j$th column is taken as the fixed mid-range of the set of numbers $\{a_{ij}^{(1)}\}_{i=1}^m$ with $j$ fixed. This is considered as one complete iteration.

### 3.1.1. Example

$$A^{(0)} = \begin{bmatrix} 5 & -3 & 1 \\ 6 & -2 & 7 \\ -4 & 8 & -1 \end{bmatrix}$$

**1st iteration**

<table>
<thead>
<tr>
<th>$a_{11}^{(1)}$</th>
<th>$\beta_j^{(1)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5/2</td>
</tr>
<tr>
<td>-4</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

**Row polish**

- $5 - 3 = 2$  
- $6 - 2 = 4$  
- $-4 + 8 = 4$  
- $-3 + 1 = -2$  
- $1 - 3 = -2$  
- $-1 + 3 = 2$

**Column polish**

- $7/2$  
- $-9/2$  
- $9/2$  
- $-6$  
- $6$  
- $-3$
Polishing the rows of $A^{(2)}$ one obtains $A^{(3)}$ and so on ... . In general

\[
2k+1 \quad 3k+1 = a_{ij} - a_{i1} ,
\]

\[
2k+2 \quad 2k+1 = a_{ij} - b_{ij} ,
\]

$k = 0, 1, 2, \ldots$ .

For example (3.1.1),

\[
A^{(4)} = \begin{bmatrix}
5 & -5 & -1 \\
5 & -5 & 4 \\
-5 & 5 & -4
\end{bmatrix}.
\]
We will give the criterion by which one can check if the matrix has reached its minimum $L_\infty$-norm or not.

The optimal weights (or effects) can be calculated as:

$$\alpha_i^* = \sum_k \alpha_i^{(k)} \quad i = 1,2,\ldots,m,$$

$$\beta_j^* = \sum_k \beta_j^{(k)} \quad j = 1,2,\ldots,n.$$

For our example:

<table>
<thead>
<tr>
<th></th>
<th>$-9/8$</th>
<th>$7/8$</th>
<th>$7/8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>8</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

leads to

$$A^* = \begin{bmatrix} 5 & -5 & -1 \\ 5 & -5 & 4 \\ -5 & 5 & -4 \end{bmatrix} = (a_{ij} - \alpha_i^* - \beta_j^*)_{3\times3}.$$

Note: The MRP algorithm may take an infinite number of steps to converge.

The following example illustrates the infinite convergence:

3.1.2. Example

$$A = \begin{bmatrix} 3 & 7 & 5 \\ -2 & 1 & 4 \\ 7 & 3 & 7 \end{bmatrix}$$
1st iteration

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>7</th>
<th>5</th>
<th>5</th>
<th>-1/2</th>
<th>0</th>
<th>3/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_0$</td>
<td>-2</td>
<td>1</td>
<td>4</td>
<td>1 +</td>
<td>-3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>-2</td>
<td>2</td>
</tr>
</tbody>
</table>

2nd iteration

<table>
<thead>
<tr>
<th></th>
<th>-3/2</th>
<th>4/2</th>
<th>-3/2</th>
<th>1/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>-5/2</td>
<td>0</td>
<td>3/2</td>
<td>-2/4 +</td>
</tr>
<tr>
<td></td>
<td>5/2</td>
<td>-1/2</td>
<td>1/2</td>
<td>1/4</td>
</tr>
</tbody>
</table>

3rd iteration

<table>
<thead>
<tr>
<th></th>
<th>-15/8</th>
<th>16/8</th>
<th>-15/8</th>
<th>1/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_2$</td>
<td>-17/8</td>
<td>6/8</td>
<td>15/8</td>
<td>-2/16 +</td>
</tr>
<tr>
<td></td>
<td>17/8</td>
<td>-16/8</td>
<td>1/8</td>
<td>1/16</td>
</tr>
</tbody>
</table>

4th iteration

<table>
<thead>
<tr>
<th></th>
<th>-63/32</th>
<th>64/32</th>
<th>-63/32</th>
<th>1/64</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_3$</td>
<td>-65/32</td>
<td>30/32</td>
<td>63/32</td>
<td>-2/64 +</td>
</tr>
<tr>
<td></td>
<td>65/32</td>
<td>-64/32</td>
<td>1/32</td>
<td>1/64</td>
</tr>
</tbody>
</table>

\[ Aj = -\frac{5}{2} 0 \frac{3}{2} -\frac{2}{4} ]
5th iteration

\[
\begin{bmatrix}
-255/128 & 256/128 & -255/128 \\
-257/128 & 126/128 & 255/128 \\
257/128 & -256/128 & 1/128 \\
\end{bmatrix}
\]

\[A_4 = \begin{bmatrix}
-257/128 & 126/128 & 255/128 \\
257/128 & -256/128 & 1/128 \\
\end{bmatrix}
\]

One can show that for \( n > 1 \),

\[
A_n = \begin{bmatrix}
-2 + \frac{1}{2^{2n-1}} & 2 & -2 + \frac{1}{2^{2n-1}} \\
-2 - \frac{1}{2^{2n-1}} & 2^{2n-1} - 2 & 2 - \frac{1}{2^{2n-1}} \\
2 + \frac{1}{2^{2n-1}} & -2 & \frac{1}{2^{2n-1}} \\
\end{bmatrix}
\]

As \( n \to \infty \), \( A_n \to A^* \), where

\[
A^* = \begin{bmatrix}
-2 & 2 & -2 \\
-2 & 1 & 2 \\
2 & -2 & 0 \\
\end{bmatrix}
\]

which will later be shown to be the correct answer.

3.2. Properties of Mid-Range Polish Operator

We can consider RMRP as an operator \( R \), \( R : A^{mxn} \to A^{mxn} \) defined on the space of all \( m \times n \) matrices \( A^{mxn} \) in such a way that \( R(A) = \overline{A} \) where \( \overline{A} \) is RMRP matrix obtained by subtracting the midrange of each row from the elements of that row.
Also, we can consider CMRP as an operator \( C : \mathbb{R}^{m \times n} \mapsto \mathbb{R}^{m \times n} \) in such a way that \( C(A) = A_1 \) where \( A_1 \) is CMRP matrix obtained by subtracting the midrange of each column from the elements of that column.

Then, one complete iteration on a matrix \( A \) can be considered as a composition of the two operators \( \text{CoR} \), i.e., the 1st iteration

\[ \text{CR}(A) = A_1, \]

a CMRP-matrix. The 2nd iteration

\[ (\text{CR})^2A = (\text{CR})(\text{CR})A = \text{CR}(A_1) = A_2 \]

where \( A_2 \) is CMRP-matrix, and the nth iteration

\[ (\text{CR})^n(a) = (\text{CR})A_{n-1} = A_n, \]

a CMRP-matrix. Therefore, the polishing procedure produces a sequence of CMRP-matrices \( \{A_n\} \) where

\[ \{A_n\}_{n=1}^{\infty} = \{A_1, A_2, \ldots, A_n, \ldots\} \]

\[ = \{\text{CR}(A), (\text{CR})^2A, \ldots, (\text{CR})^nA, \ldots\} \]

3.2.1. Lemma

No step of the mid-range polish algorithm can increase the \( L_\infty \)-norm of the matrix.

Proof: Let \( x = (x_1, x_2, \ldots, x_i, \ldots x_u, \ldots, x_n) \) be any typical row (or column) in the matrix \( A \), where
\[ x_l = \min_{1 < i < n} \{ x_i \} \quad \text{and} \quad x_u = \max_{1 < i < n} \{ x_i \} \]

Let \( R: \mathbb{R}^n \to \mathbb{R}^n \) be the corresponding component of the RMRP-operator, then

\[ R(x) = (\tilde{x}_1, \tilde{x}_2, \ldots, -s, \ldots, s, \ldots, \tilde{x}_n) \]

where \( \tilde{x}_i = x_i - \frac{x_u + x_l}{2} \quad i \neq l, u \)

\[ S = \frac{x_u - x_l}{2} > 0 \]

Now since \(|\tilde{x}_i| < |s|\), we have

\[ \|R(x)\|_\infty = s = \frac{|x_u - x_l|}{2} < \frac{|x_u| + |x_l|}{2} \]

\[ < \frac{2 \max\{|x_u|, |x_l|\}}{2} = \|x\|_\infty. \]

The same argument works for CMRP operator. \( \blacksquare \)

3.2.1. Corollary

Let \( s_k = \|A_k\|_\infty \), i.e., \( \{s_k\} \) is the sequence of numbers representing the \( \|\cdot\|_\infty \)-norm of the matrix at each iteration of the midrange polish algorithm. Then \( \lim_{k \to \infty} s_k \) exists.

Proof. By Lemma 3.2.1, \( \|A_n\|_\infty < \|A_{n-1}\|_\infty \) for all \( n \); hence \( \|A_n\|_\infty < \|A\|_\infty \). Thus \( \{s_k\} \) is a bounded non-increasing sequence of non-negative numbers, implying the sequence always converges to its greatest lower bound. \( \blacksquare \)
3.2.2. Lemma

The Mid-Range Polish is a continuous operator.

Proof. It is sufficient to prove that the Mid-Range Polish of a vector \( a \in \mathbb{E}^n \) is continuous. Let \( R : \mathbb{E}^n + \mathbb{E}^n \) be the component of the RMRF operator corresponding to the polishing of the vector 

\[ a = (a_1, a_2, \ldots, a_n). \]

Therefore, \( R(a) = a - a^* \) where

\[ a^* = \frac{\max\{a_k\} + \min\{a_k\}}{2} \quad \text{and} \quad 1 = (1, 1, \ldots, 1) \in \mathbb{E}^n. \]

Let \( \varepsilon > 0 \) be given, and take \( \delta = \varepsilon/2. \) If \( a, b \in \mathbb{E}^n \) with \( \|a - b\|_\infty < \delta \) then

\[ \|R(a) - R(b)\|_\infty = \|(a - a^*) - (b - a^*)\|_\infty \]

\[ = \|(a - b) + (a^* - b^*)\|_\infty \]

\[ < \|a - b\|_\infty + |a^* - b^*|. \]

Recall that

\[ (\max_{i \leq k} a_k - \max_{i \leq k} b_k) < (\max_{i \leq k} a_k - \max_{i \leq k} b_k) < \|a - b\|_\infty, \]

\[ (\min_{i \leq k} b_k - \min_{i \leq k} a_k) > (\min_{i \leq k} b_k - \min_{i \leq k} a_k), \quad \text{and} \quad \]
\[
\begin{align*}
&\left(\min_{k} a_{k} - \min_{k} b_{k}\right) = -\left(\min_{k} b_{k} - \min_{k} a_{k}\right) < -\min_{k}(b_{k} - a_{k}) \\
&= \max_{k}(a_{k} - b_{k}).
\end{align*}
\]

But
\[
\alpha_{a} - \alpha_{b} = \frac{\max_{k} a_{k} + \min_{k} a_{k}}{2} - \frac{\max_{k} b_{k} + \min_{k} b_{k}}{2}
\]
\[
= \frac{1}{2} \left(\max_{k} a_{k} - \max_{k} b_{k}\right) + \frac{1}{2} \left(\max_{k} a_{k} - \min_{k} b_{k}\right)
\]
\[
< \frac{1}{2} \max_{k}(a_{k} - b_{k}) + \frac{1}{2} \max_{k}(a_{k} - b_{k})
\]
\[
= \max_{k}(a_{k} - b_{k}) < \|a - b\|_{\infty}.
\]

Therefore, \(|\alpha_{a} - \alpha_{b}| < \|a - b\|_{\infty}\), and

\[
\|R(a) - R(b)\| < \|a - b\|_{\infty} + \|a - b\|_{\infty} = 2\|a - b\|_{\infty}
\]
\[
< 2 \frac{\varepsilon}{2} = \varepsilon.
\]

3.2.1. Definition

The matrices \(A = (a_{ij})_{m \times n}\) and \(B = (b_{ij})_{m \times n}\) are additively equivalent if there exist a vector \((R)_{m \times 1}\) of row effects and a vector \((S)_{n \times 1}\) of column effects such that

\[
b_{ij} = a_{ij} - r_{i} - s_{j} \quad \text{for all } (i,j).
\]

3.2.2. Definition

The matrix \(A = (a_{ij})_{m \times n}\) is said to be optimal if there is no additively equivalent matrix with lower \(L_{\infty}\)-norm.
3.2.3. Definition

A loop in a tableau is a sequence of cells in the tableau that satisfies the following criteria:

(a) The sequence consists of horizontal and vertical segments, arranged so that the direction of the segments alternate.

(b) Each segment joins exactly two cells.

(c) The first cell in the sequence is the last and no other cell is used twice.

It is clear that a loop has an even number of cells in each row and column.

3.2.4. Definition

For any matrix $A$ such that $\|A\|_\infty = u$, define

$$O_A = \{(i,j) \mid \text{where the } L_\infty \text{ norm of the matrix } A \text{ is attained, i.e. } |a_{ij}| = u\}.$$

The set $O_A \neq \emptyset$ plays an important role in deriving a criterion which is necessary and sufficient for $A$ to be optimal.

The next theorem gives this optimality criterion.

3.2.1. Theorem

A matrix $A$ with $\|A\|_\infty = u$ is optimal if and only if there exists a subset $L$ of $O_A$ such that $L$ can be arranged into a loop which alternates the values $u$ and $-u$. 
Proof: Suppose $A$ is optimal and $O_A$ contains no such loop, then there exist $(i_0, j_0) \in O_A$ such that \( a_{i_0 j_0} = u \) and no entry \( a_{i_0 k} \) in row \( i_0 \) (or column \( j_0 \)) has the value \(-u\).

Then a row polish results in all entries in row \( i_0 \) being less than \( u \) in absolute value. Therefore, \( O_A \) has at least one fewer element. In a finite number of steps each \( u \) or \(-u\) entry has been lowered. This contradicts the assumption that \( A \) is optimal.

Conversely, suppose \( O_A \) contains a loop \( L \), where \( L = \{(i_0, j_0), (i_0, j_1), \ldots, (i_k, j_0)\} \) whose cells alternate the values \( u \) and \(-u\). Construct the weights \((w_{ij})_{m \times n}\) of Theorem 2.1.1 in the following way:

\[
    w_{ij} = \begin{cases} 
    \frac{1}{k} & \text{If } a_{ij} = u > 0, \ (i,j) \in L, \\
    -\frac{1}{k} & \text{If } a_{ij} = -u < 0, \ (i,j) \in L, \\
    0 & \text{otherwise},
\end{cases}
\]

then \((w_{ij})_{m \times n}\) satisfies the conditions of Theorem 2.1.1, so the matrix \( A \) is optimal. 

3.2.2. Corollary.

A fixed point matrix of the mid range polish operations (R and C), is an optimal matrix.
Proof: A fixed point matrix of the mid range polish operator is a matrix \( \mathbf{A} \) all of whose rows and columns are MRP. If \( \| \mathbf{A} \|_\infty = u \) and \( a_{ij} = u \), then there exists \( k_1 \) such that \( a_{ik_1} = -u \) since row \( i \) is polished.

There exists \( k_2 \) such that \( a_{k_2k_1} = u \) since column \( k_1 \) is polished. Similarly there exists \( k_3 \) such that \( a_{k_2k_3} = -u \) since \( k_2 \) is polished. If \( k_3 = j \), we are done, else continue. In a finite number of steps some index must repeat. Therefore, \( O_A \) contains a loop which alternates the values \( u \) and \( -u \). By Theorem 3.2.1 the matrix \( \mathbf{A} \) is optimal. 

3.3. Convergence of Mid-Range Polish Algorithm

Consider the sequence of column polished matrices produced from the MRP algorithm

\[
[A_n]_{n=1}^\infty = \{A_1, A_2, \ldots, A_n, \ldots\}
\]

where \( A_n = (CR)^n(A) \).

By Lemma 3.2.1 this sequence is bounded, i.e.,

\[
\| A_n \|_\infty < K \text{ for all } n.
\]
So there exists a subsequence \( \{A_n^k\} \) of \( \{A_n\} \) which is convergent (say) to \( A^* \), i.e., \( A_n^k \to A^* \). We will use this subsequence for the rest of this section.

### 3.3.1. Lemma

Let \( \{B_n\} \) be a sequence of CMRP matrices, and let \( B_n \to B^* \), then \( B^* \) is CMRP matrix.

**Proof:** Let \( b_{nk}^n = (b_{1k}^n, b_{2k}^n, \ldots, b_{mk}^n) \) be the \( k \)th column of \( B_n \), and let \( b_{nk}^* = (b_{1k}^*, b_{2k}^*, \ldots, b_{mk}^*) \) the corresponding column in \( B^* \).

We want to show that \( b_{nk}^* \) is column polished. Let

\[
\max_{i} b_{ik}^* = b_{uk}^*, \\
\min_{i} b_{ik}^* = b_{zk}^*, \\
\max_{i} |b_{ik}^*| = |b_{sk}^*|. \\
\]

and

\[
\max_{i} |b_{ik}^*| = \frac{1}{2} |b_{sk}^*|, \\
\]

Let \( \delta = \frac{1}{4} |b_{uk}^* - b_{sk}^*| > 0 \), and let \( 0 < \varepsilon < \delta \) be given. Since \( b_{nk}^n + b_{nk}^* \), there exist \( N_1 \) such that \( |b_{ik}^n - b_{ik}^*| < \frac{\varepsilon}{2} \) if \( n > N_1 \), \( i = 1,2,\ldots,m \). In particular, \( (i = u) \)

\[
|b_{uk}^n - b_{uk}^0| < \frac{\varepsilon}{2} \quad \text{if} \quad n > \hat{N}. \\
\]

Therefore \( b_{ik}^* - \frac{\varepsilon}{2} < b_{ik}^n < b_{ik}^* + \frac{\varepsilon}{2} \), and \( b_{uk}^* - \frac{\varepsilon}{2} < b_{uk}^n < b_{uk}^* + \frac{\varepsilon}{2} \) for \( n > \hat{N} \).
By the choice of $\delta$ and $0 < \varepsilon < \delta$ one gets

$$b_{ik}^n < b_{uk}^n \quad \text{for} \quad i \neq u \quad \text{and} \quad n > N.$$ 

Thus $b_{uk}^n$ is the maximum element in $b_{\cdot k}^n$ for $n > \hat{N}$.

By a similar argument, there exists $N^*$ such that $b_{\cdot k}^n$ is the minimum element in $b_{\cdot k}^n$ for $n > N^*$.

Take $N = \max(N, N^*)$. For $n > N$

$$|b_{uk}^* + b_{\cdot k}^*| < |b_{uk}^* - b_{uk}^n| + |b_{\cdot k}^* - b_{\cdot k}^n| + |b_{uk}^n + b_{\cdot k}^n|.$$ 

The last term is zero, since $b_{\cdot k}^n$ is column polished. Therefore,

$$|b_{uk}^* + b_{\cdot k}^*| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$ 

Since $\varepsilon$ is arbitrary, $b_{\cdot k}^*$ is column polished. So $B^*$ is column polished.

Now since $A_{\cdot k}^* + A_{\cdot k}^*$ and $\{A_{\cdot k}^n\}$ are CMRP matrices, by the previous lemma, $A^*$ is CMRP.
3.3.2. Lemma

$A^*$ is an optimal matrix.

Proof. Suppose not, then its $L_\infty$-norm can be further reduced by CR, see Corollary 3.2.2. Assume $\| A^* \|_\infty = u$, then there exist $k \in \mathbb{Z}^+$ such that $\| (CR)^k A^* \|_\infty = v$ where $u - v = \varepsilon > 0$. Since there exists a subsequence \{A_{n_k}\} which converges to $A^*$, we can find $A_{n_{k_0}}$ such that

$$\| A_{n_{k_0}} - A^* \|_\infty < \frac{\varepsilon}{3}.$$

Consider

$$\| (CR)^k A_{n_{k_0}} \|_\infty = \| (CR)^k A_{n_{k_0}} - (CR)^k A^* + (CR)^k A^* \|_\infty \leq \| (CR)^k A_{n_{k_0}} - (CR)^k A^* \|_\infty + \| (CR)^k A^* \|_\infty.$$

By Lemma 3.2.2 \((CR)^k\) is a continuous operator, so

$$\| (CR)^k A_{n_{k_0}} - (CR)^k A^* \|_\infty < \frac{\varepsilon}{3} \text{ whenever } \| A_{n_{k_0}} - A^* \|_\infty < \frac{\varepsilon}{3}.$$

Then $\| (CR)^k A_{n_{k_0}} \|_\infty < \frac{\varepsilon}{3} + v = \frac{\varepsilon}{3} + u - \varepsilon = u - \frac{2\varepsilon}{3}$. This is a contradiction. □

Now, by Theorem 3.2.1 $O^*$ contains a loop $L$ which alternates the $A$ values $u$ and $-u$. The following Lemma will give a complete description of the orientation of the elements of $O_A$. 
3.3.3. Lemma

If $\|A\|_\infty = u$ then elements of $A^*$ are oriented in the form of loops which alternate the values of $u$ and $-u$.

Proof. Suppose not, then there exist $(i_0,j_0) \in A^*$ such that $(i_0,j_0)$ is not in a loop alternating the values $u$ and $-u$. Assume $a_{i_0,j_0}^* = u$ and there is no entry $a_{i_0,k}^*$ in row $i_0$ with the value $a_{i_0,k}^* = -u$. Take

$$\delta = \min_{(i,j) \notin A^*} \{u - |a_{ij}^*|\}.$$ 

Now, since there exist $\{A_{n_k}\}$ such that $A_{n_k} + A^*$, then for $0 < \varepsilon < \delta$ there exist $N_0$ such that

$$|a_{ij}^* - a_{ij}| < \frac{\varepsilon}{32} \text{ for all } n_k > N_0.$$

Fix $n_0 > N_0$, and without loss of generality we can assume that

$$A^* = \begin{bmatrix}
    u & u & h_2 & h_1 \\
    -u & u & -u & -u \\
    -h_2 & u & u & -h_1 \\
    u & -h_1 & u & u
\end{bmatrix}, \quad \|A\|_\infty = u.$$
Then the column mid-range polished matrix $A_{n_k0}$ can be written as:

$$
\begin{bmatrix}
\begin{array}{cc}
 a & -c \\
 y_1 & y_2 \\
-a & b \\
 -b & c & -d \\
 y_1 & d \\
 c & -y_2 \\
\end{array}
\end{bmatrix}
\begin{array}{c}
\text{Row effects} \\
\frac{a-c}{2} \\
\frac{b-a}{2} \\
\frac{c-b}{2} \\
\frac{d-y_1}{2} \\
\frac{c-y_2}{2} \\
\end{array}
$$

where

$$a, b, c, \text{ and } d \text{ belong to } (u - \frac{\epsilon}{32}, u + \frac{\epsilon}{32}),$$

some of them must belong to $[u, \frac{\epsilon}{32}]$, and the highest possible value for $y_i^*, i = 1, 2$ and $y_i^*$ is an entry not belonging to $0^*$ is given by:

$$\left|y_i^*\right| \in (u - \delta - \frac{\epsilon}{32}, u - \delta + \frac{\epsilon}{32}).$$

Claim: Any mid-range polish operation (row polish and then column polish) will reduce the values $c$ and $d$ to values strictly less than $u - \frac{\epsilon}{32}$, for any $0 < \epsilon < \delta$. 
Therefore this will be a contradiction with our assumption.

Applying the row mid range polish with entries as shown in $A_{\eta k_0}^{R^A}$

$$
\begin{array}{cccccc}
\frac{c-b}{2^2} & \frac{a-b}{2^2} & z_1 & \frac{b-a}{2^2} & z_2 & z_3 \\
\frac{a+c}{2} & -\frac{a+c}{2} & y_1 & -y_2 \\
-\frac{a+b}{2} & \frac{a+b}{2} & \frac{b+c}{2} & -\frac{2+y_1}{d} & \frac{2+y_1}{2} & \frac{c+y_2}{2} & -\frac{c+y_2}{2} \\
\end{array}
$$

where

$$\frac{a+c}{2}, \frac{a+b}{2} \text{ and } \frac{b+c}{2} \text{ belong to } (u - \frac{\epsilon}{32}, u + \frac{\epsilon}{32}),$$

since for example, $\frac{a+c}{2}$ is a convex combination of two numbers $a$ and $c$ from the convex set $(u - \frac{\epsilon}{32}, u + \frac{\epsilon}{32})$.

Also $\frac{d+y_1}{2}$ and $\frac{c+y_2}{2}$ belongs to $(u - \frac{\delta}{2} - \frac{\epsilon}{32}, u - \frac{\delta}{2} + \frac{\epsilon}{32})$ since, for example
\[ u - \frac{\varepsilon}{32} < d < u + \frac{\varepsilon}{32}, \]

\[ u - \delta - \frac{\varepsilon}{32} < y_1 < u - \delta + \frac{\varepsilon}{32} \]

and

\[ u - \delta - \frac{\varepsilon}{32} < \frac{d+y_1}{2} < u - \frac{\delta}{2} + \frac{\varepsilon}{32}. \]

Now, since \( \frac{c-b}{2} \in \left( -\frac{\varepsilon}{32}, \frac{\varepsilon}{32} \right) \) we have

\[ u - \frac{2\varepsilon}{32} < d + \frac{c-b}{2} < u + \frac{2\varepsilon}{32}. \]

But \( \| R_{A_n} \|_\infty \leq \| A_{n_k} \|_\infty \leq \left( u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32} \right) \) and

\[ d + \frac{c-b}{2} < b + \frac{c-b}{2} = \frac{b+c}{2} \in \left( u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32} \right). \]

Therefore the worst possible value of \( d + \frac{c-b}{2} \) belongs to

\( \left( u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32} \right) \). Also, \( |y_1| < u - \delta + \frac{\varepsilon}{32} \) by convexity.

Applying now the column mid range polish as shown at the head of the columns of \( R_{A_n} \) we have:
which can be written as:
where, \( a_1, b_1, c_1 \) belong to \( (u - \frac{e}{32}, u + \frac{e}{32}) \) for the same reason, for example, \( a_1 = \frac{a+c}{2} + \frac{a+b}{2} \) is the convex combination of the two numbers \( \frac{a+c}{2} \) and \( \frac{a+b}{2} \) in the convex set \( (u - \frac{e}{32}, u + \frac{e}{32}) \). And \( \bar{c}_1 = \frac{c+y_2}{2} + \frac{a-b}{4} \) belongs to \( (u - \frac{\delta}{2} - \frac{3}{2} \frac{e}{32}, u - \frac{\delta}{2} + \frac{3}{2} \frac{e}{32}) \) since,

\[
\begin{align*}
u - \frac{\delta}{2} - \frac{e}{32} < \frac{c+y_2}{2} < u - \frac{\delta}{2} + \frac{e}{32}, \\
- \frac{e}{64} < \frac{a-b}{4} < \frac{e}{64}, \text{ and} \\
u - \frac{\delta}{2} - \frac{3}{2} \frac{e}{32} < \bar{c}_1 < u - \frac{\delta}{2} + \frac{3}{2} \frac{e}{32}.
\end{align*}
\]

Therefore

\[
|\bar{c}_1| < u - \frac{\delta}{2} + \frac{3}{2} \frac{e}{32} < u - \frac{e}{32}, \quad \forall 0 < e < \delta.
\]

Also, \( \tilde{d}_1 = \frac{(d+\frac{c-d}{2}) + \frac{d+y_1}{2}}{2} \) belongs to \( (u - \frac{\delta}{4} - \frac{e}{32}, u - \frac{\delta}{4} + \frac{e}{32}) \) since

\[
\begin{align*}
\frac{\delta}{2} - \frac{e}{32} < d + \frac{c-b}{2} < u + \frac{e}{32}, \\
u - \frac{\delta}{2} - \frac{e}{32} < \frac{d+y_1}{2} < u - \frac{\delta}{2} + \frac{e}{32}, \text{ and} \\
u - \frac{\delta}{4} + \frac{e}{32} < \tilde{d}_1 < u - \frac{\delta}{4} + \frac{e}{32}.
\end{align*}
\]
Therefore,

\[ |\tilde{a}_1| < u - \frac{\delta}{4} + \frac{\varepsilon}{32} < u - \frac{\varepsilon}{32}, \quad \forall 0 < \varepsilon < \delta. \]

And the highest possible value for \( y_i \), \( i = 1,2 \) can be found by

\[ u - \frac{\delta}{2} - \frac{\varepsilon}{32} < \tilde{y}_i < u - \frac{\delta}{2} + \frac{\varepsilon}{32}, \quad \text{and} \]

\[ u - \frac{3\delta}{4} - \frac{\varepsilon}{32} < y_i < u - \frac{3\delta}{4} + \frac{\varepsilon}{32}. \]

Therefore, \( |y_i| < u - \frac{3\delta}{4} + \frac{\varepsilon}{32} \).

Now, given the matrix \((CR)^n_{A_{\kappa_0}}\) with \( a_1, b_1, c_1 \in (u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32}) \) and the estimate for

\[ |\tilde{d}_n| < u - \frac{3\delta}{4} + \frac{\varepsilon}{32} < u - \frac{\varepsilon}{32}, \quad \forall 0 < \varepsilon < \delta, \quad \text{and} \]

\[ |\tilde{c}_n| < u - \frac{\delta}{2} + \frac{3\varepsilon}{32} < u - \frac{\varepsilon}{32}. \]

I will prove by induction that the matrix \((CR)^n_{A_{\kappa_0}}\) will satisfies these estimates implying that \( \tilde{d}_n \) and \( \tilde{c}_n \) will never reach the value \( u \), which is a contradiction.
Now, assume the matrix

$$(CR)^n A_{nk_0} = \begin{bmatrix}
  a_n & -c_n \\
  -a_n & b_n \\
  y_1^n & y_2^n \\
  -y_1^n & y_2^n \\
  \tilde{c}_n & -y_2^n
\end{bmatrix}$$

is such that $|\frac{a_n-c_n}{2}| = |\frac{a-c}{2^{2n-1}}| < \frac{\varepsilon/32}{2^{2n-2}}$, $|\tilde{d}_n| < |\tilde{d}_1| + \sum_{i=1}^{n-1} \frac{1}{2^{2i-1}} \frac{\varepsilon}{32}$, and $|\tilde{c}_n| < |\tilde{c}_1| + \sum_{i=1}^{n-1} \frac{1}{2^{2i-1}} \frac{\varepsilon}{32}$,

and $a_n, b_n, c_n \in \left( u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32} \right)$. 

row effects

$$\frac{a_n-c_n}{2} = \frac{a-c}{2^{2n-1}}$$
$$\frac{b_n-a_n}{2} = \frac{b-a}{2^{2n-1}}$$
$$\frac{c_n-b_n}{2} = \frac{c-b}{2^{2n-1}}$$
$$\frac{\tilde{d}_n-y_1^n}{2}$$
$$\frac{\tilde{c}_n-y_2^n}{2}$$
We want to show that \((CR)^{n+1}\) satisfies the same estimate, applying the row polish, we have

\[
\begin{array}{cccccc}
\frac{c-b}{2n} & \frac{a-c}{2n} & z_1 & \frac{b-c}{2n} & z_2 & z_3 \\
-\frac{a+c}{n} & -\frac{a+c}{n} & -\frac{b+c}{n} & -\frac{b+c}{n} & -(\frac{d+c-b}{2n-1}) & \\
\frac{d_n+y_1^n}{2} & \frac{d_n+y_1^n}{2} & \frac{c+y_2^n}{2} & \frac{c+y_2^n}{2} & & \\
\end{array}
\]

where

\[
\frac{a+c}{n}, \frac{a+b}{n} \text{ and } \frac{b+c}{n} \text{ belong to } (u = \frac{\xi}{32}, u + \frac{\xi}{32})
\]

by convexity, and applying the column polish, we have
which can be written as

\[
(CR)^n A_{nk_0}^{n+1} = \begin{bmatrix}
  a_{n+1} & c_{n+1} \\
  y_{n+1} & y_{2n+1} \\
  -b_{n+1} & -d_{n+1} \\
  -y_1 & d_{n+1} \\
  c_{n+1} & d_{n+1} \\
  -y_{n+1} & y_{2n+1}
\end{bmatrix}
\]

where

\[
a_{n+1}, b_{n+1}, c_{n+1} \in \left( u - \frac{\varepsilon}{32}, u + \frac{\varepsilon}{32} \right) \text{ by convexity,}
\]

and

\[
\tilde{d}_{n+1} = \frac{\tilde{d}_n + \frac{c-b}{2^{2n-1}} + \frac{y_1 + \tilde{c}_n}{2}}{2}
\]

\[
\tilde{d}_n = \frac{3\tilde{d}_n}{4} + \frac{y_1}{4} + \frac{c-b}{2^{2n-1}}
\]

\[
|\tilde{d}_{n+1}| < \frac{3}{4} |\tilde{d}_n| + \frac{1}{4} |y_1| + \frac{1}{2^{2n-2}} \frac{|c-b|}{2}
\]

\[
< \frac{3}{4} |\tilde{d}_n| + \frac{1}{4} |y_1^n| + \frac{1}{2^{2n-2}} \frac{\varepsilon}{32}, \text{ but } |y_1| < |\tilde{d}_n| \text{ so,}
\]
\[
|\tilde{d}_{n+1}| < |\tilde{d}_n| + \frac{1}{2} \frac{\varepsilon}{2^{n-2}} 32 < |\tilde{d}_n| + \frac{n}{2^{2n-1}} \frac{\varepsilon}{32}
\]

\[
< |\tilde{d}_n| + \frac{2}{3} \frac{\varepsilon}{32} < u - \frac{\delta}{4} + \frac{\varepsilon}{32} + \frac{2}{3} \frac{\varepsilon}{32} < u - \frac{\varepsilon}{32}
\]

for \(0 < \varepsilon < \delta\).

Also,
\[
\tilde{c}_{n+1} = \frac{c_n + y_2^n}{2} - \frac{b-a}{2^{2n}}
\]

\[
|\tilde{c}_{n+1}| < \frac{|c_n|}{2} + \frac{|y_1^n|}{2} + \frac{|b-a|}{2^{2n-1}} , \text{ but } |y_2^n| < |\tilde{c}_n| \text{ so,}
\]

\[
|\tilde{c}_{n+1}| < |\tilde{c}_n| + \frac{1}{2} \frac{\varepsilon}{2^{2n-1}} 32 < |\tilde{c}_n| + \frac{n}{2^{2n-1}} \frac{\varepsilon}{32}
\]

\[
|\tilde{c}_{n+1}| < |\tilde{c}_n| + \frac{3}{2} \frac{\varepsilon}{32} < u - \frac{\delta}{2} + \frac{3}{2} \frac{\varepsilon}{32} + \frac{3}{2} \frac{\varepsilon}{32} < u - \frac{\varepsilon}{32},
\]

for \(0 < \varepsilon < \delta\).

Therefore the values of \(c\) and \(d\) will be reduced to values less than \(u - \frac{\varepsilon}{32}\), for any \(0 < \varepsilon < \delta\), which is a contradiction.

Therefore the elements of \(O^*\) are oriented in the form of loops \(A\) which alternate the values \(u\) and \(-u\). 

3.3.2. Theorem "Convergence Theorem"

The sequence \(\{A_n\}_{n=1}^\infty\) produced by MRP algorithm converges to an optimal matrix \(A^*\), with \(\|A^*\|_\infty = u\), in the sense that \(A^*\) defines
\[ \delta = \min_{(i,j) \in O^*} \{u - |a_{ij}^*|\} > 0, \]

for any \(0 < \epsilon < \delta\), \(\exists N_0\) s.t. \(\forall n > N_0\)

\[ |a_{ij}^n - a_{ij}^*| < \epsilon \quad \forall (i,j) \in O^* \]

\[ |a_{ij}^n| < u - \delta \quad \forall (i,j) \in O^* \]

**Proof.** Let \(0 < \epsilon < \delta\) be given. Since there exists a subsequence \(\{A_{n_k}\}\) such that

\[ A_{n_k} \to A^* \quad \exists N_0 = \max\{N_{11}, N_{12}, \ldots, N_{mn}\} \]

s.t. \(\|A_{n_k} - A^*\| < \frac{\epsilon}{32} \quad \forall n_k > N_0\).

By Lemma 3.3.3 \(0^*\) is oriented in the form of loops. Consider \(A_{n_{k_0}}, n_{k_0} > N\) and look at \((CR)^k A_{n_{k_0}}\). According to Lemma 3.3.3 the row effect for any loop's row \(i\)

\[ |r_{ik}^i| < \frac{a-b}{2^k-1} < \frac{\epsilon/32}{2^k-2}. \]

Therefore \(\sum_{k=1}^{\infty} r_{ik}^i\) converges, and similarly for the loop's columns.

Therefore, \(\{a_{ij}^n\} \to a_{ij}^*\) for all \((i,j) \in O^*\). For any element \(a_{ij}^n\)
outside the loop positions we have

\[ |a_{ij}^n| < u - \frac{\delta}{2} + \frac{\varepsilon}{32} \quad (i,j) \notin A^* \]

\[
\therefore \exists \text{ subsequence } \{a_{ij}^n\} \text{ such that }
\]

\[ a_{ij}^n + a_{ij}^* \quad (i,j) \in A^* , \]

therefore \[ a_{ij}^* < u - \frac{\delta}{2} + \frac{\varepsilon}{32} \]. And since \( \varepsilon \) is arbitrary,

\[ |a_{ij}^*| < u - \frac{\delta}{2} \quad \text{for } (i,j) \notin A^* . \]

3.3.2. Example

\[
A = \begin{bmatrix}
3 & 7 & 5 \\
-2 & -2 & 4 \\
7 & 3 & 7
\end{bmatrix}
\]

\[
\begin{array}{ccc|c}
3 & 7 & 5 & 5 \\
\hline
\end{array}
\begin{array}{ccc|c}
\alpha_{i1}^{(1)} & & & \\
\hline
3 & 7 & 5 & 5 \\
-2 & -2 & 4 & 1 \\
7 & 3 & 7 & 5 \\
\hline
-\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} & \beta_{j1}^{(1)} \\
\hline
\end{array}
\]

or

\[
\begin{array}{ccc|c}
-\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} & \beta_{j1}^{(1)} \\
\hline
-2 & 2 & 0 & \\
\hline
\end{array}
\begin{array}{ccc|c}
RA = & & & \\
\hline
3 & 7 & 5 & 5 \\
-2 & -2 & 4 & 1 \\
7 & 3 & 7 & 5 \\
\hline
-\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} & \beta_{j1}^{(1)} \\
\hline
\end{array}
\]

\[
\begin{array}{ccc|c}
3 & 7 & 5 & 5 \\
-2 & -2 & 4 & 1 \\
7 & 3 & 7 & 5 \\
\hline
-\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} & \beta_{j1}^{(1)} \\
\hline
\end{array}
\begin{array}{ccc|c}
CRA = & & & \\
\hline
-3 & -3 & 3 & \\
2 & -2 & 2 & \\
\hline
\end{array}
\]


\[ A_1 = \begin{bmatrix}
-\frac{3}{2} & \frac{5}{2} & -\frac{3}{2} \\
-\frac{5}{2} & -\frac{5}{2} & \frac{3}{2} \\
\frac{5}{2} & -\frac{3}{2} & \frac{1}{2}
\end{bmatrix} \]

\[ RA_1 = RCRA = \begin{bmatrix}
-\frac{3}{2} & \frac{5}{2} & -\frac{3}{2} \\
-\frac{5}{2} & -\frac{5}{2} & \frac{3}{2} \\
\frac{5}{2} & -\frac{3}{2} & \frac{1}{2}
\end{bmatrix} \]

\[ A^* = \begin{bmatrix}
-2 & 2 & -2 \\
-2 & -2 & 2 \\
2 & -2 & 0
\end{bmatrix} \]

optimal.

i.e. for \( X = (\frac{11}{2}, \frac{1}{2}, \frac{11}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{3}{2}) \) we have

\[ A = \begin{bmatrix}
-\frac{1}{2} & -\frac{1}{2} & 3 \\
3 & 7 & 5 \\
-2 & -2 & 4 \\
\frac{11}{2} & \frac{11}{2} \\
7 & 3 & 7
\end{bmatrix} \]
which gives

\[
A^* = \begin{bmatrix}
-2 & 2 & -2 \\
-2 & -2 & 2 \\
2 & -2 & 0
\end{bmatrix}.
\]
4. FINITE ALGORITHM FOR THE $L_\infty$-PROBLEM

4.1. Introduction

Theorem (2.1.1) suggests a finite algorithm for determining an optimal matrix of residual

$$A^* = (a_{ij}^*) = (a_{ij} - r_i^* - s_j^*) .$$

If the primal problem is

$$\text{max} \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} w_{ij},$$

when

$$\sum_{i=1}^{m} \sum_{j=1}^{n} |w_{ij}| = 1, \sum_{i=1}^{m} w_{ij} = 0 \quad j = 1, \ldots, n,$$

and

$$\sum_{j=1}^{n} w_{ij} = 0 \quad i = 1, 2, \ldots, m,$$

and its dual is

$$\text{min} \ u \quad \text{when}$$

$$r, s, u$$

$$-u < a_{ij} - r_i - s_j < u \quad i = 1, 2, \ldots, m,$$

$$j = 1, 2, \ldots, n .$$

If we keep the solution primally feasible (satisfies Condition (A) of Chapter II) and satisfying the complementary slackness (Conditions (B) and
(C) of Chapter II) until the dual feasibility is attained (Condition (D) of Chapter II). This can be done in two phases, a Phase I Algorithm and a Phase II Algorithm.

In Phase I we try to find a basic solution which satisfies Conditions A, B and C, i.e., a solution that maintains primal feasibility and complementary slackness. Then check if dual feasibility is achieved. If not, it prepares the matrix $A = (a_{ij})$ for phase II.

In Phase II, we try to achieve the dual feasibility by using a Labeling Algorithm which either increases the value $u$ through a Loop Finding Algorithm or changes the basic cells in such a way that the $L^\infty$ norm of the matrix is reduced by using the $L^\infty$-norm Reducing Algorithm.

The finiteness of Phase II Algorithm will be explained in Lemma (4.4.2) later in this chapter.

The following flow chart explains the sequence of operations, where the logical variable "Flag" from Phase I Algorithm indicates if the output matrix $A = (a_{ij})$ is optimal (true). But the logical variable "Delta" from the Labeling Algorithm is true if it has found a loop with a higher value of $u$, and false if it has found a "block" which means that the $L^\infty$-norm of the matrix can be reduced.

The logical variable "Gamma" out of either loop finding algorithm or $L^\infty$-norm reducing algorithm is true if the output matrix is optimal.
Input matrix
\[ A = (a_{ij}) \]

Phase I Algorithm

\[ A = (a_{ij}) \]
\[ G = L \cup T \]

Labeling Algorithm

\[ L^\infty \text{ norm} \]
\[ \text{Reducing Algorithm} \]

is True

is Gamma true

Loop Finding Algorithm

\[ \text{Block Loop} \]
\[ \text{Optimal} \]

Loop

\[ \text{Delta true} \]
4.2. Phase I Algorithm

The problem is to find a starting basic feasible solution to the primal problem which satisfies conditions A, B and C.

Let a matrix $A_0 = (A_{ij})_{m \times n}$ be given.

4.2.1. Example

Let $A_0 = \begin{bmatrix} -2 & -4 & -5 & 3 \\ 6 & -3 & -5 & 6 \\ -4 & 1 & -1 & 4 \\ -7 & 9 & 1 & 4 \end{bmatrix}$

Step 0: Initiate an $m \times n$-vector

$$X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n) = 0$$

Step 1: Find a cell $(p, q)$ such that $a_{pq} = \max_{i,j} |a_{ij}|$, and circle this cell.

Let $A_0 = \begin{bmatrix} -2 & -4 & -5 & 3 \\ 6 & -3 & -5 & 6 \\ -4 & 1 & -1 & 4 \\ -7 & 9 & 1 & 4 \end{bmatrix}$
Step 2: Construct a basic 1-tree rooted at \((p,q)\) and an associated sign pattern \((\overline{w}_{ij})\) in the following way:

i) Construct a spanning tree containing \((p,q)\), for a submatrix \(\bar{A}_0 = (a_{ij})_{k \times k}\) where \(k = \min\{m,n\}\).

a) If \(a_{pq} > 0\) (\(a_{pq} < 0\)) choose the next cell to circle as the minimum (maximum) element in row \(p\) and column \(q\).

\[
\begin{array}{cccc}
-2 & -4 & -5 & 3 \\
6 & -3 & -5 & 6 \\
-4 & 1 & -1 & 4 \\
-7 & 9 & 1 & 4 \\
\end{array}
\]

Say \((p,q)\), then choose the next cell as the maximum (minimum) element in column \(q_1\).

\[
\begin{array}{cccc}
-2 & -4 & -5 & 3 \\
6 & -3 & -5 & 6 \\
-4 & 1 & -1 & 4 \\
-7 & 9 & 1 & 4 \\
\end{array}
\]

Say it is \((p_1,q_1)\).
b) Assign a sign pattern \( \{ w_{ij} \} \) to this right angle shape, starting with \( w_{pq} = \text{sign}(a_{pq}) \) then \( a_{pq_1} = -w_{pq} \) and \( w_{p_1q_1} = -w_{pq_1} \).

\[
\begin{array}{cccc}
-2 & -4 & -5 & 3 \\
6^+ & -3 & -5 & 6 \\
-4 & 1 & -1 & 4 \\
-7 & 9^+ & 1 & 4 \\
\end{array}
\]

c) for the current tree, we have a row leaf \((p, q)\) and a column leaf \((p_1, q_1)\). We add cells two at a time to the current tree, first by choosing an element in the row or column emanating from a leaf of the tree. If the leaf is a row leaf, then the first new element adds a column leaf to the tree and the second adds the row of the new element so that the new tree again has a row for a leaf.

If the leaf is a column leaf, then one adds a row and then a column, (Fig. 4.2.1).
d) The rule for choosing the new cells is given by:

Suppose we have a row leaf \((p_{k-1}, q_{k-1})\) and column leaf \((p_k, q_k)\) we choose the first cell \((s, q_{k-1})\) as column leaf or \((p_k, t)\) as row leaf according to which one maximizes

\[
\left\{ [\bar{\omega} p_{k-1} q_{k-1} a_{p_{k-1} q_{k-1}} + \bar{\omega} s q_{k-1} a_{s q_{k-1}}] , \right. \\
\left. [\bar{\omega} p_k q_k a_{p_k q_k} + \bar{\omega} p_k t a_{p_k t}] \right\};
\]

where \((s, q_{k-1})\) ... is the minimum (maximum) element in column \(q_{k-1}\) does not make a cycle with previous cells, \((p_k, t)\) ... is the minimum (maximum) element in row \(p_k\) which does make a cycle with previous cells, and

\[
\bar{\omega} p_k t = \bar{\omega} s q_{k-1} = -\bar{\omega} p_{k-1} q_{k-1} = -\bar{\omega} p_k q_k
\]
say we select the row leaf \((p_k, t)\). We choose the second cell as the maximum (minimum) element in column \(t\) (row \(s\)) which does not make a cycle with the previous cells.

Give the 2nd element opposite sign of the first element.

i.e., the sign pattern will alternate so that if the old leaf had a + sign, the new signs are - and + in that order.

One notices that if the beginning element \(a_{pq} > 0\) (\(a_{pq} < 0\)) then the leaves will always have a + sign (− sign).

e) Stop when the number of cells in the tree is \((2k-1)\).

\[
\begin{array}{cccc}
-2 & -4^- & -5 & 3^+ \\
6^+ & -3 & -5 & 6 \\
-4 & 1 & -1 & 4 \\
-7^- & 9^+ & 1 & 4 \\
\end{array}
\]

Number of cells in the tree is \(2k-1 = 7\).
ii) Close the loop with the cell in the column and row of the leaves. This is the Loop L of the basic 1-tree G.

\[
\begin{array}{cccc}
-2 & -4 & -5 & 3^+ \\
\circ & -3 & -5^- & 6 \\
-4 & 1 & 1^+ & \circ \\
\circ & 9^+ & 1 & 4 \\
\end{array}
\]

And assign its sign by \((-\) (+) if \(a_{pq} > 0\) (if \(a_{pq} < 0\)).

iii) Now, if \(m > n\), attach the maximum element in absolute value, in each row not in the loop's rows to the loop L.

And if \(m < n\), attach the maximum element in absolute value, in each column not in the loop's columns to the loop L.

iv) By attaching these cells to the loop L, the basic 1-tree \(G = L \cup T\), where T is the added cells from step iii. (\(T = \phi\) in our example).

Let \(\bar{w}_{ij} = \text{sign}(a_{ij})\) \(\forall (i,j) \in T\). Therefore, the associated sign pattern \(\{\bar{w}_{ij}\}\) is

\[
\bar{w}_{ij} = \begin{cases} 
\pm 1 & \forall (i,j) \in L \\
\text{sign}(a_{ij}) & \forall (i,j) \in T \\
0 & \text{otherwise} \end{cases}
\]
Step 3: Define

\[ w_{ij} = \begin{cases} \frac{w_{ij}}{2k} & \forall (i,j) \in L, \\ 0 & \text{otherwise}. \end{cases} \]

Now, since \(|L| = 2k\), then \(w = (w_{ij})\) is a feasible solution for the primal problem. i.e., it satisfies Condition A.

\[
\begin{array}{cccc}
0 & -\frac{1}{8} & 0 & \frac{1}{8} \\
\frac{1}{8} & 0 & -\frac{1}{8} & 0 \\
0 & 0 & \frac{1}{8} & -\frac{1}{8} \\
-\frac{1}{8} & \frac{1}{8} & 0 & 0 \\
\end{array}
\]

i.e. \[
\sum_{i,j} |w_{ij}| = 1
\]

\[
\sum_{i=1}^{m} w_{ij} = 0, \quad j = 1, 2, \ldots, n,
\]

\[
\sum_{j=1}^{n} w_{ij} = 0, \quad i = 1, 2, \ldots, m.
\]

Step 4: Calculate

\[
u = \sum_{i,j} a_{ij} w_{ij} = \frac{1}{2k} \sum_{(i,j) \in L} a_{ij} w_{ij}
\]

For our example,
\[
u = \frac{1}{8} (9+7+6+5-1-4+3+4) = \frac{29}{8}.
\]

**Step 5:** Solve,

\[
a_{ij} - r_i - s_j = \bar{W}_{ij} u \quad \text{for} \quad (i,j) \in L \cup T,
\]

starting with \( r_p = 0 \) if \( a_{pq} > 0 \)

or

\( s_q = 0 \) if \( a_{pq} < 0 \).

And update

\[
X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n).
\]

\[
\begin{array}{cccc}
-2 & -4 & -5 & 3^+ \\
6^+ & -3 & -5^- & 6 \\
-4 & 1 & -1^+ & 4^- \\
-7^- & 9^+ & 1 & 4 \\
0 & & & \\
\end{array}
\]
Step 6: Pivot by Calculating:

\[ a_{ij} = a_{ij} - r_i - s_j \quad \text{for all (i,j)}, \]

\[ A_0 = (a_{ij}) = \]

Now, this matrix satisfies Conditions B and C.

Step 7: Check if the new matrix \( A_0 = (a_{ij})_{m \times n} \) satisfies Condition D.

If yes, stop. The given solution is optimal. If not, call "The Labeling Algorithm".

\[ x = \left( -\frac{46}{8}, \frac{46}{8}, \frac{20}{8}, 0, -\frac{27}{8}, \frac{43}{8}, -\frac{57}{8}, \frac{41}{8} \right). \]
4.3. Phase II Algorithm

The input matrix \( A_0 = (a_{ij})_{m \times n} \) for this algorithm is a matrix which satisfies Conditions A, B and C of Chapter 2, and the algorithm tries to make it satisfy Condition D of the same chapter. This can be done by calling a "Labeling Subroutine" which decides to call "a Loop Finding Algorithm" which increases the value \( u \) through a loop, or to call "\( L_\infty \)-norm Reducing Algorithm" which reduces the \( L_\infty \)-norm of the matrix.

4.3.1. Labelling Algorithm

**Given:** a basic \( l \)-tree \( L \cup T \) with a Loop \( L \) alternating the values \( u \) and \(-u\).

The \( L_\infty \)-norm of the matrix \( A = (a_{ij})_{m \times n}, L_\infty \) is such that \( L_\infty > u > 0 \).

**Example:**

\[
\begin{array}{cccc}
57 & -29 & 63 & 29 \\
8 & 8 & 8 & 8 \\
29 & -113 & -29 & -39 \\
8 & 8 & 8 & 8 \\
-25 & -55 & 29 & -29 \\
8 & 8 & 8 & 8 \\
-29 & 29 & 65 & -9 \\
8 & 8 & 8 & 8 \\
\end{array}
\]

\( L_\infty = \frac{113}{8} \), \( u = \frac{29}{8} \).
Define: \( V = \{ (i,j) / |a_{ij}| = L_\infty \} \neq \emptyset \), and

\[ x = L_\infty - u > 0 \, . \]

Initiate \( X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n) = 0 \, . \)

0 - Pick a position \((p, q) \in V\).

1 - If \( a_{pq} > 0 \) (if \( a_{pq} < 0 \)) label column \( q \) (row \( p \)) with a \( \oplus \) and the \( p \)th row (\( q \)th column) with a \( \otimes \). Put \( r_p = 0 \) (\( s_q = 0 \)) and update \( x = x \) (\( r_p = -x \)).

2 - Label any row (column) \( i \) (\( j \)) with a \( \oplus \) if \( a_{iq} < -u \) (\( a_{pj} > u \)). Update \( r_i^s \) by the value \(-x\) (\( s_j^s \) by the value \( x \)).

For our example \( x = \frac{113}{8} - \frac{29}{8} = \frac{84}{8} \).

\[
\begin{pmatrix}
\frac{84}{8} & 0 \\
\frac{57}{8} & -\frac{29}{8} & \frac{63}{8} & \frac{29}{8} \\
-\frac{84}{8} & \frac{29}{8} & \boxed{\frac{113}{8}} & -\frac{29}{8} & -\frac{39}{8} \\
-\frac{25}{8} & -\frac{55}{8} & \frac{29}{8} & -\frac{29}{8} \\
-\frac{29}{8} & \frac{29}{8} & \frac{65}{8} & -\frac{9}{8}
\end{pmatrix}
\]

3 - Label any column \( j \) (row \( i \)) with a \( \oplus \) if \( (a_{ij} - r_i - s_j) > L_\infty \)
(\( a_{ij} - r_i - s_j < -L_\infty \)) and some row \( i \) (column \( j \)) is labelled \( \oplus \).

Update \( s_j^s \) by the value \( x \) (\( r_j^s \) by the value \(-x\))
4 - Label any row i (column j) with a $\exists$ if $(a_{ij} - r_i - s_j) < -L_\infty$ 
$(a_{ij} - r_i - s_j > L_\infty)$ for some column j (row i) labelled $\exists$.
Update $r_j^S$ by the value $-\kappa$ ($s_j^S$ by the value $\kappa$) etc.

5 - Continue until one reaches to one of the following two possible cases:

**Case I:** At some point, row (column) $\star$, or any previously labeled row or column, is given a second label.
STOP labeling and go to "Loop Finding Algorithm".

**Case II:** At some point no new row or column can be labeled. Stop labeling. We have constructed "a block". Go to "$L_\infty$-norm Reducing Algorithm".

\[
\begin{array}{cccc}
\frac{84}{8} & 0 & \frac{57}{8} & \frac{29}{8} \\
\frac{29}{8} & \frac{113}{8} & \frac{63}{8} & \frac{29}{8} \\
\frac{25}{8} & \frac{55}{8} & \frac{29}{8} & \frac{29}{8} \\
\frac{84}{8} & \frac{29}{8} & \frac{65}{8} & \frac{9}{8} \\
\end{array}
\]
For our example:

\[ \begin{array}{cccc}
84/8 & 0 \\
57/8 & 29/8 & 63/8 & 29/8 \\
-84/8 & 29/8 & 113/8 & -29/8 & -39/8 \\
-84/8 & 29/8 & 55/8 & -29/8 & 29/8 \\
-84/8 & 29/8 & 65/8 & -9/8 \\
1 & * & 3 & 2
\end{array} \]

CALL "Loop Finding Algorithm".

### 4.3.2. Loop Finding Algorithm

**Given:** The row (column) labeled * is labeled K, or any previously labeled row (column) is labeled K.

0 - Initiate \( X = (r_1, r_2, \ldots, r_n, s_1, s_2, \ldots, s_n) = 0 \).

1 - If the row (column) labeled * is labeled K, go to a column (row) labeled (K-1) where there is an element less than or equal to \((-u)\) (greater than or equal to \(u\)) in row * (column *) and the column (row) labeled (K-1).
2 - Then go to a row (column) in the (K-1)-column ((K-1)-row) to find an entry in row (column) labeled (K-2) with an element greater than or equal to \( u \) (less than or equal to \(-u\)). etc.

3 - Continue until one reaches a column or row labeled \( 0 \).

4 - By adding the \( L^- \)-position cell in row (column) * and column (row) labeled \( 0 \), we construct the new loop.

N.B. in the case of previously labeled row (or column) the \( L^- \)-position will be replaced by the cell of row (column) \( K \) and column (row) labeled \( (K-1) \).

5 - Update \( L = \) the new loop.

\[
T = (L \cup T)_{old} - L - (s,t)
\]

where \( (s,t) \) is the basic cell matching the sign of \( a_{pq} \) in the row (or column) labeled \( 0 \).
Update

\[
\bar{w}_{ij} = \pi_{ij} \quad \forall (i,j) \neq \{(p,q) \cup (s,t)\}
\]

\[
\bar{w}_{pq} = \text{sign}(a_{pq})
\]

\[
\bar{w}_{st} = 0
\]

6 - Calculate the new \( u \).

\[
\text{(new } u \text{)} = \sum_{i,j} a_{ij} \bar{w}_{ij} = \frac{1}{|L|} \sum_{(i,j) \in L} a_{ij} \bar{w}_{ij} > \text{old } u
\]

by Theorem (4.4.1)

For our example

\[
u = \frac{1}{4} \left( \frac{113}{8} + \frac{29}{8} + \frac{29}{8} + \frac{29}{8} \right)
= \frac{50}{8}
\]

7 - Update \( X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n) \) by solving:

\[
a_{ij} - r_i - s_j = \bar{w}_{ij} \ u \quad \text{for all } (i,j) \in L \cup T
\]
starting with \( r_p = 0 \) if \( a_{pq} > 0 \), and
\[ s_q = 0 \] if \( a_{pq} < 0 \).

\[
\begin{array}{cccc}
\frac{42}{8} & 0 & -\frac{84}{8} & -\frac{42}{8} \\
\frac{21}{8} & \frac{57}{8} & \frac{-29}{8} & \frac{63}{8} & \frac{29^+}{8} \\
-\frac{63}{8} & \frac{29^+}{8} & \frac{-113}{8} & \frac{-29}{8} & \frac{-39}{8} \\
\frac{63}{8} & \frac{-25}{8} & \frac{-55}{8} & \frac{29^+}{8} & \frac{29^-}{8} \\
-\frac{21}{8} & \frac{-29^-}{8} & \frac{29^+}{8} & \frac{65}{8} & \frac{9}{8}
\end{array}
\]

8 - Pivot by calculating
\[ a_{ij} = a_{ij} - r_i - s_j \quad \forall(i,j) \]

and check \( L_\infty \)-norm. If \( L_\infty < u \), STOP; the given solution is optimal.
Otherwise, go to "Labeling Algorithm".

\[
\begin{array}{cccc}
\frac{-7}{8} & \frac{-50}{8} & \frac{126}{8} & \frac{50}{8} \\
\frac{50}{8} & \frac{-50}{8} & \frac{118}{8} & \frac{66}{8} \\
\frac{-130}{8} & \frac{-118}{8} & \frac{50}{8} & \frac{-50}{8} \\
\frac{-50}{8} & \frac{50}{8} & \frac{170}{8} & \frac{54}{8}
\end{array}
\]
CALL "Labeling Algorithm".

**Notes:**

1. If in a column (row) labeled (K-1) there exist more than one element < u (> u), choose the one with maximum absolute value. In case of a tie in the absolute value choose the one in a basic cell.

2. If the chosen one is not in a basic position, remove the matching basic cell from the l-tree \( G \) and enter instead the chosen one, interchange their \( w \)-values.

### 4.3.3. \( L_\infty \)-Norm Reducing Algorithm

**Given:** A block, a matrix \((a_{ij})_{mxn}\) containing a loop alternate the values \( u \) and \(-u\) and \( L_\infty > u > 0 \).

0. Initiate \( X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n) = 0 \).

1. Define \( x = L_\infty - u > 0 \)

\[
I = \{ i \mid \text{row } i \text{ is labeled} \} \\
J = \{ j \mid \text{column } j \text{ is labeled} \}
\]
2 - Update $X$ by the following way:

$$r_i = \begin{cases} 
-x & \forall i \in I \\
0 & \forall i \notin I 
\end{cases}$$

$$s_j = \begin{cases} 
x & \forall j \in J \\
0 & \forall j \notin J 
\end{cases}$$

\[ x = \frac{170}{8} - \frac{50}{8} = \frac{120}{8} \]

3 - Update $L = L$

$$T = T - (s,t) + (p,q)$$

where $(s,t)$ is the basic cell in the block row or column matching the sign of $a_{pq}$. 
4 - Pivot by calculating:

$$a_{ij} = a_{ij} - r_i - s_j \quad \forall (i,j)$$

and check the $L_\infty$-norm. If $L_\infty < u$, STOP. The given solution is optimal. Otherwise, go to "Labeling Algorithm".

**Example:**

```
  7  50  6  50
-8  50  2  50
 50-50  2  50
 50  50  50  50
```

"block" = $-\frac{80}{8}$

```
  130  118  70  50
 50  50  50  50
 50  50  50  50
```

Next step:

```
  7  50  6  50
-8  50  2  50
 50-50  2  50
 50  50  50  50
```

"block" = $\frac{16}{8}$
4.4. Justification of the Algorithm

4.4.1. Theorem

The "Loop Finding Algorithm" strictly increases the value of $u$.

Proof: The new $u = \sum_{i,j} a_{ij} \bar{w}_{ij} = \frac{1}{|L|} \sum_{(i,j) \in L} a_{ij} \bar{w}_{ij}$, where $|L| = K$ ($*$-row (or column) labeled $K$. Therefore

$$\text{new } u = a_{pq} + \frac{|(K-1) \text{ entries}|}{K} \text{ and each } (K-1) \text{ entry is } > u \text{ in}$$
absolute value by choice and \( a_{pq} > u \) implying

\[
\text{new } u > u.
\]

### 4.4.2. Theorem

**Given:** The \( L_\infty \)-norm of a matrix \( A_0 = (a_{ij})_{m \times n} \), \( L_\infty \), such that \( L_\infty > u > 0 \).

Let \( V = \{(i,j)/ |a_{ij}| = L_\infty \} \) and \( x = L_\infty - u > 0 \).

The \( L_\infty \)-norm Reducing Algorithm reduces the number of elements of \( V \) by at least one.

**Proof:** Let \( S \) be the set of elements \( a_{ij} \) in labeled columns and unlabeled rows. Then any \( a_{ij} \in S \) is such that \(-L_\infty < a_{ij} < u\), otherwise row \( i \) will be labeled.

According to the algorithm \( (\text{new } a_{ij}) = a_{ij} + x \). But

\[
-L_\infty + x < a_{ij} + x < u + x
\]

so

\[
-L_\infty + x < (\text{new } a_{ij}) < u + x \quad \text{and}
\]

\[
-u < \text{new } a_{ij} < L_\infty.
\]

Therefore

\[
-L_\infty < \text{new } a_{ij} < L_\infty.
\]

The new elements of the set \( S \) are strictly less than the \( L_\infty \)-norm in absolute value. Similarly, let \( B \) be the set of all elements \( a_{ij} \) in labeled rows and unlabeled columns. Any \( a_{ij} \in B \) is such that \(-u < a_{ij} < L_\infty \). Otherwise column \( j \) will be labeled.
According to the algorithm, \((\text{new } a_{ij}) = a_{ij} - x\). But
\[-u - x < a_{ij} - x < L_\infty - x\]
so that
\[-u - x < (\text{new } a_{ij}) < L_\infty - x\]
and
\[-L_\infty < (\text{new } a_{ij}) < u.\]

Since \(-L_\infty < (\text{new } a_{ij}) < L_\infty\), the new elements of the set \(B\)
are strictly less than the \(L_\infty\)-norm in absolute value. Now,
\(V \cap (S \cup B) \neq \emptyset\) since the row (column) labeled \(*\) belongs to \(S(B)\) and
contains an \(L_\infty\)-position.

But according to the algorithm, we reduce at least this element.
Therefore, the \(L_\infty\)-norm Reducing Algorithm reduces the number of elements
of \(V\) by at least one element, since the other elements in labeled rows
and labeled columns or unlabeled rows and unlabeled columns are not
changed by the algorithm.

4.4.1. Lemma

None of the loop's entries is changed by "the \(L_\infty\)-norm Reducing
Algorithm".

Proof: If some entry of the \(u\)-loop is in a labeled row or column, then
the entire \(u\)-loop rows and columns are labeled.

And since the \(r\)'s for labeled rows are \(-x\) and the \(s\)'s for
labeled columns are \(x\) this implies the \(u\)-loop position will not be
changed by pivoting.

If no entry of the \(u\)-loop is in labeled row or column this implies
that none of these entries will be changed by the \(L_\infty\)-norm Reducing
Algorithm, since their corresponding \(r_i\)'s and \(s_j\)'s are equal to zero.
4.4.2. Lemma

The algorithm converges in a finite number of iterations.

Proof: If the algorithm did not terminate in a finite number of iteration, it would be necessary to repeat some basic feasible position, because the number of distinct bases is finite.

Since $u$ is uniquely determined by the loop positions and there exist only a finite number of loops, then repeating one would imply that the objective function is at the same value at the start of two different loop finding iterations. This is a contradiction. With a fixed $u$-loop, "the $L_{\infty}$-norm Reducing Algorithm" reduces one $L_{\infty}$-position with value $|a_{pq}|$ to $u$ and would not create any new $L_{\infty}$-position by Theorem (4.4.2).

Since only a finite number of entries are equal to a given $L_{\infty}$ value, this implies that in a finite number of steps the labeling algorithm either $L_{\infty}$ is reduced or one has found a new loop with strictly larger value of $u$ or one has found an optimal solution.
5. ANOTHER ALGORITHM FOR THE $L^\infty$-PROBLEM FOR COMMENSURABLE DATA

5.1 Introduction

A collection of numbers is said to be **commensurable** if they can all be expressed as integral multiple of a certain "quantum" $\delta > 0$. Certainly any set of integers is commensurable ($\delta = 1$), but more generally, any finite set of rational numbers is commensurable since it can be expressed in terms of a common denominator.

Therefore without loss of generality, we may assume that the entries of the table are integers because we can multiply each entry by the least common multiple of the denominators of all table entries. This common scalar factor will not affect the basic process of the algorithm pivoting and will change neither the number of, nor the nature of the iterations.

The purpose of this chapter is to find another finite algorithm to solve the $L^\infty$-problem.
The overall flow diagram of the algorithm:

1. **start**

2. **Input matrix**
   \[ A = (a_{ij}) \]

3. **Phase I Algorithm**

4. **is Flag True**
   - YES: **stop optimal**
   - NO
      - **A = (a_{ij})**
      - **G = L \bigcup T**
      - **L_\infty > u**

5. **is Gamma true**
   - YES: Loop Finding Algorithm
   - NO: **L_\infty-norm Reducing Algorithm**

6. **is Delta true**
   - YES: Loop Finding Algorithm
   - NO: **block**

7. **Loop**
   - YES
   - NO
Let the matrix $A_0 = (a_{ij})_{m \times n}$, where all $a_{ij} \in \mathbb{Q}$, be given. To apply the next algorithm, one updates $A_0 = \lambda A_0$ where $\lambda = \lambda_0$ is the least common multiple of the denominator of all the matrix entries.

Call the Phase I Algorithm of Chapter 4 with the following modifications:

i) update $u$ from step 4 by

$$u = \lambda_1 u$$

where $\lambda_1 = 2K = |L|$.  

ii) update $A_0$ for step 5 by

$$A_0 = \lambda_1 A_0.$$  

iii) update $\lambda = \lambda_1 \lambda$.

Therefore the output vector $X$ of step 5 is a $\lambda$-multiple of the real one. Also the output residual matrix from step 6 is a $\lambda$-multiple of the real one.

One notices that the scaler factor $\lambda$ does not change the nature of the algorithm.

If Phase I Algorithm terminates with optimal message, we can write

$$X_{\text{optimal}} = \frac{1}{\lambda} X,$$

$$(A_0)_{\text{optimal}} = \frac{1}{\lambda} A_0,$$

$$(u)_{\text{optimal}} = \frac{1}{\lambda} u,$$

$$(w)_{\text{optimal}} = w.$$
Otherwise, we call "the Labeling Algorithm".

5.2. Phase II Algorithm

The input matrix $A_0 = (a_{ij})$ for Phase II Algorithm is a matrix which satisfies Conditions A, B, and C of Chapter 2, and the algorithm tries to make it satisfy Condition D of the same chapter.

This can be done by calling a "Labeling Subroutine" which decides to call either "a Loop Finding Algorithm" which increases the value $u$ through a loop, or to call "$L_\infty$-norm Reducing Algorithm" which reduces the $L_\infty$-norm of the matrix.

5.2.1. Labeling Algorithm

**Given:** A loop $L$ alternating the values $u$ and $-u$, where $0 < u < L_\infty$, the $L_\infty$-norm of the matrix $A_0 = (a_{ij})_{m \times n}$.

Let $V = \{(i,j) / |a_{ij}| = L_\infty\} \neq \emptyset$.

**Step 0:** Pick a position $(p,q) \in V$.

**Step 1:** If $a_{pq} > 0$ ($a_{pq} < 0$) label column $q$ (row $p$) with a $0$ and the $p$th row ($q$th column) with a $*$.

**Step 2:** Label any row $i$ (column $j$) with a $\leq$ if $a_{iq} < -u$ ($a_{pj} > u$).
Step 3: label any unlabeled column \( j \) (row \( i \)) with a \( 2 \) if 
\[ a_{ij} > u \ (a_{ij} < -u) \] and row \( i \) is labeled \( 1 \).

Step 4: label any row \( i \) (column \( j \)) with a \( 3 \) if \( a_{ij} < -u \)
\[ (a_{ij} > u) \) and column \( j \) (row \( i \)) is labeled \( 2 \). And so on.

Step 5: Stop if any of the following holds:

a) If at some point row \( * \) (column \( * \)) is given a label, STOP. Go to "Loop Finding Algorithm".

b) If at some point no new row or column can be labeled, we construct "a block", STOP. Go to "\( L_\infty \) norm Reducing Algorithm."

5.2.2. Loop Finding Algorithm

Given: The row (column) labeled \( * \) is labeled \( K \). Scalar factor \( \lambda \).

Step 0: Initiate \( X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n) = 0 \ast \).

Step 1: If the row (column) labeled \( * \) is labeled \( K \), go to a column (row) labeled \( (K-1) \) where there is an element \( < -u \ (\geq u) \) in row \( * \) (column \( * \)) and column (row) labeled \( (K-1) \).

Step 2: Then go through column (row) \( (K-1) \) to find an entry in a row (column) labeled \( (K-2) \) with element \( > u \ (< -u) \), etc.
Step 3: Continue until one reaches a column or a row labeled 0. By adding the $L_\infty$-position cell in the row (column) labeled * and column (row) labeled 0, we construct the new loop $L$.

Step 4: Construct a sign pattern $(w_{ij})$ associated with loop $L$ in the following way: start with $w_{pq} = \text{sign } a_{pq}$ where $|a_{pq}| = L_\infty$, and alternate the sign through the cells of the loop $L$ (number of cells of $L$ is even), and $w_{ij} = 0 \forall (i,j) \notin L$.

Step 5: Calculate the new $u$

$$(\text{new } u) = \sum_{(i,j) \in L} a_{ij} w_{ij}$$

update $\lambda = |L|\lambda$,

$A_0 = |L|A_0$, where

$|L| =$ number of cells in loop $L$.

Step 6: Update $X = (r_1, r_2, \ldots, r_m, s_1, s_2, \ldots, s_n)$ by solving

$$a_{ij} - r_i - s_j = w_{ij} u \quad \text{for all } (i,j) \in L,$$

starting with $r_p = 0$ if $a_{pq} > 0.$ and $s_q = 0$ if $a_{pq} < 0.$
Step 7: Pivot by calculating

\[ a_{ij} = a_{ij} - r_i - s_j \quad \forall (i,j), \]

and check \( L_\infty \)-norm of the residual matrix, if \( L_\infty < u \), STOP. The given solution

\[ u = \frac{1}{\lambda} u \]
\[ A_0 = \frac{1}{\lambda} A_0 \]
\[ X = \frac{1}{\lambda} X \]
\[ w = w, \] is optimal.

Otherwise go to the "Labeling Algorithm."

5.2.3. \( L_\infty \)-norm Reducing Algorithm

Given: A block, a matrix \( (a_{ij})_{m \times n} \) containing a loop, alternate the values \( u \) and \( -u \), and \( L_\infty > u > 0 \).

Step 1: If \( a_{pq} > 0 \), let

\( a_1 \) be the minimum entry in labeled columns but unlabeled rows.
\(a_2\) be the maximum entry in labeled rows but unlabeled columns.

If \(a_{pq} < 0\) switch \(a_1\) with \(a_2\).

Step 2: Let \(d = |a_p - \text{sign}(a_{pq} )u| = L_\infty - u > 0\).

Step 3: Take \(x = \min\{d, u+a_1, u-a_2\}\).

Step 4: Add \(x\) to all unlabeled columns,

\(-x\) to all unlabeled rows,

0 otherwise.

Step 5: Check the residual matrix. If \(L_\infty < u\), STOP. The given solution is optimal.

\[u = \frac{1}{\lambda} u,\]

\[A_0 = \frac{1}{\lambda} A_0,\]

\[x = \frac{1}{\lambda} X,\]

\[w = w.\]

Otherwise, go to the "Labeling Algorithm."
5.2.1. Example

\[ A_0 = \begin{bmatrix} 4^- & 5 & 6 & 6 & 7^+ \\ 3^+ & 4^- & 5 & 5 & 6 \\ 2 & 3^+ & 4^- & 5 & 6 \\ 2 & 3 & 3^+ & 4^- & 5 \\ 1 & 2 & 2 & 3^+ & 4^- \end{bmatrix} \]

**Phase I**

\[ u = 7 - 4 + 3 - 4 + 3 - 4 + 3 - 4 + 3 - 4 = -1 \]

\[ \lambda = \lambda_0 = 2K = 10 \]

\[ x = (0,0,0,0,0,0,0,0,0,0) \]

Switch the sign pattern and update \( u = 1 \) and

\[ A_0 = 10A_0 \quad \text{and solving for } r_i \text{ and } s_j. \]

\[ A_0 = \frac{1}{10} \]

\[ \begin{array}{cccccc}
32 & 24 & 16 & 8 & 0 \\
-71 & 40^+ & 50 & 60 & 60 & 70^- \\
-63 & 30^- & 40^+ & 50 & 50 & 60 \\
-55 & 20 & 30^- & 40^+ & 50 & 60 \\
-47 & 20 & 30 & 30^- & 40^+ & 50 \\
-39 & 10 & 20 & 20 & 30^- & 40^+ \\
\end{array} \]
The result from Phase I

\[ X = \frac{1}{10} (-71, -63, -55, -47, -39, 32, 24, 16, 8, 0) \]

\[ A_0 = \frac{1}{10} \]

Call Loop Finding Algorithm.

**Loop Finding Algorithm Calculation**

\[ u = 7 + 1 + 1 + 1 = 10 \]

update \( \lambda = 4\lambda = 4(10) = 40 \)

\[ A_0 = 4A_0 \]
\[ X = \frac{1}{40} (-284,-252,-208,-188,-156,128,78,58,32,0) \]

\[ A_0 = \frac{1}{40} \]

Call \( L_\infty \)-norm Reducing Algorithm.
**L∞-norm Reducing Algorithm Calculations**

\[ a_1 = -4 \quad u + a_1 = 6 \]

\[ a_2 = 6 \quad u - a_2 = 4 \]

\[ d = |32 - 10| = 22 \]

\[ x = \min \{22, 6, 4\} = 4 \]

\[ X = \frac{1}{40} \{-188, -252, -212, -192, -160, 132, 82, 62, 36, 0\} \]

**Call Labeling Algorithm**

\[ A_0 = \frac{1}{40} \]

\[
\begin{array}{cccccc}
4 & -6 & 14 & -12 & -8 & 1 \\
0 & -10 & 10 & -16 & 12 & \\
0 & -10 & 10 & 24 & 12 & * 5 Loop \\
20 & 10 & -10 & 4 & 8 & 3 \\
12 & 2 & -18 & -4 & 0 & 3 \\
\end{array}
\]

**Call Loop Finding Algorithm.**

**Loop Finding Algorithm Calculation**

\[ u = 28 + 12 + 10 + 10 + 10 + 10 = 80 \]
Update $\lambda = 6\lambda = 240$

$A_0 = 6A_0$

$x = \frac{1}{240} \begin{bmatrix} -1728, -1512, -1272, -1151, -960, 792, 492, 372, 216, 0 \end{bmatrix}$

$A_0 = \frac{1}{240} \begin{bmatrix} 24 & -36 & 84 & -72 & -48 \\ 80 & 0 & -60 & 60^+ & -96 & -72^- \\ 0 & 0 & -60^- & 60 & 144 & 168^+ \\ 40 & 120 & 60^+ & -60^- & 24 & 48 \\ 72 & 12 & -108 & -24 & 0 \end{bmatrix}$

$x = \frac{1}{240} \begin{bmatrix} -1728, -1432, -1272, -1112, -960, 792, 472, 312, 216, -88 \end{bmatrix}$

$A_0 = \frac{1}{240} \begin{bmatrix} -88 & 24 & -56 & 24 & -72 & -136 \\ -88 & 80 & 0 & 80^+ & -16 & -80^- \\ -88 & 0 & -80^- & 0 & 144 & 80^+ \\ -88 & 160 & 80^+ & -80^- & 64 & 0 \\ 0 & 72 & -8 & \boxed{-168} & -24 & -88 \end{bmatrix}$

Call $L_\infty$-norm Reducing Algorithm.
\[ x = \frac{1}{240} \{-1816, -1520, -1360, -1200, -960, 800, 560, 400, 304, 0\} \]

\[ a_2 = -\infty \quad u - a_2 = \infty \]

\[ a_1 = 72 \quad u + a_1 = 80 + 72 = 152 \]

\[ d = |-168 + 80| = 88 \]

\[ x = \min\{88, 152, \infty\} = 88 \]

\[
\begin{array}{cccccc}
0 & 80 & 80 & 80 & 80 \\
-80 & 24 & -56 & 24 & -72 & -136 \\
-80 & 80 & 0 & 80 & -16 & -80 \\
-80 & 0 & -80 & 0 & 144 & 80 \\
-80 & [160] & 80 & -80 & 64 & 0 \\
-80 & [160] & 80 & -80 & 64 & 0 \\
\end{array}
\]

\[ A_0 = \frac{1}{240} \]

Call \( L_\infty \)-norm Reducing Algorithm.

\[ x = \frac{1}{240} \{-1896, -1600, -1440, -1280, -1040, 880, 640, 480, 384, 80\} \]
\[ a_1 = 0 \quad a_1 + u = 80 \]
\[ a_2 = -\infty \quad u - a_2 = \infty \]
\[ d = |160 - 80| = 80 \]
\[ x = \min\{80, 80, \infty\} = 80 \]

\[
\begin{array}{cccccc}
8 & 8 & 8 & 0 & 8 \\
-8 & -56 & -56 & 24 & -72 & -136 \\
-8 & 0 & 0 & 80 & -16 & -80 \\
-8 & -80 & -80 & 0 & \boxed{144} & 80 \\
-8 & 80 & 80 & -80 & 64 & 0 \\
-8 & 80 & 80 & -80 & 64 & 0 \\
\end{array}
\]

\[ A_0 = \frac{1}{240} \]

Call \( L_\infty \)-norm Reducing Algorithm.

\[ x = \frac{1}{240} [-1940, -1608, -1448, -1288, -1048, 888, 648, 488, 384, 88] \]
\[ a_1 = -72 \quad a_1 + u = -72 + 80 = 8 \]
\[ a_2 = -\infty \quad u - a_2 = \infty \]
\[ d = |144 - 80| = 64 \]
\[ x = \min\{64, 8, 8\} = 8 \]

\[ A_0 = \frac{1}{240} \]

\[
\begin{array}{cccccc}
56 & 56 & 56 & 0 & 56 \\
-56 & -56 & 24 & -80 & -136 \\
-56 & 0 & 0 & 80 & -24 & -80 \\
-56 & -80 & -80 & 0 & \frac{136}{\ldots} & \frac{80}{\ldots} \\
-56 & 80 & 80 & -80 & 56 & 0 \\
-56 & 80 & 80 & -80 & 56 & 0 \\
\end{array}
\]

Call \( L_\infty \)-norm Reducing Algorithm.

\[ \alpha_1 = -24 \quad u + \alpha_1 = 80 - 24 = 56 \]

\[ \alpha_2 = 24 \quad u - \alpha_2 = 80 - 24 = 56 \]

\[ d = \left| 136 - 80 \right| = 56 \]

\[ x = \min\{56, 56, 56\} = 56 \]

\[ x = \frac{1}{240} \left[-1904, -1664, -1504, -1344, -1104, 944, 704, 544, 384, 144\right] \]
Results

\[
A_0 = \frac{1}{240}
\]

\[
\begin{bmatrix}
0 & 0 & 80 & -80 & -80 \\
0 & 0 & 80 & -80 & -80 \\
-80 & -80 & 0 & 80 & 80 \\
80 & 80 & -80 & 0 & 0 \\
80 & 80 & -80 & 0 & 0 \\
\end{bmatrix}
\]

optimal matrix

\[
A_0^* =
\begin{bmatrix}
0 & 0 & \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\
0 & 0 & \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\
-\frac{1}{3} & -\frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & 0 & 0 \\
\frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & 0 & 0 \\
\end{bmatrix}
\]

\[
u^* = \frac{80}{240} = \frac{1}{3}
\]

\[
X^* = [-7.93,-6.93,-6.27,-5.6,-4.6,3.93,2.93,2.27,1.6,0.6]
\]
Check

\[
\begin{bmatrix}
3.93 & 2.93 & 2.27 & 1.60 & 0.60 \\
-7.93 & 4 & 5 & 6 & 6 & 7 \\
-6.93 & 3 & 4 & 5 & 5 & 6 \\
-6.27 & 2 & 3 & 4 & 5 & 6 \\
-5.60 & 2 & 3 & 3 & 4 & 5 \\
-4.60 & 1 & 2 & 2 & 3 & 4 \\
\end{bmatrix}
\]

\[
A^*_0 = \begin{bmatrix}
0 & 0 & 0.34 & -0.34 & -0.34 \\
0 & 0 & 0.34 & -0.34 & -0.34 \\
-0.34 & -0.34 & 0 & 0.34 & 0.34 \\
0.34 & 0.34 & -0.34 & 0 & 0 \\
0.34 & 0.34 & -0.34 & 0 & 0 \\
\end{bmatrix}
\]

5.2.2. Example

\[
A_0 = \begin{bmatrix}
10 & 2 & 3 & 4 & 5 \\
2 & 3 & 4 & 5 & 6 \\
3 & 4 & 5 & 6 & 7 \\
4 & 5 & 6 & 7 & 8 \\
5 & 6 & 7 & 8 & 20 \\
\end{bmatrix}
\]
Applying Phase I of Chapter 4, we have

\[
\begin{array}{cccccc}
1 & -10 & -5 & 0 & 0 \\
7 & 10^+ & 2 & 3 & 4 & 5^- \\
3 & 2^- & 3 & 4 & 5^+ & 6 \\
8 & 3 & 4 & 5^+ & 6^- & 7 \\
13 & 4 & 5^+ & 6^- & 7 & 8 \\
18 & 5 & 6^- & 7 & 8 & 20^+ \\
\end{array}
\]

\[A_0 = \]

with \[u = \frac{20 - 5^- + 10 - 2 + 5 - 6 - 6 + 5 - 6}{10}\]

\[= \frac{20}{10} = 2\]

\[x = (-7, -3, -8, -13, -18, -1, +10, +5, 0, 0)\]
The input matrix for Phase II is

\[
A_0 =
\begin{pmatrix}
0 & 5 & 1 & 3 & 6 & 2 & 3 \\
-2 & 10 & 6 & 2 & -5 \\
-6 & 2 & -2 & -6 & -5 \\
-10 & 2 & -2 & -2 & 3 \\
-14 & -2 & -6 & -10 & 2 \\
\end{pmatrix}
\]

\[L_\infty = 14, \ u = 2\]

Call Loop Finding Algorithm.

\[X = (-4, -3, -8, -13, -9, -1, 10, +5, 0, -6)\]

\[u = \frac{14 + 2 + 2 + 2}{4} = \frac{20}{4} = 5\]
Call $L_\infty$-norm Reducing Algorithm

\[ d = L_\infty - u = 11 - 5 = 6 \]

\[ a_1 = 2 + u + a_1 = 5 + 2 = 7 \]

\[ a_2 = -\infty + u - a_2 = \infty \]

\[ x = \min\{d, u+a_1, u-a_2\} = \min\{6, 7, \infty\} = 6 \]

\[ X = (-10, -9, -14, -13, -15, 5, 16, 11, 6, 0) \]
0 block + call $L_\infty$-norm Reducing Algorithm

\[ d = L_\infty - u = 10 - 5 = 5 \]

\[ \alpha_1 = 6 + u + \alpha_1 = 6 + 5 = 11 \]

\[ \alpha_2 = -\infty + u - \alpha_2 = \infty \]

\[ x = \min\{5,11,\infty\} = 5 \]

\[ X = (-15,-14,-19,-18,-20,10,16,16,11,5) \]
Call $L_\infty$-norm Reducing Algorithm

\[ d = L_\infty - u = 7 - 5 = 2 \]

\[ a_1 = 2 + u + a_1 = 2 + 5 = 7 \]

\[ a_2 = -\infty + u - a_2 = \infty \]

\[ x = \min\{2, 6, \infty\} = 2 \]

\[ X = (-17, -16, -19, -20, -22, 12, 18, 18, 13, 7) \]
\[ L_\infty = 6, \ u = 5 \]

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
-1 & 5 & 3 & 4 & 0 & -5 \\
-1 & -2 & 5 & 6 & 2 & 3 \\
-1 & -4 & 3 & 4 & 0 & -5 \\
-1 & -4 & 3 & 4 & 0 & -5 \\
-1 & -5 & 2 & 3 & -1 & 5 \\
\end{array}
\]

0 block + call \( L_\infty \)-norm

Reducing Algorithm

\[ d = 6 - 5 = 1 \]

\[ a_1 = 4 + u + a_1 = 5 + 4 = 9 \]

\[ a_2 = -\infty + u - a_2 = \infty \]

\[ x = \min\{1, 9, \infty\} = 1 \]

\[ X = (-18, -17, -20, -21, -23, 13, 19, 18, 14, 8) \]
optimal matrix

Check

A_0 =

\begin{array}{ccccc}
13 & 19 & 18 & 14 & 8 \\
-18 & 10 & 2 & 3 & 4 & 5 \\
-17 & 2 & 3 & 4 & 5 & 6 \\
-20 & 3 & 4 & 5 & 6 & 7 \\
-21 & 4 & 5 & 6 & 7 & 8 \\
-23 & 5 & 6 & 7 & 8 & 20 \\
\end{array}
5.3. Justification of the Algorithm

1. With a fixed u-loop, the \( L_\infty \)-norm algorithm reduces \( |a_{pq}| \) closer to \( u \) and no other entry is further from \([-u, u]\), since only a finite number of entries are equal to a given \( L_\infty \) number, therefore in a finite number of steps the labeling algorithm either the \( L_\infty \)-norm is reduced or one has found an optimal solution or one has found a new u-loop with strictly greater value of \( u \).

2. Every time through a loop finding algorithm raises \( u \), this can only happen a finite number of times, since \( u \) is unique, determined by the loop positions and there exists a finite number of loops.
5.3.1. Lemma

If some entry of the u-loop is in a labeled row or column, then the entire u-loop rows and columns are labeled.

Proof: Clear by construction.

It follows that none of the u-loop entries are changed by the $L_\infty$ norm Reducing Algorithm, since $r_i = s_j = 0$ for labeled rows and columns. If no entry of the u-loop is in a labeled row or column, then $x$ and $-x$ are added to the u-loop positions, so they are not changed. i.e., the u-loop remains unchanged under the $L_\infty$ norm Reducing Algorithm.

5.3.2. Lemma

The Loop Finding Algorithm strictly increases the value of $u$.

Proof: If a loop is found by the Loop Finding Algorithm one has for

$$\text{new } u = \frac{a_{pq} + |(K-1)\text{entries}|}{K},$$

where each $(K-1)\text{ entries}$ is $> u$ in absolute value by choice and $a_{pq} > u$, thus

$$\text{new } u > u.$$

5.3.3. Lemma

Let $x$ be the number found in the $L_\infty$-norm Reducing Algorithm, then

$$\text{new } a_{pq} = \text{old } a_{pq} - \text{sign(old } a_{pq})x$$
Proof: If \( a_{pq} > 0 \), column \( q \) is labeled, row \( p \) is not and if \( a_{pq} < 0 \), row \( p \) is labeled and column \( q \) is not. So
\[
\text{new } a_{pq} < \text{old } a_{pq} \quad \text{if } x > 0. \quad \text{Since } x < |a_{pq} - \text{sign}(a_{pq})u|
\]
\[
\text{new } a_{pq} > u
\]
therefore \( \text{new } a_{pq} \) is closer to \( u \) than \( \text{old } a_{pq} \) provided \( x > 0 \).

5.3.4. Lemma

The \( x \) constructed in \( L_\infty \)-norm Reducing Algorithm is positive.

Proof: \( d > 0 \) by choice. If \( a_1 + u = 0 \) then there exists an entry \( a_{ij} \) in a labeled column and unlabeled row such that \( a_{ij} + u = 0 \) implying \( a_{ij} = -u \), but by the Labeling Algorithm this row would be labeled. Contradiction, therefore
\[
a_1 + u > 0
\]
If \( u - a_2 = 0 \), then there exists an entry \( a_{ij} \) in a labeled row and unlabeled column such that \( u - a_{ij} = 0 \), i.e. \( a_{ij} - u \), but by the Labeling Algorithm this column would be labeled. Contradiction, therefore
\[
u - a_2 > 0
\]
Therefore, \( x = \min\{d, u+a_1, u-a_2\} \) is positive.
5.3.5. Lemma

Let

\[ A = \{(i,j)/a_{ij} \text{ is in a labeled row and unlabeled column}\}, \]

\[ B = \{(i,j)/a_{ij} \text{ is in a labeled column and unlabeled row}\}. \]

The entries in \( A \cup B \) are closer to \([-u,u]\) after the \( L_\infty \)-norm Reducing Algorithm.

Proof: If \( a_{ij} \) is such that \((i,j) \in A\), then \( a_{ij} < u \) otherwise column \( j \) would be labeled. According to \( L_\infty \)-norm Reducing Algorithm

\[ \text{new } a_{ij} = \text{old } a_{ij} + x. \]

But by definition \( a_2 > a_{ij} \), i.e., \( -a_2 < -a_{ij} \), so

\[ x < u - a_2 < u - a_{ij} \text{ thus } x + a_{ij} < u, \text{ i.e., } \text{old } a_{ij} < \text{new } a_{ij} < u, \]

this is closer to \([-u,u]\). If \( a_{ij} \) is such that \((i,j) \in B\), then \( a_{ij} > -u \) otherwise row \( i \) would be labeled. From the \( L_\infty \)-norm Reducing Algorithm, we have

\[ \text{new } a_{ij} = \text{old } a_{ij} - x, \]
But by definition $a_{ij} > a_1$, therefore

$$a_{ij} + u > a_1 + u,$$

thus

$$a_{ij} + u > x > 0 \quad \text{or} \quad a_{ij} > x > -u$$

implies new $a_{ij}$ is closer to $[-u,u]$. ■

5.3.6. Lemma

The algorithm converges in a finite number of steps for the commensurable data.

Proof: The data throughout the algorithm iterations are commensurable, since the commensurable numbers are closed under addition and subtraction. If $|a_{pq}| = L_\infty$ of the matrix, then with a fixed $u$-loop, the $L_\infty$-norm reducing Algorithm replaces $|a_{pq}|$ closer to $u$ and no other entry is further from $[-u,u]$, and since only a finite number of entries are equal to a given $L_\infty$-number, and the reduction and the data are commensurable, therefore in a finite number of steps the Labeling Algorithm either reduces the $L_\infty$-norm $|a_{pq}|$ to $u$ or one has found a new $u$-loop with strictly greater value of $u$ (the number of different $u$-loops is finite) or one has found an optimal solution.

Therefore the algorithm converges to the optimal solution in a finite number of iterations for commensurable data. ■
5.4. How We Can Fix the Algorithm for General Data

In the $L_\infty$-norm Reducing Algorithm we have if $|a_{pq}| = L_\infty$-norm of the matrix, $a_{pq} > 0$, and

$a_1$ minimum element in labeled column but unlabeled row,

$a_2$ maximum element in labeled row but unlabeled column,

d = dist$(a_{pq}, [-u,u]) = |a_{pq} - \text{sign}(a_{pq})u|

= L_\infty - u$

Then $x = \min\{d, u+a_1, u-a_2\}$.

Case (1): If $x = d$.

By adding $x$ to unlabeled columns and $-x$ to unlabeled rows, we reduce one of the $L_\infty$ position (specifically $a_{pq}$) to the value $u$ and no other entry is further from $[-u,u]$.

Case (2): If $x = u + a_1$

Adding $-x$ to unlabeled rows will make the element in the labeled column which equal to $a_1$ equal to $-u = a_1 - x$, but then $a_{pq} - x > u$. 
So, if we modify Step 0 of the Labeling Algorithm, instead of choosing another $L_\infty$-position in the next iteration, to choose the same $a_{pq}$-position whose entry new value is $a_{pq} - x > u$. Since the $L_\infty$-norm Reducing Algorithm does not change the elements in the labeled columns and labeled rows, therefore the previous labeling will still be the same, but in addition the row containing the new value of $a_1 = a_2 - x = -u$ will be labeled. But this means that in a finite number of iterations one either has a new loop with higher value of $u$ or the value of $a_{pq} < u$ which is Case (1).

Case (3): If $x = u - a_2$

Similar to Case (2), adding $x$ to unlabeled columns will make the element in the labeled row which is equal to $a_2$ equal to $u = a_2 + x$, and the value of $a_{pq}$ becomes $a_{pq} - x > u$. We proceed as in Case (2).

5.4.1. Theorem

If the $L_\infty$-norm Reducing Algorithm is modified so that if $a_{pq}$ is used to begin the algorithm one uses it until either $|a_{pq}| < u$
or new loop with higher value of $u$ if found, then this modified algorithm converges in a finite number of iterations.

**Proof:** According to the above observations, at each iteration of the $L^\infty$-norm Reducing Algorithm where no loop is found, one new row or column is labeled, so in a finite number of steps the row or column labeled $\ast$ is labeled or $|a_{pq}| < u$ and no new elements are higher than $u$ in absolute value. And since the number of different loops is finite, the algorithm will converge in a finite number of iterations.

### 5.4.1. Example

$$A_0 = \begin{bmatrix}
1 & 3 & -9 \\
-4 & 7 & 2 \\
10 & 8 & 5
\end{bmatrix}$$

**Results from Phase I of Chapter 4:**

$$A_0 = \begin{bmatrix}
1 & 3^- & -9^+ \\
-4^- & 7^+ & 2 \\
10^+ & 8 & 5^-
\end{bmatrix}$$

$$u = \frac{1}{6} (10 + 4 + 7 - 3 - 9 - 5) = \frac{4}{6} = \frac{2}{3}$$

$$x = \left( \frac{18}{3}, \frac{10}{3}, -\frac{28}{3}, 0, -\frac{29}{3}, \frac{11}{3} \right)$$
The input matrix for Phase II:

\[
\begin{bmatrix}
\frac{31}{12} & \frac{93}{12} \\
\frac{21}{3} & -\frac{2}{3} & \frac{2}{3} \\
-\frac{2}{3} & \frac{2}{3} & \frac{27}{3} \\
\frac{2}{3} & -\frac{33}{3} & -\frac{2}{3}
\end{bmatrix}
\]

\( A_0 = -\frac{62}{12} \)

Circle 2

3 Loop + Call Loop Finding Algorithm

\( u = \frac{1}{4} \left( -\frac{33}{3} + \frac{2}{3} + \frac{2}{3} + \frac{2}{3} \right) = \frac{13}{4} = \frac{39}{12} \).

\( x = \left\langle \frac{72}{12}, -\frac{22}{12}, -\frac{112}{12}, \frac{31}{12}, -\frac{23}{12}, \frac{44}{12} \right\rangle \)

\( L_\infty = \frac{115}{12}, \quad u = \frac{39}{12} \)

Call \( L_\infty \)-norm Reducing Algorithm
\[ a_1 = \frac{8}{12} + u + a_1 = \frac{39}{12} + \frac{8}{12} = \frac{47}{12} \]

\[ a_2 = -\infty + u - a_2 = \infty \]

\[ d = L_0 - u = \frac{115}{12} - \frac{39}{12} = \frac{76}{12} \]

\[ x = \min\{\frac{47}{12}, \infty, \frac{76}{12}\} = \frac{47}{12} \]

\[ x = \left\{ \frac{25}{12}, -\frac{22}{12}, -\frac{112}{12}, \frac{31}{12}, -\frac{23}{12}, \frac{44}{12} \right\} \]

\[ A_0 = \begin{bmatrix} 0 & \frac{11}{12} \\ \frac{-20}{12} & 38 & \frac{-39}{12} \\ \frac{-9}{12} & \frac{39}{12} & \frac{46}{12} \\ \frac{39}{12} & \frac{-29}{12} & \frac{-8}{12} \end{bmatrix} \]

\[ u = \frac{1}{4} \left( \frac{68}{12} + \frac{39}{12} + \frac{46}{12} + \frac{39}{12} \right) = \frac{192}{48} = 4 \]

\[ x = \left\{ \frac{-5}{12}, -\frac{31}{12}, -\frac{112}{12}, \frac{31}{12}, -\frac{23}{12}, \frac{55}{12} \right\} \]
\[ A^* = \begin{bmatrix}
  4 & \frac{3}{2} & -4 \\
  -4 & \frac{5}{2} & 4 \\
  \frac{39}{12} & -\frac{39}{12} & \frac{1}{4}
\end{bmatrix} \text{ optimal matrix} \]

Check

\[ A_0 = -\begin{bmatrix}
  \frac{5}{12} & 1 & 3 & -9 \\
  -4 & 7 & 2 \\
  -\frac{112}{12} & 10 & 8 & 5
\end{bmatrix} \]

\[ A^* = \begin{bmatrix}
  4 & \frac{3}{2} & -4 \\
  -4 & \frac{5}{2} & 4 \\
  \frac{39}{12} & -\frac{39}{12} & \frac{1}{4}
\end{bmatrix} \]
5.4.2. Example "Temperatures"

<table>
<thead>
<tr>
<th></th>
<th>Jan.</th>
<th>Feb.</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caribou</td>
<td>8.7</td>
<td>9.8</td>
<td>21.7</td>
<td>34.1</td>
<td>48.5</td>
<td>58.5</td>
<td>64.0</td>
</tr>
<tr>
<td>Washington</td>
<td>36.2</td>
<td>37.1</td>
<td>45.3</td>
<td>54.4</td>
<td>64.5</td>
<td>73.4</td>
<td>77.3</td>
</tr>
<tr>
<td>Laredo</td>
<td>57.2</td>
<td>61.9</td>
<td>68.4</td>
<td>75.9</td>
<td>81.2</td>
<td>85.8</td>
<td>87.7</td>
</tr>
</tbody>
</table>

The result of using Phase I of Chapter 4

\[
\begin{array}{cccccccc}
-53.1 & -45.8 & -64.3 & -71.8 & -77.1 & -81.7 & -91.8 \\
31.9 & 8.7 & 9.8 & 21.7 & 34.1 & 48.5 & 58.5 & 64.0 \\
12.8 & 36.2 & 37.1 & 45.3 & 54.4 & 64.5 & 73.4 & 77.3 \\
0.0 & 57.2 & 61.9 & 68.4 & 75.9 & 81.2 & 85.8 & 97.7 \\
\end{array}
\]

\[
u = \frac{1}{6} (87.7 - 57.2 + 36.2 - 37.1 + 9.8 - 64.0) = -\frac{24.6}{6} = -4.1\]

Change the sign pattern and take \( u = 4.1 \).
The input matrix for the modified Phase II:

\[
\begin{bmatrix}
-9 & -3 \\
6 & -12.5 & -4.1 & -10.7 & -5.8 & 3.3 & 8.7 & 4.1 \\
 & -4.1 & 4.1 & -6.2 & -4.6 & 0.2 & 4.5 & -1.7 \\
0 & 4.1 & 16.1 & 4.1 & 4.1 & 4.1 & 4.1 & -4.1 \\
\end{bmatrix}
\]

Call Loop Finding Algorithm

\[\mathbf{X} = (31.9, 12.8, 0, -53.1, -45.8, -64.3, -71.8, -77.1, -81.7, -91.8)\]

The results from the Loop Finding Algorithm:

\[u = \frac{1}{4} (16.1 + 4.1 + 4.1 + 4.1) = 7.1\]

\[\mathbf{X} = (37.9, 12.8, 0, -53.1, -54.8, -64.3, -71.8, -77.1, -81.7, -94.8)\]

\[L = 14.7, \quad u = 7.1\]
Call "L∞-norm Reducing Algorithm"

\[ a_1 = 4.1 + u + a_1 = 7.1 + 4.1 = 11.2 \]

\[ a_2 = -\infty + u - a_2 = \infty \]

\[ d = L_\infty - u = 14.7 - 7.1 = 7.6 \]

\[ x = \min\{11.2, \infty, 7.6\} = 7.6 \]

\[ X = (30.3, 5.2, -7.6, -45.5, -47.2, -56.7, -64.2, -69.5, -81.7, -87.2) \]

\[ L_\infty = 9.3 \quad u = 7.1 \]
Call \( L_\infty \)-norm Reducing Algorithm

\[ a_1 = 0.2 + u + a_1 = 7.3 \]
\[ a_2 = -\infty + u - a_2 = \infty \]
\[ d = L_\infty - u = 9.3 - 7.1 = 2.2 \]
\[ x = \min\{7.3, \infty, 2.2\} = 2.2 \]

\[ X = (28.1, 3, -9.8, -43.3, -45, -54.5, -62, -69.5, -79.5, -85) \]

\[
A^* = \\
\begin{bmatrix}
-6.5 & -7.1 & -4.7 & 0.2 & 7.1 & 7.1 & 7.1 \\
-4.1 & -4.9 & -6.2 & -4.6 & -2.0 & -3.1 & -4.7 \\
4.1 & 7.1 & 4.1 & 4.1 & 1.9 & -3.5 & -7.1 \\
\end{bmatrix}
\]

\textit{Optimal matrix}
Check

\[
\begin{array}{cccccccc}
-43.3 & -45 & -54.5 & -62 & -69.5 & -79.5 & -85 \\
28.1 & 8.7 & 9.8 & 21.7 & 34.1 & 48.5 & 58.5 & 64.0 \\
3.0 & 36.2 & 37.1 & 45.3 & 54.4 & 64.5 & 73.4 & 77.3 \\
-9.8 & 57.2 & 61.9 & 68.4 & 75.9 & 81.2 & 85.8 & 87.7 \\
\end{array}
\]

\[A^* = \begin{bmatrix}
-6.5 & -7.1 & -4.7 & 0.2 & 7.1 & 7.1 & 7.1 \\
-4.1 & -4.9 & -6.2 & -4.6 & -2.0 & -3.1 & -4.7 \\
4.1 & 7.1 & 4.1 & 4.1 & 1.9 & -3.5 & -7.1 \\
\end{bmatrix}\]
6. REFERENCES

Abdelmalek, N. N. Linear $L_1$-Approximation for a Discrete Point Set and $L_1$ Solutions of Overdetermined Equations. JACM, 1971, 18, 41-47.


7. ACKNOWLEDGEMENTS

I gratefully acknowledge the guidance and the help of Dr. A. M. Fink in both this dissertation and my graduate work. The excellent typing was done by Ms. Jan Nyhus to whom I am indebted. I would also like to thank my family especially my wife Magda for her moral support.
8. APPENDIX
INTEGER MMAX, NMAX, M, N, MAXIT, ID, I_COUNT
PARAMETER(MMAX=50, NMAX=50, MAXIT=30)
REAL*8 A(MMAX, NMAX), R(MMAX), C(NMAX), U, RSUM(MMAX), CSUM(NMAX)
LOGICAL FLAG
CALL ASSIGNFILE
CALL READMAT(MMAX, NMAX, M, N, A)
DO 51 I = 1, M
   RSUM(I) = 0.0
51 CONTINUE
DO 52 J = 1, N
   CSUM(J) = 0.0
52 CONTINUE
I_COUNT = 0
ID = 0
CALL PRINTI(MMAX, NMAX, M, N, A)
WRITE(6,*),' '
CALL LOOP(MMAX, NMAX, M, N, A, U, FLAG)

C PRINT MATRIX TO SEE IF SUBROUTINE LOOP AGREES WITH REALITY
C
C IF(FLAG) GO TO 1000
IF(I_COUNT.GT.MAXIT) THEN
   ID = 1
   WRITE(*,*),' ID = 1'
STOP
END IF
I_COUNT = I_COUNT + 1
CALL ROWPOL(MMAX, NMAX, M, N, A, R)
DO 53 I = 1, M
   RSUM(I) = RSUM(I) + R(I)
53 CONTINUE
PRINT TO CHECK ROW POLISH
CALL PRINTI(MMAX, NMAX, M, N, A)
CALL LOOP(MMAX, NMAX, M, N, A, U, FLAG)
IF(FLAG) GO TO 1000
CALL COLPOL(MMAX, NMAX, M, N, A, C)
DO 54 J = 1, N
   CSUM(J) = CSUM(J) + C(J)
54 CONTINUE
WRITE(6,*),' '
WRITE(6,*),' THE OPTIMAL MATRIX IS ', U
CALL PRINTMAT(MMAX, NMAX, M, N, A, RSUM, CSUM)
WRITE(6,*),' THE L-INFINITY NORM OF THE MATRIX IS ', U
WRITE(*,*) ' '  
WRITE(*,*) 'ICOUNT = ', ICOUNT  
STOP  
END

C
C
C
SUBROUTINE ASSIGNFILE
  CHARACTER*80 FILENAME
  WRITE(*,10) 'ENTER THE FILE WITH THE INPUT MATRIX: '  
  10 FORMAT(1X,A38,$)  
  READ(8,20) FILENAME  
  20 FORMAT(A80)  
  OPEN(UNIT=8,FILE=FILENAME,STATUS='OLD')  
  RETURN  
END

C
C
C
SUBROUTINE READMAT(MMAX,NMAX,M,N,A)
  INTEGER MMAX,NMAX,M,N,I,J
  REAL*8 A(MMAX,NMAX)
  READ(8,*) M,N  
  DO 30 J=1,N  
  30 READ(8,*) (A(I,J), I=1,M)  
  RETURN  
END

C
C
C
SUBROUTINE LOOP(MMAX,NMAX,M,N,A,U,FLAG)
  INTEGER MMAX,NMAX,M,N,II(400),JJ(400), K, I, J
  REAL*8 A(MMAX,NMAX),U
  LOGICAL FLAG,DELTA
  U=A(1,1)  
  II(1)=1  
  JJ(1)=1  
  DO 55 I=1,M  
    DO 56 J=1,N  
      IF(DABS(A(I,J)).GT.U)THEN  
        U=DABS(A(I,J))  
        II(1)=I  
        JJ(1)=J  
      END IF  
    56 CONTINUE  
  55 CONTINUE  
  K=0  
  CALL RSEARCH(MMAX,NMAX,M,N,II(K),JJ(K),JJ(K+1),A,FLAG)
IF(FLAG) GO TO 11
RETURN

11     CALL CLOSLOOP(II, JJ, K, II(K), JJ(K+1), DELTA)
IF(Delta) RETURN
CALL CSEARCH(MMAX, NMAX, M, N, II(K), JJ(K+1), II(K+1), A, FLAG)
IF(FLAG) GO TO 111
RETURN

111    CALL CLOSLOOP(II, JJ, K, II(K+1), JJ(K+1), DELTA)
IF(Delta) RETURN
GO TO 50
END

SUBROUTINE RSEARCH(MMAX, NMAX, M, N, IIN, JIN, JOUT, A, FLAG)
INTEGER MMAX, NMAX, M, N, IIN, JIN, JOUT, I, J
REAL*8 A(MMAX, NMAX), U
LOGICAL FLAG
U=A(IIN, JIN)
DO 57 J=1, N
IF(DABS(U+A(IIN, J)).LT.1.D-8)THEN
   FLAG=.TRUE.
   JOUT=J
   WRITE(*,*) 'FLAG FROM ROWSEARCH =', FLAG
   RETURN
ENDIF
CONTINUE
FLAG=.FALSE.
WRITE(*,*) 'FLAG FROM ROWSEARCH =', FLAG
RETURN
END

SUBROUTINE CSEARCH(MMAX, NMAX, M, N, IIN, JIN, IOUT, A, FLAG)
INTEGER MMAX, NMAX, M, N, IIN, JIN, IOUT, I, J
REAL*8 A(MMAX, NMAX), U
LOGICAL FLAG
U=A(IIN, JIN)
DO 58 I=1, M
IF(DABS(U+A(I, JIN)).LT.1.D-8)THEN
   FLAG=.TRUE.
   IOUT=I
   WRITE(*,*) 'FLAG FROM COLSEARCH =', FLAG
   RETURN
ENDIF
CONTINUE
FLAG=.FALSE.
WRITE(*,*) 'FLAG FROM COLSEARCH =', FLAG
RETURN
END
SUBROUTINE CLOSLOOP(II,JJ,K,ICHECK,JCHECK,DELTA)
INTEGER II(400),JJ(400),ICHECK,JCHECK,L,K
LOGICAL DELTA
DO 59 L=1,K
IF(II(L).EQ.ICHECK.AND.JJ(L).EQ.JCHECK)THEN
   DELTA=.TRUE.
   WRITE(*,*),'DELTA FROM CLOSLOOP =',DELTA
   RETURN
ENDIF
59 CONTINUE
DELTA=.FALSE.
WRITE(*,*),'DELTA FROM CLOSLOOP =',DELTA
RETURN
END

SUBROUTINE ROWPOL(MMAX,NMAX,M,N,A,R)
INTEGER MMAX,NMAX,M,N,I,J
REAL*8 A(MMAX,NMAX),R(MMAX),AMAX,AMIN
DO 60 I=1,M
   AMAX=A(I,1)
   DO 70 J=1,N
      IF(AMAX.LT.A(I,J))AMAX=A(I,J)
70 CONTINUE
   AMIN=A(I,1)
   DO 80 J=1,N
      IF(AMIN.GT.A(I,J))AMIN=A(I,J)
80 CONTINUE
   R(I)=0.5*(AMAX+AMIN)
   DO 90 J=1,N
      A(I,J)=A(I,J)-R(I)
90 CONTINUE
60 CONTINUE
RETURN
END

SUBROUTINE COLPOL(MMAX,NMAX,M,N,A,C)
INTEGER MMAX,NMAX,M,N,I,J
REAL*8 A(MMAX,NMAX),C(NMAX),AMAX,AMIN
DO 100 J=1,N
   AMAX=A(1,J)
   DO 200 I=1,M
      IF(AMAX.LT.A(I,J))AMAX=A(I,J)
CONTINUE
AMIN=A(1,J)
DO 300 I=1,M
IF(AMIN.GT.A(I,J))AMIN=A(I,J)
300 CONTINUE
C(J)=0.5*(AMAX+AMIN)
DO 400 I=1,M
A(I,J)=A(I,J)-C(J)
400 CONTINUE
100 CONTINUE
RETURN
END

SUBROUTINE PRINTMAT(MMAX,NMAX,M,N,A,RSUM,CSUM)
INTEGER MMAX, NMAX, M, N, I, J, SLICE, NSLICE, NLINE,NSTAR, GAB, NSTART,
1 NSTOP
REAL*8 A(MMAX,NMAX),RSUM(MMAX),CSUM(NMAX)
C
NLINE = 5
IF(NLINE.GT.(N+1))NLINE = N + 1
NSTAR = 16*NLINE
IF(MOD((N+1),NLINE).EQ.0) THEN
  NSLICE = (N+1)/NLINE
ELSE
  NSLICE = (N+1)/NLINE + 1
END IF
GAB=NSLICE*NLINE-N
C
DO 599 SLICE = 1,NSLICE
NSTART = (SLICE - 1)*NLINE + 1
NSTOP = NSTART + NLINE - 1
IF(SLICE.EQ.NSLICE) NSTOP = NSTOP - GAB
WRITE(6,11) ('*',J=1,NSTART)
WRITE(6,10) (CSUM(J), J=NSTART,NSTOP)
10 FORMAT(1X,615.6,4(*',615.6),/)
WRITE(6,11) ('*',J=1,NSTART)
11 FORMAT(1X,80A1)
IF(SLICE.NE.NSLICE) THEN
  DO 598 I=1,M
    WRITE(6,10)(A(I,J), J=NSTART,NSTOP)
  598 CONTINUE
ELSE
  DO 597 I=1,M
    WRITE(6,10)(A(I,J), J=NSTART,NSTOP),RSUM(I)
  597 CONTINUE
END IF
599 CONTINUE
RETURN
END
C
SUBROUTINE PRINT1(MMAX, NMAX, M, N, A)

INTEGER MMAX, NMAX, M, N, I, J, SLICE, NSLICE, NLINE, NSTAR, GAB
REAL*8 A(MMAX, NMAX)

NLINE = 5
IF (NLINE .GT. N) NLINE = N
NSTAR = 16*NLINE
IF (MOD((N), NLINE).EQ.0) THEN
  NSLICE = (N)/NLINE
ELSE
  NSLICE = (N)/NLINE + 1
END IF
GAB = NSLICE*NLINE - N

DO 35 SLICE = 1, NSLICE
  NSTART = (SLICE - 1)*NLINE + 1
  NSTOP = NSTART + NLINE - 1
  IF (SLICE.EQ. NSLICE) NSTOP = NSTOP - GAB
  WRITE (6, 11) ('*', J=1, NSTAR)
  FORMAT (1X, 80A1)

10 FORMAT (1X, G15.6, 4(' *', G15.6), /)
DO 36 I = 1, M
  WRITE (6, 10) (A(I, J), J=NSTART, NSTOP)
 CONTINUE

35 CONTINUE
RETURN

END

SUBROUTINE PRINT2(MMAX, NMAX, M, N, A, R, C)

INTEGER MMAX, NMAX, M, N, I, J, SLICE, NSLICE, NLINE, NSTAR, GAB
REAL*8 A(MMAX, NMAX), R(MMAX), C(NMAX)

NLINE = 5
IF (NLINE .GT. (N+1)) NLINE = N + 1
NSTAR = 16*NLINE
IF (MOD((N+1), NLINE).EQ.0) THEN
  NSLICE = (N+1)/NLINE
ELSE
  NSLICE = (N+1)/NLINE + 1
ENDIF
GAB = NSLICE*NLINE - N

DO 37 SLICE = 1, NSLICE
  NSTART = (SLICE - 1)*NLINE + 1
  NSTOP = NSTART + NLINE - 1
  IF (SLICE.EQ. NSLICE) NSTOP = NSTOP - GAB
  WRITE (6, 11) ('*', J=1, NSTAR)
  WRITE (6, 10) (C(J), J=NSTART, NSTOP)
  FORMAT (1X, G15.6, 4(' *', G15.6))
WRITE(6,11)('**',J=1,NSTAR)
11 FORMAT(1X,80A1)
IF(SLICE.NE.NSLICE) THEN
   DO 38 I=1,M
       WRITE(6,10)(A(I,J),J=NSTART,NSTOP)
38       CONTINUE
ELSE
   DO 39 I=1,M
       WRITE(6,10)(A(I,J),J=NSTART,NSTOP),R(I)
39       CONTINUE
ENDIF
37 CONTINUE
RETURN
END