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

Software for Regional Studies: Adjustment Procedures for Integrated Balances

Lenko, M. and Kim, K.

IIASA Working Paper

WP-83-049

May 1983

SOFTWARE FOR REGIONAL STUDIES: ADJUSTMENT PROCEDURES FOR INTEGRATED BALANCES

Miloslav Lenko
Klim Kim*

May 1983
WP-83-49

*The Central Economic Mathematical Institute of the Academy of Sciences of the USSR, Moscow

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

PREFACE

The Tuscany case study is the last in the sequence of four regional case studies carried out in the Regional and Urban Development Group (RUD) at IIASA. The Tuscany study has developed as a joint effort between the RUD group and researchers from the Regional Institute for Economic Planning of Tuscany (IRPET).

One of the several efforts in the Tuscany study has been the construction of a regional accounting balance table connecting so called material and financial balances. The character of such integrated economic balance tables is discussed by Issaev and Umnov in WP-82-118.

The current paper presents an algorithm which can be used as a means to construct and adjust integrated balances. It also includes a detailed software description. In summary, the paper provides a documentation of work related to the Tuscany study.

Börje Johansson
Acting Leader
Regional and Urban
Development Group

May, 1983

CONTENTS

INTRODUCTION 1

STATEMENT OF THE PROBLEM 1

NETWORK INTERPRETATION OF THE PROBLEM 3

THE NETWORK ALGORITHM 4

USER'S MANUAL FOR BALANCE PROGRAMS
1. Definition of task and some items 7
2. Usage of programs 9
3. I/O files 10
4. Configuration, error messages, remarks 16

PROGRAMMER'S GUIDE FOR BALANCE PROGRAMS
Programming conventions 19

PROGRAMS 19

COMMON 20

ARRAYS AND VARIABLES 21

ERROR MESSAGES 24

FILES 27

DESCRIPTION OF PROGRAMS 28

Software for Regional Studies: Adjustment Procedures for Integrated Balances

Miloslav Lenko, Klim Kim

IIASA

Introduction

This report is devoted to the problem of analysis of sensitivity of balance tables, which arises by mathematical modeling, for example, economic-financial flow within a region, a trade market, a transport system and so on. The report contains both the statement of the problem and the algorithms of solution, including a well detailed software description.

Statement of the problem

The object of a formal analysis in this problem is a square matrix (NxN), elements of which are real positive numbers. An example of this table is shown in Figure 1. Everywhere further we will denote the element of the matrix placed in the i-th row and j-th column as A_{ij}.

The matrix A is called balanced, if the following relations

$$\sum_j A_{ij} = \sum_k A_{ki}$$

take place for all i=1, ... , N.

Generally speaking, considered matrixes need not be balanced, but a procedure to make them balanced is a main aim of the analysis.

Let us introduce vectors $c = (c_1, c_2, \cdots c_N)$ and $r = (r_1, r_2, \cdots, r_N)$ defined as row and column sums

$$c_i = \sum_j A_{ij} \tag{1}$$

$$r_i = \sum_k A_{ki} \tag{1}$$

for all i=1, ... , N and a vector $B = (b_1, b_2, \cdots b_N)$, where

$$b_i = r_i - c_i \tag{2}$$

It is obvious, that for a balanced matrix $c_i = r_i$ *or* $b_i = 0$ are valid.

Now we are able to state the following problem :

Problem 1: if the given matrix A is not balanced, find such values X_{ij} ,that will balance the new matrix with elements $A_{ij} = A_{ij} + X_{ij}$.

It is necessary to note that the problem does not have a unique solution. Moreover, this problem is not of practical value, because of the absence of any relations between new and old matrixes. For example, the simplest trivial solution of problem 1 is the following. Let elements of the new matrix have sufficiently great numbers H , then $X_{ij} = H - A_{ij}$ are a solution of the problem.

The statement of the problem should be defined more exactly.

The most natural additional condition is a result of a natural trend to introduce the smallest possible changes in the initial matrix.

There are two versions of the concept of **the smallest change**. Either the number of elements to be changed is kept to a minimum, or any changes in the relative values of the elements are kept as small as possible. Currently, only the second version is used in practice, although the first one may also be interesting from a theoretical point of view.

A more general description for the nonlinear case is given in (B.Issaev, A.Umnov, 1982).

Further we will consider only the following concrete formalization of the second concept, called below **uniform balancing**. Its mathematical statement is

Minimized Y *with respect all* X_{ij},

Subject to

$$abs(X_{ij}) \le Y * X_{ij}$$

for all i,j=1,2,...,N and

$$\sum_j (A_{ij} + X_{ij}) - \sum_k (A_{ki} + X_{ki}) = 0$$

for all i=1,2, ... ,N.

Using notatations (1) and (2), finally the main problem can be formulated as follows

$$\textit{Minimized } Y \textit{ with respect } X_{ij} \tag{3}$$

Subject to

$$\sum_j X_{ij} - \sum_k X_{ki} = b_i \tag{4}$$

$$X_{ij} \ge -A_{ij} * Y \tag{5}$$

$$X_{ij} \le A_{ij} * Y \tag{6}$$

It is easy to see, that the problem (3) - (6) is a linear programming problem (LP), which can be solved by a standard software tool. On the other hand, this LP-problem has a special structure. Therefore, it can also be solved by a more effective special algorithm.

In practice the choice of the software tool should be made by compromising. We have to provide possibilities of including some extra relations in the statement of the main problem. A general tool is required. But use of the model with many repetitions (for example, an interactive regime) is also possible. In these cases it is desirable to have a high speed software.

There is no doubt about what kind of general software tool may be used for the problem of the **uniform balance**. It is the standard simplex method.

A special structure of the problem (namely, a modified problem about **the minimal flow in a network**), may be a basis for developing both new, and more effective algorithm and software units.

Before begining discussion of the network interpretation of the problem to be solved, let us formulate a modified problem of the **uniform balance**.

We will suppose that the user is interested in solutions of the problem where some elements in the balanced matrix are fixed with a priori given values. Hence, we have

Problem 2: the problem of the **uniform balance with fixed elements**.

It is not necessary to give the mathematical statement of this new problem, because the statement will repeat the statement (3) - (6) of problem 1 with additional constraints $X_{ij} = 0$, subject to the initial matrix having $A_{ij} = f$, where f is the given value.

Network interpretation of the problem

The suggested approach to problem 2 is a generalization of an algorithm for solving flow problems, which are formulated for a network.

Now we give a network interpretation for problem 2.

We associate each pair of rows and columns possessing the same index with a node of a network with this same index. Each nonzero element of the matrix A_{ij} we associate with two oriented arcs (i,j) and (j, i). The nonnegative value X_{ij} of the flow along the arc (i,j) we will consider as increasing the value of A_{ij}, and the nonnegative value X_{ij} of the flow along the arc (j, i) - as decreasing the value of A_{ij}.

It will be convenient to enumerate all nonzero elements of the matrix A. The numbers will be k=1,2, ... ,M. In the same manner we will enumerate all arcs. The increasing arcs will have indexes k=1,2, ... , M and the decreasing arcs numbers k+M. The index k is the same for the pair of arcs belonging to the same element of the matrix A.

Therefore, for M elements of A we consider 2M oriented arcs, the first M of which are increasing and the last M are decreasing. Notice, that we will not take into account the diagonal elements, because they do not change balance constraints at all.

Now let us consider a small example. Let us have the following matrix

Table 1

j	1	2	3	4
1		50.	0.	0.
2	10.		40.	0.
3	0.	0.		30.
4	20.	0.	0.	

Denote as K, IB, IE, A the arrays of indexes of nonzero elements of the matrix A, the beginning and ending of the arcs, and the values of the elements. Then the total information about the network corresponding to the matrix can be collected in table 2.

Table 2

K	IB	IE	A
1	1	2	50.
2	2	1	10.
3	2	3	40.
4	3	4	30.
5	4	1	20.
6	2	1	50.
7	1	2	10.
8	3	2	40.
9	4	3	30.
10	1	4	20.

In figure 1 the picture of the network is shown.

Values b_i can be interpreted as values of a uniform flow at i-th node. If $b_i>0$, then the node is a source. If $b_i<0$, then the node is a sink.

In the given example we have $b_1=-20.$, $b_2=0.$, $b_3=10.$ and $b_9=-10.$. Hence, nodes 3 and 4 are sources and the node 1 is a sink.

The given rules of data preparing are not of great importance here, of course. But they are sufficiently convenient to explain algorithms and to interprete results. From the formal viewpoint it is sufficient to consider the problem of the **uniform balance** as the following network problem

Find the minimal value of Y , for which the feasible flow saturates all sinks of the network with channel capacities equal to $Y*A_k$.

A particular result follows from the interpretation given above. If the network has only a single pair source-sink, then the problem of the **uniform balance** can be reduced to the classical problem of the **maximal flow**.

Let us have the solution of the problem where the maximal flow equals f , subject to Y = 1 . Then it is possible to equate this value of the flow to the given b. by mean proportional changes of all flows of the arcs X_{ij} and channel capacities A_{ij} .

It gives us the solution of the special case of the **uniform balance** problem with value $Y=b_i / f$.

This result is very useful as we have a balanced matrix with a single fixed element.

One more imortant note.

In the above network interpetation of the **uniform balance** problem, some additional constraints can be introduced on the values of changes of the matrix elements. Namely, it is possible to forbid decreasing or increasing for some of the elements. To do this, we may simply adjust the corresponding channel capacities to the zero level.

The network algorithm

When the problem of the balance generates a network problem with several sources and sinks, the direct usage of the above simple algorithm is not possible. Here a transportation problem arises. The main specific feature of the problem is the fact that the channel capacities are linear functions of Y . It is necessary to find the minimal value of Y , when the problem is still feasible.

A very short description of the approach applied here is given below.

The whole process consists of two steps. At the first step a balancing flow is built without constraints on the channel capacities of the arcs. We may say that the problem is solved by a very large Y.

At the second step an optimal (with respect to Y) flow has been found. During this step the value of Y is iteratively changed. These changes are connected with a change of the structure of the optimal basis of the transportation problem.

The whole process will be out, if there is no possibility to decrease Y by changing flows in arcs.

Without going into details of proofs, we will formulate some geometrical properties of these basis solutions of the transportation problem.

If a value of Y and the corresponding distribution of the flows within the network are a basis solution of the LP-problem, then the set of flows in arcs can be described as follows.

The set of the arcs is divided into three groups

S - nonbasis arcs,
D - basis nonsaturated arcs,
G - basis saturated arcs.

The flows X_k will have the following values

$X_k = 0.$, if k $\in$ S
$0 \le X_k \le Y^* A_k$, if k $\in$ D
$X_k = Y^* A_k$, if k $\in$ G .

The most important property is that the subset D contains exactly N-2 arcs. The subset consists also of two tree-like components of connectedness D_1 and D_2 . There exists always an arc from G , which connects D_1 and D_2 . The set, which connects D_1 and D_2 , we will call **pseudo-cut** and denote as R .

In figure 2 a possible structure of a current basis is shown. S-arcs are shown by dot lines, D-arcs are shown by thin pointers and G-arcs are shown by thick pointers. R-arcs are shaded.

The above properties permit the formulation of optimal conditions for a basis solution.

A basis solution is optimal, if all nonsaturated arcs from R are oriented in the same direction (between D_1 and D_2) and all saturated arcs from R are oriented in the opposite direction.

In other words, a basis is optimal if the total channel capacity permitting the increase of the flow from D_1 to D_2 is zero. It means that the set R is a cut.

Therefore, to decrease Y (the norm of the channel capacity) it is necessary to decrease flows in all arcs from G , including from R. But to keep the balance conditions valid, all the changes must form a nonzero circulation. If such a circulation cannot be built, then the current basis solution is optimal. Figure 2 shows just this situation.

If R is not a cut, then the algorithm defines necessary circulations, in which arcs from G and at least one nonbasis arc are involved. The decreasing Y leads to increasing this circulation. It happens as long as the change of Y reaches its limit value. Then the rebuilding of basis becomes necessary.

The work of the algorithm may be illustrated for the network given in Figure 1. The sequence of iterations is shown in Figure 3.

The first balanced flow (with $Y = +\infty$) goes from node 3 to node 1 through node 4. The minimal value of Y, granted the feasibility of the flow, is
$max(X_k / A_k) = X_5 / A_5 = 20./20. = 1.$
There is a nonsaturated 5-th arc here, which divides the network into two subsets $D_1=(3,4)$ and $D_2 = (2,1)$.

Nonbasis arc 8 permits the formulation of a circulation to decrease the flow through arc 5 . Increasing this circulation leads to the saturation of arc 2 , which occured at $Y = 0.66$. And so on.

Notice, that for all nonsaturated arcs the inequality $X_k \leq Y^* A_k$ takes place. In the optimal basis solution all arcs between D_1 and D_2 consist of a cut, and this does not permit the decrease of the value of Y further.

The computer codes of the algorithm were elaborated by K.Kim in cooperation with a laboratory of mathematical programming of The Central Economic Mathematical Institute of the Academy of Science of the USSR and Regional Development Group of IIASA.

Reference

B. Issaev, A. Umnov INTEGRATED ECONOMIC BALANCE: ADJUSTMENT PROCEDURES, IIASA, WP-82-118, 1982

User's manual for balance programs

This manual describes how to use **BALK**, **BALM** programs for The Integrated Economic Balance model of Tuscany.

The manual describes how to use these programs on a **VAX 11/780** computer under the **UNIX** operating system. There are only several changes for using these programs on an **IBM** computer.

The manual contains several chapters:

1. Definition of task and some items
2. Usage, compilation and linkage
3. I/O files, their formats
4. Error messages and other recommendations

1. Definition of task and some items

Let's have a quadratic matrix, which contains n rows and n columns and which can be sparse. Let's call this matrix **balance table.**

For reasons of flexibility, divide this table equally rowwise and columnwise into several parts, let's call them **groups** , where each group can contain 1 to m sequential rows (and columns too).

The whole table consists now of several groups. Each group has its own size (nro of rows/columns). For the marking of each group we can use 1 (non-numeric) character, so the table will contain e.g. groups A,B,C etc.

Each row/column of our table - let's call it **account** should be marked by absolute number of this row/column or by the name of group and the relative number in it, e.g. B02, C01 etc. Let's call row **income account** and column **outcome account.** Similarly, each element of this table, let's call it **flow,** can be marked with the help of two numbers - number of the row and number of the column; or alternatively with the help of relative numbers in groups, e.g. B02C04, A01E04 etc. The places where groups are crossing these submatrices (which should not be quadratic) we call **blocks.**

Our first task is as follows:

Let's have one balance table, which is created from blocks. All these data create **data base.** For computing we use only some blocks. Let's call them **included blocks** according to our **scheme.** All flows from these included blocks must have the following property: The sum of data for each input account must be equal to the sum of data for the corresponding output account. Let's call this property **balancing.**

Generally, the input balance table need not be balanced. Our task is to get this balance in such a way, that the maximum relative changes for all flows must be minimized.

All this structure (division of the table into groups and so into blocks) was done from practical definition of the task. Each group and block has its own economic interpretation for this model. In order to make this construction more flexible, it is possible to change the size of groups with minor changes in input data etc.

The second task is to balance this table if we do some exogenous changes. These changes may be of the following type:

a/ let the resulting flow have exactly the initial defined value
b/ let the resulting flow increase only
c/ let the resulting flow decrease only (but still be >=0.)

At the solution of this task (and also the first task) we get new values of flows: and so a new table - let's call it **result table.** The differences between new and old values create a so called **correction table.** Of course, should the situation occur that the solution is infeasible in both cases, then the table cannot be balanced.

The third task is like the first two, but we have another additional property of data in the table. Some of this data must correspond to some other so called equations of **proportionality.**

For the solution of all these three types of problem a software package has been developed which contains, as a main part, 2 different programs for optimal balancing (because this task can be formulated as a linear programming problem). The first program is a network algorithm for optimization of flows in the network. The program was developed at the laboratory of applied software at **CEMI**, Moscow. The second program is the **MINOS** linear programming package from Stanford University.

What must the user define and prepare, and what results will he get?

1. The user must prepare all input data files in appropriate form. Here the user must divide the table into groups, giving names and sizes of each group. The user can also give the name for each group (40 chars max) which will be printed on output files. Then the user must prepare the whole data base - all blocks which contain nonzero flows. Then the user must prepare a list of included blocks and file with names for each account (also 40 chars long).

2. A so-called specification file will contain a description of our task which means names of all input and output files, some important parameters and type of task (balance only, balance with restriction, proportionality, compressed form of table etc.). If the user wants to fix some variables, he must prepare a so-called restriction file too.

3. The user gets results stored on output files, according to the options he has chosen. All output files are optional. There are e.g. input balance table file; table of correction file; results table file. All results can be stored on data file in the same format as data base file and used for further computing.

The user can also display data in so called graphical form, where each result flow is displayed as 1 character; for instance, . if it did not change; + or - if a maximum change was made, and so on (see chapter 3). Looking at this file the user can get an overview of how corrections data are distributed in the resulting table.

The next possibility is to get the file with some chosen accounts which are interesting for the user (not a complete table); e.g. input, correction and result income and outcome accounts. Finally, the user can define and get a file where input blocks will be stored in a form of matrix with subtotals for rows and columns.

4. The user has the possibility of choosing some of the following functions:

- balancing input table
- balancing input table with restrictions
- as before but with proportionality
- work with compressed version of table (first two tasks only), where each block is interpreted as 1 flow which is the sum of all flows from this block.

2. Usage of programs

All programs are written in **FORTRAN-77** language for **VAX 11/780** computer under the UNIX operating system. The programs were written in such a way that their portability is very good, as generally, they are written in **FORTRAN IV**, when only some statements must be changed. It is necessary to use virtual machine for their computing because of large volumes of arrays of variables (which depends on the type of work and amount of data) and the **MINOS** programming package.

Generally the user has 2 different program packages for solving his problems in IIASA's computer. There is a package using network algorithm (**BALK**) and a package using **MINOS** for solution (**BALM**).

User must translate all the programs with the f77 compiler and link them to executable form. There are several **FORTRAN** subroutines in one source file.

For package **BALK**, user must translate and link following files: **balnew.f bdata.f comput.f flow.f maxfld.f subr1.f vstup.f vystup.f** There is file **link.balk** where commands can be found on how to link programs.

For package **BALM**, user must translate the following files: **balprp.f bdata.f subr1.f vstup.f vystup.f gmpsp.f subp.f** and link them together with files where the **MINOS** package is. File on how to link these is **link.balm**.

Submitting the job for computing:
Before submitting, user must prepare all input files (some of them are optional), according to type of job. All files besides spec file are opened internally in program, so user must submit the job simply by typing
BALK 1=spec or **BALM 1=spec**

Choice of appropriate algorithm:
network algorithm enables solution of tasks of following types only:
- balancing non-balanced table
- balancing with restriction of type equality (fixing result flows)

This algorithm is very fast (approximately 30 times faster than **MINOS**). It serves for quick preliminary balancing of job.

MINOS enables solving of all kinds of tasks (fixing of all types of restrictions, proportionality), by generating an mps-file but solving is relatively slow.

General description of program:

Program consists of 3 parts:

- inputing of data
- optimization
- producing results.

First, program reads specification file. According to defined task program does all necessary functions, reads groups and included-blocks file. Then program reads database file. If the user chooses option to compress table, data are compressed during reading input block and each block is considered as consisting of one flow. At the moment of reading data, the output file with output blocks is also prepared (if user wishes to have such a file). Reading the names of

accounts follows (if user did not choose compress options - in that case names of accounts are equal to names of groups). If the user wants to have restrictions, restriction file is read, the same as for input account file and proportional file.

Then two programs can be used:

a/ network program - Description of this algorithm can be found in the first part of this paper. Program does the following: at first it checks whether the table is balanced. Then, according to user's wishes, it balances input matrix and/or balances it with restrictions.

b/ **MINOS** program - For using **MINOS**, a so called mps file must be ready before calling. This file is created automatically by a program **genmps**. If the user wants to use proportionality, the mps file is created slightly differently. First, a file **mpsfile** is created, where all data for non-proportional variables are stored. After that a file **mpsprop** is created where data from proportional equations only are stored. Finally, the program **merge** merges these two files producing a file **mpsall** which will be standard input mps file for **MINOS**. After calling **MINOS** results are stored in working array *z* and a program **solut** transfers these results from this array into the appropriate positions in array *res*. Output part prepares all other output files (according to user demands). First the balance file, then the correction file, result file, output account file, output database file and finally the output graph file are produced.

3. I/O files

There is a difference in managing the files in **IBM** and **VAX** computers. For **VAX** all files are opened internally within the programs. The user gives their numbers in specification file and their numbers are allocated at generation time. For **IBM** computers, files are opened externally and the numbers of files are given as they were made at the installation. The names of files in the specification file have here only information meaning. They are not used to open files.

It is possible to connect several files into one file (input as well as output files), which must be done during the installation, when the same numbers for several files are assigned. User must connect these files together and use blank line as **EOF** indicator of each file. Concerning input files, the user must know the sequence in which input files are read. That is: spec, groups, include, database, names, restriction(optional), accounts (optional) and proport(optional).

All necessary files could be divided into three parts:

a/ Input files
b/ Output files
c/ Working files

3.1. Input files:

1. specification file
2. database file
3. groups file
4. included blocks file
5. names file
6. restriction file [optional]
7. accounts file [optional]
8. proportional file [optional]
9. **specfile** for **MINOS** [optional]

3.2. Output files

All output files except **fort.9** are optional

1. standard output file **fort.9** and tty output
2. balance table file
3. corrections table file
4. results table file
5. graph file
6. output accounts file
7. output blocks file
8. output database file
9. **outfile** for **MINOS**

3.3. Working files

These files are used when **MINOS** algorithm is chosen only

1. **mpsfile**
2. **mpsprop** [only if proportionality]
3. **mpsall** [only if proportionality]
4. scratch file for **MINOS fort.8**
5. optional basis files for **MINOS**

a1/ Specification file:

The following is an example of such a file:

```
name of task        : Tuscany II.
type of algorithm   : MINOS
database file       : database
groups file         : group
names file          : names
included blocks file: incl
restriction file    : rest751
out bal. table file  : outbal
out correction file  : outcor
out result file      : outres
out graph file       : outgrp
inp & out account f.: acc       outacc
out database file   : no
proportional file   : propor
out blocks file     : no
compress            : no
balance of input     : no
restr & bal. of inp. : yes
tolerance eps       :   1.0000e-5
tolerance seps      :   1.0000e-3
tolerance epstol    :   1.0000e-3
```

The structure and position of elements are fixed. Each line starts with an explanatory text which is 22 chars long after which follow the parameter(s). All names of files are 12 characters long. Name of task (1.line) has a max of 40 chars, this occurs on output files. Type of algorithm should be keyword **Kim** or **minos** (2.line). If some file is optional and is not chosen, just type **no**. Two names of account files (input and output) must be given on line nro 12, name of output account file starts in column 35. Parameters compress (16.line), balance of input (17.line) and restr. & balance of input (17.line) must be keywords **no** or **yes**. There is a check to some extent e.g. if user does not choose balancing and wants to have output result or output correction file then error message - wrong spec file - occurs. Tolerances *eps* and *seps* serve as criterion for accuracy when balancing with network algorithm, *epstol* serves as criterion if the variable is on upper or lower bound at preparing graph file.

formats are: for lines 1-18 (22x,10a4)
for lines 19-21 (22x,e12.5)

a2/ database file

All data are stored as a sequence of data belonging to blocks. Database can also contain blocks which will not be included in scheme for computing (will not be in the list of incl. blocks). Blocks contain all nonzero flows. It is not possible to have 2 blocks with the same name in the database. This error is not checked by the program and if left it may cause strange results.

Block contains the head of block where name of block is 2 characters. If this name is not in list of included blocks, the data of this block are skipped. After head all nonzero flows follow for this block in a mode:
row index, column index, value of flow
Indexes are compared with values got from the table - dimensions of groups. All

values must be greater than 0. Values need not be sorted.
The tail contains the sequence: 0 0 0. Block can be empty.

Formats are for the head: (6x,2a1)
for flows:(2i5,e12.5)

An example follows:

```
block CD
1,2,14.
2,3,3.
4,1,11.
1,1,7.
0,0,0.
block CS
4,4,1.
0,0,0.
```

a3/ group file

File contains identifier of the group (1 character), dimension of the group and title of the group (40 chars).
Format is (a1,i3,1x,10a4)
Sequence of groups defines division of table.

a4/ included blocks file

This file contains identifiers of all blocks (2 characters), which must be taken from database and included in the table. It is possible to put character **Y** at the end of the row if you want to print this block on the output file of blocks (in matrix mode with subtotals). Blocks could be in the file in any order.
Format is (2a1,1x,a1)

a5/ names files

This file contains names (titles) for each account (40 chars max). There must be the account name at first, e.g. name of group (1 char), than index in this group, and then the title for account (40 chars.) on each line. Accounts may not be sorted. If user chooses compress option this file is not read and titles for accounts are the same as for groups.
Format is (a1,i2,1x,10a4)

a6/ restriction file

The file has similar structure to the database file. The data are grouped together according to blocks, which they belong to. For each restricted datum it is necessary to give type of restriction, which can be one of three possibilities:

- fixing datum for exact value, character is =
- character > means, that flow can increase only
- character < means, that flow can decrease only

It is not necessary to give any value at the end of the line if modes < or > were chosen. Then at each line user must define:
index of row, index of column, restr. type(1 ch), restr. value (if type is '=')
Format is (i3,i3,1x,a1,1x,e12.5)

Example:

```
block AB
1,4, = 15.6
2,4, >
0,0,0.
block UA
 31  2 = 1234.56
  2 12 <
   0  0    0.
```

a7/ accounts file

Here user defines which accounts (income and outcome) he wants to write in output accounts file. For each account it is possible to have its value for input flows, correction flows and result flows, what user defines by giving characters **I**, **C**, **R** to the fields for type of acc. The user defines on each line number of account (by means of identifier of the group and index in this group) and the type of account. User can, of course, choose **C** and **R** type only if he chooses an option for balancing input matrix (or bal with restr). If the user wants to write all accounts from one group, he uses char **0** (zero) instead of index within group. If the user wants to print sum of all accounts for the whole table, he uses character * as name of group.
Format is (a1,i2,1x,3a1)

Example:

```
* 0 ICR   (prints sums of all accounts for input,corrections and output)
A 2 I
A 3 I
A 4 I    (from group A user wants to print input accounts for rows 2,3,4 only)
C 0  CR   (print all accounts for group C, and so correction and result)
```

a8/ proportional file

This file has fixed structure.

Example:

```
A 01 31 K 01 09
A 01 31 L 01 08
A 01 31 A 01 31
C 01 01 A 01 31
C 02 02 A 01 31
P 01 01 A 01 31
```

The file has 6 lines, for each proportionality one line. These lines show which variables are included in the relations of proportionality. If user does not choose a proportionality, he must put name **X** for the name of group. There are 6 various types of proportionality, first three for columnwise proportionality are for tables AK, AL and AA respectively. Columnwise proportionality means, that if in input balance table value of one flow was 1/16. of the sum of this column of this block, the resulting flow must also be 1/16.-th of new sum of this column. The next three types of proportionality are concerning table CA (first and second row) and table PA (first row). The values of these flows are defined by the set of equations. Numbers in prop files give lower and upper limits for proportional variables, these values are checked with dimension of groups.

a9/ **specfile** for **MINOS**

If user has chosen the **MINOS** package, he must prepare a so called specification file (see **MINOS** reference manual). Its name is always **specfile**. Here user can change number of rows, columns and nro of elements according to his task. User can also define usage of basis files. Number of rows, cols and elements is approximately:

if our table has N accounts and M nonzero flows, then
nro of rows = 2 * M + N + 1
nro of cols = N + 1
nro of elements= 6 * M

b/ Output files

Output files are mostly created after solution. All files (except **fort.9**) are optional and their names are given in spec file. The **fort.9** and the standard tty output contain information about solution, e.g. number of accounts, flows, sum of flows , debalance, obj. function.
The source balance table file contains the whole table with all data from included blocks. Rows and columns are signed absolutely and relatively within groups. Only nonzero flows are displayed. Blocks are separated by lines. It is possible to glue together the table into one large table.
The correction and result table files (their format is equal to format of balance table) contain corrections and result flows respectively.
The graph table file contains the picture of the table, where each flow is presented by one of the following characters:

. - there was no change or flow was equal to zero
e - element was fixed exogenously
+ - change of this flow is on the upper bound
- - change of this flow is on the lower bound
o - change of this element is somewhere between lower and upper bounds.

It is possible to get the graph only after balancing.

If nro of accounts is > 128, graph is too big and it cannot be written.
The output accounts file contains accounts according to specifications from the input account file.
The output blocks file has all blocks written separately, with sums for all rows/columns.
The output database file has the same format as input database file. The values stored there are resulting flows. In this way this file can be used as input database file for next iteration.
The **outfile** is standard output file when **MINOS** is used. It is used for information purposes only.

c/ Working files

These files are created at the time of computing. All files have given names and all are used if **MINOS** algorithm only was chosen.
The **mpsfile** is mps file for **MINOS**
The **mpsprop** is also mps file, which is slightly different. It has only variables (rows,cols,...), which are used in proportional equations only. This file must be merged with file **mpsfile**. It produces the file **mpsall** which will be the input mps file for **MINOS** if proportionality was chosen.
The **mpsall** is produced as merging of files **mpsfile** and **mpsprop**
The **fort.8** is a scratch file for **MINOS**.

4. Configuration, error messages, remarks

a/ Program has defined some maximal sizes for some arrays:

maximal number of groups - mxlng - 30
maximal number of accounts - mxacc - 200
maximal number of incl. blocks - mxinc - 150
maximal number of flows - mxvar - 2000
maximal number of restrictions - mxres - 1000
maximal number of graph size - - 128

In the case of a larger job it is necessary to change the dimensions of these arrays and also the value of the corresponding mx... variables (see programmers' guide), to translate programs and to link them together

b/ Names of files are 12 characters long. Names of groups and accounts have a max of 40 chars. All character variables are declared as integer*4 (it means 4 characters per 1 variable). Hence for storing names of files it is array of 3 variables. For storing names of groups it is array of 10 variables. Some variables are initiated by data statements.

c/ For usage of **MINOS** single large array (of type real*8) with name **z** is necessary. The default length of this array is 50000 (variable nwcore). If this is too small, error diagnostic 28, 40 occurs.

d/ Real arrays and variables are declared as type real*8 for accuracy at inputing and computing.

e/ Tolerances are of three types:
eps, seps (1.e-3, 1.e-4) are used by network algorithm. They are used to check whether the difference income-outcome is small enough. User can change them in the spec file,
toleps (1.e-3) is used at computing variables for graph file to check if the flow is close to the upper or to the lower bound.

f/ objective function gives maximal relative difference for any flow in the table. This value can be read from the file **fort.9** or the graph file.

g/ error messages are displayed on tty output as 2 numbers. First number is code of error. Second number depends on the first one. Generally when reading input files it is the number of the line where the error occurs. At code 28 it is a program error (see Programmer's guide or **MINOS** manual). A list of error codes is at the end of this chapter. After you get an error message, you must either correct some data file or make changes in source programs. Look in Programmers guide to see what action may be taken.

h/ The following restrictions must be taken into account:

- when preparing database file all data belonging to one block must be stored in this one block
- for proportionality purposes:
  - group A must be the first group in list of groups
  - max size of group A is 31
  - max size of group K is 10
  - max size of group L is 9
  - max size of group U is 1
  - all names for groups A,K,L,C,P,S,R must be the upper case letters

i/ Restrictions for compressed form are like those for the compressed flows (not source flows).

j/ How to fix element which is not in database file (which is zero there)?
You may add this element into database with a very small value.

BUGS:

1. All output files are generated on finishing the optimization.
2. The user should beware of not having data of one block in different parts of database file.
3. There are some restrictions for the proportionality option as well.

ERROR CODES

1 [inpspc] - short or bad data on input file **balspec**
2 [inpgrp] - number of groups > mxgrp
3 [inpgrp] - size in group is <0 or >99
4 [inpgrp] - wrong data on group file (size of group)
5 [inpgrp] - group name is already used
6 [inpinc] - such a group does not exist
7 [inpinc] - block name is already used
8 [inpdat] - parameters in base file are not correct(out of range or v=0.)
9 [inpinc] - number of included blocks > mxinc
10 [inpgrp] - number of groups = 0
11 [inpdat] - number of variables > mxvar
12 [inpnam] - wrong data on name file
13 [inpnam] - group not found
14 [inpinc] - number of included blocks = 0
15 [vstup] - number of accounts > mxacc
16 [inpdat] - last block was not closed
17 [inpdat] - number of variables = 0
18 [inpres] - number of restrictions > mxres
19 [inpres] - bad data on restriction file
20 [inpres] - number of restrictions = 0
21 [absind} - element does not match
22 [inpres] - restriction has no variable
23 [pripr] - error at computing graph elements
24 [maluj] - more than 128 accounts for preparing graph
25 [comput] - number of variables > mmax
26 [comput] - fatal error bad program
27 [comput] - n or nel are too big, change dimensions
28 [comput] - error in flow
30 [comput] - number of fixed elements > nmax
31 [inpspc] - error in spec file
32 [inpinf] - error in data in input for acc file
33 [otvor] - open error
34 [outgrp] - obj < 0.000 000 01
35 [isrest] - something strange with data in restriction file & source data
36 [inpprp] - proport tables too small
37 [inpprp] - description outside limits
38 [inpprp] - such a group was not found
39 [inpprp] - such a block is not included
40 [inpinf] - groups outside limits for input acc file
41 [inpinf] - not such a group for input acc file
42 [inpprp] - error on proport. file
43 [inpspc] - algorithm is not corresponding to program used
44 [inpres] - restr. of such a type should not be solved with this algorithm.
45 [inpdat] - size of working arrays too small for storing block at preparing block output
46 [whatva] - restr. of proport. not found
47 [pprop] - bad proportional data or program
48 [inpres] - type of restriction does not match any type of < = >

Programmer's Guide for Balance Programs

Programming conventions

All programs are written in **FORTRAN-77** programming language, mostly in **FORTRAN IV** subset. Only some subroutines use new features of **FORTRAN-77** language. The portability of programs is possible. Because of large arrays and large **MINOS** package computer for implementation must be virtual. Character variables are stored in variables of type integer (resp. integer*4 which is the same) as 4 characters per variable. Some of them are initialized in data statement. Comparison for equality is often used. Names of files are 12 characters long (3 variables), texts are stored as 40 characters (10 vars) generally.

Real data are stored in real*8 variables for the reason of maximum possible accuracy at computing and reading input data.

For opening and closing files standard **FORTRAN** commands **open**, **close** are used. In some subroutines declaration character*x is used, also commands of type encode, decode (which is implemented as read or write from character variables) are used. It is used in functions **inpinf** (decode and character*2 declaration), **presn** (encode and ch*8) and **otvor** (ch*12). There is also an equivalence statement in routine **presn** and some small inconsistencies with types of parameters.

All these inconsistencies were corrected for the **IBM** version. Here commands marked as **IBM** or **VAX** must be chosen at implementation of programs for a particular computer.

There are 2 versions of programs now:
BALK - version where network algorithm for balancing is used. This is a very quick version but user can fix elements only.
link: file **link.balk**
files: **balnew.o, bdata.o, comput.o, flow.o, maxfld.o, subr1.o, vstup.o, vystup.o**
BALM - version where **MINOS** is used for balancing. The user can use all types of restrictions and proportionalities.
link: file **link.balm**
files: **balprp.o, bdata.o, subr1.o, vstup.o, vystup.o, gmpsp.o, subp.o** and **MINOS** subroutines

PROGRAMS:

Main programs, subroutines and functions are grouped into several files. The following files contain the following subroutines:
balnew.f - balnew, err
balprp.f - balprp, err
bdata.f - block data
comput.f - comput
flow.f - flow
maxfld.f - maxfld
subr1.f - absind, indabs, sums, cpf
vstup.f - vstup, inpspc, inpgrp, inpnam, inpinc, inpdat, inpres, inpinf, inpipr, otvor, zatvor
vystup.f - vystup, outbal, outcor, outres, zaptab, outgrp, pripr, maluj, outsum, outtab, head, indexy, vyber, presn, tlacri, outacc
gmpsp.f - ries, genmps, isrest, solut, calcfg
subp.f - pprop, isprop, genprp, merge, isresp, whatva

COMMON

```
      common /prop/ lna,izc1,izc2,ikcr,izk,ikk,izl,ikl,izp1,iu,
     *              izs,iks,izr,ikr,iprflg(6)
```

in pgms: **genprp,pprop,isresp,whatva,inpipr,solut**

lna - length of group A
izc1 - absolute index starting of group C
izc2 - absolute index for account C2
ikcr - absolute index end of group C
izk - absolute index for start of group K
ikk - absolute index for end of group K
izl - absolute index for start of group L
ikl - absolute index for end of group L
izp1 - absolute index for start of group P
iu - absolute index for start of group U
izs - absolute index for start of group S
iks - absolute index for end of group S
izr - absolute index for start of group R
ikr - absolute index for end of group R
iprflg(6) - flags which proportionality is chosen 1=yes/0=no

```
      common /dsi/ ndspc,ndmpsa,ndoldb,ndnewb,ndspcf,ndoutf,ndtty,
     *             ndf9,ndgrp,ndincl,ndnam,ndprop,ndf8,nddatb,ndrest,
     *             ndacc,ndobal,ndocor,ndores,ndogrp,ndoacc,ndoblk,
     *             ndodat,ndmps,ndmpsp
```

in pgms: **balnew,balprp,block data,err,comput,flow,maxfld,cpf,inpspc,inpgrp, inpnam,inpinc,inpdat,inpres,inpinf,inpipr,outbal,outcor,outres, outgrp,outacc,ries,genmps,genprp,merge**

All variables contain identifiers of files used in read/write statements. These values are assigned to them with the help of data statement and must be prepared at the installation of both programs. Identifiers are for following files:

ndspc - spec file for balans;
ndmpsa - **mpsall** file;
ndoldb - old basis file for **MINOS**;
ndnewb - new basis file for **MINOS**;
ndspcf - specfile for **MINOS**, it must be equal to 5;
ndoutf - **outfile** for **MINOS**, it must be equal to 6;
ndtty - standard tty output, it is usually 6;
ndf9 - list output file **fort.9**;
ndgrp - groups file;
ndincl - included blocks file;
ndnam - names file;
ndprop - proportional file;
ndf8 - scratch file for **MINOS**, it must be 8;
nddatb - data base file;
ndrest - restrictions file
ndacc - input accounts file;
ndobal - output balance table;
ndocor - output corrections table;
ndores - output results table;
ndogrp - output graph table;
ndoacc - output accounts file;
ndoblk - output blocks file;
ndodat - output database file;
ndmps - **mpsfile**
ndmpsp - **mpsprop** file

next 4 commons are from **MINOS** subroutines and are used in program **ries** only

```
common /lpcom / krhs,ns1,maxr,ierr,jr1
common /intcom/ itn,itnitm,nphs,kmodlu,kmodpi
common /fxcom / fx,sinf,wtobj,minimz,nfx,ninf,obj,nprob
common /core  / kz1,kz2,kz3
```

ARRAYS AND VARIABLES

ind1 [mxvar] - absolute row index of flow
ind2 [mxvar] - absolute column index of flow
val [mxvar] - input flow
res [mxvar] - result flow
group [mxlng] - identifiers of groups (1 char)
lgr [mxlng] - length of groups
grpnam [10,mxlng] - names of group parts 40 chars
blinc1 [mxinc] - identifiers of included blocks 1.part (1 char)
blinc2 [mxinc] - identifiers of included blocks 2.part (1 char)
inclpr [mxinc] - output this block to file or not (1 yes/0 no)
accnam [10,mxacc] - names of accounts (40 characters each)
infacc [mxacc] - information which account to print
(1=input balance, 2=correction, 4=results, other their sum)
sincom [mxacc] - income of account (sum of the row)
soucom [mxacc] - outcome of account (sum of the column)
subory [3,15] - names of all files (12 character each)
solnam [10] - name of our task (40 characters)

iflg [12] - flags
irest [mxres] - absolute row index for restriction
jrest [mxres] - absolute column index for restriction
nrest [mxres] - type of restriction 0= -1< 1>
vrest [mxres] - fixed value of restriction if type is =
ip1z [10] - absolute row index start of proport var
ip1k [10] - absolute row index end of proport var
ip2z [10] - absolute column index start of proport var
ip2k [10] - absolute column index end of proport var
cut,head,headr,stk,dfr,df0,df [mxacc] - arrays for network algorithm
a [2*mxvar] - array for network algorithm
arc,list,fl [4*mxvar] - arrays for network algorithm
z [nwcore] - working array for **MINOS**
alfa [31,31] - proport coefficients for block AA
beta [31,9] - proport coefs for block AK
gama [31,10] - proport coefs for block AL
sa [31] - sum of columns for block AA
sk [9] - sum of columns for block AK
sl [10] - sum of columns for block AL
c1a [31] - first row from block CA
c2a [31] - second row from block CA
p1a [31] - first row from block PA
zzz [31] - sum of column group A - AU(transp)
x [31] - zzz() - RA - SA
y [31] - x() - C2A - C3A - C4A
zz [] = z(kz1) array where solution variables are stored after **MINOS** solution
tb [50,50] - temporary array for storing block variables at preparing output block file
s1 [50] - temp array for sum of the row for block
s2 [50] - temp array for sum of the column for block
sub [3] - name of file (12 chars)
prvky [128] - temp array for taking vars
grupy [13] - names of groups (1 char) when printing out tables
oddel [13] - separator after accounts (blank or |) when printing out table
irel [13] - relative number within group when printing out tables
prem [2,13] - various according to usage (value of variable at double precision or character format
nwcore - length of array z
ipflag - flag if proportionality yes=1/ no=0
kz1 - index where results after **MINOS** solution start in array z()
imt - max length of arrays tb,s1,s2 default = 50
nmax - = mxacc
mmax - = mxvar
mmax2 - = mmax * 2
mmax4 - = mmax * 4
mx... - maximal length of some arrays
eps - tolerance for network program
seps - tolerance for network program
epstol - epsilon tolerance at preparing graph
obj - objective function after network algor.
istate - state of solution
infall - which account of sums to print
lngvar - actual number of flows
lnggr - actual number of groups

lnginc - actual number of included blocks
lngacc - actual number of accounts
lngres - actual number of restrictions
lngip - actual length of propor
minkim - what algorithm is used 1=**MINOS** 2=network program

ERRORS:

code of error [subroutine from which is called] - /subcode/ explanation
A: Action to correct error

1 [inpspc] - /line/ short file or bad data on input spec file
A: correct spec file
2 [inpgrp] - /line/ number of groups > mxgrp
A: change dimension of arrays group(), lgr(), grpnam() and variable mxlng in main program
3 [inpgrp] - /line/ size in group is <0 or >99
A: correct groups file
4 [inpgrp] - /line/ wrong data on groups file (size of group)
A: correct groups file
5 [inpgrp] - /line/ group name is already used
A: correct groups file
6 [inpinc] - /line/ such a group does not exist
A: correct included-blocks file or add such group into groups file
7 [inpinc] - /line/ block name is already used
A: correct included-blocks file
8 [inpdat] - /line/ parameters in database file are not correct (indexes out of range or v=0.)
A: correct database file
9 [inpinc] - /line/ number of included blocks > mxinc
A: change dimension of arrays blinc1(), blinc2(), inclpr() and mxinc in main program
10 [inpgrp] - /line/ number of groups = 0
A: correct groups file (it is empty)
11 [inpdat] - /line/ number of variables > mxvar
A: change dimension of arrays ind1(), ind2(), val(), res() value mxvar and other arrays depending on it for network algorithm in main program
12 [inpnam] - /line/ wrong data on names file
A: correct names file
13 [inpnam] - /line/ group not found
A: correct names file
14 [inpinc] - /line/ number of included blocks = 0
A: correct included-blocks file (it is empty)
15 [vstup] - /0/ number of accounts > mxacc
A: change dimension of arrays accnam(), infacc(), sincom(), soucom() and mxacc and other arrays for network algorithm in main program
16 [inpdat] - /line/ last block at reading database file was not closed
A: correct database file
17 [inpdat] - /line/ number of variables = 0
A: correct database file (no input datum is included)
18 [inpres] - /line/ number of restrictions > mxres
A: change dimension of arrays irest(), jrest(), nrest(), vrest() and mxres and also some arrays for network algorithm in main program
19 [inpres] - /line/ bad data on restriction file (e.g. block not closed or fixing at negative number)
A: correct restriction file

20 [inpres] - /line/ number of restriction = 0
A: correct restriction file (it was empty)
21 [absind} - /line/ element does not match (wrong group name or outside limits)
A: correct database or restriction file
22 [inpres] - /line/ you want to make restriction on non-existent flow
A: correct restriction file
23 [pripr] - /0/ error at computing graph elements division by zero
A: consult with author
24 [maluj] - /0/ more than 128 accounts for preparing graph
A: not possible to have graph file, remove from spec file
25 [comput] - /0/ number of variables > mmax
A: consult with author
26 [comput] - /0/ fatal error bad program
A: something strange with program, consult with author
27 [comput] - /0/ n or nel are too big, change dimensions
A: consult with author
28 [comput] - error in optimization algorithm, second number gives code of err
for network program:
1 - problem is unbounded
2 - problem cannot be solved
3 - y is too big
4 - wrong data
A: change task (change restrictions etc)
[ries]
for **MINOS**
1 - infeasible (change task)
2 - unbounded (change task)
3 - too many iterations (increase number of iterations on **specfile**)
>=4 - another error condition (look into **MINOS** manuals)
30 - not enough core at reading mps file (correct dimension of z() and nwcore in ries program)
40 - fatal error at reading mps file (perhaps increase number of rows, columns,elements ...)
30 [comput] - /0/ number of fixed elements > nmax
A: consult with author
31 [inpspc] - /0/ error in spec file definition of task is incorrect
A: correct spec file
32 [inpinf] - /line/ you want to have output correction or result accounts but no balancing function was chosen
A: correct input accounts file
33 [otvor] - /0/ open error
A: file cannot be opened, look for name of file, consult with system programmer
34 [outgrp] - /0/ obj < 0.000 000 01, it is not possible get graph file
A: release out graph filename from spec file
35 [isrest] - /0/ something strange with data in restriction file & source data
A: consult with author
36 [inpipr] - /line/ proportional tables too small
A: consult with author
37 [inpipr] - /line/ index in group outside limits at reading proportional file
A: correct proportional file
38 [inpipr] - /line/ such a group was not found at reading proportional file
A: correct proportional file
39 [inpipr] - /line/ such a block is not included at reading proportional file

A: correct proportional file

40 [inpinf] - /line/ index of group outside limits at reading input account file
A: correct input accounts file

41 [inpinf] - /line/ such a group does not exist at reading input accounts file
A: correct input accounts file

42 [inpipr] - /line/ error on proportional file - wrong number
A: correct proportional file

43 [inpspc] - /line/ algorithm is not corresponding to program used
A: change algor. on spec file of used **BALM** instead of **BALK** or vice versa

44 [inpres] - /line/ restr. of such a type should not be solved with this algorithm
A: correct restriction file

45 [inpdat] - /line/ size of working arrays too small for storing block at preparing block output at reading database file
A: change dimension of arrays tb(), s1(), s2() and imt in program **inpdat**

46 [whatva] - /0/ restr of proport not found; strange error
A: consult with author

47 [pprop] - /code/ bad proportional data or program
0 - group A is not first in list
1 - size of group A is > 31
2 - size of group C is < 2
3 - size of group K is > 9
4 - size of group L is > 10
5 - size of group U is not = 1
6 - 11 - limits on proportional variables does not match
A: change task (size of groups) or change program (consult with author) or do not use proportionality

48 [inpres] - /line/ type of restriction does not match to any type of < = >
A: correct restriction file

FILES:

O - open (identifier of file), C - close, R - read, W - write

All files are opened in subroutine **otvor**, but O means the subroutine from which **otvor** is called; same for close.

Identifiers are assigned by user at installation time. There is standard assigning for **VAX** version in this paragraph. Files are opened and closed internally for **VAX** version, but externally for **IBM** version. In this case if more files have the same identifier for **VAX**, they are still separate files. For **IBM** it means that several files are concatenated to one file, where blank line serves as **EOF** separator.

For **VAX** version there are some files where names are given in program (user cannot define their names in spec file). They are files: **specfile, fort.8, fort.9, mpsfile, mpsprop, mpsall.**

spec file:
 O: user(1), C,R: **inpspc**
database file
 O,C,R: **inpdat** (1)
groups file
 O,C,R: **inpgrp** (1)
incl.blocks file
 O,C,R: **inpinc** (1)
names file
 O,C,R: **inpnam** (1)
restriction file
 O,C,R: **inpres** (1)
accounts file
 O,C,R: **inpinf** (1)
proportional file
 O,C,R: **inpipr** (1)
specfile for **MINOS**
 O,C: **ries** (5), R: **MINOS**
standard tty output
 O: system (6); W: **flow,maxfid,ries,err,comput**
standard output file **fort.9**
 O,C: system (9), W: **flow,err,maxfid,ries,comput**
balance table file
 O,C: **outbal** (2), W: **outtab,head,tlacri**
correction table file
 O,C: **outcor** (2), W: **outtab,head,tlacri**
results table file
 O,C: **outres** (2), W: **outtab,head,tlacri**
graph file
 O,C: **outgrp** (2), W: **maluj,head**
output accounts file
 O,C: **outacc** (2), W: **outacc,head,outsum**
output blocks file
 O,C,W: **inpdat** (13)
output database file
 O,C,W: **zaptab** (2)

outfile for **MINOS**
O,C: **ries** (6), W: **MINOS**
mpsfile for **MINOS**
O,C,W: **genmps** (1), R:**MINOS**, O,C,R: **merge** (2), O,C,R: **cpf** (1)
mpsprop file for **MINOS**
O,C,W: **genprp** (3), O,C,R: **merge** (3)
mpsall file for **MINOS**
O,C,W: **merge** (4), O,C: **ries** (2), R: **MINOS**, O,C,W: **cpf** (2)
scratch file for **MINOS fort.8**
O,C: system (8), W,R: **MINOS**

Some remarks concerning how output files are prepared:

a/ preparing output balance, correction and result files: calling sequence of subroutines for preparing these files is the same, only input data are different. For balance file input data are got from array *val*, for result file it is array *res* and for correction it is array *res*, which is difference *res=res- val* and which is corrected back to previous values after preparing correction file. Calling sequence is: **sums** prepares sums for whole columns. Then **outtab** for each page (which contains 13 output accounts) prints heading (name of task), page number - subroutine **head**) and names for each output accounts (it prepares subroutine **indexy**, name of account **grupy** and relative numbers within these groups are *irel* and also separators after end of group in array *oddel* which is either blank or |. Then it prints for each input account values for corresponding 13 output accounts. These output accounts for each row are formed with a help of subroutine **vyber**. Then these numbers are stored in maximal possible accuracy into output array *prem* with a help of routine **presn** and are printed within a subroutine **tlacri**. The whole cycle is repeated for each page after the whole table is printed.

b/ preparing graphical output: sequence of routines is **pripr** , which prepares array *res* in such a way, that it stores values 0,1,2,4,8 for each variable according to whether this variable is no change(0), is on upper band(1), on lower bound(2), between(4) or fixed(8). Then routine **maluj** is called. Within this routine at first heading is printed and then for each row values are taken for these variables for the whole row from array *res* with the help of routine **vyber**. Then each variable according to its code is replaced with its corresponding graphical symbol (.+-oe) and is printed on line. Because of length of line of prints, where a maximum of 132 characters is possible, the maximal size of 128 accounts was chosen, so as not to split one row between several lines on the printer

Description of programs:

For Parameter section: most parameters are described in ARRAYS & VARIABLES section, here we give only parameters which are not described there, or need more explanation.

For Method section: there is no description if method is trivial.

BALPRP

program balprp
Call: vstup ries vystup
Called from: -
Purpose: main program for **BALM** where **MINOS** is used

ABSIND

subroutine absind(bl1,bl2,in1,in2,iab1,iab2,group,lgr,lnggr,line)
Call: err
Called from: inpdat inpres
Purpose: from relative indexes and group names gives absolute indexes for flow
Parameters: bl1,bl2 - group names (1 char) for row and column
in1,in2 - relative indexes within group
iab1,iab2 - absolute indexes
Method: At first computes absolute index of starting row for group, then index for particular row; the same for column

BLOCK DATA

block data
Purpose: initiates variables in common /dsi/

CALCFG

subroutine calcfg(mode,n,x,f,g,nstate,nprob)
Call: -
Called from: **MINOS** subroutine
Purpose: dummy subroutine for **MINOS**, it will never be called
Parameters: all are dummy variables

CPF

subroutine cpf
Call: -
Called from: ries
Purpose: copies file **mpsfile** to **mpsall** if no proportionality was chosen

ERR

subroutine err(i,j)
Call: -
Called from: absind inpdat inpgrp inpinc inpinf inpipr inpnam inpres inpspc isrest maluj otvor pprop pripr ries vstup comput
Purpose: prints error codes on **fort.9** and standard tty output and STOP
Parameters: i - code of error; j - subcode of error

GENMPS

subroutine genmps(ind1,ind2,val,lngvar,irest,jrest,nrest,vrest,lngres,
sincom,soucom,lngacc,ipflag,ip1z,ip1k,ip2z,ip2k,lngip)
Call: otvor isprop isrest zatvor
Called from: ries
Purpose: prepares mps file **mpsfile** for **MINOS**
Method: row, column, rhs and bound parts are prepared sequentially.
If flow is proportional (call of routine isprop is = 1) skip
commands which generate part of mpsfile for these variables. Similarly
skip commands if flow is fixed.
For flow with indexes i,j the column 'aiii.jjj' is generated
(here iii=100+i jjj=100+j). Two rows 'loiiijjj' & 'upiiijjj' are
also generated for each flow. Sequence of flows is the same as if they are
stored in arrays ind1,ind2,val. For each account i a row is generated
with name 'bal.iii'. All input flows are going with coef 1. into this
row and -1. for output accounts.
Objective function has name 'aim' objective variable has name 'u'.
Rhs has name 'side' and bounds 'bnd'.
For bounds part for restricted variables we generate following rows:
if type < then 2 rows: lo bnd aiii.jjj 0.
up bnd aiii.jjj fixed_value
if type = then 1 row: fx bnd aiii.jjj fixed_value
if type > then 1 row: lo bnd aiii.jjj fixed_value

GENPRP

subroutine genprp(ind1,ind2,val,lngvar,ip1z,ip1k,ip2z,ip2k,lngip,alfa,
beta,gama,sa,sk,sl,c1a,c2a,p1a,zzz,x,y,irest,jrest,
nrest,vrest,lngres)
Call: otvor isresp isrest isprop whatva zatvor
Called from: ries
Purpose: prepares proportional mps file **mpsprop**
Method: File **mpsprop** will contain only data for proportional variables
Also here row, column, rhs & bounds are prepared. This subroutine was
written especially for our 6 types of proportionality cases. For each
type of proportionality corresponding rows, columns and bounds are
generating only if particular type of proportionality was chosen.
For propor. in blocks AA AK AL names of variables will be a100.jjj which is
sum for the whole column of this block.

GO

subroutine go(z,nwcore,lpi)
Call: **MINOS** subroutines
Called from: ries
Purpose: main program for **MINOS** package

HEAD

subroutine head(odsi,sub,solnam,page)
Call: -
Called from: maluj outacc outtab
Purpose: Prints heading to output balance, correction, result and graph file
Parameters: odsi - identifier of file
page - page number of output file

INDABS

subroutine indabs(iab1,iab2,bl1,bl2,in1,in2,group,lgr,lnggr)
Call: -
Called from: maluj zaptab
Purpose: Converts absolute indexes of flow to indexes relative in block
Parameters: see routine absind
Method: at first it finds group to which indexes belong and then indexes

INDEXY

subroutine indexy(iab2z,iab2k,group,lgr,lnggr,grupy,oddel,irel)
Call: -
Called from: outtab
Purpose:for each page it computes names of groups, accounts and their relative separators
Parameters: grupy - names of corresponding 13 accounts for this page
oddel - separators - for end of group it is | otherwise blank
irel - array of relative indexes for corresponding 13 accounts
iab2z - starting absolute index of column
iab2k - ending absolute index of column

INPDAT

subroutine inpdat(sub,ind1,ind2,val,lngvar,mxvar,blinc1,blinc2,
lnginc,group,lgr,lnggr,iflag,iflgb,sub1,inclpr)
Call: otvor err absind zatvor
Called from: vstup
Purpose: Reads data from database file and prints output block file if necessary
Parameters: sub1 - name of output block file

INPGRP

subroutine inpgrp(sub,group,lgr,lnggr,mxlng,lngacc,grpnam)
Call: otvor err zatvor
Called from: vstup
Purpose: Reads data from groups file

INPINC

subroutine inpinc(sub,group,lnggr,blinc1,blinc2,lnginc,mxinc,inclpr)
Call: otvor err zatvor
Called from: vstup
Purpose: Reads data from include file

INPINF

subroutine inpinf(sub,infacc,lngacc,infall,group,lgr,lnggr,iflg)
Call: otvor err zatvor
Called from: vstup
Purpose: Reads data from input accounts file

INPIPR

subroutine inpipr(sub,ip1z,ip1k,ip2z,ip2k,mxip,lngip,group,lgr,
lnggr,blinc1,blinc2,lnginc)
Call: otvor err zatvor
Called from: vstup
Purpose: Reads data from input proportionality file

INPNAM

subroutine inpnam(sub,group,lgr,lnggr,accnam,lngacc)
Call: otvor err zatvor
Called from: vstup
Purpose: Reads data from names file

INPRES

subroutine inpres(sub,group,lgr,lnggr,ind1,ind2,lngvar,irest,jrest,
nrest,vrest,lngres,mxres,natype)
Call: otvor absind err zatvor
Called from: vstup
Purpose: Reads restriction file
Parameters: natype - type of algorithm user wants to use for solving his problem (1 for **MINOS**, 2 for network program)

INPSPC

subroutine inpspc(subory,solnam,iflg,eps,seps,epstol,minkim,natype)
Call: err zatvor
Called from: vstup
Purpose: Reads spec file

ISPROP

function isprop(ii,jj,ip1z,ip1k,ip2z,ip2k,lngip)
Call: -
Called from: genmps genprp solut
Purpose: If flow is of proportional type returns 1, otherwise 0
Parameters: ii,jj - absolute indexes for flow

ISRESP

subroutine isresp(ii,jj,irest,jrest,nrest,vrest,lngres,ifl1,ifl2,value)
Call: -
Called from: genprp
Purpose: If flow is in list of restricted & it is proport., returns ifl1=1 and ifl2 & value according to parameters in list of restricted variables for proportionality for types 1,2 or 3 it returns also index
Parameters: ii,jj - absolute indexes for flow
ifl1 - returns 1 if restricted, 0 otherwise
ifl2 - returns type of restriction
value - returns restricted value

ISREST

subroutine isrest(ii,jj,irest,jrest,nrest,vrest,lngres,ifl1,ifl2,
value,ind1,ind2,val,lngvar)
Call: err
Called from: genmps genprp
Purpose: tests if flow is in list of restricted variables, output is same as for routine isresp

MALUJ

subroutine maluj(odsi,ind1,ind2,res,lngvar,lngacc,solnam,sub,obj,
epstol,group,lgr,lnggr,blinc1,blinc2,lnginc)
Call: err head vyber indabs
Called from: outgrp
Purpose: routine prints output graph file
Method: at first prepares head, gets absolute indexes.
For each accounts it takes all flows(subr. vyber). According to their values assign one character for each flow (,+,-,e,o,.) and prints one line. Because of the limitation of paper width the whole graph can be on one page (not divided into several pages) the maximum number of accounts is limited to 128. If there are more accounts, it is not possible to have a graph output.

MERGE

subroutine merge
Call: otvor zatvor
Called from: ries
Purpose: Merges **mpsfile** with **mpsprop** producing thus **mpsall** as input file for **MINOS**

OTVOR

subroutine otvor(idsi,menof)
Call: err
Called from: genmps genprp inpdat inpgrp inpinc inpinf inpipr inpnam inpres merge outacc outbal outcor outgrp outres ries zaptab
Purpose: Opens file with identifier idsi and name menof. This routine is empty for 3IBM version.

OUTACC

subroutine outacc(sub,ind1,ind2,val,res,lngvar,sincom,soucom,lngacc,
infacc,infall,accnam,group,lgr,lnggr,epstol,solnam,grpnam)
Call: otvor head outsum zatvor
Called from: vystup
Purpose: writes ouput account file

OUTBAL

subroutine outbal(sub,ind1,ind2,val,lngvar,sincom,soucom,lngacc,
solnam,accnam,group,lgr,lnggr)
Call: otvor sums outtab zatvor
Called from: vystup
Purpose: writes output balance file

OUTCOR

subroutine outcor(sub,ind1,ind2,val,res,lngvar,sincom,soucom,lngacc,
solnam,accnam,group,lgr,lnggr)
Call: otvor sums outtab zatvor
Called from: vystup
Purpose: writes output correction table

OUTGRP

subroutine outgrp(sub,ind1,ind2,val,res,lngvar,irest,jrest,nrest,
vrest,lngres,solnam,lngacc,obj,epstol,group,lgr,
lnggr,blinc1,blinc2,lnginc)
Call: otvor pripr maluj zatvor
Called from: vystup
Purpose: writes output graph file

OUTRES

subroutine outres(sub,ind1,ind2,res,lngvar,sincom,soucom,lngacc,
solnam,accnam,group,lgr,lnggr)
Call: otvor sums outtab zatvor
Called from: vystup
Purpose: writes output results file

OUTSUM

subroutine outsum(ins,odsi,in1,in2,var,lngvar,sincom,soucom,lngacc,
accnam,group,lgr,lnggr,epstol,grpnam)
Call: sums
Called from: outacc
Purpose: routine prints sums or income/outcome accounts; it is called separately for input data, for results and corrections
Parameters: ins - if it is 0 then produces sums of all accounts, otherwise there are accounts for row/column ins
in1,in2,var - indexes and values for input, correction or results data

OUTTAB

subroutine outtab(odsi,in1,in2,var,lngvar,group,lgr,lnggr,sub,
accnam,lngacc,solnam,sincom,soucom)
Call: head indexy vyber presn tlacri
Called from: outbal outcor outres
Purpose: prints output table
Method: Array var contains values to print, which may be input data, corrections or results (it depends from what subroutine it is called). A first routine prints heading. It must get absolute and relative addresses for each output account and separators after flows (call routine indexy).Then sequentially 13 output accounts are printed on one page. For each row it takes flows (call vyber), prepares values in maximum possible accuracy (call presun) & prints one row (call tlacri).

PPROP

subroutine pprop(ind1,ind2,val,lngvar,soucom,lngacc,alfa,beta,gama,sa,
sk,sl,c1a,c2a,p1a,zzz,x,y,group,lgr,lnggr,ip1z,ip1k,
ip2z,ip2k,lngip)
Call: err
Called from: ries
Purpose: prepares necessary arrays & variables if proportionality was chosen for later usage

PRESN

subroutine presn(prvky,prem,kon)
Call: -
Called from: outtab
Purpose: prepares data from real*8 mode to formated mode in maximum possible accuracy
Parameters: prvky - max 13 values of flows (real*8)
prem - array (2,13) of 4 characters (implemented as integer*4) where values in formated mode will be
kon - number of arrays prvky (<=13)
Method: This routine is strongly dependent on **FORTRAN** dialect. The best way to transfer data from real*8 form to formated form is with a decode statement, which is not available for **IBM FORTRAN**. For **IBM** it is done with the help of file 8 (Data are written to file and read as character

PRIPR

subroutine pripr(ind1,ind2,val,res,lngvar,irest,jrest,nrest,vrest,
lngres,obj,epstol)
Call: err
Called from: outgrp
Purpose: prepares array res for graph output in such a way that it stores numbers depending on whether the flow exists, if the change was to the upper or lower bound, if it was fixed etc.

RIES

subroutine ries(ind1,ind2,val,res,lngvar,irest,jrest,nrest,vrest,lngres,
istate,obj,sincom,soucom,lngacc,ipflag,ip1z,ip1k,ip2z,
ip2k,lngip,group,lgr,lnggr)
Call: sums genmps pprop genprp merge otvor go zatvor solut err cpf
Called from: balprp
Purpose: controls solution with **MINOS** algorithm
Method: at first it prepares file **mpsfile**.If proportionality was chosen it prepares file **mpsprop**. Then these two files merge to produce thus **mpsall**, which is input mps file for **MINOS**. It copies **mpsfile** to **mpsall** if no proportionality. Then **MINOS** is called and after solution gets results (call solut).

SOLUT

subroutine solut(zz,ind1,ind2,res,val,lngvar,ipflag,alfa,beta,gama,sa,
sk,sl,ip1z,ip1k,ip2z,ip2k,lngip,c1a,c2a,p1a)
Call: isprop
Called from: ries
Purpose: gets results from working array z()

SUMS

subroutine sums(ins,in1,in2,var,lngvar,sincom,soucom,lngacc,sinp,sout)
Call: -
Called from: outbal outcor outres outsum ries
Purpose: returns accounts or sum of accounts
Parameters: ins - 0 for sum of accounts, integer number for corresp. account
sinp - sum of the whole input account
sout - sum of the whole output account

TLACRI

subroutine tlacri(odsi,i,grp,irl,prem,oddel,kon)
Call: -
Called from: outtab
Purpose: prints one line of output data, correction or results tables
Parameters: i - row number = absolute account number
grp - relative account name
irl - relative account number
prem - values of flows in character mode
oddel - separators after each flow (blank of | after group)
kon - number of columns

VSTUP

subroutine vstup(subory,solnam,iflg,group,lgr,blinc1,blinc2,accnam,
ind1,ind2,val,irest,jrest,nrest,vrest,infacc,infall,
mxvar,mxlng,mxinc,mxacc,mxres,lngvar,lnggr,lnginc,
lngacc,lngres,eps,seps,epstol,ip1z,ip1k,ip2z,ip2k,
mxip,lngip,grpnam,minkim,inclpr)
Call: inpspc inpgrp err inpinc inpdat inpnam inpres inpinf inpipr
Called from: balprp balnew
Purpose: calls sequentially all input routines (if necessary) to read input data

VYBER

subroutine vyber(in1,in2,var,lngvar,i1,ia2z,ia2k,prvky)
Call: -
Called from: maluj outtab
Purpose: searches in arrays in1,in2,var for flows between interval ia2z ia2k for input account i1
Parameters: i1 - absolute input account number
ia2z - absolute output account starting interval
ia2k - absolute output account ending interval
prvky - values moved from array var

VYSTUP

subroutine vystup(ind1,ind2,val,res,lngvar,group,lgr,lnggr,blinc1,
blinc2,lnginc,accnam,sincom,soucom,lngacc,infacc,
infall,irest,jrest,nrest,vrest,lngres,subory,solnam,
iflg,epstol,obj,grpnam)
Call: outbal outcor outres outacc zaptab outgrp
Called from: balprp balnew
Purpose: Calls main output subroutines for preparing output files

WHATVA

subroutine whatva(ii,jj,ind1,ind2,val,lngvar,alfa,beta,gama,oldval,value)
Call: -
Called from: genprp
Purpose: returns value of restricted variables in tables AA, AK, AL

ZAPTAB

subroutine zaptab(sub,ind1,ind2,res,lngvar,group,lgr,lngg)
Call: otvor indabs zatvor
Called from: vystup
Purpose: writes data (results flows) to output database file

ZATVOR

subroutine zatvor(idsi)
Call: -
Called from: genmps genprp inpdat inpgrp inpinc inpinf inpipr inpnam inpres inpspc merge outacc outbal outcor outgrp outres ries zaptab
Purpose: closes file with identifier idsi - empty for **IBM** version

BALNEW

program balnew
Call: vstup comput vystup
Called from: -
Purpose: main program for network version, it calls input module, then subroutines for solving problem with network algorithm and then output module

COMPUT

subroutine comput(cut,head,headr,stk,dfr,df0,df,a,arc,list,fl,ind1, ind2,val,res,lngvar,irest,jrest,nrest,vrest,lngres, num,cc,y,eps,seps,nmax,mmax,mmax2,mmax4)
Call: err flow
Called from: balnew
Purpose: controls solution with network algorithm - checks balance, balances input data without or with restrictions

FLOW

subroutine flow(n,e,edge,fl,a,df,y,cc,df0,cut,head,list,dfr,stk,headr, nmax,mmax,mmax2,mmax4,eps,seps)
Call: maxfld
Called from: comput
Purpose: finds minimax flow
Parameters & method: see comments in program and another part of this manual

MAXFLD

subroutine maxfld(n,e,edge,fl,df,cut,flow,r,code,head,list,dfr,stk, headr,nmax,mmax,mmax2,mmax4,eps)
Call: -
Called from: flow
Purpose: constructs maximal flow in directed network
Parameters & method: see comments in program and another part of this manual

Remarks concerning MINOS

Changes into standard version of **MINOS 3.4**

1. instead of #implicit real*8(c-g,o-z)# use #implicit real*8(a-g,o-z)# in all **MINOS** subroutines.
2. in subroutine **initlz** set #nwordr=1#
3. change #abs# to #dabs# in routines: **crash, factor, forms, insert, invert, loadn, mps, rgitn, setx, solprt**
4. in subroutine **unpack** set #real*8 a#
5. in subroutine **mpsin** change parameters ne,m,krhs to ne2,m2,krhs2 before #call mps(3,...# instead of last 3 lines insert lines:

```
ne2 = 2 * ne
mm2 = 2 * m
krhs2 = 2 * krhs
call move (...,0,ne,ne2)
call move (...,krhs2,n,mm2)
call move (...,krhs2,n,mm2)
```

6. in subroutine **go** description is #subroutine go(z,nwcore,lpi)#
7. in subroutine **go** save ierr parameter:
   before label 100 insert line : icnt=0
   after label 100 add line : icnt=icnt+1
   after call minos add line : if(icnt.eq.1) iersol=ierr
   after label 900 add line : ierr=iersol
8. remove tabulator from all programs
9. remove commands concerning measuring of time consumed by **MINOS**

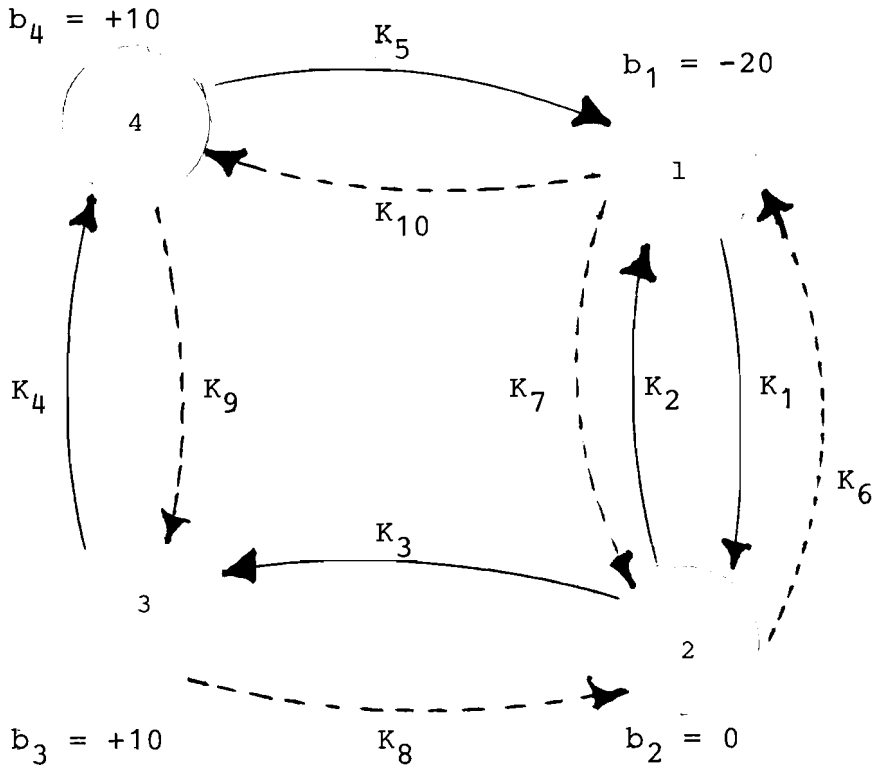


Figure 1.

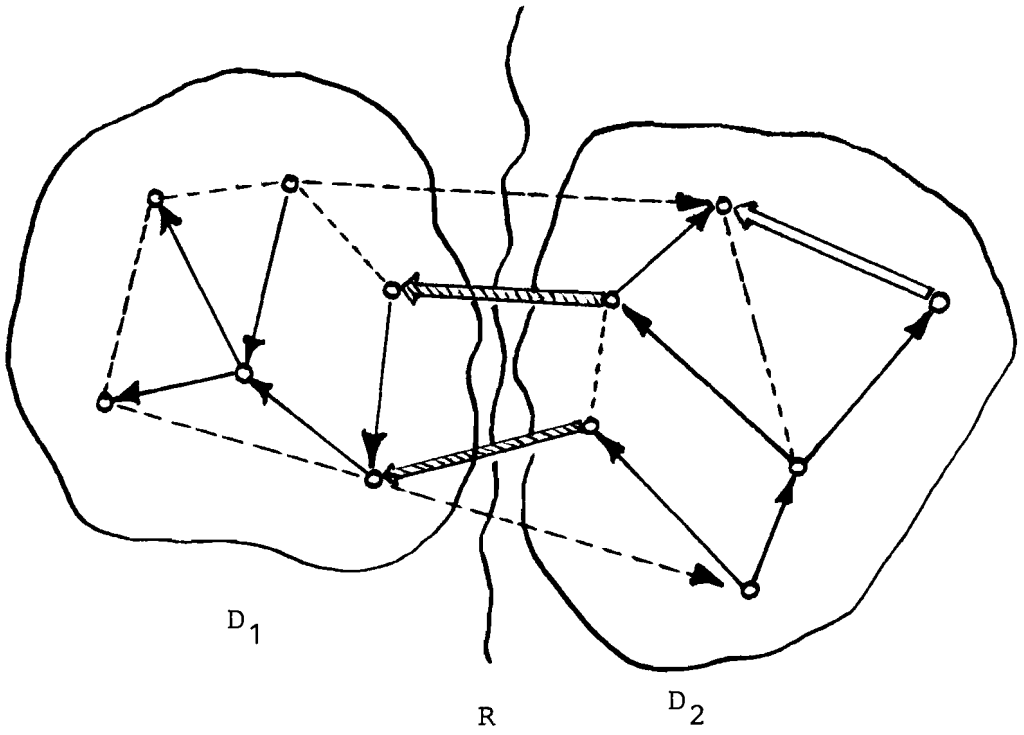
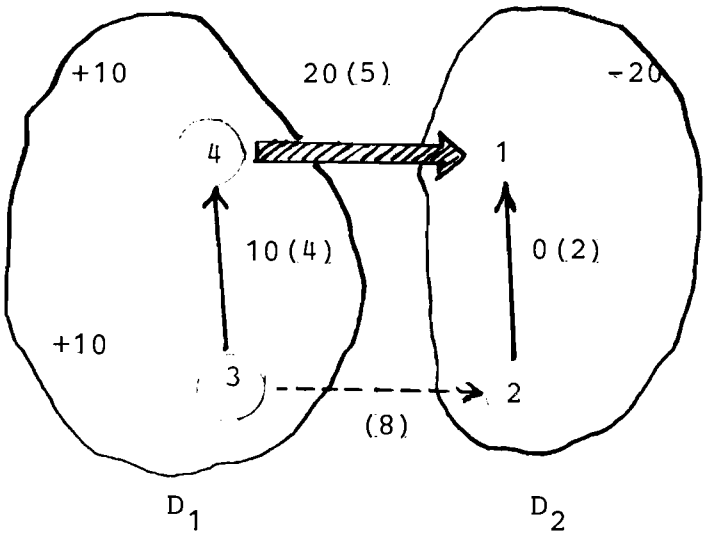


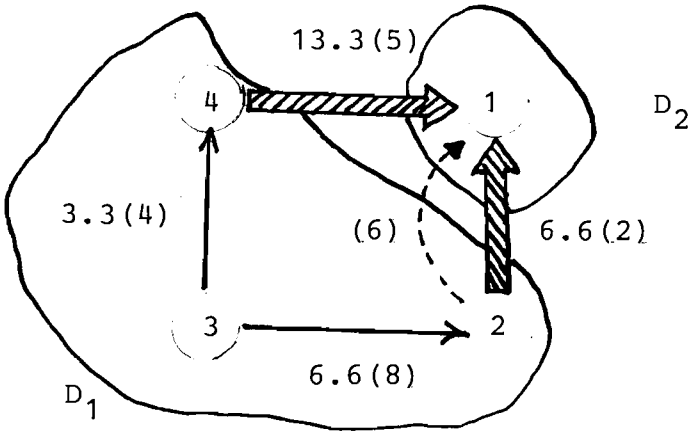
Figure 2.



Initial basis

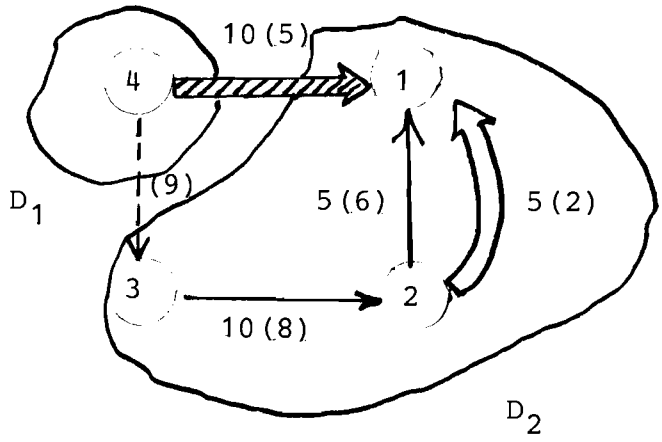
Y = 1

X(5) = 20 = Y·A(5)



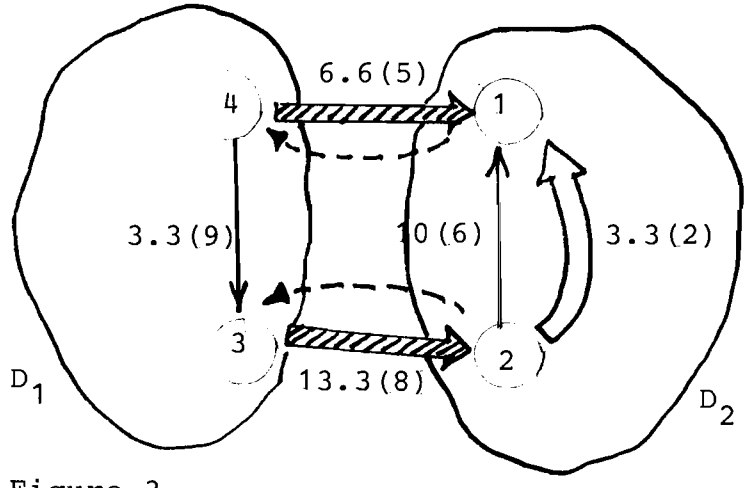
Y = 0.66

X(5) = 13.3 = Y·A(5)
X(2) = 6.6 = Y·A(2)



X(5) = 10 = Y·A(5)
X(2) = 5 = Y·A(2)

Y = 0.5



Optimal basis

Y = 0.33

X(5) = 6.6 = Y·A(5)
X(2) = 3.3 = Y·A(2)
X(8) = 13.3 = Y·A(8)

Figure 3.