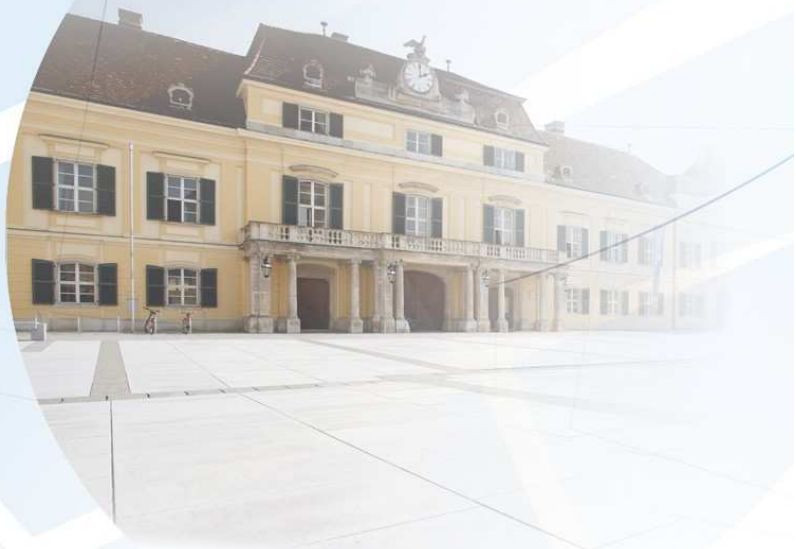
International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

# Implementation of an Algorithm for Scaling Matrices and Other Programs Useful in Linear Programming

Makowski, M. and Sosnowski, J.S.

IIASA Collaborative Paper
December 1981



# IMPLEMENTATION OF AN ALGORITHM FOR SCALING MATRICES AND OTHER PROGRAMS USEFUL IN LINEAR PROGRAMMING

Marek Makowski and Janusz Sosnowski

Systems Research Institute,
Polish Academy of Sciences

December 1981
CP-81-37



INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

ABSTRACT

Matrix scaling is an operation in which the rows and columns of a matrix are multiplied by positive scalars such that the elements of the scaled matrix are similar in absolute magnitude. This can be very important when dealing with models in which matrix elements can take a wide range of values, especially since certain mathematical codes, such as MINOS, require that the absolute values of all non-zero elements should be "reasonably" close to one.

This paper describes the implementation of an optimal scaling method proposed by Curtis and Reid. The algorithm is outlined and a users' guide to the program implemented on the VAX at IIASA is given. (The scaling algorithm has been linked to the MINOS package at IIASA by Zenon Fortuna, who has also developed a rescaling routine linked to this package.)

Two other programs useful in linear programming are also described: the first is designed to print out a matrix by rows, and also gives diagnostic aid; the second makes it possible to merge up to five LP models into one, generating a resultant MPSX file that may be used to solve multicriteria problems.

# IMPLEMENTATION OF AN ALGORITHM FOR SCALING MATRICES AND OTHER PROGRAMS USEFUL IN LINEAR PROGRAMMING

Marek Makowski and Janusz Sosnowski

Systems Research Institute,
Polish Academy of Sciences

## 1. INTRODUCTION

Matrix scaling is an operation in which the rows and columns of a matrix are multiplied by positive scalars such that the elements of the scaled matrix are similar in absolute magnitude. Many techniques have been proposed; these include geometric-mean scaling, arithmetic-mean scaling, and optimal scaling methods.

Tomlin (1975) surveys the use of scaling in LP problems and presents a comparison between several scaling algorithms. His conclusion is that scaling is not necessary if the LP problem is formulated by an experienced model builder who uses sensible units and avoids unnecessarily large or small matrix elements. However, this is only true for a few models, and building a model which is well-scaled can be a time-consuming process. We gained some experience of this problem while constructing a model for water system development in the Noteć region of Poland (Makowski 1981). Since observed data were used in this model, we had to deal with a wide range of values and hence a problem-oriented scaling procedure was necessary. Moreover, certain mathematical programming codes, for example MINOS (Murtagh and Saunders 1977), require that the absolute values of all non-zero elements should be "reasonably" close to one. However, in practice these coefficients are often considerably greater than one.

The advantages of scaling can be briefly summarized as follows:

1. Reduction in the number of simplex iterations required to solve the problem.
2. Improvement in the numerical behavior of the simplex method (through a decrease in the condition number of the basic matrix).
3. Simplification of checking procedures and the setting up of tolerances.

This paper describes an implementation of the optimal scaling method proposed by Curtis and Reid (1972), and discusses two other programs useful in the examination and formulation of LP problems.

## 2. OPTIMAL SCALING METHOD

Let us consider a linear program with constraints

$$A\,x = b \tag{2.1}$$

$$d \leq x \leq q \tag{2.2}$$

The row representing the goal function can be treated (for consistency and to ease implementation) in the same way as any other row, because, for a medium-sized problem, this would not reduce the scaling effect significantly.

Scaling involves the substitution of the set of equivalent constraints (2.3)-(2.5) for the original constraints (2.1) and (2.2).

$$(RAC)x' = Rb \tag{2.3}$$

$$C^{-1}d \leq x' \leq C^{-1}q \tag{2.4}$$

$$x = C\,x' \tag{2.5}$$

Here $R = \text{diag}\,(r_1,\ldots,r_m)$ and $C = \text{diag}\,(c_1,\ldots,c_n)$ are two diagonal matrices with $r_i > 0$, $c_j > 0$. In other words, we are multiplying

the ith row by $r_i > 0$ and the jth column by $c_j > 0$. If $a_{ij}$ is an element of matrix A, then the corresponding element $a'_{ij}$ of matrix RAC is given by

$$a'_{ij} = r_i c_j a_{ij} \tag{2.6}$$

Let $J_i$ be the index set of non-zero elements in the ith row, and $h_i$ be the number of non-zero elements in this row.

$$J_i = \{j | a_{ij} \neq 0\} \quad , \quad h_i = \dim J_i \quad , \quad i = 1,\ldots,n \tag{2.7}$$

Similarly, we define for the columns

$$I_j = \{i | a_{ij} \neq 0\} \quad , \quad m_j = \dim I_j \quad , \quad j = 1,\ldots,m \tag{2.8}$$

According to Curtis and Reid (1972), matrix A can be described as well-scaled if

$$\sum_{i=1}^{m} \sum_{j \in J_i} (\log|a_{ij}|)^2 \leq v \tag{2.9}$$

for some acceptable v. Note that only non-zero elements are included in the left-hand side of (2.9).

A matrix may be adjusted to fulfill criterion (2.9) through the use of scaling coefficients. These may be determined by minimizing the following function:

$$f(r_1,\ldots,r_m, c_1,\ldots,c_n) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j \in J_i} (\log_2 r_i c_j |a_{ij}|)^2 \tag{2.10}$$

Note that for numerical reasons it is advisable to use 2 as a logarithmic base and $r_i$ and $c_j$ should be integer powers of 2.

Let us define

$$w_i = \lg_2 r_i \qquad i = 1,\ldots,m \tag{2.11}$$

$$z_j = \lg_2 c_j \qquad j = 1,\ldots,n \tag{2.12}$$

Then minimization of (2.10) is equivalent to minimization of

$$F(w_1,\ldots,w_m,\ z_1,\ldots,z_n) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j\in J_i} (w_i + z_j + \lg_2|a_{ij}|)^2 \qquad (2.13)$$

We start by determining real values of $w_i$ and $z_j$ which are later rounded to the nearest integer value. The function F is quadratic with a non-negative definite Hessian with respect to $w = (w_1,\ldots,w_m)$ and $z = (z_1,\ldots,z_n)$. The gradient of the function F is defined by the simple formulae

$$\frac{\partial F}{\partial w_i} = n_i w_i + \sum_{j\in J_i} + \sum_{j\in J_i} \lg_2|a_{ij}| \qquad i = 1,\ldots,m \qquad (2.14)$$

$$\frac{\partial F}{\partial z_j} = \sum_{i\in I_j} w_i + m_j z_j + \sum_{i\in I_j} \lg_2|a_{ij}| \quad j = 1,\ldots,n \qquad (2.15)$$

The simple formulation of the gradient makes it possible to use a special version of the conjugate gradient method to minimize the function F. After minimization, the elements of the matrix A' are close to $\pm$ 1 and have only a small variance.

Consider the original problem:

$$\min \sum_{j=1}^{n} p_j\ x_j \qquad (2.16)$$

$$\sum_{j=1}^{n} a_{ij}\ x_j \le b_i \qquad i = 1,\ldots,m \qquad (2.17)$$

$$d_j \le x_j \le q_j \qquad j = 1,\ldots,n \qquad (2.18)$$

The scaled problem now takes the form

$$\min \sum_{j=1}^{n} p'_j\ x'_j \qquad (2.19)$$

$$\sum_{j=1}^{n} a'_{ij} x'_j \leq b'_i \qquad i = 1,\ldots,m \tag{2.20}$$

$$d'_j \leq x'_j \leq q'_j \qquad j = 1,\ldots,n \tag{2.21}$$

where

$$a'_{ij} = r_i c_j a_{ij} \quad , \; p'_j = p_j c_j \quad , \; b'_i = b_i r_i \quad ,$$

$$d'_j = d_j/c_j \quad , \quad q'_j = q_j/c_j \tag{2.22}$$

If $\hat{x}'_j$ and $\hat{\lambda}'_i$ are the primal and dual solutions of the scaled problem, then the primal and dual solutions of the original problem may be calculated easily using the relations

$$\hat{x}_j = c_j \hat{x}'_j \quad , \quad \hat{\lambda}_i = \hat{\lambda}'_i/r_i \tag{2.23}$$

## 3. THE ALGORITHM

The code developed for scaling is composed of two routines. The first handles the input of an MPSX file and the output of the scaled MPSX file. It also performs formal and consistency checks on the input files (see the users' guide in Section 4). This routine should be modified if linked directly with an LP package. The second routine performs the actual scaling.

Function (2.13) may be minimized by solving a system of linear equations but this problem would have dimensions $(n + m) \times (n + m)$ compared with only $n \times m$ for a scaled problem. Hence, and because of the structure of the problem, it seems reasonable to use a specialized conjugate gradient method, and we have therefore adopted one such method proposed by Tomlin (1975). We will not describe this method in detail; it is sufficient to note that no precise minimization of (2.13) is required. Curtis and Reid (1972) have suggested rounding the results to the nearest integer, and we have followed this approach to ensure that scaled and rescaled problems are identical,

even at the expense of increasing v (although this increase should not be significant).

Several applications of the program confirmed an obvious expectation that the value of function (2.13) (v) should decrease rapidly during the first few iterations. A typical run (274 columns, 250 rows) is presented as an illustration in Table 1. The following notation is used: gn is the square of the gradient norm; v represents the value of function (2.13) (i.e., the sum of the squares of the logarithms of the absolute values of the non-zero matrix elements), here divided by the number of elements considered; var represents the variance; min. coeff. and max. coeff. give the values of the smallest and largest matrix elements, respectively. We do not use the gradient norm as the stop criterion; we have chosen instead

$$\text{stop if } (v^k/v^{k-1} \geq \varepsilon) \tag{3.1}$$

where $v^k$ is the value of (2.13) in the kth iteration and $\varepsilon$ is a number slightly less than one. We found that less than 10 iterations are usually required to obtain a solution with $\varepsilon \sim 0.97$, irrespective of the size of the problem; this was also reported by Tomlin (1975). Also, if changes have been made to the matrix, the program may be started using scaling coefficients obtained in a previous run, and a satisfactory solution can often be obtained after only 2-4 iterations.

Murtagh and Saunders (1977) have suggested that the optimal value of the variables is less than 100, but at the same time not too close to zero - presumably to deal with nonlinear problems. In addition, there is no unique solution for scaling factors, despite the fact that the scaled matrices obtained are the same. We have therefore introduced an option that makes it possible to scale the non-zero bounds by the same factor as the corresponding column. This is achieved by replacing (2.13) by

$$F = \frac{1}{2} \sum_{i=1}^{m} \sum_{j \in J_i} (w_i + z_j + \lg_2|a_{ij}|)^2 + \frac{n}{2\lg_2(100)} \sum_{j \in B} (\lg_2|bnd_j| - z_j)^2 \tag{3.2}$$

Table 1. A typical run of the algorithm

| iter | gn | v | var | min. coeff. | max. coeff. |
|---|---|---|---|---|---|
| 1 | .20977+006 | 67.352 | .13760+007 | .14380-004 | .37330+008 |
| 1 | 94430. | 11.427 | 18133. | .25970-004 | .40538+006 |
| 2 | 61623. | 5.3163 | 21.325 | .42776-003 | 208.52 |
| 3 | 37230. | 2.6221 | 27.075 | .39398-002 | 360.18 |
| 4 | 14181. | 1.8025 | 18.839 | .20074-001 | 258.64 |
| 5 | 7309.9 | 1.0908 | 2.6722 | .15036-001 | 27.820 |
| 6 | 4430.6 | .84754 | 2.0175 | .21887-001 | 17.189 |
| 7 | 1061.5 | .69083 | 2.3809 | .40653-001 | 23.570 |
| 8 | 325.92 | .62609 | 1.8142 | .45586-001 | 18.333 |
| 9 | 145.86 | .59842 | 1.9773 | .37408-001 | 22.794 |
| 10 | 121.04 | .58523 | 1.8956 | .48150-001 | 21.110 |
| 11 | 43.111 | .57784 | 1.9173 | .37568-001 | 23.233 |
| 12 | 26.163 | .57360 | 1.8912 | .42186-001 | 22.261 |
| 13 | 38.864 | .57117 | 1.8778 | .39494-001 | 21.851 |
| 14 | 44.528 | .56992 | 1.8784 | .38709-001 | 21.791 |
| 15 | 27.583 | .56712 | 1.8997 | .38330-001 | 21.792 |
| 16 | 30.357 | .56329 | 1.8520 | .31154-001 | 20.227 |
| 17 | 36.233 | .56021 | 1.8234 | .27431-001 | 18.884 |
| 18 | 31.218 | .55898 | 1.8139 | .26274-001 | 18.239 |
| 19 | 20.215 | .55661 | 1.8150 | .24535-001 | 16.787 |
| 20 | 19.159 | .55436 | 1.8044 | .23443-001 | 14.981 |
| 21 | 16.702 | .55206 | 1.7956 | .22364-001 | 15.098 |
| 22 | 11.132 | .55055 | 1.7839 | .20904-001 | 15.740 |
| 23 | 19.509 | .54940 | 1.7663 | .19637-001 | 15.352 |
| 24 | 5.1783 | .54903 | 1.7595 | .19179-001 | 15.225 |
| 25 | 2.8303 | .54864 | 1.7477 | .18479-001 | 15.375 |
| 26 | 4.7975 | .54814 | 1.8083 | .18001-001 | 15.975 |
| 27 | 1.6822 | .54794 | 1.7514 | .17199-001 | 15.416 |
| 28 | 1.9503 | .54765 | 1.7702 | .16381-001 | 15.722 |
| 29 | 2.3080 | .54756 | 1.7722 | .16105-001 | 15.683 |
| 30 | 1.0800 | .54746 | 1.7684 | .15715-001 | 15.554 |
| 31 | .41004 | .54735 | 1.7766 | .15392-001 | 15.738 |
| 32 | .41714 | .54730 | 1.7744 | .15314-001 | 15.713 |
| 33 | .40357 | .54729 | 1.7751 | .15284-001 | 15.720 |
| 34 | .12791 | .54726 | 1.7776 | .15203-001 | 15.755 |
| 35 | .12263 | .54725 | 1.7751 | .15238-001 | 15.717 |
| 36 | .72795-001 | .54724 | 1.7777 | .15046-001 | 15.765 |

where B is the set of indices for the non-zero bounds and $bnd_j$ are the values of the bounds.

Hence, instead of (2.15) we have

$$\frac{\partial F}{\partial z_j} = m_j z_j + \sum_{i \in I_j} (w_i + \lg_2 |a_{ij}| - \frac{\eta \cdot s_j}{\lg_2(100)} \qquad (3.3)$$

where

$$s_j = \begin{cases} 0 & \text{if } bnd_j = 0 \\ \lg_2 |bnd_j| & \text{otherwise} \end{cases} \qquad (3.4)$$

The original program is not reproduced in this paper, but is available on the VAX (see Section 4); details may be obtained from the authors.

## 4. USERS' GUIDE

The program is written in FORTRAN ASCII and has been implemented on the VAX at IIASA. Both the source code and the compiled program are available on /uc/makowski/lp; the source code is on file "scale.f" (or "scale.f.C" if compacted), while the compiled program is on file "scale".

### 4.1 *Dimensions*

The program can handle problems of the following dimensions in its present form:

| | |
|---|---|
| No. of matrix elements | 8000 |
| No. of columns | 1000 |
| No. of rows | 700 |
| No. of bound sec. cards | 1000 |
| No. of ranges | 100 |

If it is wished to solve problems larger than this, certain changes must be made (see comment in source problem).

4.2 *Input/output units*

The following list gives the units used for various files:

2 - Matrix to be scaled (in MPSX format)
3 - Control cards (nine items, first seven in free format)
    1. Given starting (or optimal) point? (logical)
    2. Optimal scaling desired? (logical)
    3. Iteration log requested? (logical)
    4. Check for parallel rows requested? (logical)
    5. Maximum number of iterations (integer)
    6. Epsilon (stop criterion) (real)
    7. Eta (weight coefficient) (real)
    8. Right-hand side set (format A8)
    9. Bounds set (format A8)
4 - Starting or optimal point (scaling coefficient)
6 - Diagnostics of MPSX file and information on program flow
7 - Output of optimal scaling coefficients
8 - Iteration log (if requested)
9 - Scaled matrix (in MPSX format)

4.3 *Suggested values of control parameters*

The program uses the nine parameters listed in the previous section; a sample set of parameters is available on /uc/makowski/lp/consc. The following points should be noted when setting the parameters:

1. It is best to use previously obtained scaling coefficients as the starting point (in the same way that an old basis is used in LP).

2. Optimal scaling should not be suppressed unless there is some special reason for it.

3. The iteration log is useful only in certain applications, and hence it is generally better to avoid it.

4. A check of parallel rows should be performed only during the first run and after major changes have been made to the problem.

5. The maximum number of iterations should be about 15.

6. The recommended value of the stop criterion is slightly less than one (0.95 - 0.97).

7. The weight coefficient is concerned with the values of bounds and can be either zero or non-zero. Non-zero values of eta are recommended for problems that have significantly large (or small) bounds. There is no universal rule for choosing eta, and hence we advise starting with eta = 0.5 and going up to eta = 5; should the values of the bounds still be unsatisfactory, eta should be increased even further. (However, note that very high values of eta may reduce the efficiency of scaling.) On the other hand, if the values of the bounds are satisfactory then eta should be set at zero in the next scaling of the problem, provided that the same starting point is used.

8. The values of the right-hand side set and bounds set should obviously correspond to the desired sets of relevant values.

4.4 *Formal and consistency checking performed by the program*

The program checks for the following errors or inconsistencies:

- formal errors in the input file (also garbage and premature eof)
- illegal sequence of sections
- duplicated or undefined names (columns, rows) in all sections
- illegal indicators (describing rows and bounds)
- split columns
- inconsistency in the section dealing with bounds
- parallel rows (if requested)
- columns and rows that have zero or only one entry
- zero elements

4.5 *Information provided by the program*

All of the computer printout is self-explanatory, and hence there is no need to describe it in detail. It is only necessary to define the variables: v represents the value of (2.13) divided by the number of non-zero elements; var represents the variance; rhs represents s; ranges, bounds, coeff. refer to the numbers of the corresponding quantities that are non-zero.

4.6 *Reliability*

The program has been operational only since July 1981, and therefore only a small number of problems have been investigated. This also means that we may not have traced all of the bugs, and we would be grateful if future users could let us (or Zenon Fortuna at IIASA) know of any problems that they encounter.

The responses of the program to various situations that may occur in practice are summarized below.

1. Any error in the control cards (see Section 4.3) will cause the program to terminate.
2. The program continues reading the MPSX input file even if errors are detected, unless the errors are very serious (for example, if part of a row is missing).
3. If major errors are detected while the input file is being processed, the program terminates when processing is complete.
4. If errors are detected in the file of starting points (previously computed scaling coefficients) a new starting point is assumed and the process is continued; this is also done if the dimensions of the problem are inconsistent with those of a given starting point.
5. If the determination of scaling factors is interrupted on reaching the iteration limit, the current coefficients are stored.

### 4.7 *Calling statement*

Assume that the input MPSX file is on file "data". The calling sequence is then as follows:

```
/uc/makowski/scale 2=data 3=ctr 4=old 7=new 9=fil-9
```

File "ctr" should contain control cards - see Sections 4.2 and 4.3 and the example on uc/makowski/lp/consc. Files "old" and "new" contain scaling coefficients (previously computed values and current values), and file "fil-9" contains the MPSX file ready for use by MINOS.

The scaling algorithm has recently been linked to the MINOS package at IIASA by Zenon Fortuna, who has also developed a re-scaling routine linked to this package. (Please contact Computer Services for information on using these programs.)

### 4.8 *Other programs useful in LP*

There are two other programs that are useful in LP problems, and both are available in /uc/makowski/lp. The first not only provides the diagnostic help described in Section 4.4, but also makes it possible to print out the matrix by rows. This is the most concise way of printing matrices and is also the most useful format for examining them. We recommend that the matrix should be examined in this way after any major changes have been introduced into the problem. The printout generated should be self-explanatory.

The following list gives the units (note control cards on unit 3) used for various files:

- 3 - Control cards (four items, first two in free format)
  1. Matrix printout by rows requested? (logical)
  2. Checking for parallel rows requested? (logical)
  3. Right-hand side set (format A8)
  4. Bounds set (format A8)
- 4 - Matrix printout (by rows)
- 6 - Diagnostics and information on program flow
- 11 - Matrix to be examined (in MPSX format)

The program can handle problems of the following dimensions in its present form:

| | |
|---|---|
| No. of matrix elements | 8000 |
| No. of columns | 1000 |
| No. of rows | 700 |
| No. of bound sec. cards | 1000 |
| No. of ranges | 100 |

If it is wished to solve problems larger than this, certain changes must be made (see comment in source problem). The calling statement for this program is:

```
/uc/makowski/lp/byrows 3=ctr 4=matr 11=data
```

The second program makes it possible to merge up to five LP models into one, generating a resultant MPSX file that may be used to solve multicriteria problems, and/or aggregating rows selected according to the second character in the name of the row. This program can also print out the final matrix by rows.

It is possible to generate a problem equivalent to solving

$$x = \arg\max \min_i \frac{1}{\alpha_i \bar{q}_i} (q_i - \bar{q}_i)$$

$$q_i = \sum \beta_i x_i$$

where x belongs to a set defined by the aggregated model, $q_i$ is the ith criterion, $\bar{q}_i$ is the reference point for the ith criterion, $\alpha_i$ is the corresponding weighting coefficient (positive if the criterion is to be maximized and negative if it is to be minimized), and $i \leq 50$.

To solve this problem we must prepare a file which defines the criteria and reference points used. The first and second lines of the file contain the name of the first criterion (which must not be used as a row or column name in any of the other models or files to be read), and the values of the corresponding reference point $\bar{q}_i$ and weighting coefficient $\alpha_i$, respectively. The third and fourth lines give the corresponding information for the second criterion, and this process is continued until the names, reference points,

and weighting coefficients of all criteria have been listed. An asterisk is entered on the next line to indicate that this part of the file has been completed.

The next three lines give the name of the first criterion, the name of one of the variables included in this criterion ($x_i$), and the corresponding coefficient $\rho_i$. This information is repeated for all of the variables in the first criterion, and then for all variables in all other criteria. The list is once again concluded with an asterisk.

As an illustration, consider the following problem. We have two criteria

$$\text{goal } 1 = 0.5x_1 + 0.9x_2 - 0.3x_3$$

$$\text{goal } 2 = 0.3x_1 - 0.4x_4$$

and wish to maximize the first and minimize the second with (17,9) as the reference point. The file should be prepared as follows:

```
goal 1
17,1
goal 2
9,-1
*
goal 1
x1
0.5,
goal 1
x2
0.9,
goal 1
x3
-0.3,
goal 2
x1
0.3,
goal 2
x4
-0.4,
*
```

Note that this approach may be used to define a new goal function without involving any change in the aggregated model. Note also that this approach cannot be used simply by supplying an empty file on unit 2.

All of the actions performed by the program (including the diagnostic aid discussed above) are reported clearly and unambiguously. The only question requiring any comment is the problem of duplicated columns. In this case, if we allow the same name to be used for columns in different models, the output matrix must be sorted. However, this should not be confused with the problem of splitting columns in one model, which is always reported as an error.

The following list gives the units used for various files:

2 - Definition of multicriteria problem (if required)

3 - Control cards (11 items, in free format if not specified)

1. Title of aggregated model (format A8)
2. Matrix printout by rows requested? (logical)
3. Checking for parallel rows requested? (logical)
4. Number of models to be merged (integer)
5. Right-hand side set (format A8)
6. Bounds set (format A8)
7. Duplicated columns allowed? (logical)
8. Number of characters indicating aggregation (integer)
9. Characters (second in row name) indicating aggregation (format 1X,5A1)
10. Number of characters indicating change of row type (integer)
11. Characters (must be subset of that described in 9) indicating change of row type and a new type of row (format 5(1X,2A1))

4 - Matrix printout (by rows)

6 - Diagnostics and information on program flow

9 - Output matrix in MPSX format

11 - First matrix to be merged (in MPSX format)

12 - Second matrix

13 - Third matrix

14 - Fourth matrix

15 - Fifth matrix

The program can handle problems of the following dimensions in its present form:

| | |
|---|---|
| No. of merged models | 5 |
| No. of matrix elements | 8000 |
| No. of columns | 1000 |
| No. of rows | 700 |
| No. of bound sec. cards | 1000 |
| No. of ranges | 100 |

If it is wished to solve problems larger than this, certain changes must be made (see comment in source program). The calling statement for this program is:

/uc/makowski/lp/merge 3=ctr 4=matr 9=fil-9 11=data1 12=data2 etc.

where data1, data2, etc. contain the MPSX input files, one for each model. It is possible to merge up to five models, in which case units 11-15 are used to input the data.

## 5. VERIFICATION AND EVALUATION OF THE ALGORITHM

Two problems have been used to test the scaling algorithm discussed in this paper. The first involved a model for water system development in the Upper Noteć region, which was constructed using observed data with widely varying magnitudes (largest $3.37 \times 10^{+7}$, smallest $4.77 \times 10^{-6}$). The specialized scaling program has been used on the model both in Poland and at IIASA. We have found it impossible to solve the problem by MINOS without scaling. After several trials using different tolerances (some of which were really unacceptable), we were forced to give up, although the MPSX file is still available for other trials*. However, the scaled problem has been solved without difficulty. Our second example was a model describing individual farms. Although the difference between the largest and smallest elements is not very great (800 and 0.001, respectively) the application of scaling has cut down the number of iterations by almost half (1257 without scaling compared with 623 after scaling), only slightly decreasing the dif-

---

*The data for the LP problem that cannot be computed by MINOS without scaling are stored on the file uc/makowski/lp/sample.

ference between the largest and smallest coefficients (43.47 and 0.012, respectively). Other advantages of scaling will be reported in a separate publication in the near future.

## 6. CONCLUDING REMARKS

It is very easy to use the MINOS package in combination with the scaling algorithm: it is only necessary to specify the units as described above. Furthermore, it is hoped to employ the specs file in the near future to make scaling techniques even more accessible to the user.

More generally, our experience has shown that it is really worth scaling and verifying a matrix after any change in the problem being solved. Although it may seem ridiculous, it often happens that when introducing a simple modification the user makes an unconscious change, producing an error that is very difficult to detect, especially if the LP code declares: "status - optimal solution".

## 7. ACKNOWLEDGMENTS

Thanks are due to the System and Decision Sciences Area and the Food and Agriculture Program for support which accelerated the research, and to Dr. Z. Fortuna for helpful suggestions.

REFERENCES

Curtis, A.R. and J.K. Reid (1972) On the automatic scaling of matrices for Gaussian elimination. Journal of the Institute of Mathematics and its Applications, 10, pp. 118-124.

Makowski, M. (1981) A model for planning water system development in the Upper Noteć Region, Poland. Paper submitted to the Noteć-Silistra-Skåne Workshop, IIASA (forthcoming).

Murtagh, B.A. and M.A. Saunders (1977) MINOS. A large-scale non-linear programming system (for problems with linear constraints). User's Guide. Technical Report, System Optimization Laboratory, Stanford University.

Tomlin, J.A. (1975) On scaling linear programming problems. Mathematical Study 4. North-Holland Publishing Company, Amsterdam.