4. THE MPS FILE

An MPS file is required for all problems to specify names for the variables and constraints, and to define the constraints themselves. In contrast to the relatively free format allowed in the SPECS file, a very fixed format must be used for the MPS file. (This means that each item of data must appear in specific columns.)

Various “header cards” divide the MPS file into several sections as follows:

```
NAME
ROWS

COLUMNS

RHS

RANGES (optional)

BOUNDS (optional)

ENDATA
```

Each header card must begin in column 1. The intervening card images (indicated by “." above) all have the following data format:

```
Columns  2-3  5-12  15-22  25-36  40-47  50-61
Contents  Key  Name0  Name1  Value1  Name2  Value2
```

In addition, “comment” cards are allowed; these have an asterisk “*” in column 1 and any characters in columns 2-22.

MPS format has become the industry standard. Files of this kind are recognized by all commercial mathematical programming systems (including MPS/360, MPSX, MPSX/370 and MPS III on IBM systems; APEX III and IV on CDC machines; FMPS on Univac systems; TEMPO on Burroughs systems). They may be created by hand, by your own special-purpose program, or by various commercial “matrix generators”, such as GAMMA, MAGEN and OMNI.

Beware that variations are inevitable in almost any “standard” format. Some restrictions in the format accepted by MINOS are listed later. Some extensions are also needed for nonlinear problems.

4.1 The NAME Card

```
NAME      MODELO01  (for example)
```

This card contains the word NAME in columns 1-4, and a name for the problem in columns 15-22. (The name may be from 1 to 8 characters of any kind, or it may be blank.) The name is used to label the solution output, and it appears on the first card of each basis file.

The NAME card is normally the first card in the MPS file, but it may be preceded or followed by comment cards.
4.2 The ROWS Section

ROWS
E FUN01
G FUN02 (for example)
L CAPITAL1
N COST

The general constraints are commonly referred to as rows. The ROWS section contains one card for each constraint (i.e., for each row). Key defines what type the constraint is, and Name0 gives the constraint an 8-character name. The various row-types are as follows:

<table>
<thead>
<tr>
<th>Key</th>
<th>Row-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>=</td>
</tr>
<tr>
<td>G</td>
<td>&gt;=</td>
</tr>
<tr>
<td>L</td>
<td>&lt;=</td>
</tr>
<tr>
<td>N</td>
<td>Objective</td>
</tr>
<tr>
<td>N</td>
<td>Free</td>
</tr>
</tbody>
</table>

(The 1-character Key may be in column 2 or column 3.)

Row-types E, G and L are easily understood in terms of a linear function \( a^T x \) and a right-hand side \( \beta \). They would be used to specify constraints of the form

\[
a^T x = \beta, \quad a^T x \geq \beta \quad \text{and} \quad a^T x \leq \beta
\]

respectively. (Nonzero elements of the row-vector \( a \) will appear in appropriate parts of the COLUMNS section, and if \( \beta \) is nonzero it will appear in the RHS section.)

Row-type N stands for “Not binding”, also known as “Free”. It is used to define the objective row, and also to prevent a constraint from actually being a constraint. (Note that \(-\infty \leq a^T x \leq +\infty \) is not really a constraint at all. Type N rows are implemented by giving them infinite bounds of this kind.)

The objective row is a free row that specifies the vectors \( c \) and \( d \) in the objective function \( F(x) + c^T x + d^T y \). It is taken to be the first free row, unless some other free row is specified by the OBJECTIVE keyword in the SPECS file.

The ROWS section need not contain any free rows if \( c = d = 0 \). If there are some nonlinear objective variables, the objective function will then be \( F(x) \) as defined by subroutine FUNOBJ. Otherwise, no objective function exists and MINOS will terminate at the first point that satisfies the constraints.

If the ROWS section does contain free rows but none of them is intended to be an objective row, then some dummy name such as OBJECTIVE = NONE should be specified in the SPECS file to prevent the first free row from being selected. (If the objective function is \( F(x) \) with no linear terms, OBJECTIVE = FUNOBJ would be a mnemonic reminder.)

Row-names for Nonlinear Constraints

The names of nonlinear constraints must be listed first in the ROWS section, and their order must be consistent with the computation of the array \( F(*) \) in subroutine FUNCON.

In particular, the objective row (if any) must appear after the list of nonlinear row names. For simplicity we suggest that potential objective rows be placed last:
4.3 The COLUMNS Section

ROWS
G FUNO1 nonlinear constraints first
G FUNO2
E LINO1 now linear constraints
E LINO2
N COSTO1 objective rows last
N COSTO2

4.3 The COLUMNS Section

<table>
<thead>
<tr>
<th></th>
<th>5. . . . 12</th>
<th>15. . . . 22</th>
<th>25. . . . . 36</th>
<th>40. . . . 47</th>
<th>50. . . . . 61</th>
<th>(fields)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMNS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X01</td>
<td>FUNO8</td>
<td>1.0</td>
<td>ROW09</td>
<td>-3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X01</td>
<td>ROW08</td>
<td>2.5</td>
<td>ROW12</td>
<td>1.123456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X01</td>
<td>ROW03</td>
<td>-11.11111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X02</td>
<td>FUNO2</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X02</td>
<td>COSTO1</td>
<td>5.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each variable \( x_j \) (say), the COLUMNS section defines a name for \( x_j \) and lists the nonzero entries \( a_{ij} \) in the corresponding column of the constraint matrix. The nonzeros for the first column must be grouped together before those for the second column, and so on. If a column has several nonzeros, it does not matter what order they appear in (as long as they all appear before the next column).

In general, Key is blank (except for comments), Name0 is the column name, and Name1, Value1 give a row name and value for some coefficient in that column. If there is another row name and value for the same column, they may appear as Name2, Value2 on the same card, or they may be on the next card.

If either Name1 or Name2 is blank, the corresponding value is ignored.

Values are read by MINOS using Fortran format E12.0. This allows values to be entered in several forms; for example, 1.2345678, 1.2345678E+0, 123.45678E-2 and 12345678E-07 all represent the same number. It is usually best to include an explicit decimal point.

Beware that spaces within the value fields are the same as 0's (on most computer systems). In particular, this means that if an exponent like E-2 appears then it must be right-justified in the value field. For example, the two values

\[
\begin{align*}
1.23E-02 \\
1.23E-2
\end{align*}
\]

are not the same if the decimal point is in column 30 in both cases. The second value is actually \(1.23E-20\).

In the example above, the variable called X01 has 5 nonzero coefficients in the constraints named FUNO8, ROW09, ROW08, ROW12 and ROW03. The row names and values may be in an arbitrary order, but they must all appear before the entries for column X02.

There is no need to specify columns for the slack variables; they are incorporated implicitly.
Nonlinear Variables

Nonlinear variables must appear first in the COLUMNS section, ordered in a manner that is consistent with the array \( X(*) \) in the user subroutines FUNOBJ and/or FUNCON. In the example

\[
\begin{align*}
\text{minimize} & \quad (x + y + z)^2 + 3z + 5w \\
\text{subject to} & \quad x^2 + y^2 + z = 2 \\
& \quad z^4 + y^4 + w = 4 \\
& \quad 2x + 4y \geq 0 \\
& \quad z \geq 0, \quad w \geq 0
\end{align*}
\]

we have three nonlinear objective variables \((x, y, z)\), two nonlinear Jacobian variables \((x, y)\), one linear variable \(w\), two nonlinear constraints, one linear constraint, and some simple bounds. The nonlinear constraints and variables should always be ordered in a similar way, at the top left-hand corner of the constraint matrix. The latter is therefore of the form

\[
A = \begin{pmatrix} J_k & A_1 \\ A_2 & A_3 \end{pmatrix}
\]

where \( J_k \) is the Jacobian matrix. The variables associated with \( J_k \) and \( A_2 \) must appear first in the COLUMNS section, and their order must be consistent with the array \( X(*) \) in subroutine FUNCON. Similarly, entries belonging to \( J_k \) must appear in an order that is consistent with the array \( G(*) \) in subroutine FUNCON.

For convenience, let the first \( n_1 \) columns of the constraint matrix be

\[
\begin{pmatrix} J_k \\ A_2 \end{pmatrix} = \begin{pmatrix} j_1 & j_2 & \ldots & j_{n_1} \\ a_1 & a_2 & \ldots & a_{n_1} \end{pmatrix},
\]

where \( j_1 \) is the first column of \( J_k \) and \( a_1 \) is the first column of \( A_2 \). The coefficients of \( j_1 \) and \( a_1 \) must appear before the coefficients of \( j_2 \) and \( a_2 \) (and so on for all columns). Usually, those belonging to \( j_1 \) will appear before any in \( a_1 \), but this is not essential. (If certain linear constraints are made nonlinear at a later date, this means that entries in the COLUMNS section will not have to be reordered. However, the corresponding row names will need to be moved towards the top of the ROWS section.)

If JACOBIAN = DENSE, the elements of \( J_k \) need not be specified in the MPS file. If JACOBIAN = SPARSE, all nonzero elements of \( J_k \) must be specified. Any variable coefficients should be given a dummy value, such as zero. These dummy entries identify the location of the elements; their actual values will be computed later by subroutine FUNCON or by finite differences.

If all constraint gradients are known (DERIVATIVE LEVEL = 2 or 3), any Jacobian elements that are constant may be given their correct values in the COLUMNS section, and then they need not be reset by subroutine FUNCON. This includes values that are identically zero—such elements do not have to be specified anywhere (in the MPS file or in FUNCON). In other words, Jacobian elements are assumed to be zero unless specified otherwise.

Note that \( X(*) \) need not have the same dimension in subroutines FUNOBJ and FUNCON (i.e., the parameter \( N \) may differ), in the event that different numbers are specified by the NONLINEAR OBJECTIVE and NONLINEAR JACOBIAN keywords. However, the shorter set of nonlinear variables must occur at the beginning of the longer set, and the ordering of variables in the COLUMNS section must match both sets.

A nonlinear objective function will often involve variables that occur only linearly in the constraints. In such cases we recommend that the objective variables be placed after the Jacobian variables in the COLUMNS section, since the Jacobian will then be as small as possible. (See the variable \( z \) in the example above.)
4.4 The RHS Section

1 5......12 15......22 25......36 40......47 50......61

RHS

RHS01  FUN01  1.0  ROW09  -3.0
RHS01  ROW08  2.5  ROW12  1.123456
RHS01  ROW03  -11.111111
RHS02  FUN02  1.0
RHS02  FUN04  5.0

This section specifies the elements of $b_1$ and $b_2$ in (2)-(3). Together these vectors comprise what is called the right-hand side. Only the nonzero coefficients need to be specified. They may appear in any order. The format is exactly the same as in the COLUMNS section, with Name0 giving a name to the right-hand side.

If $b_1 = 0$ and $b_2 = 0$, the RHS header card must appear as usual, but no rhs coefficients need follow.

The RHS section may contain more than one right-hand side. The first one will be used unless some other name is specified in the SPECS file.

4.5 The RANGES Section (Optional)

1 5......12 15......22 25......36 40......47 50......61

ROWS

E  FUN01
E  FUN02
G  CAPITAL1
L  CAPITAL2

COLUMNS

RHS

RHS01  FUN01  4.0  FUN02  4.0

RANGES

RANGE01  FUN01  1.0  FUN02  -1.0
RANGE01  CAPITAL1  1.0  CAPITAL2  1.0

Ranges are used for constraints of the form

$$ l \leq a^T z \leq u,$$

where both $l$ and $u$ are finite. The range of the constraint is $r = u - l$. Either $l$ or $u$ is specified in the RHS section (as $b$ say), and $r$ is defined in the RANGES section. The resulting $l$ and $u$ depend on the row-type of the constraint and the sign of $r$ as follows:

<table>
<thead>
<tr>
<th>Row-type</th>
<th>Sign of $r$</th>
<th>Lower limit, $l$</th>
<th>Upper limit, $u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>+</td>
<td>$b$</td>
<td>$b +</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>$b -</td>
<td>r</td>
</tr>
<tr>
<td>G</td>
<td>+ or -</td>
<td>$b$</td>
<td>$b +</td>
</tr>
<tr>
<td>L</td>
<td>+ or -</td>
<td>$b -</td>
<td>r</td>
</tr>
</tbody>
</table>
The format is exactly the same as in the COLUMNS section, with Name0 giving a name to the range set. The constraints listed above will have the following limits:

\[
\begin{align*}
4.0 & \leq \text{FUN01} & \leq 5.0, \\
3.0 & \leq \text{FUN02} & \leq 4.0, \\
4.0 & \leq \text{CAPITAL1} & \leq 5.0, \\
3.0 & \leq \text{CAPITAL2} & \leq 4.0.
\end{align*}
\]

The RANGES section may contain more than one set of ranges. The first set will be used unless some other name is specified in the SPECS file.

4.6 The BOUNDS Section (Optional)

\[
\begin{align*}
1 & \quad 5 \ldots 12 & 15 \ldots 22 & 25 \ldots \ldots 36 \\
\text{BOUNDS} & & & \\
\text{UP BOUND01} & \text{X01} & 4.0 \\
\text{UP BOUND01} & \text{X02} & 4.0 \\
\text{LO BOUND01} & \text{X04} & -1.0 \\
\text{UP BOUND01} & \text{X04} & 4.0 \\
\text{FR BOUND01} & \text{X06} & 4.0 \\
\text{UP BOUND01} & \text{X06} & 4.0
\end{align*}
\]

The default bounds on all variables \(x_j\) (excluding slacks) are \(0 \leq x_j \leq \infty\). If necessary, the default values 0 and \(\infty\) can be changed in the SPECS file to \(l \leq x_j \leq u\) by the LOWER and UPPER keywords respectively.

If uniform bounds of this kind are not suitable, any number of alternative values may be specified in the BOUNDS section. As usual, several sets of bounds may be given, and the first set will be used unless some other name is specified in the SPECS file.

In this section, Key gives the type of bound required, Name0 is the name of the bound set, and Name1 and Value1 are the column name and bound value. (Name2 and Value2 are ignored.)

Let \(l\) and \(u\) be the default bounds just mentioned, and let \(x\) and \(b\) be the column and value specified. The various bound-types allowed are as follows:

<table>
<thead>
<tr>
<th>Key</th>
<th>Bound-type</th>
<th>Resulting bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>LO</td>
<td>Lower bound</td>
<td>(b \leq x \leq u)</td>
</tr>
<tr>
<td>UP</td>
<td>Upper bound</td>
<td>(l \leq x \leq b)</td>
</tr>
<tr>
<td>FX</td>
<td>Fixed variable</td>
<td>(b \leq x \leq b) (i.e., (x = b))</td>
</tr>
<tr>
<td>FR</td>
<td>Free variable</td>
<td>(-\infty \leq x \leq +\infty)</td>
</tr>
<tr>
<td>WI</td>
<td>Minus infinity</td>
<td>(-\infty \leq x \leq u)</td>
</tr>
<tr>
<td>PL</td>
<td>Plus infinity</td>
<td>(l \leq x \leq +\infty)</td>
</tr>
</tbody>
</table>

The effect of the examples above is to give the following bounds:

\[
\begin{align*}
l & \leq \text{X01} \leq 4.0 \\
l & \leq \text{X02} \leq 4.0 \\
-1.0 & \leq \text{X04} \leq 4.0 \\
-\infty & \leq \text{X06} \leq 4.0
\end{align*}
\]

Note that types FR, WI, or PL should always be used to specify "infinite" bounds; they imply values of \(\pm 10^{20}\), which are treated specially at certain times.
Nonlinear Problems

It is often essential to use bounds to avoid singularities in the nonlinear functions. For example, if an objective function involves $\log x_j$, a bound of the form $x_j \geq 10^{-4}$ may be necessary to avoid evaluating the objective function at zero or negative values of $x_j$. (Subroutine FUNOBJ is usually not called until a feasible point has been found. Note that $x$ is regarded as feasible if it satisfies its bounds to within the FEASIBILITY TOLERANCE $\epsilon$. Thus, it would not be safe to specify the bound $x_j \geq 10^{-6}$ if $\epsilon$ retained its default value $\epsilon = 10^{-8}$.)

Beware that subroutine FCN sometimes will be called before the nonlinear variables satisfy their bounds. If this causes difficulty, one approach is to specify feasible values for the offending variables in the INITIAL bounds set described next.

The INITIAL Bounds Set

In general, variables will initially have the value zero, if zero lies between the associated upper and lower bounds. Otherwise, the initial value will be the bound closest to zero.

The name INITIAL is reserved for a special bounds set that may be used to assign other initial values. The INITIAL bounds set must appear after any normal bound sets (if any); a warning is given if it is the first set encountered after the BOUNDS card.

The INITIAL bounds set also influences CRASH during construction of an initial basis. Broadly speaking, CRASH favors certain variables, ignores certain others, and treats the remainder as neutral. The following example illustrates the various cases:

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FX INITIAL</td>
<td>X1</td>
<td>1.0</td>
</tr>
<tr>
<td>FX INITIAL</td>
<td>X2</td>
<td>2.0</td>
</tr>
<tr>
<td>LO INITIAL</td>
<td>X3</td>
<td></td>
</tr>
<tr>
<td>UP INITIAL</td>
<td>X4</td>
<td></td>
</tr>
<tr>
<td>MI INITIAL</td>
<td>X5</td>
<td>5.0</td>
</tr>
<tr>
<td>PL INITIAL</td>
<td>X6</td>
<td>6.0</td>
</tr>
</tbody>
</table>

1. During gradient checking and evaluation of the initial Jacobian, the value of X1 will be 1.0. X1 will then be favored by CRASH for inclusion in the initial basis. (Free rows and columns will also be favored.)

2. X2 will initially be superbasic at the value 2.0. (If the number of FX INITIALs has already reached the SUPERBASICS LIMIT, X2 will initially be nonbasic at the same value 2.0.)

3. X3 and X4 will initially be nonbasic at their respective lower and upper bounds (or at value zero if both bounds are infinite).

4. X5 and X6 will initially be nonbasic at the specified values 5.0 and 6.0.

The last five bound types (FX, LO, UP, MI, PL) prevent the associated variables from being included in the initial basis.

FX INITIAL or FX INITIAL should be used if good values are known for variables that are likely to lie between their bounds in an optimal solution. (Type FX is preferred if many such values are to be specified; however, the values may change when the basic variables are reset to satisfy $Ax + Is = 0$. Type FX guarantees the specified starting value, but should not be used excessively if the optimal solution is likely to be close to a vertex.)

LO INITIAL or UP INITIAL should be used for variables that are likely to be on their lower or upper bound at a solution.

MI INITIAL and PL INITIAL are included for completeness.
As with normal bound sets, variables may be listed in any order. (For each entry a linear search is made through the column names, starting at the name on the previous entry. Thus, for large problems it helps to follow the order of the variables in the COLUMNS section, at least to some extent.)

The INITIAL bounds set is ignored if a basis file is supplied.

4.7 Comment Cards

Any card in the MPS file may contain an asterisk "*" in column 1 and arbitrary data in columns 2–61. Such cards will be treated as comments. They will appear in the printer listing but will otherwise be ignored.

4.8 Restrictions and Extensions in MPS Format

1. Blanks are significant in the 8-character name fields. We recommend that all names be left-justified with no imbedded blanks. In particular, names referred to in the SPECS file must be left-justified in the MPS file; for example, OBJECTIVE = COST02 specifies an 8-character name whose last two characters are blank.

2. Comments ideally should use only columns 1–61 as noted above.

3. Scale factors cannot be entered in the ROWS section.

4. It does not matter if there is no row of type M.

5. There must be at least one row in the ROWS section, even for problems with no general constraints. (It may have row-type M.)

6. Nonlinear constraints must appear before linear constraints in the ROWS section.

7. Markers such as INTORG and INTEND are not recognized in the COLUMNS section.

8. Numerical values may be entered in E or F format. Spaces within the 12-character fields are treated as if they were 0's.


10. If RANGES and BOUNDS sections are both present, the RANGES section must appear first.

11. In the BOUNDS section, if an UP entry specifies a zero upper bound, the corresponding lower bound is not affected. (Beware—in some MP systems, the lower bound is converted to \(-\infty\).)

12. The bounds name INITIAL has a special meaning.